UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS


# Research and Development of a General Purpose Instrument DAQ-Monitoring Platform applied to the CLOUD/CERN experiment

*" Documento Definitivo"*

**Doutoramento em Engenharia Física**

António Miguel da Cruz Baptista Dias

Tese orientada por:
Prof. António Joaquim Rosa Amorim Barbosa
Prof. António Rodrigues Tomé

Documento especialmente elaborado para a obtenção do grau de doutor

2020

UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS


Ciências
ULisboa

# Research and Development of a General Purpose Instrument DAQ-Monitoring Platform applied to the CLOUD/CERN experiment

**Doutoramento em Engenharia Física**

António Miguel da Cruz Baptista Dias

**Tese orientada por:**

**Prof. António Joaquim Rosa Amorim Barbosa**

**Prof. António Rodrigues Tomé**

Júri:

Presidente:

- Doutor José Manuel de Nunes Vicente e Rebordão, Investigador Coordenador
  Faculdade de Ciências da Universidade de Lisboa

Vogais:

- Doutor Helmut Wolters, Investigador Auxiliar
  Faculdade de Ciências e Tecnologia da Universidade de Coimbra;
- Doutor Jaime Enrique Villate Matiz, Professor Auxiliar
  Faculdade e Engenharia da Universidade do Porto
- Doutor José Manuel de Nunes Vicente e Rebordão, Investigador Coordenador
  Faculdade de Ciências da Universidade de Lisboa;
- Doutor António Joaquim Rosa Amorim Barbosa, Professor Catedrático
  Faculdade de Ciências da Universidade de Lisboa (orientador)
- Doutor António Casimiro Ferreira da Costa, Professor Associado
  Faculdade de Ciências da Universidade de Lisboa
- Doutor Nuno Filipe Fiuza de Barros, Professor Auxiliar Convidado
  Faculdade de Ciências da Universidade de Lisboa.

Documento especialmente elaborado para a obtenção do grau de doutor

Partially supported by a Marie Curie Initial Training Network Fellowship of the European Community's Seventh Framework Programme under contract number (PITN-GA-2012-316662-CLOUD-TRAIN.)

2020

# Acknowledgments

A number of acknowledgements are required. While there is only one author on this thesis, it fails to capture the valuable and (most importantly) consistent effort of those who believe and wished to see this work bear fruit.

First off a warm thank you to my supervisors, Professor António Amorim and Professor António Tomé, who provided valuable guidance and constantly adjusted the rudder of this journey, while providing a freedom to navigate these often unforgiving waters found in the world of research and development. A safe port would never have been reached without these expertly timed contributions.

A special thanks should also be given to all the CLOUD team at CERN, specifically Dr. Jasper Kirkby, Serge Mathot and Antti Onella. To continue the sailing analogy, they were the actual rudder, it was with their generosity and patience with my usage the resources available in CLOUD that allowed me to find a direction for this work. Inside the CLOUD collaboration, a very special thanks is also sent to the CLOUD-TRAIN project, specifically Katja Ivanova and Joachim Curtius, with constant guidance in this bureaucracy-ridden world of ours.

To all my companions in CLOUD that I've had the privilege to work with during my time spent there (and beyond!), you've provided me with experiences that consistently changed me and allowed me to be a part of the human side of science. I specifically direct my thanks to Sebastian Ehrhart, Alex Vogel, Christina Williamson, Martin Heinritzi, Andrea Wagner, Rima Baalbaki, Chao Yan, Anne Bermhammer, Dominik Soltzenburg, Robert Wagner, Jasmin Troestl, Leonid Nichman, Claudia Fuchs, Kamalika Sengupta, Carla Frege, Emma Jervinen, Hamish Gordon, Xucheng He, Changyuk Kim, Jonathan Duplissy and Mao Xiao; You were the hull of this ship and, either willingly or unwillingly, kept me above water the whole time. A very special and personal thank you goes out to Mario Simon and Lubna Dada, Mario for being CLOUD's concierge and a good friend through all of those dreary late nights manning the controls and to Lubna for never missing an opportunity to keep me humble and for their capacity for consistently remind me that in science (as with most things in life) it's not just about the results it's also about the journey.

To my family, for their patience and sacrifice, their unconditional belief in my abilities and the sheer will they poured into this endeavor alongside me, I am infinitely indebted. You are my main sail without which no amount of currents, wind or planning could result in a good result. Specific thanks to my mother Ana Dias, for always being there and being the buffer for all my frustrations. To my brother, Duarte Dias, one of the smartest and level-headed I know, for the time lent to turn all my ramblings into actionable content. To my aunt, Professor Margarida Cruz, for all the work put into ensuring the work performed to the required standards.

Lastly a very special thanks to all my friends who have been present in my life and have been there for this period and hopefully for longer. You ALL know who you are; you are the wind on my sails, you give me the strength to go through all these sanity draining efforts and my only hope is that I am able to contribute individually to your

well being as much as you contribute to mine. You make the highs higher and the lows not as low. Specific thanks to Ricardo ”Hellgardia” Silva, João ”Moontouched” Branco, João ”Kekulé” Vitorino, Pedro ”Polyphonia” Neves, João ”Jacav” Valverde, Rui ”Enyozjeh” Pedras, Marta ”Sardaniscas” Pontes, Pedro ”Ordep” Lucas, thanks for all the games, jokes and times well spent; to Luís Rodrigues, Rita Pinto, Joaquim Pereira, Joana Lopes and Mariana Duarte, thanks for all the outings, trips, laughs and support.

It was with this boat built by the contributions of all the aforementioned individuals, that I managed to navigate the challenges of this work and was able to produce this thesis. Thank you everyone!

This work was partially supported by a Marie Curie Initial Training Network Fellowship of the European Community's Seventh Framework Programme under contract number (PITN-GA-2012-316662-CLOUD-TRAIN.)

# Resumo

O processo de investigação científica moderno requer que tanto experimentalistas como administradores de sistemas dediquem uma parte significativa do seu tempo a criar estratégias para aceder, armazenar e manipular instrumentos científicos e os dados que estes produzem. Este é um desafio crescente considerando o aumento de colaborações que necessitam de vários instrumentos, investigação em áreas remotas e instrumentos científicos com constantes alterações. O DAQBroker é uma nova plataforma desenhada para a monitorização de instrumentos científicos e ao mesmo tempo fornece métodos simples para qualquer utilizador aceder aos seus dados. Os dados podem ser guardados em uma ou várias bases de dados locais ou remotas utilizando os gestores de bases de dados mais comuns (MySQL, PostgreSQL, Oracle). Esta plataforma também fornece as ferramentas necessárias para criar e editar versões virtuais de instrumentos científicos e manipular os dados recolhidos dos instrumentos, independentemente do grau de conhecimento que o utilizador tenha com o(s) instrumento(s) utilizado(s). Séries temporais guardadas numa base de dados DAQBroker beneficiam de um conjunto de métodos estatísticos para a classificação, comparação e detecção de eventos, determinação das séries com maior influência e os sub-períodos experimentais com maior actividade. Esta tese apresenta a arquitectura da plataforma, os resultados de diversos testes de esforço efectuados em ambientes controlados e um caso real da sua utilização na experiência CLOUD, no CERN, Suíça. São estudados também os métodos de análise de séries temporais, tanto singulares como multivariadas aplicados na plataforma.

**Palavras-chave:** Estatística de Séries Temporais, Desenvolvimento de Software, Arquitectura de Bases de Dados, Applicações Web, Aquisição de Dados

# Abstract

The current scientific environment has experimentalists and system administrators allocating large amounts of time for data access, parsing and gathering as well as instrument management. This is a growing challenge since there is an increasing number of large collaborations with significant amount of instrument resources, remote instrumentation sites and continuously improved and upgraded scientific instruments. DAQBroker is a new software designed to monitor networks of scientific instruments while also providing simple data access methods for any user. Data can be stored in one or several local or remote databases running on any of the most popular relational databases (MySQL, PostgreSQL, Oracle). It also provides the necessary tools for creating and editing the meta data associated with different instruments, perform data manipulation and generate events based on instrument measurements, regardless of the user's know-how of individual instruments. Time series stored in a DAQBroker database also benefit from several statistical methods for time series classification, comparison and event detection as well as multivariate time series analysis methods to determine the most statistically relevant time series, rank the most influential time series and also determine the periods of most activity during specific experimental periods. This thesis presents the architecture behind the framework, assesses the performance under controlled conditions and presents a use-case under the CLOUD experiment at CERN, Switzerland. The univariate and multivariate time series statistical methods applied to this framework are also studied

**Keywords:** Time series statistics, Software development, Database architecture, Web applications, Data Acquisition

# Resumo Alargado

A criação de sistemas de aquisição de dados aplicados a conjuntos de instrumentos científicos é uma área pouco aprofundada e para a qual existem poucas soluções integradas desde a colecção de dados até download/exibição dos dados a um utilizador final. Estas soluções muitas vezes variam em execução, devido às idiossincrasias da arquitectura utilizada o que limita o seu uso fora dos ambientes para os quais foram desenvolvidas. Outras soluções são desenvolvidas em ambiente fechado com o objectivo de explorar financeiramente a plataforma, limitando o número de utilizadores e colaborações que as usam. O aumento do número de colaborações científicas e a necessidade de melhor compreender processos fisico-químicos cria a necessidade de empregar um número cada vez maior e mais variado de instrumentos científicos para o seu estudo. Ao mesmo tempo, a proliferação de dispositivos ligados à Internet e os valores e conceitos promovidos pelo movimento *Internet of Things* (IoT) promovem o desenvolvimento de soluções integradas que facilitem o acesso de utilizadores tanto aos dados dos seus instrumentos, como aos dados de outros instrumentos associados, sejam eles processados ou não. Todas as necessidades apresentadas tornam essencial a existência de uma plataforma integrada em todos os aspectos da monitorização de instrumentos científicos, fornecendo soluções universais ao nível do armazenamento de informação e monitorização e primitivas para a o acesso a dados e controlo dos instrumentos monitorizados.

Para a criação de uma ferramenta integrada de monitorização de instrumentos científicos, é necessário garantir o funcionamento de várias funcionalidades: comunicação, identificação de fontes de dados, armazenamento de dados e acesso a dados. Para o aspecto de comunicação da plataforma são estudados vários conceitos possíveis de utilização, focando a relevância no seu grau de utilização e flexibilidade de acesso. Devido à necessidade de possivelmente garantir comunicação com utilizadores em qualquer parte do mundo foi establecido que a comunicação por TCP/IP fosse utilizada, sendo introduzido um protocolo de comunicação entre diferentes utilizadores para a actualização de informação. Quanto à identificação de fontes de dados, foi introduzido o conceito da representação virtual do instrumento científico, dividindo o instrumento em três blocos distintos que se colocam dentro de cada um. Começando num bloco geral que descreve o instrumento e o seu objectivo, dentro do qual se inserem blocos intermédios que identificam conjuntos de dados que são recolhidos através da mesma interface e dentro dos quais se inserem canais de dados que descrevem um *stream* individual de uma variável no tempo que está a ser recolhida. Relativamente ao aspecto de armazenamento de dados, foram estudados diferentes sistemas de armazenamento de dados, focando-se não só na flexibilidade de armazenamento mas também na necessidade de apresentar uma estructura uniforme que permita a separação distinta entre diferentes esforços científicos. O sistema escolhido acabou por ser o relacional, que apesar de fornecer uma estructura rígida de armazenamento, permite uma clara separação entre esforços científicos, permite pesquisas complexas sobre a estructura e está optimizada para realizar pedidos de elevado volume. Finalmente, relativamente ao aspecto de acesso e manipulação de dados, introduziram-se várias alternativas para fornecer ferramentas para os utilizadores criarem, editarem e manipularem as representações virtuais dos seus instrumentos, focando-se na necessidade de criar ferramentas flexiveis, de desenvolvimento contínuo e que possam ser utilizadas por qualquer utilizador em qualquer máquina. Como tal, foi escolhido o desenvolvimento de uma aplicação web para fornecer a interface básica de representação do estado dos dados guardados. Esta aplicação fará uso de um *Application Programming Interface* (API) usando um modelo de *REpresentational*

*State Transfer* (REST) fornecida pela plataforma para realizar alterações na estructura de dados e representações virtuais de instrumentos.

Um factor importante na avaliação da viabilidade da plataforma consiste em medir o seu desempenho em diferentes arquitecturas para determinar os limites da sua utilização. Para tal, são introduzidas duas arquitecturas com diferentes níveis de potência computacional, dentro da qual é instalada a plataforma para realizar um conjunto de testes para identificar os limites de monitorização de instrumentos científicos. O primeiro teste consiste em testar o efeito que o número de instrumentos monitorizados tem sobre o funcionamento tanto da plataforma no geral, como na arquitectura onde foi instalada. Para este teste são analisados o tempo de aquisição de dados, a utilização de CPU, RAM e utilização de disco para periodos semelhantes usando um número crescente de instrumentos. O segundo teste consiste em estudar os limites de armazenamento de dados em tempo real da plataforma. Para tal é calculado o diferencial de tempo existente entre os dados armazenados e os dados produzidos por instrumentos com diferentes granularidades temporais. O último teste realizado consiste em testar os limites de armazenamento de dados. Para tal é medido o tempo de pedidos de dados de um instrumentos com um crescente número de dados armazenados.

Para além de avaliar o desempenho da plataforma num ambiente controlado, também se torna necessário avaliar o uso da plataforma num exemplo real. Para tal, a plataforma foi instalada na experiência CLOUD, no CERN, Suíça, para monitorizar os dados recolhidos pelos instrumentos durante o período experimental do Outono de 2017. Durante este período foram identificadas e avaliadas as principais funcionalidades fornecidas, focando a análise nas limitações da flexibilidade da aplicação e na identificação de pontos onde o desenvolvimento poderá ser melhorado. Paralelamente foram identificados um conjunto de análise de dados *offline* e monitorização em tempo real que foram facilitados com o uso da plataforma.

A plataforma apresentada descreve um método de recolher, armazenar e manipular dados de diversos instrumentos de uma maneira universal. Esta natureza universal dos daods monitorizados permite que seajm criados e aplicados diversos algoritmos para a análise de séries temporais sobre esses dados com o objectivo de fornecer aos utilizadores informações sobre a qualidade dos dados recolhidos, a relevância dos dados recolhidos, descrever as relações entre os dados recolhidos por um instrumento e os dados recolhidos por outros instrumentos e realizar uma análise de um determinado periodo experimental. Para tal, são estudados um conjunto de métodos de análise de séries temporais, métodos esses que são aplicados sobre os dados recolhidos durante o Outono de 2017 pela experiência CLOUD para mostrar os resultados obtidos quando aplicados a um conjunto de dados reais. Inicialmente são apresentados um conjunto de testes de hípotese que pretendem classificar séries temporais individuais sobre um diverso conjunto de características. Seguidamente são estudados e apresentados diversos métodos de comparação de séries temporais, focando-se não só na qualidade da comparação, mas também na eficiência computacional do método usado, visto que a utilização de tais métodos poderão ser utilizados por vários utilizadores um número variado de vezes e o uso de um algoritmo pouco eficiente por vários utilizadores poderá por em causa o bom funcionamento da plataforma. Outro ângulo de análise de séries temporais foca-se na identificação de eventos em diferentes séries temporais. A natureza universal dos dados guardados na plataforma obriga a análise a utilizar

métodos de detecção não supervisionados, focando-se num método que permite a determinação de alterações de estado de séries temporais baseado na alteração de entropia da série entre diferentes intervalos de tempo. A análise de séries temporais culmina na implementação de um método de análise de séries temporais multivariadas que tem como objectivo a realização de um resumo de um período experimental baseado em testes estatísticos aplicados a um conjunto de séries temporais. O resumo pretendido está divido em três partes: identificação das séries temporais estatisticamente mais relevantes, análise das relações entre as diferentes séries temporais e determinação dos sub-períodos experimentais mais relevantes. A primeira parte da análise consiste em identificar as séries temporais mais relevantes segundo a sua distribuição de frequências. A segunda parte consiste em identificar e exibir as relações entre as diferentes séries temporais através dos diversos métodos de comparação estudados anteriormente. A última parte consiste na determinação dos sub períodos experimentais mais relevnates através de um algorítmo de detecção de eventos baseado no algoritmo utilizado anteriormente para séries individuais. Também é estudada a integração deste resumo como uma adição à plataforma e quais os passos necessários a realizar para o conseguir.

# Contents

# List of Tables

# List of Figures

# Nomenclature

**Acronyms**

ACF Auto Correlation Function

ACID Atomicity, Consistency, Isolation, Durability

ADF Augmented-Dickey-Fuller

AJAX Asynchronous Javascript And XML

AP Alpha-pinene ($\alpha$-pinene)

API Application Programming Interface

ARIMA Autoregressive Integrated Moving Average

ASCII American Standard Code for Information Interchange

AWS Amazon Web Services

CAN Controller Area Network

CDN Content Delivery Network

CERN European Organization for Nuclear Research

CIMS Chemical Ionization Mass Spectrometry

CLOUD Cosmics Leaving Outdoor Droplets

CPU Central Processing Unit

CSS Cascading Style Sheets

DAQ Data Acquisition

DB Database

DBMS Database Management System

DCS Distributed Control System

DDBMS Distributed Database Management System

DFT Discrete Fourier Transform

DTW Dynamic Time Warping

FFT Fast Fourier Transform

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

IT Information Technology

JSON JavaScript Object Notation

KPPS Kwiatkowski–Phillips–Schmidt–Shin

LSS Local Similarity Statistic

ME Maximum Entropy

MK Mann-Kendall

MS Mass Spectrometry

MVSG Multivariate Summary Generator

NTP Network Transfer Protocol

PACF Partial Autocorrelation Function

PCA Principal Component Analysis

PSM Particle Size Magnifier

RAM Random Access Memory

RDBMS Relational Database Management System

RDS Relational Database Service

REST Representational State Transfer

RH Relative Humidity

ROM Read Only Memory

RPS Reads Per Second

SA Sulphuric Acid

SCADA Supervisory Control and Data Acquisition

SMPS Scanning Mobility Particle Sizer

SQL Structured Query Language

SVD Single Value Decomposition

TCP Transport Control Protocol

UDP User Datagram Protocol

USB Universal Serial Bus

UV Ultra Violet

VME Versa Module Europa

WPS Writes Per Second

XSRF Cross-Site Request Forgery

XSS Cross-Site Scripting

**Symbols**

$\alpha$ Maximum Entropy Distance Limit

$\mathbb{C}$ Cost Matrix

$\mathbb{E}$ Excpected Value

$\mathscr{C}$ Coherence

$\mathscr{F}$ Fourier Transform

$\mathscr{G}$ Cross-spectral Density

$\pi^+$ Pion

$\psi$ Base Function

$\sigma$ Variance Explained Limit

$\mathscr{C}$ Coherence Matrix

$\mathbf{X}$ Mualtivariate Time Series

$ACF$ Autocorrelation Function

$AR$ Autoregressive Process

$d$ Distance Measure

$E$ Entropy

$J$ Nucleation Rate

$N$ Time Series Sample Size

$O$ Computational Complexity

$Q$ Modularity of a Network

$r$ Correlation Coefficient

| | |
|---|---|
| $s$ | Similarity Measure |
| $w$ | Differenced Time Series |
| $X$ | Time series |
| $x_i$ | Time Series Value at Sampling Time $i$ |
| $xr$ | Cross-correlation |

# Chapter 1

# Introduction

Data acquisition (DAQ) systems are a constantly evolving medium that is ubiquitous in any field of science [1–5]. All modern scientific instruments either possess or require some sort of DAQ system ranging from simple hardware buffers [6] to fully implemented control and monitoring systems [7]. Often, instruments with different measurement purposes are used together to produce a better understanding of the underlying process being measured [8–10]. This reality is becoming more and more common as collaborations between different scientific institutes are increasing [11, 12]. With the proliferation of the Internet and the recent trend of the Internet-of-Things (IoT) enabled devices, information is now expected to be available with only a few clicks [13, 14]. However, the growing number of instruments, multi-instrument sites and collaborations will often have data being stored in instrument files [15, 16] and using specific software tailored only to the instrument set available for the collaboration [17–19]. Often companies offer closed source DAQ solutions for their own products [20, 21], limiting the spectrum of data analysis and the research carried out to that of the companies' technologies, products and/or instruments.

An alternative approach to these closed DAQ systems must exist to provide the following functionalities for any instrument user[22]:

- **Universal data storage** - identify, collect and store data from any instrument and ideally from any data source,
- **Universal data monitoring** - provide users with the tools to visualize the instrument's data output even if they are not familiar with the instrument,
- **Primitives for instrument data access** - provide functions to integrate with existing systems to access other instruments' data,
- **Primitives for instrument control** - allow sending specific messages, ideally via any imaginable medium, that trigger changes to an instrument's settings.

A universal data storage format can be achieved by creating virtual instrument representations, transferable between sites using the same DAQ system, or stored in a global repository for later use, thus facilitating future data acquisition efforts. It also eases the experience of data monitoring, making it easier for users to request visualiza-

tion (or data) from other instruments they are not experienced with. The data access and control primitives would allow the creation of third-party systems that could deal with the specific data processing and control needs of the instrument set that is used, while still maintaining the same known format for all users. A DAQ system that provides these functionalities would be an advantage on a variety of instrument sets and individual instruments, facilitating the sharing of data and information between different users. Providing such functionalities in an open format would allow users to build upon the existing framework and provide support for more complicated instruments which could immediately be shared with other researchers. It would also simplify the moving of instruments from one experimental site to another, as the infrastructure for data acquisition would already exist (assuming both sites agree to adopt the same DAQ system).

## 1.1 Problem Overview

DAQ systems are typically referred to as the systems that provide means of extracting information from a physical phenomenon, converting it into a digital value and either storing or presenting that information for an end-user to analyze[23]. This is a somewhat broad definition that allows for many interpretations, thus the existence of many stages of data acquisition:

- Transduction of a physical quantity to an electrical signal
- Sampling of said signal via specific circuitry
- Analog-to-digital conversion of the sampled signal
- Collection, parsing and storage of the digital signal

For the purposes of this thesis, only the last step of data acquisition will be relevant, as for all other steps, the approaches to data acquisition are highly dependent on the instrument and the quantity being measured [24] and as expressed earlier, all modern scientific instruments possess the ability to perform most, if not all, steps of data acquisition.

### 1.1.1 Classes of DAQ systems

At this level of the DAQ process, most systems can be categorized into one of two classes that, while similar, still possess distinctive properties[25, 26].

#### 1.1.1.1 Distributed Control Systems

A Distributed Control System (DCS) is a control architecture that oversees multiple integrated sub-systems, each controlling specific parts of a process. DCS systems are often implemented in industrial settings where the process being controlled has small geographical extension. DCS are robust since the control is highly distributed through small systems and can resist single processor failures. DCS systems are process oriented, being made to control

specific industrial processes, thus making them less flexible to changes and not being specifically designed for data acquisition.

#### 1.1.1.2 Supervisory Control and Data Acquisition Systems

Supervisory Control and Data Acquisition (SCADA) systems, contrary to DCS, often consist of geographically dispersed control systems, monitored by a centralized unit, employing high fidelity communication infrastructure to connect each individual control system to the centralized unit, which employs a human-machine interface which allows operators to issue commands to each control unit. SCADA processes are event driven, meaning that they require more emphasis in data acquisition and more importantly data manipulation to accurately report events to human operators.

Historically, DCS and SCADA systems were employed in different areas of industrial control but today with the advancements in communication and Information Technologies (IT), there is little difference between DCS and SCADA systems [27]. The heavier DAQ side of the SCADA architecture as well as the centralized supervising of many (and often geographically disperse) processes, makes it a better approach for a DAQ system developed for scientific instrument data acquisition.

### 1.1.2 Data Acquisition Software

Several software packages exist that can be used for scientific instrument data acquisition. However, they suffer from problems that make them unfit to be used as universal scientific instrument DAQ systems, be it limited scope, closed source, or not being designed for scientific instruments. Some examples will now be discussed.

#### 1.1.2.1 Zabbix

The Zabbix monitoring platform [28] is a highly scalable enterprise solution for monitoring IT networks, servers and services. It provides a framework for assessing the status and performance of connected resources. It provides server applications with centralized data storage as well as agents for handling larger networked resources and ausers to edit the code in order to expand functionalities for more specific applications. However, while it has seen efforts in its specific use in scientific experiments [29–31] it is not designed with scientific instruments in mind and thus presents challenges such as visualization generation and data parsing which limits its use as a universal scientific instrument DAQ system. In the specific case of DAQ systems in the CLOUD experiment, the Zabbix ecosystem was not only used to ensure network communication between the central servers collecting data and the individual instrument machines but had its functionalities extended to provide tools and specialized observation interfaces for the experiments in question.

#### 1.1.2.2 LabView

LabView [20] is a platform provided by National Instruments (NI) that allows users to collect data from NI hardware via a simple graphical programming language. It is extremely popular for use in individual instrument data acquisition due to easy interfacing with NI hardware, which can be found in all fields of scientific study today [32–34]. Its simple graphical language also allows for fast development of graphical user interfaces and automated state management for operators which are not fluent with programming nor with the instruments electronics. While it is used for industrial and single instrument data acquisition solutions, its closed source and limited compatibility only with NI hardware make it unsuitable for creating a universal scientific instrument DAQ system. In the case of the CLOUD experiment at CERN, Labview is used extensively on individual instruments employing NI hardware.

#### 1.1.2.3 Matlab

Matlab [35] is a data analysis and programming language provided by MathWorks. It includes a widely encompassing set of data processing techniques and visualization possibilities as well as packages for instrument monitoring and is very actively used within the scientific community[36–38]. MathWorks invests heavily on exposing this product to students, providing free student licenses and testing suites, thus making MatLab a very common tool among students and, by extension, early stage and often experienced researchers, even though such popularity is beginning to be overshadowed in favor of other open source data analysis software, such as Python or R [39]. However, besides being a closed source environment, it does not provide a framework for multiple instrument DAQ and is mostly optimized for post-processing of data, limiting its use with a general purpose set of instruments. In the case of the CLOUD experiment, Matlab is very popular as a data processing tool for instrument data.

#### 1.1.2.4 Grafana

Grafana [40] is an open source time series analytics software. It provides an intuitive interface for displaying analytics collected from different databases. Grafana is already used for a variety of visualization solutions [41–43]. Grafana, however, provides only visualizations of data from already collected and uniform data, not providing the tools for acquiring and storing data from instruments. This makes Grafana unsuitable as a single tool for collecting and serving data from multiple scientific instruments. In the case of the CLOUD experiment, Grafana was briefly considered as a tool for handling data visualization in a proposed update to the existing DAQ system.

#### 1.1.2.5 OPeNDAP

OPeNDAP (**O**pen-source **P**roject for a **N**etwork **D**ata **A**ccess **P**rotocol) [44] is an open source protocol for requesting and providing data access across the applications that enable the use of such protocol. It provides resources and defines standards for data storage servers and data access across many different platforms. It is currently used by leading organizations worldwide [45–47]. While providing many needed communication primitives and standard storage solutions, it still lacks the tools for handling individual scientific instrument data acquisition, as the

handling of each individual instrument must be built outside of OPeNDAP.

While other software frameworks exist, the above examples illustrate that the current scientific instrument environment lacks a common architecture for handling and presenting the data acquisition of different scientific instruments, whether being operated by a single individual or institute or being operated by multiple individuals or collaborations. Thus, developing a single, open-source, device independent application for handling, collecting, storing and serving data from a universal set of instruments is not only a worthwhile endeavor, it will eventually become a necessity in a scientific world with massively increasing data loads.

## 1.2 Proposed Solution

This thesis documents the creation of a framework that allows the data acquisition of a set of instruments with varied data sources and formats, collects and stores said data in a centralized and universal format and provides the tools for data access, manipulation and visualization in a fast, open and intuitive way. The created framework has been named DAQBroker and it was successfuly implemented at the CLOUD experiment at CERN. DAQBroker has been tested under machines of different computational power as well as under different suites of instruments and database setups, the results of which will be discussed on this thesis.

Chapter 2 introduces the reader to the current approaches for instrument data sources, communication protocols, data storage techniques that were considered while designing DAQBroker, as well as the statistical principles of time series analysis that are used in some more advanced features of DAQBroker. Chapter 3 details the design of DAQBroker, focusing on its three main components: Storage, communication and user interface. This chapter also describes and presents results of performance studies performed under different experimental conditions and host machines. Chapter 4 details the implementation of DAQBroker in a real-world experiment - the CERN CLOUD experiment - highlighting the needs and limitations found as well as providing examples of several studies, online and offline, that are facilitated by the use of DAQBroker. Chapter 5 presents a more advanced set of general purpose measurements that can be achieved via statistical methods of time series analysis of a generalized dataset and proof-of-concept results with data collected during the last CLOUD experimental campaign in the Fall of 2017. Chapter 6 provides a brief synopsis of the different achievements of the work of this thesis, the main interpretation of the results of performance tests and time series analysis and provide a set of improvements and future work to apply to the DAQBroker framework.

# Chapter 2

# Background

This chapter introduces the concepts behind the development and data processing aspects of DAQ systems, all of which were considered during the development of DAQBroker. These concepts will be used in chapters 3 and 5. Section 2.1 will introduce the technical concepts behind building the framework itself, such as communication, storage and data sources. Section 2.2 will introduce the statistical concepts of time series analysis considered for advanced aspects of single and multiple variable time series analysis.

## 2.1 Technical concepts

This section introduces the concepts that are behind the technical development of DAQBroker, from the data output of an experimental instrument up to, and including, the storage and handling of said data to other uses. In some aspects, the development of DAQBroker is not unlike that of a content delivery network (CDN) [48, 49]. CDNs consist of a network of (often geographically distributed) servers, that oversee and provide access to a specific type of data or content [50]. DAQBroker can be seen as the software that provides the service of collecting, overseeing and providing access to the data from a network of scientific instruments, be it in a centralized or distributed fashion. The development of such a software must take into account:

- **Communication** - proper communication must be ensured between instruments and the software,
- **Data sources** - all possible data sources of an instrument data must be accessible,
- **Storage** - at least one data storage solution is available and functioning,
- **Access** - user access to data of the instrument network must be ensured.

These aspects are discussed in the following subsections

### 2.1.1 Instrument communication

The ability to communicate with an instrument is extremely important in any scientific field[51–53]. Whether for simple data acquisition [54, 55] or to control an experiment [56, 57], digital instrument communication plays a major role in all modern scientific experiments and laboratories. The most popular means of digital instrument communication follow.

#### 2.1.1.1 Serial communication

One very popular means of communication between computers and peripherals is the serial communication via serial ports. This communication consists in transferring information sequentially one bit at a time [58]. An integrated circuit inside the computer manages the data timing and framing. The widespread availability of serial ports (historically starting with the RS-232 standard connector) allowed for the production of many instruments' data transfer via serial communication [59]. Today, most serial communications are made via USB and to a lesser extent FireWire [60], which are more complex communication standards that require faster processing speeds. In the context of modern scientific instruments, serial communications are mixed between the USB and RS-232 standards [61–66]. The CAN bus is also an example of a more specialized serial communication for industrial settings with improvements over traditional serial communication systems. Some communication standards such as Bluetooth[67] will emulate a serial communication channel by providing the computer they are connected to with a virtual serial port through which communication can occur[68].

In order to generate useful information, serial data transfers are defined by the instrument via a series of settings, often user controlled, which must be matched by the computer reading the instrument's data [69] (figure 2.1):

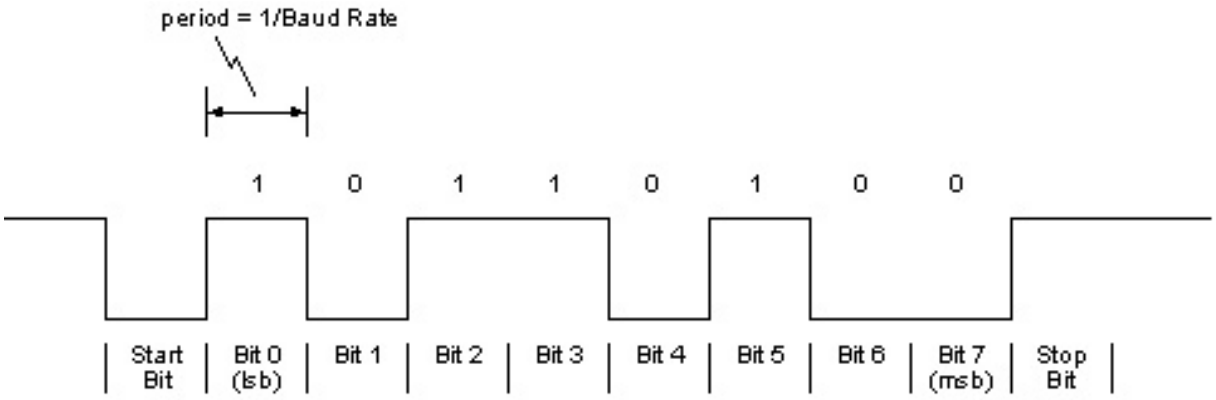



*Figure 2.1: Example digital message via serial communication highlighting the different relevant settings (parity bit not represented). Source - University of Texas at El Paso.*

- ***Baud Rate*** - The rate at which bits are transmitted via the serial port. The value is usually measured in bits per second. Some popular baud rate values include 9600, 19200 and 115200 bit/s,
- ***Data bits*** - The number of bits of each character to be sent. The current accepted values are 5,6,7 and 8 bits, the last one consists of byte-sized characters being the most popular,

- ***Parity*** - Determines the type of error checking (if any) to be done by the reading software. This value can be set to *none* (N), *odd* (O), *even* (E), *M* or *space* (S). No parity bit is the most usual form of communication, allowing the error handling be made further after the decoding of the message.
- ***Stop bits*** - Defines the number of bits to send at the end of every character for the hardware to be able to detect and resynchronize with the character stream. One stop bit is usually the amount of bits used, however some specific devices use one-and-one half or even two stop bits.

#### 2.1.1.2 Network communication

The physical distance from the instrument or the sheer amount of instruments to collect data from usually make serial communication impossible [70, 71] since serial cables without any form of signal boosting will have a range limit of about 15 meters[1][72]. When this limitation exists, the choice is to connect instruments in a network. Although instrument networks can be complicated and fairly customized, most modern instrument networks follow the OSI network model which divides the network into layers with different levels. A high level layer responds to requests from a lower layer and issues requests to the upper level layer [73] (Illustrated in figure 2.2):

1. **Physical Layer** - Provides the physical and electrical specifications of the connections. This means pin numbers, voltages, signal timing, frequency, transmission mode and more. For the context of scientific instruments, this layer defines cabling, hubs and repeaters that allow the physical communication between instruments.
2. **Data Link Layer** - Handles node-to-node data transfer. It defines the protocols to handle connections between two physically connected devices. This layer is also responsible for detecting and (if possible) correcting errors that occur in the physical layer. Of note in the context of scientific instruments, the Ethernet and 802.11 (WIFI) protocols operate in this layer of abstraction, which allows connection between computers and scientific instruments of the network.
3. **Network Layer** - Provides definitions of switching and routing technologies to ensure optimal paths for transmitting data between nodes.
4. **Transport Layer** - Provides the means of transferring variable-length data sequences from a source to a destination. Defines the protocols that handle the sending of said sequences as well as providing acknowledgment of successful transmission if no errors occurred (if required by the underlying protocol). In the context of a network of scientific instruments, the most relevant protocols are **T**ransport **C**ontrol **P**rotocol (TCP) and **U**ser **D**atagram **P**rotocol (UDP). TCP provides a highly reliable, ordered and error-checked message streaming service as messages require both end points of the message retrieval to go through a complex series of state changes[74]. On the other hand, UDP is a less safe protocol, where there is no end point handshaking, no guarantee of message delivery and no error checking, making it a flexible protocol to be used in cases where data loss is not a concern and error checks, handshaking, etc., are implemented at a higher level (video streaming for example).[75].

[1]This is only true for traditional serial communication, specialized buses like the CAN bus can reach up to 40 m.

5. **Session Layer** - Establishes, manages and terminates connections between applications. This layer is responsible for the graceful closing of connections, one feature of the TCP protocol.

6. **Presentation Layer** - Responsible for providing a translation to and from the application layer. It handles encryption and decryption of data, character conversions, data compression and graphic handling.

7. **Application Layer** - The layer with most contact with the end-user. This layer interacts with the applications that implement the communicating component, the applications themselves being not a part of the OSI model. In the context of scientific instruments the language used to communicate via a network interface is responsible to provide the methods for each host to ensure that the communication is used and displayed.

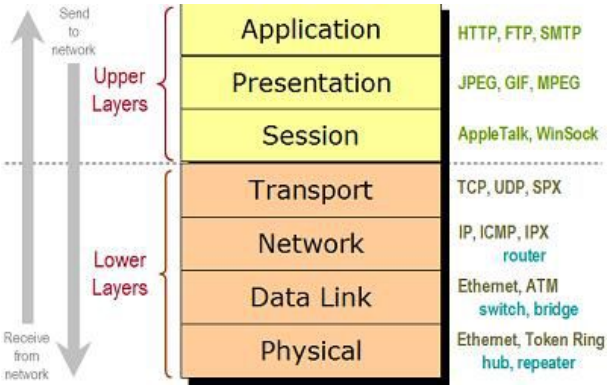



*Figure 2.2: Illustration of the OSI model layers of a computer network. Source Lifewire*

Using this model, many instruments and data acquisition systems can transmit data and be controlled via general network communication. In the context of data acquisition software, an instrument will usually have a reserved network port (identified by an integer number) where its data is either sent periodically or presented when a proper command is specified [76]. It is thus required to provide the used port number and transport protocol in order to effectively communicate with the network instrument.

#### 2.1.1.3 Specialized buses

Apart from the general purpose communication methods outlined previously, other instrument communication and instrument network protocols exist. These communication and network methods are often employed in industrial settings but can also be seen in specific scientific experiments [77, 78]. Two examples of such communication protocols are VME [79] and CAN[80] buses. These often require either proprietary or custom solutions to be implemented into the data acquisition system which can be handled by providing an open-source data acquisition system and allowing users to create specific solutions for such buses. Development of such solutions are not the focus of this thesis and thus will not be discussed further.

### 2.1.2 Data sources

The evolution of personal computing has had a deep impact in the development of scientific instruments and the way they produce data [81]. Scientific instruments rely less and less on specialized communication buses and

will often be advertised to be compatible with most commercially available personal computers [82, 83]. Digital instrument data creation is thus more and more accessible via general purpose sources [84–86]. This subsection will discuss the most used general purpose data sources in the scientific instrument landscape.

#### 2.1.2.1 Data files

Files currently make up a large amount of scientific data sources and are today perhaps the most popular source of scientific instrument data[87, 88]. The reason for this popularity is the file's universal compatibility with commercial and private operating systems (operating systems are required to read and write data to files), the non-volatility of a file (a file remains in a computer's storage until willfully deleted) and the ease of use (when learning any programming language, writing and reading to files are among the first required skills[89]). Data files can be divided into two groups depending on the encoding of the data in the file[90]:

1. **Text file** or ASCII file or plain text file - consists of a file which can be read directly by a person as an electronic book. A set of bits in these files encodes each character according to a specific encoding (ASCII is the most often used encoding, hence text files are often called ASCII files). In the context of most scientific instruments, text data files are structured to provide the type of measurement, value of said measurement and the time at which that measurement was taken[91, 92], and can be accompanied by a leading or trailing set of text providing additional secondary information for data classification referred to as the header. The format of this data is easy to understand, however, the lack of a unified table and/or header format makes it extremely complicated to programatically gather data from a large set of instruments[93].

2. **Binary file** - A formal treatment defines a binary file as any file that contains at least some data that consists of sequences of bits that do not represent plain text. These files cannot normally be read by users unless a specific (and often proprietary) piece of software is used to decode the non-representable sequences of bits. In the context of scientific instruments a significant number of instruments produce said files. This practice is used when instrument results require some degree of post-processing to produce meaningful data (such as de-noising or filtering). However, some binary formats have been more frequently used within the scientific community due to the capability of efficiently storing information, saving space compared to a structured table of measurement data [94, 95]. The secondary information is often referred to as *meta-data* and is of high importance in classifying instrument data [96, 97]

#### 2.1.2.2 Serial port

As pointed out in 3.1.1, serial communication plays a great part in controlling and relaying scientific information. Instruments often provide a true or virtual serial communication channel - called the *serial port* - for data acquisition and/or control. In the case of data acquisition via serial port, data can be transferred in one of two ways:

1. **Per request** - users establish a connection to the instrument's serial port and also provide a specific command before the data is returned to the user. This allows instruments to compartmentalize relay of data by providing

data, only when it is requested[98].

2. **Streaming** - instrument continuously sends data to the serial port. It is the data acquisition system on the other end of the serial communication channel that gathers and parses the data provided via streaming. This method allows faster sending of data since no request for the data needs to be made; however, the computing work increases when multiple data is sent via this method[99].

#### 2.1.2.3 Network port

In much the same way as in serial communication, instruments often communicate and are controlled via network communication (most times via TCP or UDP)[100, 101]. In the context of software, the communication is made through a network port, identified by a specific number. Again much like serial ports, the data transfer can also be made via request or streaming[102, 103].

### 2.1.3 Data storage

Storage of digital data is still today a highly investigated topic[104–106]. As stated in 2.1.2, instruments often store their data in files. While manageable for individual instruments, the task of sifting through files becomes cumbersome when faced with a (potentially large) set of instruments[2]. While specific software can be created to automatically search through the aforementioned files, find and return the required information, there is still the question of how to store this newly acquired data for later use[107, 108]. A common resource is required to ensure that data is properly stored and accessed, the most popular being called a database. Databases and database management systems (DBMS) have been under constant evolution since their introduction in the 1960s and are still today a topic of intense discussion[109–111]. A number of databases exist and all vary in terms of reliability, scalability and speed[112–114]. This subsection discusses several aspects of databases and the pros and cons of choosing specific DBMS and languages over others.

#### 2.1.3.1 Relational vs Non-relational databases

Databases were popularized in the 1970s with the introduction of a relational database[115]. A relational database follows the relational model of data, first appearing in 1969[116]. This model introduces the following concepts:

- **Tables** or relations - Related data is stored in a table. Each table has rows that represent a record and columns that represent the attributes of each record.
- **Tuple** or row - a row of a table, this row describes a record with its many attributes (table columns),
- **Attribute** or column - a column of a table, attribute represented by a value, character or string that is related to each other via the same row,
- **Relation Schema** - describes the table designation and each table's column type and names.

[2]Instrument data from serial or network ports could theoretically be collected onto files, resulting in another file data source.

For every table (relation) the following constraints, exist:

- **Key constraints** - In a table, there must be at least one subset of columns - called a key - which identifies each a row uniquely. This forces each key to be unique and not to contain NULL values,
- **Domain constraints** - Each column is supplied with a domain depending on the real world value it depicts, meaning each column is limited to values in a specific range (ex: age as a number; telephone numbers cannot contain digits above 9),
- **Referential integrity constraints** - Requires that the same data presented in different tables must be unique. In terms of key constraints, for the same key to exist in two tables, it must contain the same value in both tables.

In modern relational databases, none of the above constraints are necessarily enforced, however, ensuring them provides faster development and mostly in the case of key constraints, processing speed[117]. A database system that provides storage options according to the aforementioned model is called a **R**elational **D**atabase **M**anagement **S**ystem (RDBMS). In order to insert, delete, edit and select data from different tables in a database, a query must be performed[118]. All RDBMS have the advantage of sharing a common query language called **S**tructured **Q**uery **L**anguage (SQL)[119]. This language allows changes to be made to relational databases regardless of the system behind them.

In the past decade, there has been a growing resurgence of non-relational databases (more often called noSQL databases, a moniker for '*no SQL*' or '*not only SQL*'), incentivized by the needs of Web 2.0 companies such as Facebook, Google and Amazon[120, 121] as well as its intensive use by the scientific community[122], with a very active example being the BaBar experiment[123]. These databases, as their name state, do not focus on building relations and most of them don't benefit from using the SQL language[124]. Their rise in popularity came from their ease of use and the ability of *horizontal scaling*[125, 126], which is an inherent challenge for relational databases since an RDBMS requires several systems to ensure their transactions preserve the relational model, which is harder to ensure for multiple machines[127]. There are several different types of noSQL database systems available for use today[121], most of them being open source, and some provided by cloud computing companies[128–131]:

1. **Key-Value Store** - This type of database is the simplest concept of noSQL database. It defines a key and assigns it a value, a varying-length binary string containing whatever was decided to be stored, to be understood by the application that stored it. Key-value stores are useful for storing session data, user profiles and preferences (ex. online shopping cars). However, these databases are not optimized for carrying out queries for specific values, ensure relationships between data values and operate on multiple keys and persistent value updates.[132]
2. **Document Store** - While similar to the key-value store model, the document store model stores data associated with a key in a structured document, this structured document contains data that can be queried against using methods provided by the database management system (DBMS) which stored it. Document store databases are useful for storing unstructured data and make it easier to write application logic. However, document stores should be avoided if complex queries and/or multiple operation transactions are needed[133].

3. **Column Store** - This concept departs from the previous two, in that data is stored in columns and families of columns. It comes quite close to the relational model, however the concept of rows is not defined and each column is serialized together and stored in the same disk entry for faster access. Column store databases provide very efficient compression, more optimized aggregate queries (sums,averages,etc...), provide good scalability and fast load times[134].

4. **Graph Base** - Employ the concept of Nodes, Edges and Properties to create a relationship between data[135]. A node can be seen as a row in a relational database or a document in the document store model. The edge of a graph connects different nodes together, it is an abstraction concept that is not implemented in other database models. Properties are related to nodes and can be seen as the columns of a relational database. The concept of the edge is what distinguishes the graph model from other database models as it provides a simplified solution to complex queries. The value of graph databases is diluted when there is a simple underlying structure to the collected data[136].

While many different non relational database systems exist, the main difference between relational and non relational databases comes down to the ability to handle data that have a complex, changing or even non existent underlying structure[137]. Relational databases can efficiently and reliably store, edit and provide data when their structure and relationships is uniquely identified, while non relational databases are built to store, edit and provide whatever data is supplied to them with minimal setup requirements[138]. The choice of using either database systems often depends on the amount of data to handle, as regardless of database model, the challenges for handling the data increase with data size. In the light of the development of a data acquisition system for a potentially large set of instruments, each of which can produce itself a large amount of data, the choice of DBMS is one of crucial importance for the handling, sorting and serving of instrument data to users.

#### 2.1.3.2 Scaling and Database Distribution

As discussed previously, the choice of the database system is intrinsically connected to the amount of data to be handled and the resources available. The more data is expected to be handled, the more the DBMS is required to *scale*, meaning it needs to perform more operations with the resources available[139]. If a database system is running on a single machine and that machine's hardware is not changed, there will be a limit to how much data can be handled[107]. There are two possible solutions to this problem[140, 141]:

- **Vertical Scaling** - Provides more resources to a DBMS in order to handle larger computational loads. This can be done via hardware (more CPU cores, RAM, storage) or via software (via settings of the DBMS),
- **Horizontal Scaling** - Provides more machines to handle the extra load. These machines must run a variation of DBMS called a distributed DBMS (DDBMS).

Choosing to scale your system vertically or horizontally has both advantages and disadvantages and should be carefully chosen by correctly identifying the expected amount of data to handle[142].

If one chooses to *vertically* scale their system, there is no need to change the underlying DBMS, however there is an eventual limit of data to be handled, since there is no hardware produced today that provides infinite computational

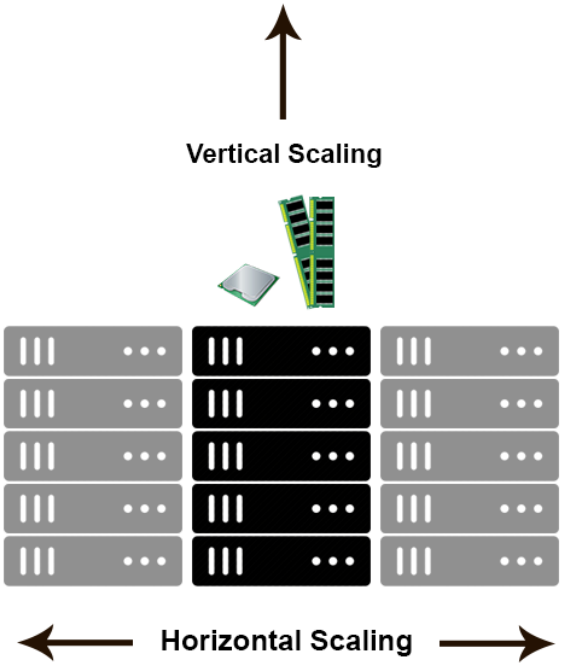



*Figure 2.3: Illustration of the different methods of scaling a DBMS. Source Stack Overflow*

power on a single machine[143].

If the choice is to *horizontally* scale a system, a distributed DBMS must be installed on a network of computers. This DDBMS will allow the handling of more data but will be inherently slower, due to the overhead of consistency checks over the network of computers[144]. DDBMS are subjected to the CAP theorem, which states that no distributed system is safe from network failures, thus network partitioning[3] must be ensured by the DDBMS[146]. In the case of network failure, the DDBMS must guarantee that either consistency or availability is maintained. If a system chooses to ensure consistency, then every node in a distributed system must return the most recent successful write, otherwise an error is returned (strong consistency)[147]. If, however, a system chooses to ensure availability, then every node must return its own available most recent write, without errors but also without the guarantee that the values are the most recent[148].

In the case of the development of a multi instrument data acquisition system, the choice of scaling model can be a hybrid between vertical and horizontal, since most DDBMS systems offer APIs that are compatible with the non-distributed DBMS used in single machines[149, 150]. Using a DDBMS might provide better solutions for large sets of instruments but can suffer from slower response times if the set is sufficiently small to be handled by a single DBMS.

#### 2.1.3.3 Data storage types

In order to optimize the storage, query speed and responsiveness of the system, an appropriate representation of the data to store - often called a *schema* - must be created in the DBMS. While building a specific schema depend strongly on the data to be stored, there are some general models that can be applied to a set of data[151]:

[3]The ability for the DDBMS to continue despite an arbitrary number of messages being dropped[145]

- **Star Schema** - It is the simplest type of data warehousing schemas[152] and consists on separating the data into facts and dimension containers[4][153]. A fact container stores measurable information about an event or object and key constraints to dimension containers which contain more detailed description of a specific fact[154].
- **Snowflake Schema** - A more generalized version of the star schema, a snowflake schema contains a central fact container whose dimensions are distributed into their own star schema, meaning that each main dimension container is divided into other dimension containers[155]. Snowflake schemas trade more complex query formulation for storage savings[156].

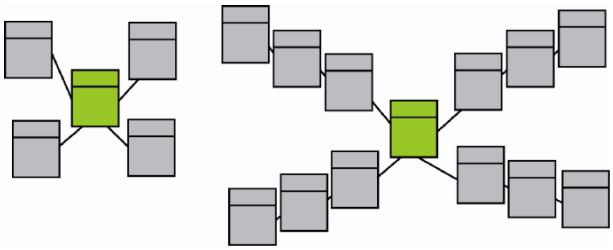


*Figure 2.4: Illustration of the star (left) and snowflake (right) database schemas. Source ResearchGate*

### 2.1.4 Data access and handling

When a suitable storage system is in place and instrument data is being properly stored, decisions must also be made in terms of data access. Apart from general tools for providing data access, what medium is chosen to provide the data to the users is important. Issues of authentication and security must also be addressed.

#### 2.1.4.1 Web vs Native applications

With the rapid growth of web technologies[157], it is now viable to not only provide data-intensive applications as downloadable and installable programs, but also server applications via web browsers (web applications)[158]. The use of native applications takes advantage of the full features of the device it is being run on and often perform better due to more optimized processes[159]. However, they are harder to maintain, since they need to be altered and compiled for different operating systems. Web applications on the contrary, have the advantage of being supplied to the users directly from a common source, the web browser, which contains a common code base between all operating systems (i.e: Javascript, HTML) making web applications much easier to maintain[160]. The limiting factor of serving web applications is the unavailability of certain features on the machine it is being run as well as of optimization of the processes [161].

One concern of web applications that is not always a concern of the native application is handling of data in an inherently decentralized system[162]. While native applications often have the choice of storing critical data locally and accessing it at will, web applications do not, not only due to the ephemeral nature of the application but also due to security concerns[163, 164]. This means that persistence of data in web applications must be handled

[4]In a relational database one can see dimensions as a table, but in non relational databases that may not be the case.

on the server side (back-end). To this end, web applications are created with the expectation of a back-end service that allows the requesting, adding, altering and removing of server held data, or resources. These services are often referred to as Web API (**A**pplication, **P**rogramable, **I**nterface)[165]. The most popular architecture of Web APIs is the **Re**presentational **S**tate **T**ransfer, which provides a set of operations that can receive a textual representation of the action to be taken on a specific web resource[166]. REST APIs use HTTP to communicate between client and server and will often employ a common way of providing data to and from a request, the most popular of which are e**X**tensible **M**arkup **L**anguage (XML) and **J**ava**S**cript **O**bject **N**otation (JSON)[167, 168]. REST APIs also have the advantage of being independent of the client using them, requiring one that clients send a request via HTTP via one of the available actions (GET, HEAD, POST, PUT, PATCH, DELETE, CONNECT, OPTIONS or TRACE) to a resource[169], meaning that REST APIs, while designed with web applications in mind, can also be used for native applications as well.

#### 2.1.4.2 Monitoring & Visualization

A critical aspect of data handling and one that is particularly critical when designing a system meant to handle data from scientific instruments is the question of monitoring and visualization[16, 170]. When considering monitoring of a set of instruments, tools and interfaces for the following cases must be considered:

- **Instrument Status** - when the measurement status of an instrument measurement is changed, whether by direct action of the operator, or by unexpected changes in the network (i.e: instrument's measuring state is changed, network communication is ceased - disconnection, shutdown),
- **Event Detection** - when the data collected by the instrument shows a behaviour that is outside of what is considered normal operation, whether by experimental action or unexpected actions (i.e: experiment stage change, power failure/surge)

In any of these cases, programmable automated tools must exist to either simply inform users that (un)expected actions were taken that changed the overall state of the experiment or to take automatic actions to safeguard the instrumentation[171].

When considering visualization of data, several libraries exist to handle data visualization, regardless of considerations of the type of application used[172]. One interesting factor of visualization is its role on providing interactive monitoring information, specifically when it comes to event detection[173, 174]. An interface that provides scientific instrument data handling must provide tools to visualize instrument data.

#### 2.1.4.3 Security concerns

With open software, which is often the case in the scientific world, there come inevitable security concerns of data breaches. Several steps can be implemented to ensure proper user authentication and to prevent malicious individuals from obtaining access through known vulnerabilities[175]:

- **User authentication** - The most common means of authenticating users in repositories, often required by the DBMS themselves for access [176].

- **Access control** - A separation of users by privileges is also often used (i.e: A guest can only access data visualizations, while an instrument operator can manipulate said visualizations)[177].
- **Injection attack prevention** - Injection attacks target unprotected segments of code where user input is required. This input is usually altered to provide more information than expected or affect the database (i.e: delete complete databases, add/delete specific entries, etc...). While only common in DBMS that have an underlying query system (i.e: SQL), injection attacks should also be a concern for noSQL databases[178]. To prevent such attacks, all expected user input must be properly handled by the server application that processes user input to the DBMS, a topic of much discussion [179, 180].
- **Data encryption** - In order to prevent a malicious party to monitor the exchange of data between a server and an end-user, data will often be encrypted between the user and the application by use of several algorithms that ensure that decrypting data without knowing the appropriate keys is not completed in a small time scale[175].

## 2.2 Time Series Analysis

Scientific data collection requires the observation of physical processes, occasionally for long periods of time. The observed process will often contain a probabilistic component, either contained in the underlying process or induced by the method of observation used (instrument)[181, 182]. This means that every time data is collected from said process, the same exact data is not collected. These *stochastic* processes [183] are extremely prevalent in data acquisition systems and current scientific experiments[184, 185]. Data collected during a time interval of a realization of these processes is called a *time series*[186].

The collection of time series from different instruments generates a time series data set related to the time interval of the collection. Several statistical techniques can be applied to data from one or various instruments in order to extract information from the environment being sampled, reveal statistical relevance of specific time periods or finding relationships between different variables. This section will introduce statistical concepts used for analysis of time series. Subsection 2.2.1 will focus on methods to analyze individual time series. Subsection 2.2.2, will focus on methods for analyzing clusters of time series.

### 2.2.1 Univariate Time Series Analysis

A time series ($X$) consists of a series of measurements of the same variable, executed during a specific time interval and indexed in time order[187]. Time series can be continuous or discrete, with regards to their sampling method. Often, for instruments only discrete time series are analyzed[188]. Time series collected using the same sampling frequency can be conceptually defined as follows:

$$X = \{x_1, x_2, ..., x_{n-1}, x_n\} \tag{2.1}$$

Where $N$ is the time series *sample size*. The following subsections will discuss the statistical concepts applied to single and sets of time series to extract meaningful information.

#### 2.2.1.1 Classifying Time Series

Time series of a single process contain information about the process to be extracted (i.e: mean, variance, trend, etc...). In order to correctly extract that information, there are several statistical properties of a time series that must be evaluated[189]:

- Stationarity,
- Integration,
- Trends,
- Seasonality.

This subsection discusses these statistical properties and provides ways of testing whether or not these properties are ensured.

**2.2.1.1.1 Stationarity** A stationary time series is a special case of a time series where a certain number of its properties do not vary over time[190]. A time series can have several types of stationarity, depending on which properties are allowed to vary over time:

1. **Strong Stationarity** - As the name states, it is the strongest and hence the hardest type of stationarity. Strongly stationary time series have the same joint distribution of any smaller collection of time/value pairs. Formally that property can be defined as follows[191]:

$$F(\{x_{1+\tau}, ..., x_{k+\tau}\}) = F(\{x_1, ..., x_k\}), \forall \tau, k \in \mathbb{Z}_{>0} \tag{2.2}$$

   where F(...) represents the joint distribution of a specific time series. As a consequence statistical properties such as mean and variance of the time series do not change with time;

2. **Weak Sense Stationary** - A common form of stationarity in signal processing that only requires that the first moment (mean) and the autocovariance of the series do not vary with time. Formally, weak sense stationarity can be defined as [192]:

$$\mathbb{E}(\{x_{1+\tau}, ..., x_{k+\tau}\}) = \mathbb{E}(\{x_1, ..., x_k\}) = m_x(t), \forall \tau, k \in \mathbb{Z}_{>0} \quad \text{(Mean)} \tag{2.3}$$

$$ACF_x(t_1, t_2) = ACF_x(t_1 - t_2, 0) \quad \text{(Autocovariance)} \tag{2.4}$$

   where $\mathbb{E}$ represents the expected value and $ACF(t_1, t_2)$ represents the autocorrelation function of the distribution $x$ with regards to the times $t_1$ and $t_2$. It comes from this definition that the autocovariance function only depends on the time difference specified. This definition can also be extended to include non-varying moments up to a specific order $m$, such a time series is thus considered to be "stationary of order $m$"[193]

3. **Trend Stationarity** - Corresponds to the existence of a trend in a time series which can be removed, the remaining time series being stationary. The corresponding trend needs not be linear and must only be a function of time. Formally, time series $(X_t)$ with trend stationarity can be defined as[190]:

$$X_t = f(t) + e_t \tag{2.5}$$

Where $t$ is time, $f$ is a function $f(\mathbb{R}) - - > \mathbb{R}$ and $e$ is a stationary time series.

To detect stationarity in a time series, one can perform a variety of statistical tests[194]. The data is considered to follow a specific distribution (*null hypothesis*) against another possibility which will usually be another distribution (*alternate hypothesis*). The test consists in calculating a a specific value or *test statistic*, associated with assuming the null hypothesis[195]. The resulting value of the test statistic will allow the tester to decide whether to accept the assumed null hypothesis or reject it in favor of the alternate hypothesis. The value of the statistic is often associated with a probability value called *p-value*, which expresses the probability of the tester being wrong by rejecting the null hypothesis. It is common to consider a p-value of 5 or 1% to reject the null hypothesis[196]. The use of statistical tests is not fool proof as it is highly dependent on the sample being tested. A small or improperly pre-processed sample might result in the wrong conclusion, whether it be incorrectly accepting the null hypothesis - false positive or type I error - or incorrectly rejecting the null hypothesis - false negative or type II error[197].

The statistical test used in this thesis to test stationarity of a time series is the **K**wiatkowski–**P**hillips–**S**chmidt–**S**hin test (KPSS). This test is one of the most popular tests for determining stationarity of a time series[198]. It tests the null hypothesis of the tested time series being sampled from a trend-stationary process, against the alternate hypothesis of the process possessing a unit root[199], defined below.

**2.2.1.1.2 Integrated time series** There are many processes with non-stationary underlying time series [200]. The most common process represented in non-stationary time series are *integrated processes*[201]. The definition of a time series of an integrated process is the one that can be transformed into a stationary process by *differencing* it. Considering a time series $x_t$, its differenced series $w_t$ is defined as:

$$w_t = \Delta x_t = x_t - x_{t-1}, \tag{2.6}$$

where $\Delta$ is the differencing operator and $x_{t-1}$ is the value of time series in the previous time value. If the time series $w_t$ is considered to be stationary (constant mean and variance), the original time series $x_t$ is considered to be *integrated of order 1*. If a number $d$ differentiations are required to recover a stationary time series, the process is considered to be *integrated of order* $d$[190].

The most well known integrated process is called the *random walk process*. It is formally defined as a time series $z_t$ with the following representation[202]:

$$z_t = z_{t-1} + \epsilon_t, \tag{2.7}$$

where $z_{t-1}$ is the time series value in the previous time and $\epsilon_t$ is a value of a random distribution at time $t$. The random walk is a special case of a more general representation of stochastic processes that are linearly dependent on earlier values, the *autoregressive model*, which is defined by the series[203]:

$$z_t = a + \sigma z_{t-1} + \epsilon_t, \tag{2.8}$$

where $a$ is a constant and $\sigma \in \mathbb{R}$. It can easily be seen that equation 2.7 can be recovered from 2.8 with $a = 0$ and $\sigma = 1$. A random walk process can be easily shown to be an integrated process of order 1 by applying the differencing operator ($\Delta$) into equation 2.7, since the remaining stochastic term of equation 2.7 ($\epsilon_t$) is by definition stationary. A process which is integrated of order 1 can also be stated as having a *unit root*[204].

In the context of instrument data acquisition and scientific data, it is not uncommon to find unit root processes[205, 206]. Unit root processes should not be mistaken for trends, even though they exhibit at first glance similar properties[207].

To detect integrated time series, more specifically, unit root time series, the Augmented Dickey-Fuller (ADF) test is used in this work. ADF tests the null hypothesis of whether a unit root is present against the alternate hypothesis that no unit root is present, making the process trend stationary. It can be performed for 3 different types of unit root[208]:

- The process is integrated of order $p$: $\Delta z_t = \sum_{i=1}^{i=p} \delta_i \Delta z_{t-(i+1)} + \epsilon_t$,
- The process is integrated of order $p$ with drift: $\Delta z_t = \alpha + \sum_{i=1}^{i=p} \delta_i \Delta z_{t-(i+1)} + \epsilon_t$,
- The process is integrated of order $p$ with drift and deterministic trend: $\Delta z_t = \alpha + \beta t + \sum_{i=1}^{i=p} \delta_i \Delta z_{t-(i+1)} + \epsilon_t$.

For the purposes of this thesis, only the first type of ADF tests will be used, as a test for trend and also for trend stationarity.

**2.2.1.1.3 Trend** A stochastic process that contains a trend has its values changed by a deterministic constant value with time. It can be described as follows[209]:

$$z_t = a + \beta(t)t + \epsilon_t, \tag{2.9}$$

where $\beta$ is a deterministic value that may or may not be constant over time. By subtracting this trend from the time series one can retrieve a stationary time series[190]. The main difference between trended and integrated processes is their behaviour to *shocks*, or sudden changes in the environment being measured[210]. This is clear in figure 2.5 which compares two stochastic processes, one containing a trend and one containing a unit root. After a shock, processes that contain a trend will revert to their mean value and thus are called *mean reverting*, while processes which are integrated will not[211].

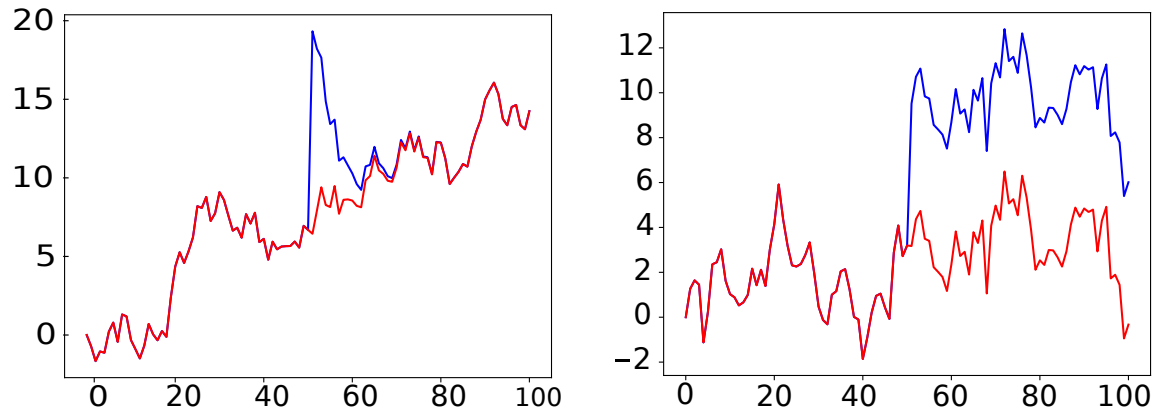



*Figure 2.5: Example of two stochastic processes (red) and their shocked equivalents at $t = 50$ (blue). The left figure shows a trended stochastic process while the right process shows a unit root process. Notice that on the left image, after the shock is applied the left image reverts to its non-shocked equivalent (mean reverting) while the right image does not.*

The previously introduced autoregressive model for a stochastic process can be naturally complemented with the addition of a trend term to 2.8:

$$z_t = a + \beta(t)t + \sigma z_{t-1} + \epsilon_t \tag{2.10}$$

In the context of instrument data acquisition and scientific data, trends in time series are more important than integrated time series in most cases [212]. It is thus extremely important to identify and extract trends from time series, whether it is for post processing, detrending and/or smoothing or forecasting[213, 214].

In order to detect trends in a time series, the Mann-Kendall (MK) trend test[215, 216] is used in this work. This test assumes a null hypothesis of a non-existent monotonic trend, which may or may not be linear against the alternate hypothesis that a trend is present.

**2.2.1.1.4 Seasonality** Seasonality consists of patterns in the data that are repeated over time[217]. From the previous definition two types of periodic patterns can be deduced (illustrated in figure 2.6):

- **Cyclic patterns** - When the time elapsed between repeated patterns is not well defined (e.g: extinction events),
- **Seasonal patterns** - When the time elapsed between repeated patterns is well defined, also often referred to as *periodicity* (e.g: solar cycle,heart beat, etc...).

Seasonality of a time series is analysed using the **A**uto**c**orrelation **F**unction (ACF) and **P**artial **A**uto**c**orrelation **F**unction (PACF)[218]. The ACF is calculated by taking the time series ($x_t$) and correlating it with the same time series shifted in time by a certain interval, defined as *lag*. Formally, taking a time series the calculation of its autocorrelation function is:

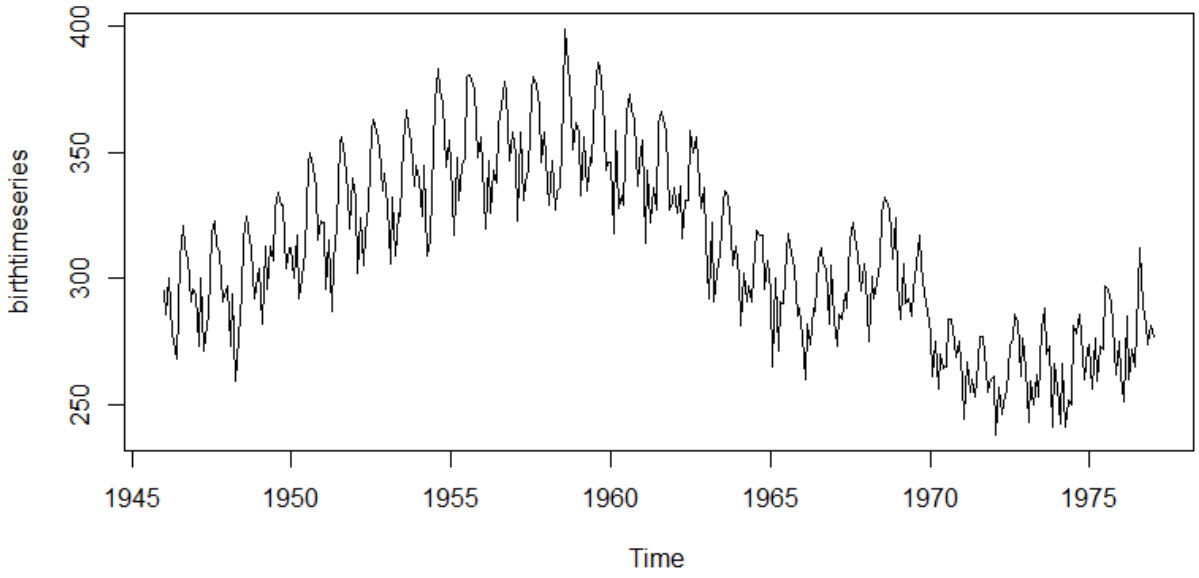



*Figure 2.6: A plot of US birth rates as a function of time, the plot shows a highly reapeatable (seasonal) yearly pattern of higher and lower births. The plot also shows a slow increase of overall births over a decade proceeding to drop off over the following decade, while the last decade in the plot shows an increase of births over 5 years followed by a decrease in the next 5 years (cyclic pattern). Source Grroups*

$$ACF_x(n) = Corr(x_t, x_{t+n}) = \frac{\sum_{i=1}^{N-n}[x(i) - \mathbb{E}(x)][x(i+n) - \mathbb{E}(x)]}{\sum_{i=1}^{N}[x(i) - \mathbb{E}(x)]} \tag{2.11}$$

Where $N$ is the total number of samples of the time series and $x(i)$ is the value of the time series on index $i$. The number of indexes, $n$, defines the lag interval.

The ACF at a particular lag value will include contributions from all other smaller lag values (i.e: ACF at lag 10 will have contributions from lag 0 to 9). In order to properly calculate the single contribution of a lag value, the aforementioned contributions must be removed, thus calculating the PACF[219]. The PACF is obtained for lag $n$ by producing a regression analysis of the time series considering an autoregressive model of order $n$ defined as:

$$AR(n) = \sum_{i=1}^{n} \sigma_i x_{t-i} + \epsilon_t \tag{2.12}$$

Where $\sigma_i$ is the autoregressive coefficient of order $i$. The resulting coefficient of order $n$ assuming the process is $AR(n)$ will be the value of the PACF at lag $n$.

The distinction between seasonality and a trend in the time series is highly dependent on the time series length, as well on the sampling frequency chosen. In the context of scientific instruments, the identification of seasonal patterns is important whether for modeling efforts [220] or removing from the general trend[221]. Cyclic patterns are usually the result of unexpected changes to the measured environment and can be used for prediction of incoming events[222], thus are also important to identify.

In order to detect periodicities in a time series, the Fisher periodicity test is used. This test assumes a null hypothesis of non-existent periodicity and an alternate hypothesis of periodicity[223].

#### 2.2.1.2 Comparing Time Series

In most experiments and especially in ones employing sets of scientific instruments, it is often the case that more than one distinct time series exists for different measured quantities. In such cases, it is important to distinguish time series with similar behaviors to one another, as it allows to infer on possible causal relationships between the series[224, 225]. This can be done through visual inspection of the time series, a slow process which often requires heavy data processing and knowing beforehand which time series will behave similarly[226, 227]. If the behaviors of time series are not known *a priori* and the number of time series prohibits visual inspection of each pair of series, then a different method must be used to compare their behaviors and, when comparison is complete, should provide a shorter pool of similar time series, from which data processing and visual inspection can be employed with less user effort.

Two major types of measurements exist for comparing time series. The first are *similarity measurements* that compare how alike two time series ($X$ and $Y$) are. They provide a number in the range of $[0, 1]$, 0 being absolutely different and 1 being absolutely equal. The second types of comparison measurements are dissimilarity or *distance measurements*, which provide a number in the range $[0, \infty]$ where 0 means the series are equal and the larger the value the more different the time series are[228]. Of important note, dissimilarity measurements ($d(X, Y)$) values can be converted into similarity measurement ($s(X, Y)$) values using the following formula[229]:

$$s(X, Y) = \frac{1}{1 + d(X, Y)} \tag{2.13}$$

Other important properties of both similarity and dissimilarity measurements are[230]:

- **Symmetry** - $d(X, Y) = d(Y, X)$,
- **Reflexivity** - $d(X, Y) = 1$ (or 0 for dissimilarity) $\iff X = Y$,
- **Triangle inequality** - $d(X, Y) \leq d(X, Q) + d(Q, Y)$ with $Q$ being any other time series.

When the time series to be compared are well constrained in time and a reasonably controlled environment is established, it is often common for the time period to be ignored in favor of transforming the time series into a set of components in a frequency space, often by means of the Fourier Transform[231]. The Fourier Transform ($\mathscr{F}$) of a time series is defined as:

$$\mathscr{F}_X(\omega) = \int_{-\infty}^{\infty} X_t e^{-2i\pi\omega t} dt, \tag{2.14}$$

where $X_t$ is the time series value at time $t$. This transformation turns $X$, a function of time, into a function of frequency ($\omega$) and allows the exploitation of each frequency contributions to the overall time series. For discrete

data, the Discrete Fourier Transform (DFT) is generally used:

$$\mathscr{F}_X(k) = \sum_{n=0}^{N-1} X_n e^{-2i\pi kt} dt \qquad k = 0, ..., N-1, \tag{2.15}$$

where $N$ is the number of elements in time series $X$ (sample size). Computationally, the discrete Fourier transform is calculated via Fast Fourier Transform algorithms (FFT), which greatly decrease the computational effort for the calculation[232]. Working with the frequency components of time series provides several avenues for time series comparison[233–235]

This work focuses and uses the most popular measures in literature, more specifically: correlation (similarity), dynamic time warping (distance), local similarity (similarity) and coherence (similarity)

**2.2.1.2.1 Correlation** Correlation coefficient is a similarity measurement that can be applied to two samples in order to calculate the degree of linear correlation between the two corresponding variables[236]. The Pearson correlation coefficient is probably the most popular measure to infer the similarity of two samples [237]. Considering the specific case of time series, if two samples have different distributions with the same size ($N$) - $X_t = \{x_1, ..., x_N\}$ and $Y_t = \{y_1, .., y_N\}$ - the formal calculation of the Pearson correlation coefficient ($r$) is calculated as:

$$r = \frac{\sum_{i=1}^{N}(x_i - \overline{X})(y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{Y})^2}}, \tag{2.16}$$

where $\overline{q}$ is the mean sample value of the sample $q$. The result of equation 2.16 is a value between -1 and 1. If the value of the coefficient is 1, then $X$ and $Y$ are perfectly related to each other (correlation). If the value is -1 then there is a relationship between $X$ and $Y$ where if $X$ increases, then $Y$ decreases and *vice-versa* (anti-correlated). If the value of the coefficient is 0 then there is no relationship between the samples[238].

An immediate limitation of the Pearson correlation coefficient is the need for both samples to be of the same size[239]. This makes it impossible to calculate the coefficient between two time series with different sample frequencies; while this limitation can be fixed by interpolation of the largest frequency sample or suppression of values from the smallest frequency sample, these methods can induce unwanted artifacts[240]. The Pearson correlation coefficient is also sensitive to scaling along the y-axis and warping of the time axis (i.e: not sensitive to delayed responses, accelarated/decelarated processes)[241].

A possible alternative to the Pearson correlation coefficient similar to correlation is the cross-correlation of signals. This method is popular in image processing to find similar images, and can be used for 1-dimensional signals It consists of multiplying elements of one of the signals by the other shifted by a certain time values. It can be formally defined for discrete time series as[242]:

$$\mathscr{R}_{XY}(k) = \sum_{i=1}^{N} X_i \times Y_{i+k} \tag{2.17}$$

This measurement is similar to the convolution of signals, except that the first signal precedes the second signal, which does not happen for a convolution. The cross-correlation measurement can deal with delayed time signals but is now not bound by the similarity measurement limits. It can recover those limits by normalizing the time series by the norm of their auto-correlation[243].

**2.2.1.2.2 Dynamic Time Warping (DTW)** This is a distance measurement which accounts for different sequences which change in speed[244]. DTW attempts to find the optimal alignment between two different time sequences ($X$ adn $Y$) of length $N$ and $M$ (respectively) by producing an NxM cost matrix ($C_{DTW}$) where each element of said matrix is a simple distance measure such as the Eucledian distance and finding a path through that matrix that minimizes the cost of traversing the matrix. Formally this cost matrix can be defined as[245]:

$$\mathbb{C}_{DTW}(X,Y) = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,M-1} & c_{1,M} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,M-1} & c_{2,M} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{N-1,1} & c_{N-1,2} & \cdots & c_{N-1,M-1} & c_{N-1,M} \\ c_{N,1} & c_{N,2} & \cdots & c_{N,M-1} & c_{N,M} \end{bmatrix} \tag{2.18}$$

where $c_{i,j}$ is an appropriate binary distance measure (eg: Eucledian, Mahalanobis or cosine distances) of element $i$ of time series $X$ and element $j$ of time series $Y$. In order to traverse the matrix with minimal cost the optimal *warping path* must be discovered. The warping path is a set $p$ of a number $L$ of pairs of indexes from the $X$ and $Y$ time series ($p = \{p_1, \ldots, p_L\} = \{(x_1, y_1), \ldots, (x_L, y_L)\}$) and must satisfy the following conditions[246]:

- *Boundary condition* - $p1 = (1,1)$ and $p_l = (N, M)$
- *Monotonicity condition* - $x_1 \leq x_2 \leq \cdots \leq x_{L-1} \leq x_L$ and $y_1 \leq y_2 \leq \cdots \leq y_{L-1} \leq y_L$
- *Step size condition* - $p_{i+1} - p_i \in \{(1,1), (1,0), (0,1)\}$ for $i \in [1 : L-1]$

For a warping path ($p$) properly defined, DTW will minimize the total cost function over said path ($c_p$) which is defined by:

$$c_p(X,Y) = \sum_{i=1}^{L} c(x_{p_i}, y_{p_i}), \tag{2.19}$$

where $c$ is the aforementioned distance measure selected for the definition of the cost matrix in 2.18.

The classical DTW contains one critical disadvantage, since it requires searching through a matrix of $N \times M$ for elements, making the calculation of the optimal path and respective cost function a computational problem of complexity $O(n^2)$[247][5]. However, several restrictions can be applied to the warping path calculation to increase the computational speed of the method, as with the Sakoe-Chiba band or the Itakura parallelogram, both represented in figure 2.7. These methods restrict the choice of optimal warping path to certain (user-defined) regions in the cost matrix thus limiting the number of choices of possible warping paths [248, 249]. This limitation, while in

[5] An algorithm implementing this method would require $n^2$ computations to finish where $n$ is a length representing the size of the arrays used. This is one of the most inefficient types of algorithmic complexities.

some scenarios might limit the distance metric to higher values, will in most chosen signals with appropriate limits allow the choice of a semi-optimal path that should properly define the distance between time series[250].

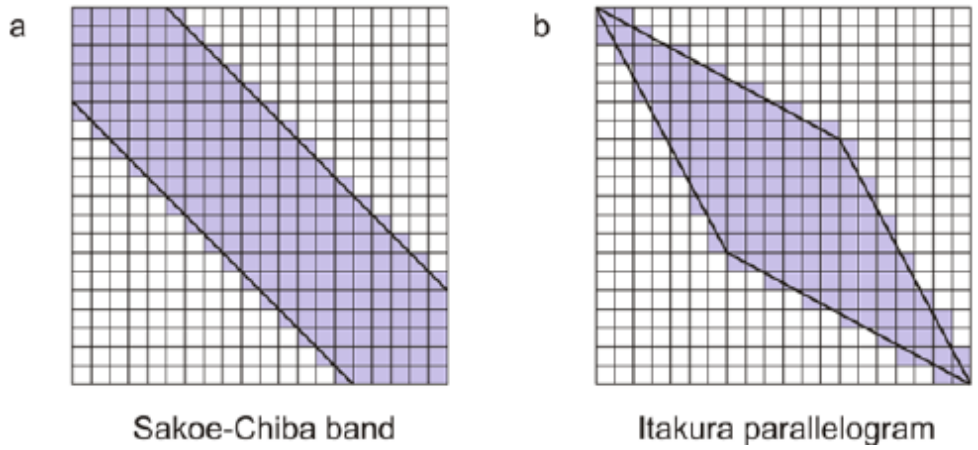



*Figure 2.7: Illustration of the Sakoe-Chiba and Itakura parallelogram over a generic cost matrix. The shaded regions represent the allowed regions for warping path calculation[251].*

Another possibility for making DTW more efficient is to apply dimensionality reduction by using coarse approximations of the time series, and thus building a smaller overall cost matrix. One such example is the fastDTW algorithm[252].

**2.2.1.2.3 Local Similarity** Measurements of local similarity were first introduced for testing similarity between biological time series[253–255]. The need for this introduction came from the fact that very often the effect of one tested compound or tested behavior would vary from series to series not allowing relevant similarity measurements by using simple correlation. It is an ideal method to use with a set of time series where little knowledge is known *a priori* for each individual channel. Local similarity provides, as the name states, a similarity measurement, that results from the application of successive similarity measurements in localized subsets of the time series, limited by a user-defined maximum delay parameter[256].

One algorithm for calculating a local similarity statistic is presented here, adapted from [257].

To note that the returned value from this algorithm, LSS, can be either positive or negative, depending on which score matrix dominates in absolute value, thus providing not only a similarity measurement, but also a sense of correlation or anti-correlation. Also of note is that for a large enough delay, this algorithm will be of complexity $O(n^2)$. This means that steps must be taken to limit the available delay. The fact that this measurement provides the positive aspects of both correlation measurements (signal of the measurement) and dynamic time warping (flexibility with the time scale) justifies its use in this work.

**2.2.1.2.4 Coherence** The coherence of two time series is a measure of how related the time series are at a certain frequency[258]. The coherence ($\mathscr{C}$) of two time series ($X$ and $Y$) is defined as:

**Data:** Time series to compare ($X$ and $Y$), delay ($D$) in time
**Result:** Local Similarity Statistic (LSS)
$m$=len($X$); /* Length of time series */
$n$=len($Y$); /* Length of time series */
$P$=zeroes($m$,$n$); /* Positive score matrix declaration */
$N$=zeroes($m$,$n$); /* Negative score matrix declaration */
**for** $i = 1$, $i++$, *while* $i < m$ **do**
  **for** $j = 1$, $j++$, *while* $j < n$ **do**
    **if** $|i - j| <= D$ **then**
      $P_{i+1,j+1} = max(0, P_{i,j} + X_i * Y_j)$;
      $N_{i+1,j+1} = max(0, N_{i,j} - X_i * Y_j)$;
    **end**
  **end**
**end**
$\hat{P} = max_{|i-j|<=D}(P_{i,j})$;
$\hat{N} = max_{|i-j|<=D}(N_{i,j})$;
LSS = $sign(\hat{P} - \hat{N}) \frac{max(\hat{P},\hat{N})}{\sqrt{n*m}}$

**Algorithm 1:** Local similarity statistic calculation for two time series.

$$\mathscr{C}_{XY}(\omega) = \frac{|\mathscr{G}_{XY}(\omega)|^2}{\mathscr{G}_{XX}(\omega)\mathscr{G}_{YY}(\omega)}, \tag{2.20}$$

where $\mathscr{G}$ is the cross-spectral density of two time series, defined as[259]:

$$\mathscr{G}_{XY}(\omega) = \mathscr{F}_{\mathscr{R}_{XY}}(\omega) = \int_{-\infty}^{\infty} \mathscr{R}_{XY}(t) e^{-2i\pi\omega t} dt, \tag{2.21}$$

with $R_{XY}$ being the cross-correlation function of $X$ and $Y$, defined in equation 2.17.

Coherence can be seen as a the cross correlation function in the frequency domain and provides a measure of each frequency component relation for the compared time series.

A relevant result to facilitate the estimation of coherence comes from the definition of the Fourier Transform when applied to the convolution of two functions. The convolution of two functions $f$ and $g$ ($fg$) is defined as:

$$(fg)(t) = \int_{\infty}^{\infty} f(\tau) g(t - \tau) dt \tag{2.22}$$

Which draws many parallels with the cross-correlation function, in fact applying this definition the cross correlation function can be written as convolution of two time series:

$$\mathscr{R}_{XY}(t) = X^* Y \tag{2.23}$$

Where $*$ denotes complex conjugate (which in practical terms will not matter for this thesis, since all signals are of real values). It can also be proven that the Fourier Transform of a convolution of two functions is equal to the multiplication of the Fourier Transforms of each function[260]. Thus, the cross spectral density can be calculated as:

$$\mathscr{G}_{XY}(\omega) = \mathscr{F}_X^*(\omega)\mathscr{F}_Y^*(\omega) \tag{2.24}$$

#### 2.2.1.3 Event detection

Event detection is one actively purpose of time series analysis [261–263]. It has many uses in the scientific community allowing simple activities like identification of change of state of an experiment [263], but can also be used for critical event detection to maintain the health of a site or experiment[264]. There are many types of event detection algorithms, most of then falling into one of two main categories[265]:

- **Supervised detection** - The algorithm receives a set of events (positive, negative or both, depending on the complexity of the algorithm) and by studying the underlying statistical properties and structure of the set it compares that information to the one corresponding to new data and identify events [266].
- **Unsupervised detection** - The algorithm is only provided with a method of extracting the structure of the data and from it, the algorithm has a process to decide whether an event has happened or not[267]

Usually, supervised event detection will more accurately detect events with a smaller amount of false positive events. However, large sets of events (training sets) must be provided to these algorithms beforehand[268]. If a type of event is not provided to the training set, more than likely it will not be detected in the live data. Unsupervised algorithms by not requiring training will require less computational effort and will not be restricted to a possibly limited set of training events. However, if not properly tuned, unsupervised algorithms may often provide many false positives and also false negatives[269].

This work focuses on *unsupervised* algorithms that detect events while the data is being gathered, called *online detection* algorithms. Specifically, three different event detection methods are explored: threshold detection, ARIMA model detection and symbolic ensemble wavelet decomposition event detection.

**2.2.1.3.1 Threshold detection** This technique consists in defining one (or more) values that are considered anomalous (threshold), if the values of the series is exceeded, then an event is considered to be under way (figure 2.8)[270]. More complex versions of this algorithm exist such as adaptive thresholds[271] or cumulative sum thresholds[272] to deal with signal seasonality or spurious values. Threshold methods require a stable signal that is not expected to have multiple states[273].

**2.2.1.3.2 Windowed detection** One variation of this method consists in the detecting events inside a certain sub-period of time inside the available time series. The event detection is evaluated by examining the point-by-point statistics of each sub-period, assuming that a specific distribution is achieved during each sub-period. The most popular method used is the rolling z-score, which assumes that a point in a sub-period is anomalous and is thus classified as an event if it deviates more than a user-specified amount of standard deviations of the sub-period. This method thus assumes that there is a certain degree of stationarity on each window, which, given a combination of narrow windows and slow processes could be true. However, the use of a z-score assumes that the distribution of the signal at each window is normally distributed, which is not always the case[274].

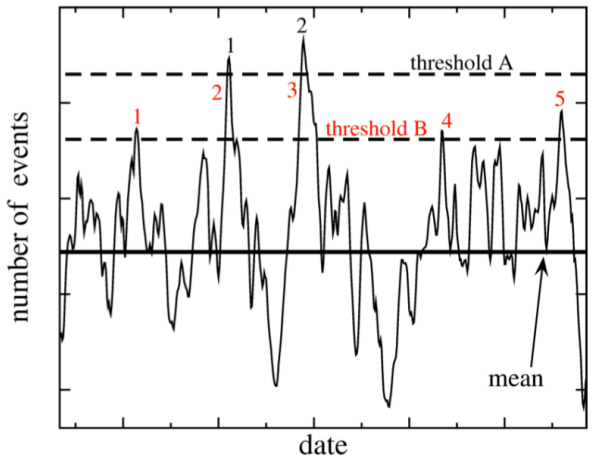



*Figure 2.8: Illustration of threshold event detection. This particular method uses two threshold values (A and B). Source .*

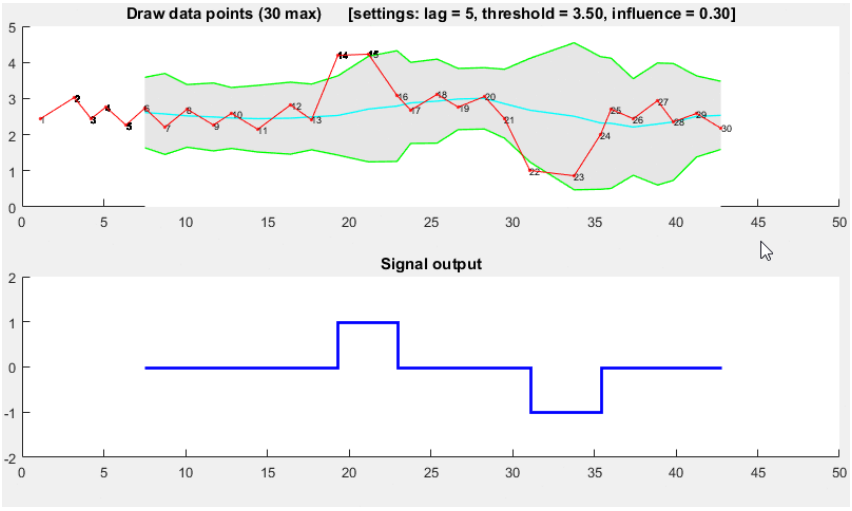



*Figure 2.9: Illustration of event detection using a rolling z-score method. The top image shows in red the data points and in green the limit of detection (3.5) given by the previous 5 data points. The bottom plot shows the response of the event detection (0 if no event and [1, -1] if the signal is over the user-defined number of standard deviations.*

**2.2.1.3.3 ARIMA modeling** Statistical modeling of time series is a highly investigated area[275, 276]. **A**uto-**r**egressive **I**ntegrated **M**oving **A**verage (ARIMA) models are highly popular methods for modeling and forecasting time series[277]. These models can also be used for event detection by searching for time periods where the data deviates from the forecast values[278]. As the name states, the model consists of optimizing an expression with 3 distinct terms [279]:

1. **Autoregressive** - A term that depends on the previous values of the time series, this term may also contain (non linear) trend terms to be fitted,
2. **Integrated** - These are not terms to be fitted but merely an indication that the data used is subtracted by their previous values. This method is used to remove any existing unit roots from the data,
3. **Moving Average** - A term that depends on the previous values of a previously chosen error distribution.

With the forecasting power of the ARIMA models comes the ability to distinguish when a signal deviates from its

normal behavior, thus signaling a possible event (figure 2.10) [280]. ARIMA models with seasonal terms can also be used to include existing seasonality in time series. While ARIMA modeling is a powerful and proven method for time series forecasting and event detection, a large amount of data points is required in order to generate a proper model of the underlying process [281].

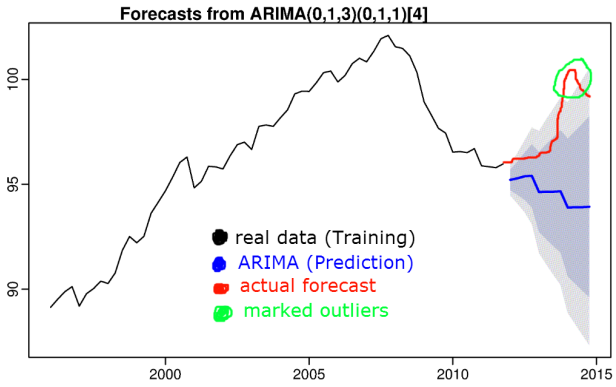



*Figure 2.10: Illustration of threshold event detection. This particular method uses two threshold values (A and B). Source .*

**2.2.1.3.4 Symbolic wavelet partitioning** This time series analysis method is based on several techniques and is thoroughly explained in [282–284]. The ones used in this work are: Wavelet transform and symbolic analysis.

The wavelet transform is a time-to-frequency transformation, in which a time series is represented in the frequency domain by the components in a set of base functions, much like the Fourier Transform[285]. The set ($\psi$) of functions used to represent the data in frequency space must be an orthonormal system such that:

$$\langle \psi_j, \psi_k \rangle = \delta_{jk}, \tag{2.25}$$

where $\psi_j$ represents one base function of the set and $\delta_{jk}$ represents the Kronecker delta[286]. In the case of the wavelet transform, instead of functions with shifting frequencies as in the Fourier Transform, the orthonormal set consists in a series of functions that are shifted in both time and frequency (scale) called *wavelets*. A set of wavelets is defined by a *mother wavelet* and a *scaling function* that is responsible for moving and stretching/compressing (scale) the mother wavelet in time. This means that the wavelet transform not only provides information of the time series in the frequency space, it also maintains time space information[287]. A wavelet is a class of function that has similar characteristics to waves, but is limited in time depending on the *decomposition level* of transformation chosen [288]. The *decomposition level* of the transformation defines how fine a frequency analysis is required. A larger decomposition level uses a scaling function that reaches smaller scales, meaning better representation of finer features of the signal (figure 2.11).

The result of a wavelet transform is a matrix of $time \times frequency$ points. In order to analyze these points, a symbolic ensemble analysis is performed. This method consists in ordering the content of a signal in a meaningful set of representative symbols, often called an *alphabet*, by a procedure called called *partitioning*[289]. Partitioning

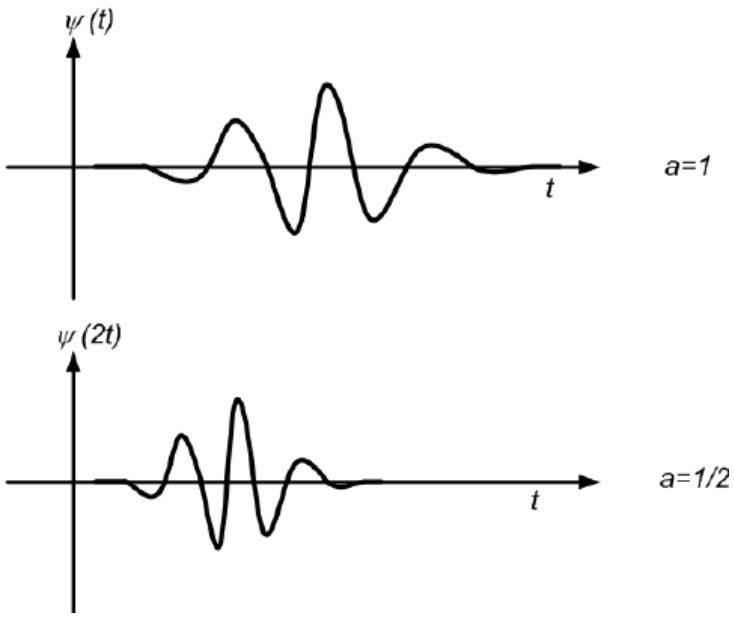



*Figure 2.11: Illustration of the scaling of a wavelet, the top shows the normal wavelet function and the bottom function shows a scaled wavelet function by a scale factor of $\frac{1}{2}$. Source .*

consists in dividing the time series data in subsets and assigning a symbol to each one[6]. There are many partitioning methods[290], this work focuses on *maximum entropy* (ME) partitioning. ME partitioning allows an automatic choice of alphabet size (number of subsets of data to consider) by maximizing the information (entropy) that each alphabet segment provides. When the provided information of an alphabet segment is lower than a certain provided threshold, the entropy is considered to be maximized and the ideal alphabet size.

Once the ideal alphabet size is defined a symbolic representation of the time series is achieved. The aforementioned alphabet can now be applied to other time series of similar length to create a new symbolic representation, the comparisons between the two representations allow to decide whether or not an event occurred[291].

### 2.2.2 Multivariate time series analysis

When considering multiple instruments, often the use of combined data from those instruments allows to increase the knowledge of the overall process being studied. The application of a statistical treatment to several time series is part of the domain of *multivariate statistics*, more specifically in this case, *multivariate time series statistics*. A multivariate time series ($\mathbf{X}$) is simply a set of two or more time series, which are either realizations of the same process measured by different instruments, or measurements made from the same instrument, only shifted in time[292, 293]. Extending the definition of equation 2.1, a multivariate time series can be defined as follows:

$$\mathbf{X} = X_1, ..., X_M \tag{2.26}$$

[6] An alphabet does not require to have specific symbols, it only requires to contain a set intervals that encompass all the data points in a time series

Where $X_i$ consists of a single time series (i.e: the measurement of a single variable, as defined by equation 2.1). In this case the multivariate time series contains $M$ individual time series, and is thus M-dimensional.

In the context of multivariate time series applied to a universal data acquisition and monitoring solution, two aspects are important: dimensionality reduction and network visualization. Each of these aspects will be discussed in the following subsections.

#### 2.2.2.1 Dimensionality reduction

The concept of dimensionality reduction consists in removing from a data set specific data that is considered to be redundant to the available information, maintaining data that meaningfully explains the behavior in the initial data set[294]. This allows reducing noise in the data and/or alleviates the computational strain of algorithms applied further down in the processing pipeline[295, 296].

Dimension reduction can be extended to multivariate time series, however, it is important to take into account that when considering time series, the order of the measurements with respect to time must be preserved[297]. When applied to multivariate time series, the goal is to find a subset of time series that encapsulate the behavior of all the time series.

While there are many different methods of dimension reduction, by far the most popular one consists in a form of *P*rincipal *C*omponent *A*nalysis (PCA)[298]. This method transforms the data set from the space of measurements to the space of the data points and in this space, chooses the set of points that represent most of the variance of the whole data set[299]. It can be easily extended to the analysis of multivariate time series, considering the components of the multivariate time series Fourier transform, which don't suffer from the ordering requirement of the time space[300].

For a m-dimensional multivariate time series ($\mathscr{F}_{\mathbf{X}}$):

$$\mathscr{F}_{\mathbf{X}}(k) = \begin{bmatrix} \mathscr{F}_{X_1}(k) \\ \mathscr{F}_{X_2}(k) \\ \vdots \\ \mathscr{F}_{X_m}(k) \end{bmatrix}, \tag{2.27}$$

where $\mathscr{F}_{X_i}$ is the Fourier transform of the i-th time series, given by equation 2.15. This matrix consists of an $m$ sets of $\frac{n}{2}$ Fourier components which can now be analyzed to determine which set of data sources and Fourier components best represent the variance of the whole matrix data. Matrix $\mathscr{F}_{\mathbf{X}}$ can be decomposed using *S*ingle *V*alue *D*ecomposition (SVD)[301]:

$$\mathscr{F}_{\mathbf{X}}(k) = \mathbf{\Sigma\Lambda\Omega}, \tag{2.28}$$

Where $\mathbf{\Sigma}$ is an $m \times m$ matrix containing an orthogonal set of vectors that transforms the data sources space, $\mathbf{\Omega}$ is an $\frac{n}{2} \times \frac{n}{2}$ matrix containing an orthogonal set of vectors that transforms the frequency space and $\mathbf{\Lambda}$ is an $m \times \frac{n}{2}$

diagonal matrix which contains a set of scaling coefficients in descending order. It can also be shown that equation 2.28 can be represented as:

$$\mathscr{F}_{\mathbf{X}}(m) = \sum_{i=1}^{r} \lambda_{\mathbf{i}} \mathbf{S}_i, \tag{2.29}$$

Where $\lambda_i$ is the i-th scaling component in the $\mathbf{\Lambda}$ diagonal matrix and $\mathbf{S}_i$ is an $m \times \frac{n}{2}$ matrix obtained by the outer product of the i-th column vectors of the $\Sigma$ and $\Omega$ matrices, respectively. The i-th element of the sum in equation 2.29 consists in the i-th *principal component* of the data set[302]. Equation 2.29 is thus the result of a PCA analysis of a multivariate time series and by truncating the sum at a specified number of ($r < m$) principal components, the data set is thus simplified into a smaller subset of sources and frequency components that meaningfully represent the initial data set. To do that, one needs only to look at the components of $\mathbf{\Lambda}$ which contain the information on how each principal component contribute to explaining the original data. The contribution ($\sigma_i$) of each principal component can be calculated from its respective scaling factor ($\lambda_i$) as:

$$\sigma_i = \frac{\lambda_i^2}{\sum_{k=1}^{m} \lambda_i^2} \tag{2.30}$$

Since the SVD of the multivariate Fourier matrix presents its scaling factors in descending order, a maximum value of $r$ can be imposed in equation 2.29 such that:

$$\sum_{i=1}^{r} \sigma_i \geq \sigma_L, \tag{2.31}$$

where $\sigma_i$ is given by equation 2.30 and $\sigma_L$ is a limiting number between 0 and 1 that represents the percentage of the variance of the Fourier components explained by the $r$ first principal components. This value is usually user defined and lies within the range of 90-100% for most applications of PCA.

By decomposing the initial data set via PCA where $\sigma_L$ of the variance is explained, the initial data set is reduced to a smaller and more easily treatable data subset.

#### 2.2.2.2 Graph visualization

Graph visualization is a method of visualizing data that has recently gained popularity[303–305]. By showing data as a set of nodes interconnected with a specific kind of relation, the whole data set can be visualized as a network of interactions and several algorithms of graph theory can be applied to highlight relevant nodes or emphasize communities of closely related data[306]. Graph visualizations can be analyzed according to the number and strength of connections between each nodes in a subset of graph theory called network theory. Prominent uses of graph and network theory in science include condensed matter physics and quantum field theory (see figure 2.12)[307]

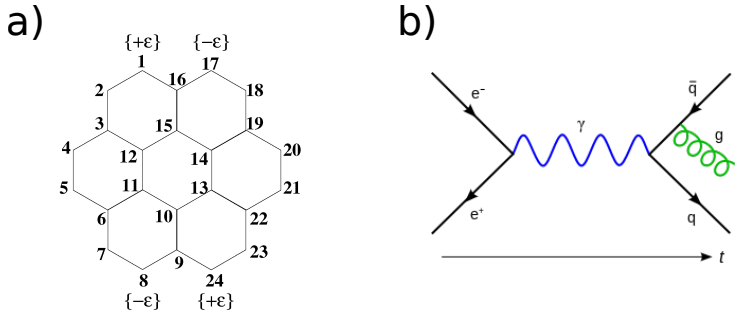



*Figure 2.12: Two examples of graph representations used in separate fields of science: a) Nanoflake graph representation; b) Feynman diagram*

**2.2.2.2.1 Interaction measurement** In order to study a network of time series, a relationship measure must be defined to interrelate the sources of data. Since the measurements being discussed are time series, either of the similarity methods discussed in 2.2.1.2 are potential candidates for use as an interaction measurement between pairs of series. A similarity matrix can be built:

$$\mathbb{S} = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,N-1} & s_{1,N} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,N-1} & s_{2,N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ s_{N-1,1} & s_{N-1,2} & \cdots & s_{N-1,M-1} & s_{N-1,N} \\ s_{N,1} & s_{N,2} & \cdots & s_{N,N-1} & s_{N,N} \end{bmatrix} \tag{2.32}$$

Where $s_{x,y}$ is the similarity measurement between series $x$ and $y$ and can be viewed as the interaction measurement between nodes in a similarity time series graph visualization.

**2.2.2.2.2 Graph representation** When an interaction measurement is chosen, a graph visualization can be created. A graph is a diagram where textual labels (nodes) are connected via line segments (edges). Graph edges can be directed or undirected, depending if the relationship is asymmetrical or symmetrical, respectively. Directed graphs can have two edges connecting two nodes, one for one direction of the relationship and another for the inverse direction (hence the name *directed*), while undirected graphs can have only one connection between two nodes. In this work, the nodes in a graph represent a single signal and the edges are measurements derived from either coherence or partial coherence analysis. All presented graphs are undirected.

Graphs can be presented in a variety of forms:

- Force-based or force-directed layouts: where the strength of the connection between nodes is represented by a force that repels or clusters nodes. These graphs can easily convey the strength of the connections and also provide good interactivity and flexibility[308].
- Hierarchical layouts: where nodes are placed in layers often in a descending order, generating more complex connections as the layer number increases. These layouts are more popular when listing dependencies such as in software debugging [309].

- Circular layouts: where nodes are placed in the perimeter of a circle, and edges are usually connections made inside the circle. These layouts are very popular for use in communications networks[310].

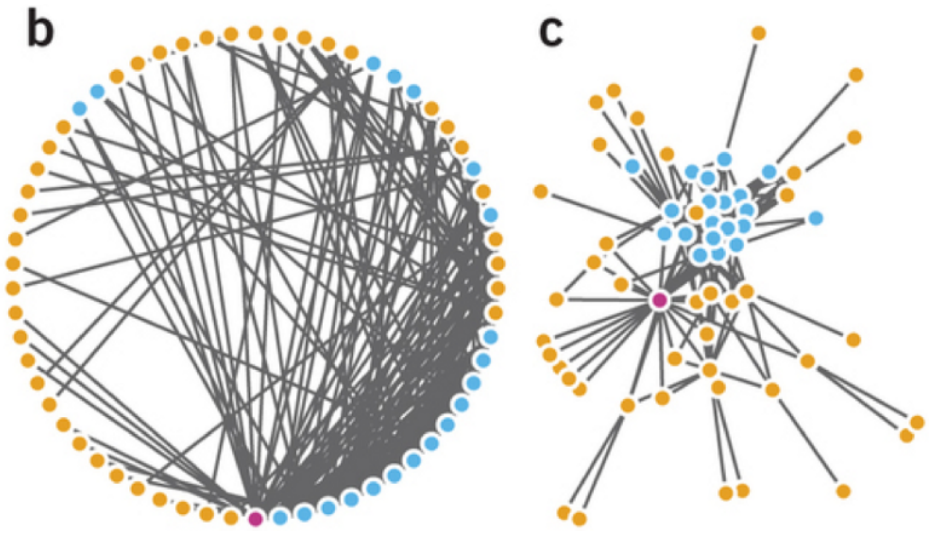



*Figure 2.13: Illustration of two types of graph layouts (circular on the left and force-based on the right). The right graph is more effective at presenting the dependency of individual nodes[311]*

The most appropriate graph to visualize the strength of the connection between an arbitrary number of nodes are the force-directed graphs (as shown in figure 2.13). From these graphs, network theory can be applied and relevant information can be extracted from the nodes and their interactions. This work will focuses on two important pieces of information: centrality and community measurements.

**2.2.2.2.2.1 Centrality** Graph centrality measurements attribute a specific value to each node according to a specific function which often evaluates the amount and weight of the edges linked to the node on the graph[312]. Several centrality measurements exist, 3 specific ones will be considered:

- Degree centrality: the simplest form of centrality measurement, consists in calculating the number of edges connected to a node. It is often normalized to the total amount of edges in the graph. This measure is used to find the most important (most connected) nodes of the network. A more complex measure of degree centrality is *eigenvector centrality* ($\epsilon$) which takes into account not only the number of connections of a node but also the number of connections to connected nodes, which allows the calculation of the most influential nodes in a network[313].

- Betweenness centrality: this measure of centrality focuses on shortest paths. It is calculated by dividing the number of shortest paths between adjacent nodes that pass through the target node by all the shortest paths between the adjacent nodes. This measure allows finding nodes that control the flow of information in a network graph[314].

- Closeness centrality: this measure of centrality focuses on the distance between a node's adjacent nodes. It is calculated by dividing the number of connections of a node by the sum of the distances between adjacent nodes. This measurement is used to find nodes which can quickly influence most of the network[315].

**2.2.2.2.2.2 Communities** Another important aspect of graph analysis is the grouping of different sets of nodes into communities. These communities of nodes are useful not only to aid the user in understanding the structure in the network, but to highlight specific sets of nodes that may have similar behavior and should be studied together when assessing the influence in the overall system[316]. Several algorithms exist to calculate the existence of communities in a graph[317]. For this work, the Louvain algorithm is chosen, which focuses on optimizing a quantity called *modularity* ($Q$) with respect to the number of communities and their nodes:

$$Q = \frac{1}{2m} \sum_{i,j} \left[ A_{ij} - \frac{k_i k_j}{2m} \right] \delta c_i, c_j \tag{2.33}$$

Where $A_{i,j}$ is the edge weight between nodes $i$ and $j$, $k_n$ is the sum of weights of edges attached to node $n$, $m$ is the sum of all edge weights in the graph and $c_n$ is the community of node $n$. By finding shifting nodes between initially assigned communities and calculating the shift in modularity an optimal arrangement of nodes in communities can be made that maximize the modularity of the network[318].

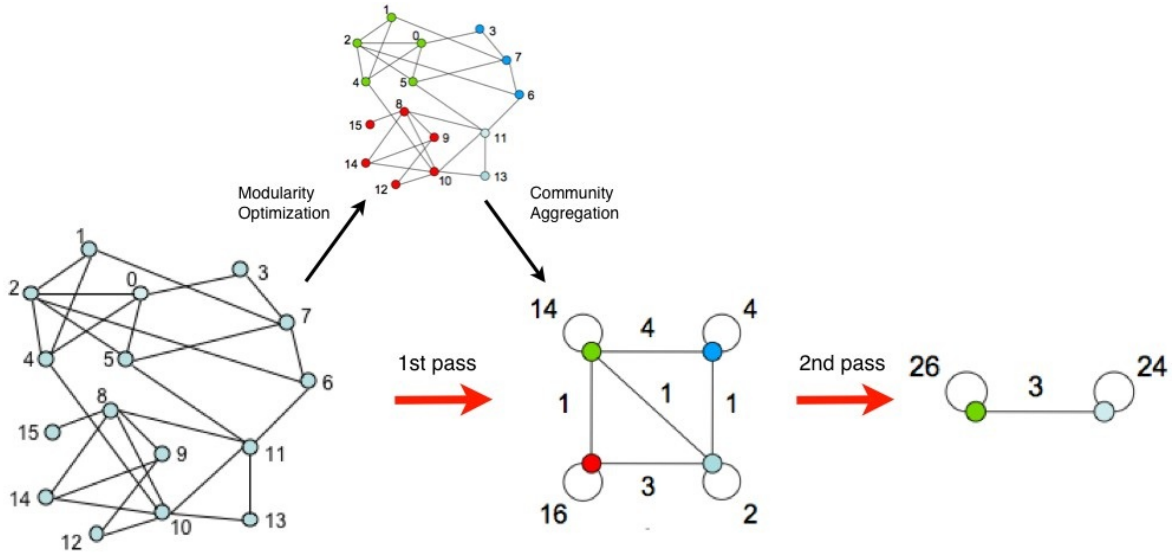



*Figure 2.14: Illustration of the Louvain algorithm. A first step of community assignment is performed by modularity optimization is performed and a second step of community aggregation is made. These steps are repeated until a maximum step of modularity is achieved (source[318]).*

# Chapter 3

# DAQBroker framework & performance

DAQBroker is the proposed solution to the challenge of providing an open source framework for instrument monitoring and control. In this chapter the first section describes the main components of DAQBroker and the second section evaluates the performance of DAQBroker on a working example of the framework on controlled infrastructures of varying computational power.

## 3.1 DAQBroker framework

DAQBroker is a python-based web framework that focuses on instrument monitoring and aims to provide monitoring and control methods for any type of instrument. The framework itself focuses on three major components:

- **Communication** - strategies for conveying information from an instrument to a centralized repository,
- **Storage** - methods for collecting, organizing and storing instrument data,
- **Interface** - solutions for data access and manipulation.

While interconnected, each component presents clearly defined challenges in the context of scientific instrument DAQ. Most of DAQBroker's code is written in Python with the exception of the supplied web interface, which uses the standard HTML, Javascript and CSS. The following subsections will thoroughly define and justify each of the choices made for the aforementioned components. The last subsection is devoted to security mechanisms, access control and data access in DAQBroker.

### 3.1.1 Communication

The current version of DAQBroker allows instrument data to be gathered from either a single machine or from a network of machines. The network communication is implemented via a server/agent architecture that emphasizes

on minimizing the load on the client machine[1]. The protocol used was built over TCP sockets, but provides a more relaxed communication scheme to handle slow or malfunctioning machines.

#### 3.1.1.1 Server/agent architecture

Since the source of data (different scientific instruments) is varied, similar requests require instructions to be tailor made for each instrument, hampering remote procedure calls. The use of the Internet and local area networks has made wide-spread the usage of networks and the paradigm of network communication introduced in 2.1.1.2, pushing the use of specialized buses into a secondary alternative. For those reasons, DAQBroker is prepared to be deployed in a TCP/IP network environment, where multiple computers are acquiring data from multiple instruments. Its structure consists of two python-based applications that communicate in a networked server-client architecture:

- **Server** - focuses on connecting to the different associated database engines - remote or local - and analyzing the different DAQBroker databases that exist in those engines. With this information this application identifies the connected instruments and linked computers in the network (nodes). It then periodically requests information from the different nodes running the agent application. This strategy is called a push communication method[319]. Though this method is not the most scalable for large numbers of machines[320], message size and quantities are optimized and as small as possible, to prevent network overflow. This application is also responsible for the main experiment data flow, ensuring that the most recent instrument data is stored in each of the connected database engines (refer to 3.1.2 for more details on storage)
- **Agent** - focuses on maintaining communication with one or more server applications and provides answers to specific requests issued by those servers. The use of an agent application allows it to be personalized for the specific instrument control tasks. Apart from providing new data from the scientific instruments on request, the agent also provides health information of the machine running it, an important factor to keep in mind on a network of instruments.

While not being the most scalable option, a server-agent architecture with push communication provides the least amount of extra load on the different client machines, which are expected to be running computationally intensive programs for individual instrument data acquisition. The use of an agent application also provides customization opportunities for instrument control, a feature that can not be guaranteed in a network architecture based on remote procedure calls (see for instance[321]).

#### 3.1.1.2 Communication protocol

The communication between the server and each agent application is based on TCP sockets run over a high level distributed messaging library called ZeroMQ[322]. ZeroMQ provides a special class of high level sockets that fulfill specific communication paradigms with performance and concurrency. In DAQBroker, these sockets are used to send personalized messages between the server and the agents. The agents interpret the orders from

[1] As the client machine may be using most of its resources to run important instrument application(s)

the server and provide a response. However, the ZeroMQ socket communication is still very rigid and a single malfunctioning agent machine could slow down the server machine due to connection loss or high agent loads. To overcome this problem a parallel push-pull method of communication is implemented in which both the server and agent application contain dedicated message receiver and sender loops, which interpret and send messages between each other, ensuring concurrency in both cases. In order to ensure that all machines that are assigned instruments are available on the network, a server-wide message is sent to the network via the local network's broadcast port. Machines running the agent application in the local network will receive this message and reply to the server with specific information about the machine, including a dedicated message listener port and the network unique identifier of the machine. Agents not on the local network of the server machine run an active secondary communication loop, periodically contacting the server machines to convey the same information. After a machine is deemed operational (i.e: running the agent application and reachable through the network), requests are sent via the following steps (illustrated in figure 3.1):

1. A (ZeroMQ) TCP socket (S1) is established between the dedicated server message sender and the agent message listener port,

2. The message is encoded and sent to the agent and the socket (S1) is closed. In this message a number for a dedicated server message listener is also provided, along with the unique identifier(s) of the instrument(s) and the respective database the message is aimed at,

3. The agent receives the message via the dedicated listener, decodes and interprets the message. The agent executes the order in the message and creates a new socket to the dedicated server listener (S2),

4. The response message is encoded and sent back to the server with the unique identifier(s) of the instrument(s) and respective database the message was aimed at, the socket (S2) is then closed,

5. The server receives the response message, interprets the request, instrument and database it was aimed at and updates the respective database accordingly.

This system allows for multiple sockets to be sent in succession, without need for waiting for the remote machine to interpret orders and provide replies, since the socket is immediately closed after the message is sent. This ensures that a single server/client machine can run without being interrupted or delayed by slow or malfunctioning client/server machines.

### 3.1.2 Storage

Two types of data storage are used in DAQBroker. User and server configuration data are stored in a local file database and instrument data is stored in each user provided database running any type of major consumer SQL engine. This subsection will introduce this storage strategy as well as the existing methods for data gathering and the main storage process loop.

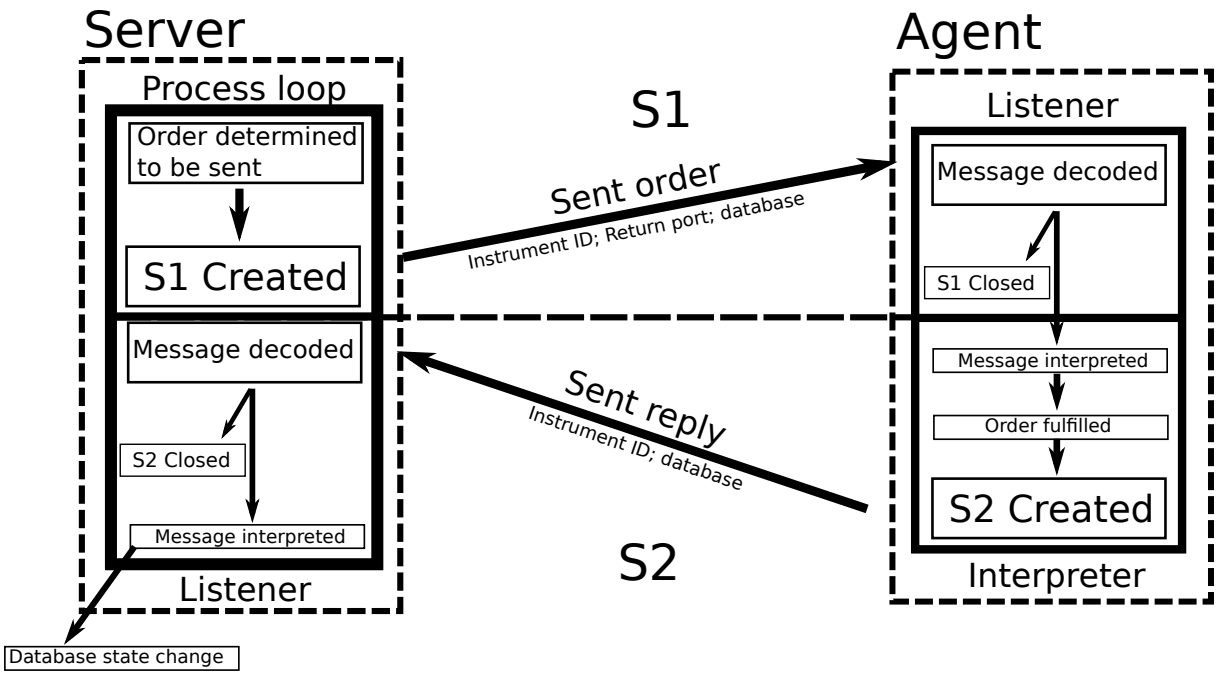



*Figure 3.1: Illustration of communication between a server and agent applications. The communication requires the creation of 2 sockets (S1 & S2) and a total of 4 independent processes evenly divided between the server and agent applications (refer to 3.1.2.5 for more information).*

#### 3.1.2.1 Data partitioning

The first data storage concern before considering individual instrument data is to define how data belonging to different scientific efforts is to be partitioned. The main concerns in choosing how to partition data are the following:

1. A flexible format that allows long single experimental efforts to be recorded as well as small and staggered ones,
2. Clear separation between periods of different experimental parameters,
3. Hierarchical structure to allow separating periods of both major and minor changes in experimental parameters.

To that end the structure used in the CLOUD experiment was implemented, consisting of a hierarchical structure of *Campaigns*, which contain experimental *Runs*, which in turn contain *Stages*:

- Campaign - this is the largest data partitioning block, which represents a period of scientific experiments where an effort is made to study one or several effects using a particular set of instruments. Using the example of the CLOUD experiment, a campaign represents a time period where the chamber is in one of its two main operational modes (see 4.1.2),
- Run - an experimental effort where the influence of a particular set of parameters is studied. This data partition defines a period of time during which several experimental parameters are changed to study the development of certain phenomena. In the particular example of the CLOUD experiment an experimental run will consist of a single particle formation event, where experimental parameters are changed to study

the chemical dependency of newly formed particles or a cloud formation run where the composition of the atmosphere is changed to study the formation and lifetime of clouds.

- Stage - the smallest data partitioning block that consists in minor changes (often and ideally a single change) of experimental parameters to study the immediate influence of those parameters on the overall experimental effort. In the particular example of the CLOUD experiment, an experimental stage consists in simple changes to experimental parameters to change the generation of new particles or clouds (e.g: turning on UV lights, injection of seed particles or changing of gas concentration levels).

This data partitioning hierarchy is implemented in DAQBroker by separating each campaign into separate databases, while each database contains an internal record of its experimental runs and stages. This partitioning model should provide enough flexibility to be used in most scientific data gathering efforts (see appendix A for more information).

#### 3.1.2.2 Instrument set as a data warehouse

Instrument data comes in a wide variety of formats, sources and granularity. However, there is one common quantity to all data gathering instruments, the time at which measurements are taken. Time allows data from different instruments to be compared and, when required, to be manipulated to create new measurements. Time is thus the best parameter to base the data storage and searches for a single or multiple instruments. The different sources and formats of instrument data consist of a top-down design concept of instrument data warehousing is introduced where the instrument is the at the top. The instrument block is presented in figure 3.2 and is divided into smaller more specific blocks:

- **Instrument block** - The most general block that encompasses all others and defines the basic information about an instrument (name, operator info, contacts, etc...). This block allows separation between different instruments' data.

- **Data source block** - Provides information about a specific data source from an instrument. This block allows DAQBroker to decide what data gathering method to use and what information to feed to the corresponding method (ex: file data gathering requires a folder, file format and extension info).

- **Data channel block** - The most specific block of an instrument, each channel is associated to a data source block and is related to a single stream of unit information (ex: number or string). Data for each channel are associated with a specific time value or range.

An instrument model as the one described allows most instruments to be properly defined by providing the users a means of supplying each block. In order for data gathering to be performed in DAQBroker, an instrument (already defined by its main block) is required to have at least one data source and channel blocks. With the instrument identification as well as a common data attribute for storage and searches to be performed, it is clear that the approach of DAQBroker is to create a relational database of instrument data.

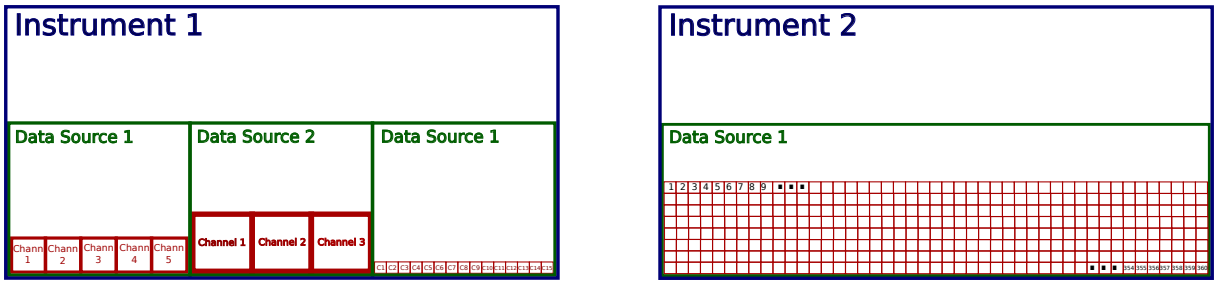



*Figure 3.2: Illustration of the instrument data warehouse model with two examples. Left instrument with 3 different data sources, each one having a varying number of channels - 5, 3 and 15. Right instrument with a single data source and 360 data channels.*

#### 3.1.2.3 The case for noSQL

At first glance, the reader could be inclined to consider that noSQL databases would be the best approach for this kind of universal data storage approach, since as stated previously, these databases are much better suited to handle heterogeneous data and would also facilitate the horizontal scalability of the application. However, two key features of relational databases win out over noSQL databases. The first is the ability for the underlying database management system to make operations that ensure the continuity of the database in case of failure (often called ACID transactions[323]). The other feature is the ability to preform complex queries over the underlying data. While noSQL databases allow for a more heterogeneous set of data structures, they severely limit the range of inspection over that data by the user, making it necessary to create third-party applications for the handling of this heterogeneous data. While a solution for DAQBroker could be implemented in terms of a noSQL database, that was not the selected option, the process of implementation would be wildly different and the instrument concept would have to be revised.

#### 3.1.2.4 DAQBroker database schema

As described previously, DAQBroker uses a relational storage model for instrument data. A DAQBroker user may choose to use one or many databases for storing their data, and a single database should contain the instruments that gather data for the same scientific objective. A single instrument database follows a hybrid star/snowflake *schema*. In figure 3.3 an example database without any inserted instrument is provided. The *schema* contains the following tables that define the instrument using the previously defined model (a full description of the schema is provided in appendix A):

- **Instruments** - Contains the information of the main instrument block. The attributes of each instrument are as follows:
    1. *Name* - String defining the name of the instrument. This name is unique and cannot be repeated.
    2. *instid* - Integer uniquely defining the instrument.
    3. *active* - Boolean defining whether an instrument is in active data gathering or not,
    4. *description* - Blob that contains a description of what the instrument is supposed to do,

5. *username* - String relating the instrument to the DAQBroker user that created it,
6. *email* - Blob containing contact information for the instrument operator (does not have to be just email),
7. *insttype* - Integer that defines the type of instrument storage format,
8. *log* - JSON encoded blob that contains a public electronic instrument log that users can fill out to help keep track of changes made to the instrument.

- **Instmeta** - Contains the information of each instrument's data source blocks. The attributes of said blocks are as follows:
  1. *clock* - Integer defining the last edit timestamp of the data source,
  2. *name* - String identifying the data source,
  3. *metaid* - Integer uniquely identifying the data source,
  4. *instid* - Integer relating the data source with its instrument (see Instruments - instid),
  5. *type* - Integer defining the type of data source (allows specific data gathering methods to be chosen),
  6. *node* - String defning the network node associated with this data source,
  7. *remarks* - JSON encoded blob containing relevant information for data gathering methods for the type of data source chosen,
  8. *sentRequest* - Boolean that tests whether a remote data gathering routine is under way (for time out purposes),
  9. *lastAction* - Integer defining the timestamp of the last automated action preformed on this data source,
  10. *lasterrortime* - Integer defining the timestamp of the last error encountered when preforming data gathering for this data source,
  11. *lastError* - Blob containing information of the last error encountered when preforming data gathering for this data source.
  12. *lockSync* - Boolean to test whether there is a data gathering action currently taking place for this data source (used with *sentRequest* for timeout purposes).
- **Channels** - Contains the information of each instrument's data channels. The attributes each channel are:
  1. *Name* - String containing the name of data channel,
  2. *channelid* - Integer uniquely identifying the data channel,
  3. *channeltype* - Integer that defines the type of data channel (used for data manipulation),
  4. *valuetype* - Integer that defines the type of data returned by this channel (i.e: string, numbers),
  5. *units* - String containing the physical units of the data provided by this data channel,
  6. *instid* - Integer relating the data channel with its instrument (see Instruments - instid),

7. *description* - Blob containing a description of the data provided by the data channel,
8. *active* - A Boolean that tests whether the data channel is actively providing data,
9. *remarks* - JSON encoded blob containing relevant information for automated handling of the specific data channel,
10. *metaid* - Integer relating the data channel with its data source (see Instmeta - metaid),
11. *lastclock* - Integer defining the timestamp of the last gathered data value,
12. *lastValue* - String containing the last gathered value on the channel,
13. *fileorder* - Integer defining an order for the data channel to be gathered (only used on specific data gathering methods),
14. *alias* - String defining an alias of the channel to be used in certain data gathering methods,
15. *firstClock* - Integer defining the timestamp of the first gathered data value (can be updated if newer values are gathered).

These 3 tables contain the relevant information that fully define an instrument and preform data gathering. When an instrument is created via DAQBroker, two new tables are created:

- ***_data** - Contains the data gathered from the instrument (the '*' in the name of the table is replaced by the instrument name). The attributes of this table are the names of the non-custom data channels,
- ***_custom** - Contains the custom data gathered from the instrument. This includes data that is a direct result of manipulation of data between one or various instruments. The attributes of this table are the names of the custom data channels of the instrument.

With the creation of these final two tables for each instrument, data from instruments can be stored and queried via the snowflake. Other tables are also created to allow the storage of information from other relevant DAQBroker services and are not part of the aforementioned snowflake thus the hybrid star/snowflake *schema* definition.

The main advantage of this schema for instrument data storage is that an individual instrument's data injection can be made in parallel of other instruments under the same RDMS, meaning that, under the same campaign, one instrument will not be responsible for locking data storage of other instruments since only that instrument's table is locked for data injection.

While a network of computers will often be connected to a single machine running a DAQBroker server (or more if one server machine is not powerful enough or the network is too large), it is possible that multiple servers connect to the same database engine and store instrument data on that database. Thus a DAQBroker database is not geographically bound to a location as long as different instrument locations have access to a specific database engine (e.x: AWS RDS, institution server).

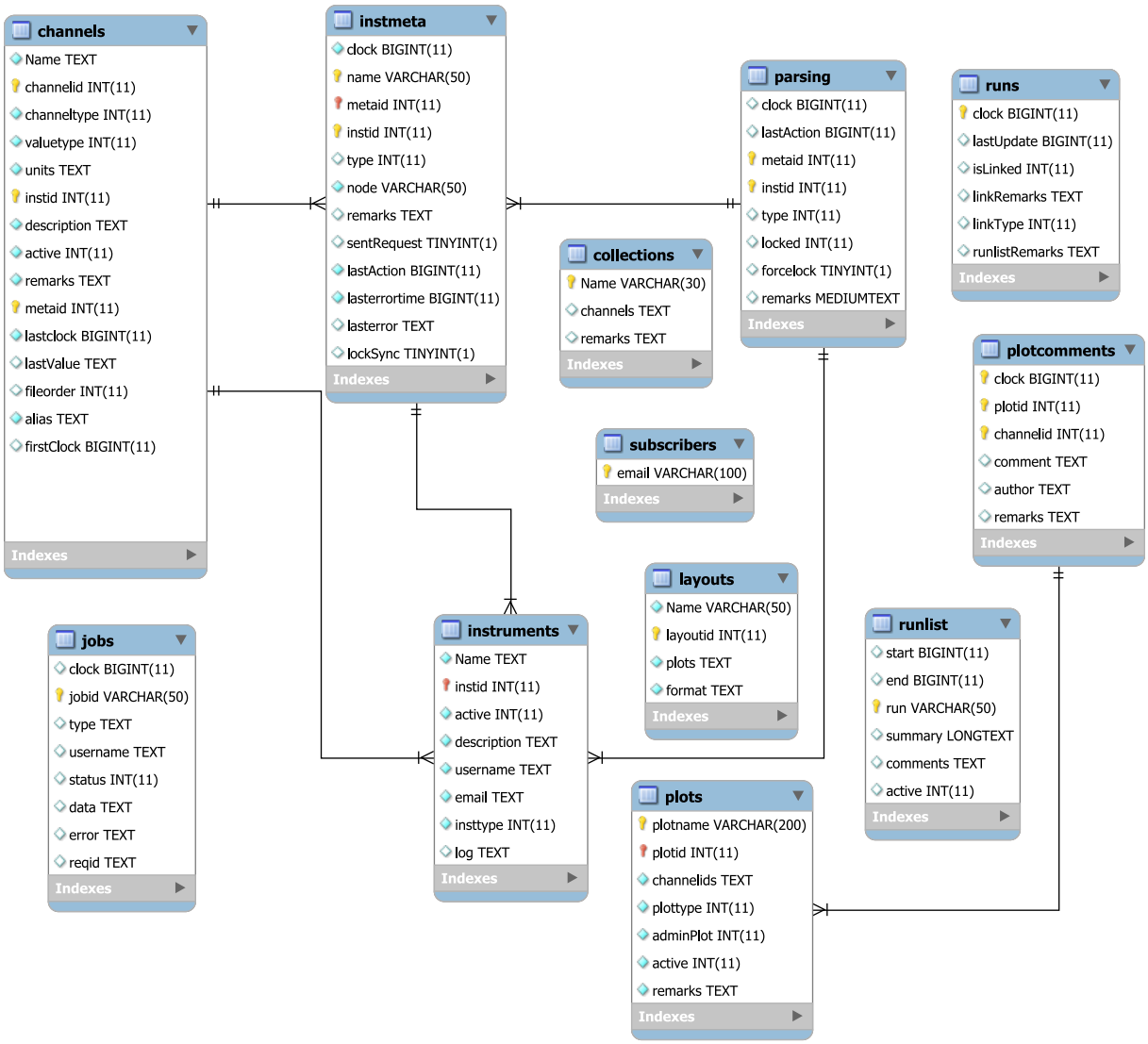



*Figure 3.3: Enhanced Entity Relationship (EER) diagram of an empty DAQBroker database. A full description of all tables can be found in appendix A*

### 3.1.2.5 Database monitor

Each DAQBroker server allows one or more local or remote SQL database engines to be used to store instrument data. For each user-provided engine, the server creates a process that continuously queries the database instruments' data sources to evaluate when they require actions to be preformed. When an action is to be preformed on an instrument's data source, it must be properly identified and its relevant information must be conveyed to the appropriate node so that the proper data acquisition action is taken. This subsection will define the main process loop for continuously monitoring a database engine as well as the different available instrument data gathering methods.

**3.1.2.5.1 Process loop** The process loop for each user-provided database engine is depicted in algorithm 2:

This loop allows for continuous monitoring of the database engine and sends update messages only from active

```
Data: Engine login information
Result: Continuous database engine analyser
connection=dbConnect();                                  /* Create initial engine connection */
while True do
    connection.update();                                 /* Updates the connection object */
    nodes=nodeCheck();            /* List of local available machines - separate process */
    bookeeping();                                        /* Run global engine book keeping functions */
    for db in connection.databases do ;                  /* list of active databases */

        msgToSend=[] ;                                   /* List of messages to send */
        DBbookeeping();                  /* Run specific database book keeping functions */
        for inst in db.instruments do ;                  /* List of active instruments */

            for src in inst.dataSources do ;             /* List of active data sources */

                if (src.node included in nodes) and (src.lastAction-currentTime < src.actionPeriod) then
                    msgToSend[node].append(createMsg(src));          /* Update node message */
                end
            end
        end
        for message, node in msgToSend Where message not empty do
            sendMessage(message,node) ;                  /* Send messages to appropriate nodes */
        end
    end
end
```

**Algorithm 2:** Main process loop for monitoring each database.

databases with instruments that are active and have active data sources, to nodes that are contactable by the server application. An instrument from one database can contain data sources from machines in different networks that are being managed by different server machines. While it can be argued that a database engine that contains several DAQBroker-generated databases will become increasingly slower, the operations preformed in this loop are aimed at an overall small number of target rows[2] and thus one database loop should not take more than several milliseconds on an appropriately designed machine.

**3.1.2.5.2 Data gathering methods** When it is determined by DAQBroker's database engine process loop that an instrument is required to provide new data, a new process is initiated to gather new instrument data, order it into each individual data channel and store it in the respective database (either locally or remotely depending on the assigned network machine of the data source). DAQBroker currently supports 3 methods of data gathering and one method for custom data gathering:

- **File parser** - A large amount of instruments produce data in the form of data files. Those files will often contain extremely free form formats. DAQBroker allows users to define the format of an instrument's output file by providing the channel separator, an option of providing an example file to extract individual channel names and specify the format of the timestamp (figure 3.4). This parser allows virtually any ASCII instrument ASCII output file to be immediately read out to a DAQBroker database, as well as provides the option

[2] A single instrument is not expected to have on average more than 5 data sources - even though very specific instruments can exceed that value - and a single database containing more than 100 instruments is already considered to be a large scale database (that should consider partitioning and multiple server machines). Thus the total number of operations for each nested database loop should not exceed 300 and this can be quickly preformed by any low-mid range machine.

to back up the created files, for later use by the users,

- **Serial port reader** - Some instruments' DAQ provide information via serial communication. DAQBroker allows such instruments' data to be gathered and stored. The user has full control over the settings of the serial port as well as the serial port to use (either local or remote). Users can also choose to send a command via the same serial port to request new data in the case that the instrument produces data by request,
- **Network port reader** - Apart from serial ports, some instruments' DAQ will provide data via network ports. DAQBroker provides users the ability to set up data sources that listen or even provide commands to network ports much in the same way as serial ports. Network data requests are sent to the machine of the network port and the request is made locally to emulate the serial port, such that the load of processing the request is left to the data machine,
- **Custom data gathering** - Custom data refers to data created from one or several channels from one or several instruments. The current iteration of the data manipulator allows for data from different channels to be gathered and manipulated into a new value which is a function of the different chosen data channels. This function can come in the form of a simple mathematical expression but can also contain conditionals and loops as functions of channel values using the Python language. In the near future further functions will be added, such as means (weighted and not), derivatives and string handlers.

The decision to use different data gathering methods depends on the type of data source (refer to 3.1.2.4). The open source nature of DAQBroker's code allows for easy integration of new data gathering methods as well as expansion of the existing ones by adding new methods to the project.

### 3.1.3 Interface

Interface is the last major component of the DAQBroker framework and of great importance, as it provides users with the tools and visual methods of interacting with the instrument environment provided by DAQBroker. Since in a shared instrument environment, most users will not be familiar with all instruments, a simple interface is required to access the relevant data from each instrument in a universal way. In order to provide off-the-shelf functionalities to users from a wide spectrum of programming experience, a web application was designed to query and interact with DAQbroker databases. This application contains a dynamic HTML interface based on AJAX (**A**synchronous **JA**vascript and **X**ML) requests [324] with an underlying RESTful API built with Python and Flask. This subsection will cover the user interface and underlying API.

#### 3.1.3.1 Front-end user interface

The main user interface is presented in figure 3.5. It contains several different tabs on the left side that provide users with the ability to query and interact with the instrument environment visible by the server machine, and tabs on the right side that provide users with settings and/or administrative tools. Depending on the type of user that is connected, some tabs may have limited functionalities or even be off limits, raising an error to the user in case he

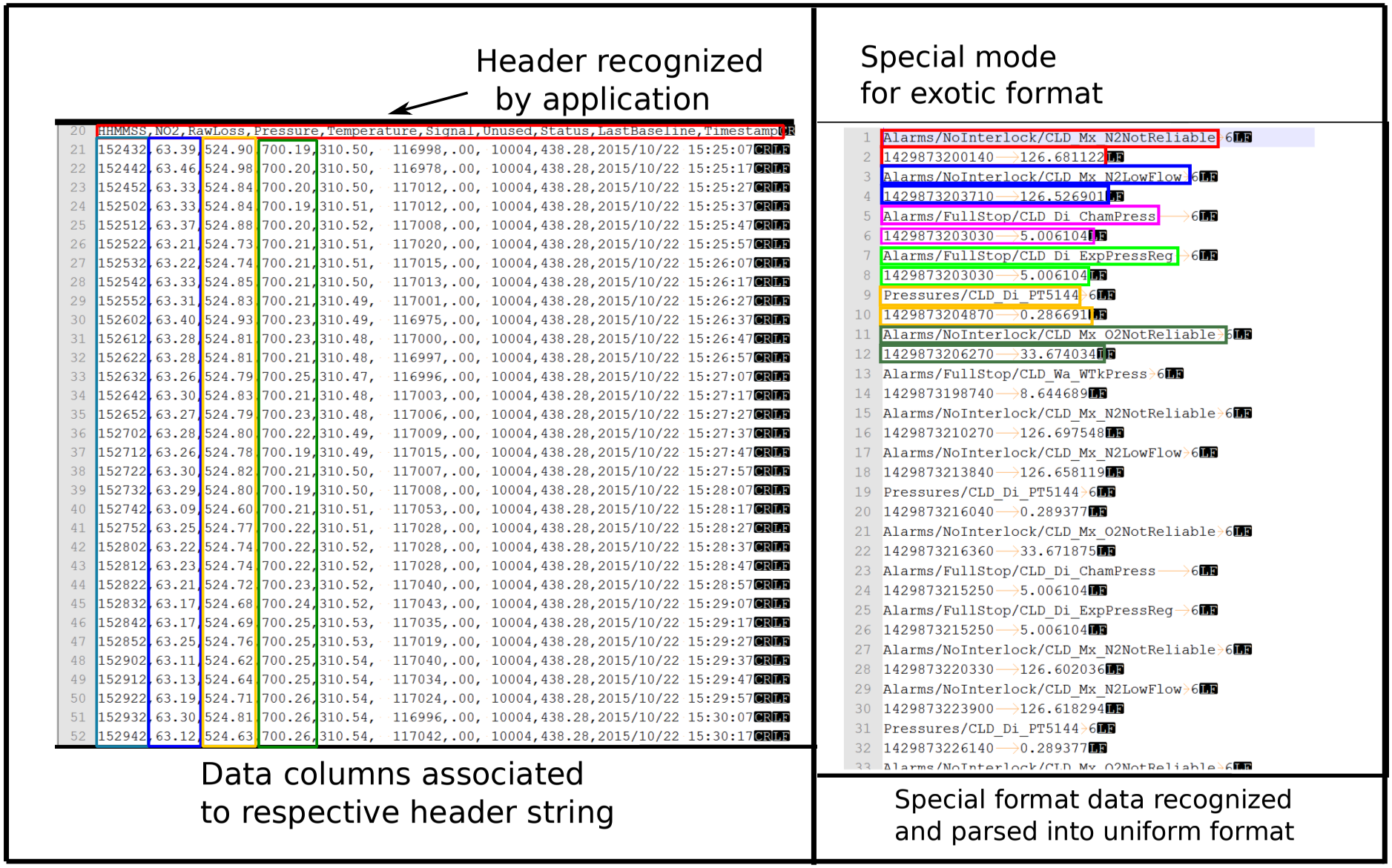



*Figure 3.4: Example of file data gathering from two different formats.*

decides to attempt access (refer to 3.1.4 for more information regarding users and access control). The left tabs are the following:

- **Instruments** - The default tab of DAQBroker. Users can access this tab to create/edit/view existing instruments, users can access instrument logs and access data for individual channels. To create an instrument the user is prompted to go through a 3-step form to provide information for each of the blocks of the instrument (refer to 3.1.2). Created instruments are bound to the user that created the instrument and can only be edited/deleted by said user or an administrator.

- **Data** - Provides users with the ability to access and manipulate instrument data. Users can create and share visualizations of channels from different instruments or even create visualizations of linear functions of existing channels. Users can also use this tab to access visualizations created by other users and insert comments on existing visualizations to provide interpretations for specific features in the data. This tab can also be used to download data from different instruments directly into a file to be used by other data analysis programs.

- **Runs** - Using this tab, users can keep a record of different experimental periods of the data collected in a database. Each experimental period is called a run and each run can be divided into any number of stages. Only specific users can create and stop new runs/stages but any user can insert comments into a run/stage to instruct other users on specific events that occurred during the run. Runs and stages can be shown in every data visualization to provide users with information when the experimental period changed states or they can be downloaded into a file for use in other data analysis programs. On first use of the tab, users are prompted to create a set of experimental parameters to change between each experimental run/stage. This list can be edited in case a new parameter is added later to the experiments.

- **Nodes** - This tab provides a list of all machines connected to the server machine. Users can access this tab to acquire health information about network machines. They can also use this node to gain access to the agent application across many available platforms, so that it can be installed and run on other machines. Administrator users can access this tab to insert new machines into the network by providing a pairing string that can be used to find the agent machine in the network. Administrator users can also decide whether or not an NTP routine is applied to specific machines for time synchronization. This tab is specific to a single machine running the DAQBroker server application. If a user accesses a machine in a different network, even if connected to the same database, this tab will provide a different list of nodes.

The right hand tabs are the following:

- **Database chooser** - A drop-down menu that provides users with a list of databases present in the connected engine. Choosing a different database will change the information provided to the user on most left-side tabs

- **Database administrator** - Provides administrator users access to database engine administrator tools. These users can access a list of databases and users and change their settings. Databases can be set to active or inactive, providing or removing the ability to collect new instrument data. Users can be changed between their many types. Administrators can also create and delete existing users and/or databases.

- **Local administrator** - Administrator users can access this tab to edit local settings of the server machine. These users can edit the existing paths for backup and import, set and provide NTP servers for time synchronization, change the network ports for communication and logging and access the local logging information for specific time periods.
- **Logout** - This tab logs the current user out of the application and returns a login prompt for users to provide new login credentials.

#### 3.1.3.2 Back-end RESTful API

DAQBroker provides a comprehensive API based on HTTP calls to manage and modify the instrument environment. Each call requires the proper user authentication before any changes can be attempted. The most recent version of the API can be found at http://daqbroker.com/documentation.html.

### 3.1.4 Security

One final important aspect of DAQBroker and of any web-based application is security. With off-the-shelf software and HTTP APIs come inevitable security concerns and possible avenues for malicious interventions. While rarely the target of these attacks, the scientific community should have prime responsibility in implementing security methods because scientific products often bleed out to industry and individual use. This subsection will discuss the common web-based attacks that could affect DAQBroker and the security measures implemented to prevent said attacks.

#### 3.1.4.1 Role Based Authentication

In order to control the power each user has over the underlying DBMS, DAQBroker was built with a role-based authentication (RBA) with 4 different user types ordered below by increasing privileges in access to changing the instrument environment:

- **Guest** - The user with most basic privileges. Can only view instrument data and is not allowed to create or edit instruments, runs or provide comments. The intended use is to provide data access to external collaborators,
- **Operator** - This user type is assigned to instrument operators, who are in charge of one or more instruments and/or even a network machine. This user can create and edit the instruments created by itself as well as create visualizations and provide comments to runs and visualizations,
- **Run coordinator** - This user is assigned to control room operators or experiment coordinators. Apart from having the same privileges as an operator, a run coordinator can create/edit/remove experimental runs/stages and edit experimental parameters,

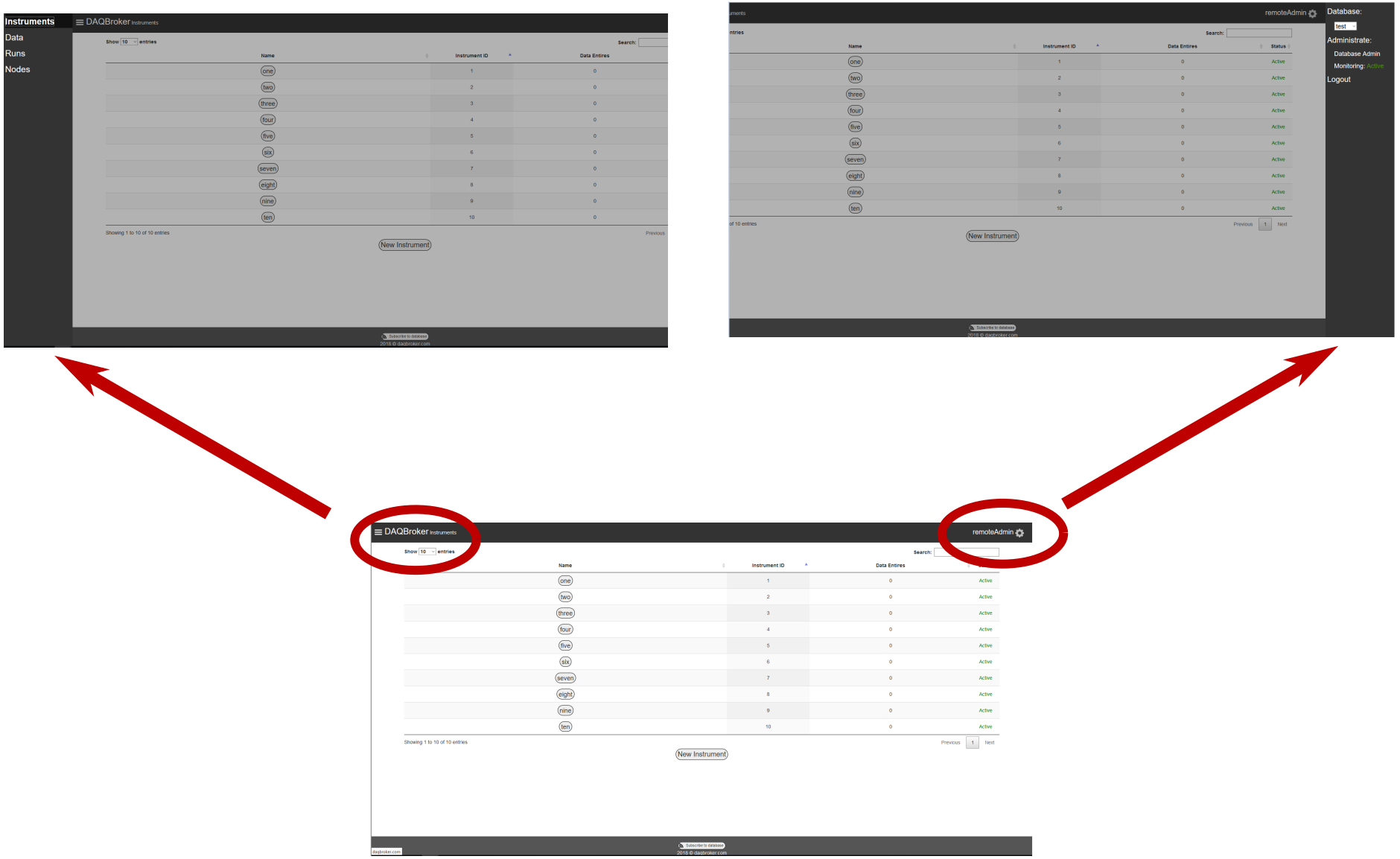



*Figure 3.5: DAQBroker home screen (bottom) and different tabs (top). By interacting with the title and settings buttons (red), the user can access the different utilities of the software, while the majority of the screen is free to show the tools of the current tab.*

- **Administrator** - The user with the most privileges. An administrator user has control over all created instruments, runs/stages and connected machines. An administrator user is also responsible for creating new users and databases.

This privilege hierarchy minimizes the risk of compromising instrument data when a user has their credentials compromised. Operator users only have full control over their own instruments' data and are only able to view (and not edit) other users' data. Only the run coordinator and administrator accounts can inflict changes that affect global information on the database. It is thus expected that only a few administrator and run coordinator accounts exist per DAQBroker instance, thus limiting the number of accounts that, if compromised, can alter/remove multiple users' data. The operators are thus responsible for keeping their own accounts secure at the risk of compromising only their own instruments' data.

#### 3.1.4.2 Injection/XSS/XSRF prevention

Web-enabled technologies give users with malicious intent the opportunity to execute unexpected commands via the provided API and/or user interfaces. With the use of SQL as a common querying language, input that is not properly protected can be a target of *SQL injection* and if improperly handled by the server application can be used to run unintended commands for other clients accessing the user imported data (ex: Running a key logger to capture a user's credentials). This is called *Cross-site scripting* (XSS). The improperly authenticated user sessions maliciously '*hijacked*' is *called cross site request forgery* (XSRF).

DAQBroker's database connections and requests are handled by the a popular open-source database toolkit SQLAlchemy (see 3.2.2). This toolkit contains bindings for user input that dramatically reduce the risk of SQL injection. Additions to the software, however, must be properly tested to ensure that no avenues for SQL injection are inadvertently created. DAQBroker itself is created under the Flask framework, that uses the Jinja2 language[325] to generate website templates. This language already checks for arbitrary HTML attempts to be injected via templates. In the case of DAQBroker's back-end API, all requests generate a unique id that is created at the start of a session and is written to all templates and sent to all requests in order to validate the user making the request, which drastically reduce the risk of XSRF. Similarly, all requests requiring user input are also thoroughly analyzed and escaped to prevent unwanted code execution via that avenue.

## 3.2 DAQBroker architectural choices

### 3.2.1 Communication

As stated in 3.1.1 the communication protocol for DAQBroker was built over TCP with safeguards in place to handle remote instruments with poor/without connection to allow graceful termination of communication attempts. The protocol was designed on top of ZeroMQ message sockets, which are essentially TCP sockets with the possibility of handling bad connections and timing out communication attempts. This is basis for the internal commu-

nication that is used between DAQBroker agent and server applications to handle data requests from instruments. For external communication with a DAQBroker application other options are provided (see 3.2.3)

### 3.2.2 Storage

Storage in DAQBroker follows a hybrid snowflake data warehouse relational model as described in 3.1.2. To provide support for all popular RDBMS, a popular open-source python **O**bject-**R**elational **M**apper (ORM) called SALAlchemy[326] was used. This ORM abstracts the particular needs of each RDBMS into classes objects used by the underlying programming language (in this case Python, which uses classes to define the different tables and their relations). This provides a dependency injection development pattern when dealing with the definition of the schema and the connection to the RDBMS itself.

### 3.2.3 Interface

In 3.1.3, it is stated that DAQBroker provides a web interface via a REST API. This API is responsible for handling user authentication, instrument **C**reation, **R**eading, **U**pdating and **D**eleting (CRUD) operations as well as data access (data requests for specific channels). This API is built under the Flask framework, which is an extensible framework under Python that provides methods for handling all the aforementioned interactions. To provide users with a graphical way of interacting with this API, a front-end web application is also provided with DAQBroker that allows users to easily create, edit, delete instruments and access/manipulate instrument data, experimental runs using the Flask API. This application is built using standard web tools (Javascript, HTML) along with some graphical frameworks such as jQuery for DOM manipulation.

## 3.3 DAQBroker performance

As discussed in the previous chapters, an application with the scope of monitoring sets of scientific instruments requires not only a comprehensive and simple interface, but also the possibility to handle a changing and often growing set of instruments. Several choices made for DAQBroker to be user-friendly can affect its performance. In this section the performance of the current version of DAQBroker is evaluated by applying a set of tests designed to mimic examples of real-world use. The focus is on testing the performance of the database monitoring process under changing sets of instruments and available computational power.

Two machines were chosen for the following tests to represent low (LP) and high (HP) computational power machines. The specifications of each machine are highlighted in figure 3.6:

When relevant, tests will be duplicated first using a local database engine and second a remote AWS (**A**mazon **W**eb **S**ervices) free tier database[327] to study the effect of decentralizing the database engine. All machines are running on the same network served by a TP-LINK AC1200 Gigabit Router[328]. Randomized instrument data will be collected from a single low power Raspberry pi machine running the DAQBroker Agent application. This

Low Computing Power (LP)

- CPU: 4× ARM Cortex-A53, 1.2GHz
- RAM: 1GB LPDDR2 (900 MHz)
- Net: 10/100 Ethernet
- ROM: SanDisk Ultra® microSD UHS-I 16GB

High Computing Power (HP)

- CPU: Intel Core i7 5960X (8x HT) @ 3.00GHz
- RAM: GSKill 32GB DDR4 (3200 MHz)
- Net: Intel Gigabit LAN
- ROM: Crucial MX200 500GB

*Figure 3.6: Specifications of the two machines used to test the DAQBroker software. Considering the number of possible threads that can be created at the same time and the speed difference between them, the high power computer has around 10 times more computational power. However, there is always demand for low power machines to handle long running monitoring processes on small sets of instruments that update data at very low rates. For those cases, a less powerful machine provides lower power consumption.*

machine will not be monitored as it does not accurately represent a single or multiple instrument machine. Each test and their results are presented in the following subsections.

### 3.3.1 Number of instruments

The test intends to study the server machine response to an increasing set of instruments, it can also be seen as a scalability test. For the LP machine, each *instrument* produces a single source 10 channel file and updates data every second. For the HP machine, another single source, with 100 channels, is considered. Since all data sources are treated separately, regardless of the instrument they belong to, this test also evaluates the behavior of a server to an increasing set of data sources on a single instrument. To emulate an instrument already containing data, each instrument data container is pre-filled with entries of artificially generated data and is set up to collect new data every 10 seconds. For each set of instruments, 30 minutes of continuous monitoring is preformed, during which relevant machine parameters are recorded, such as CPU utilization, RAM, ROM and disk IO.

#### 3.3.1.1 Collection time

Figure 3.7 presents the collection time of instrument data as a function of the number of instruments being monitored. DAQBroker attempts to collect new data twice per data collection interval. Hence an instrument whose data is to be collected every 10 seconds should have its data collected every 5 seconds in practice. This figure shows that under a local database engine, there is a decrease in the average collection time of instrument data when more instruments are monitored, with that decrease being more pronounced on the LP machine. This behavior is counter-intuitive to the expected behavior of the application, although the variability of the collection time increases with the number of monitored instruments. This indicates that the numberof instruments up to 19 does not limit the collection time in local DB.

Under a remote database engine there is no discernible trend as a function of monitored instruments for the LP machine, while in the HP machine there seems to be a gradual increase of the collection time. This behavior is more in line with the expected behavior of the application but may be influenced by the underlying cloud service and the limitations of the database engine and storage tier used.

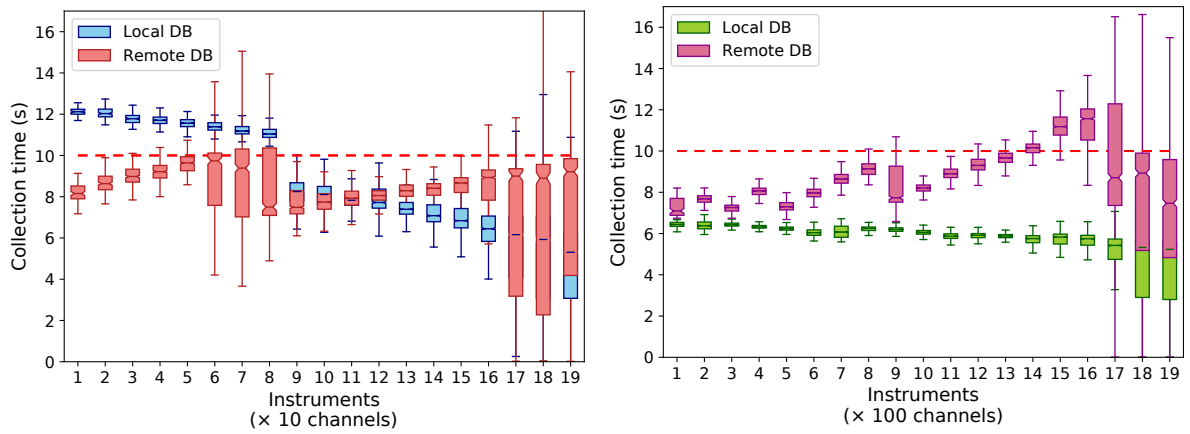



*Figure 3.7: Instrument collection time versus number of monitored instruments in a single DAQBroker server on the LP machine (left) and the HP machine (right).*

#### 3.3.1.2 CPU utilization

Figure 3.8 shows the CPU utilization of the underlying machine as a function of monitored instruments. Regardless of the number of instruments and database engine used, there seems to be no existing trend for CPU utilization, averaging around 25% for the LP machine and around 7% for the HP machine. This behavior is in line with the expected behavior of DAQBroker, as not particularly heavy computational efforts are requested for the application itself, most of the computation being spent in organizing lists resulted from database requests.

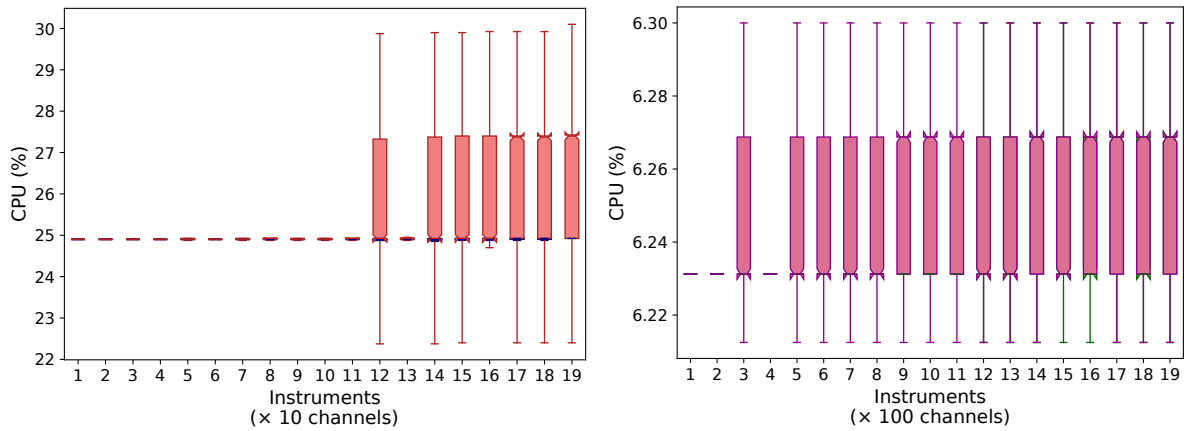



*Figure 3.8: CPU utilization versus number of monitored instruments in a single DAQBroker server on the LP machine (left) and the HP machine (right).*

#### 3.3.1.3 RAM

Figure 3.9 shows the RAM used by DAQBroker as a function of monitored instruments. There is a marginal upward trend with more monitored instruments however the used RAM tends to stay constant at around 1.25% for the HP machine and 18 and 19% for the LP machine with local and remote database engines, respectively. This behavior is in line with the expected behavior of DAQBroker since little amount of data is kept stored in buffered memory, most of it being sent directly to the database engine. While not enough time was spent with each set of instruments, a total of 10 hours of monitoring was made and clearly no meaningful amount of memory changed during that time, thus it is safe to assume that no visible memory leaks exist in the current version of this software.

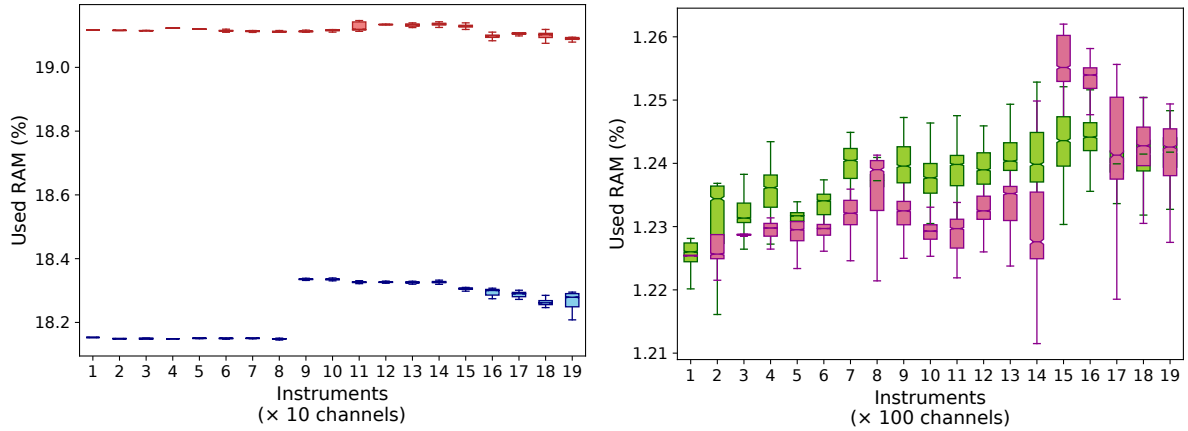



*Figure 3.9: Used RAM versus number of monitored instruments in a single DAQBroker server on the LP machine (left) and the HP machine (right).*

#### 3.3.1.4 Disk usage

Figure 3.10 shows the disk usage of as a function of monitored instruments. Regardless of machine and database engine, a clear increasing trend exists for both read and writes per second (RPS and WPS, respectively), with a local database producing more variability and overall larger values of both metrics, as expected for using local resources over remote ones. This trend seems to show that DAQBroker is IO-bound for both machines. This is not surprising, as the bulk of DAQBroker's operations are reading and writing to files or file-like resources. The actual values of the disk operations vary from machine to machine but the trend is present in both.

### 3.3.2 Rate of data generation

For DAQBroker to be able to monitor an instrument, it must be able to retrieve and store its information in the database in a timely fashion. Otherwise, new data will take progressively longer to arrive and real time monitoring will eventually be impossible to perform. In order to test the limits of DAQBroker when it comes to instrument data generation rates, the following test is proposed. A single instrument from a remote machine with a single file source and 10 channels is created with a variable data creation periods from 1 to $10^{-3}$ s. For each period a fixed

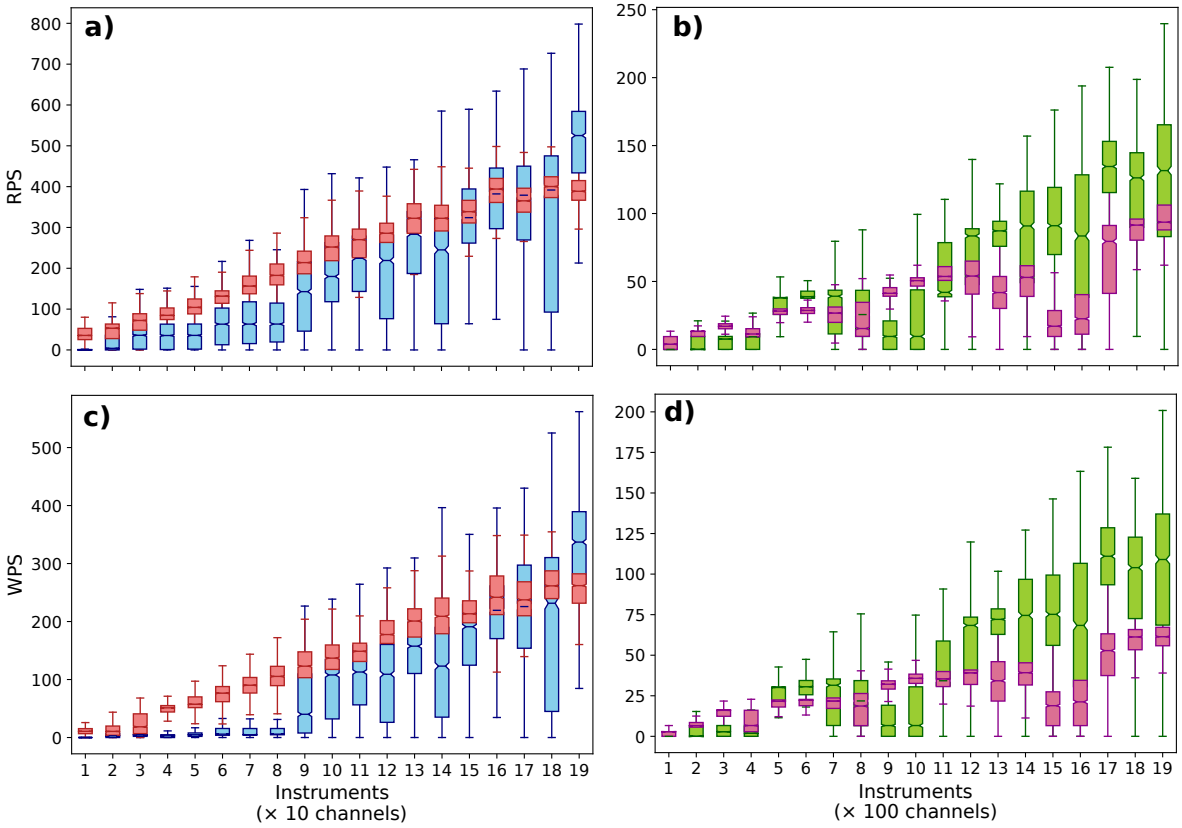



*Figure 3.10: Disk usage versus number of monitored instruments in a single DAQBroker server on the LP machine (left) and the HP machine (right). The top plots show writes per second and the bottom plots show reads per second.*

10 minutes of monitoring is preformed using DAQBroker. During this time a program running on the instrument's machine will simultaneously record every second the timestamp of the last record in the reference file and the timestamp of the last value stored in the DAQBroker database. While DAQBroker is able to store the instrument's data in a timely fashion, the difference between the aforementioned timestamps should always be smaller than the data collection period of that instrument.

Figure 3.11 shows the timestamp differences as a function of time for the different data generation rates. It can be seen that for a LP machine, a local database engine struggles to monitor instruments with a periods equal and lower than $10^{-2}$ seconds (100 Hz). This limitation is improved by the use of a remote database and data generation periods of $10^{-2}$ seconds can now be monitored. For a HP machine, however, a local database engine struggles to keep up with periods as low as $10^{-3}$ seconds (1000 Hz) while a remote database engine simply cannot handle the same data generation period.

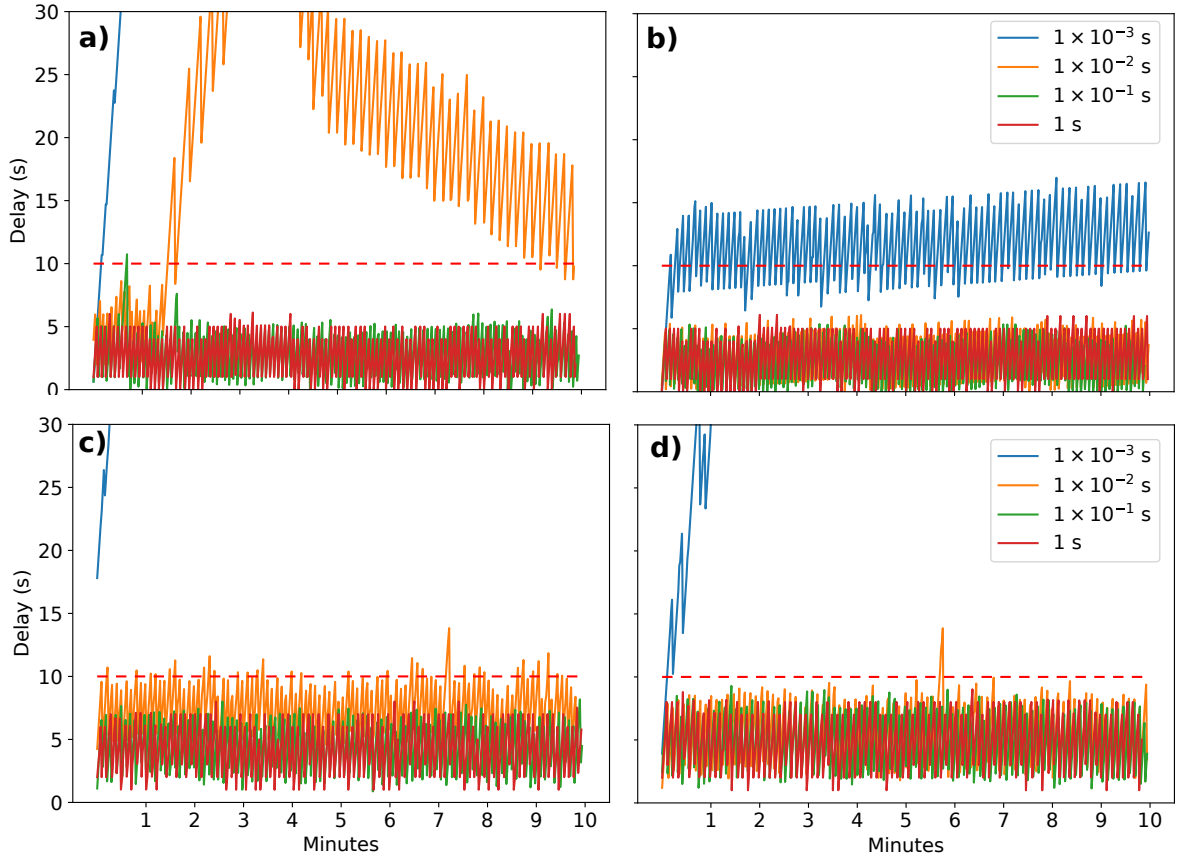



*Figure 3.11: Current to stored timestamp differences versus time for different machines and data generation periods. The left plots characterize the LP machine performance and the right plots the HP machine performance. The top plots refer to the use of a local database engine while the bottom plots refer to the use of a remote database engine.*

Figure 3.11 illustrates the need for carefully choosing the database engine position. A remote engine, while freeing local resources to handle more computation costs, trades more CPU performance for less network performance which can be at times more damaging for the monitoring of very fast data generating instruments.

It should also be noted that the data generation periods presented in these figures are quite fast to require real time monitoring. Frequently data generated at this speed will require some intermediate step of processing before producing data to monitor. However, DAQBroker can handle most of the presented speeds of data generation so that the data processing step can be made with DAQBroker itself (see section 4.3.2 for specifics on data processing with DAQBroker).

### 3.3.3 Data retrieval

Apart from instrument data injection, tests must also be preformed on the server's ability to handle data requests. These tests serve to validate the algorithms and queries used for serving data to users, specifically for online visualizations, which require timely showing of data. The algorithms used for online visualizations use the width of the user's screen to prevent too much data from being delivered to the user that requested it. Thus, no matter what the time length a user requests to visualize, only a maximum number of *(time,value)* pairs (up to the pixel width of the screen used) will be delivered, as this is the maximum resolution that can be seen in the used display.

The proposed test for data requests uses similar queries. A single instrument with 10 data channels is created and one million entries are introduced in it (a little over 10 days for an instrument sampling frequency of 1 Hz). Once done, a certain amount of data is requested (1, 5 and 10 days respectively, assuming the instrument produces data every second), that data is delivered considering a screen with 1200 pixels of width[3]. The request is made several times with the same length but different start times in order to produce a statistically significant set of request time values and not use the same and often cached values from previous requests. Once done, another million data entries are added to the instrument's table and the requests are again made. This process is repeated until 100 million entries are placed in the instrument (a little over 3 years for sampling at 1 Hz). These tests measure the degradation of the serving of data, as an instrument chooses from more and more data.

Figure 3.12 shows the request time versus number of entries in the instrument for both machines. While for the HP machine all requests seem to take a similar amount of time to complete, the LP machine requests take longer to complete, saturating at around 1 second at containers above 40M entries. While the LP requests take significantly longer to process than the HP machine, these requests are within a perfectly reasonable waiting time for remote monitoring and web technologies.

[3]This width lies in the range of HD, which is a low end when compared to already existing Full HD and 4K displays. However, for the amount of data points requested, the total number of points returned would be only a factor of 4 higher, at maximum.

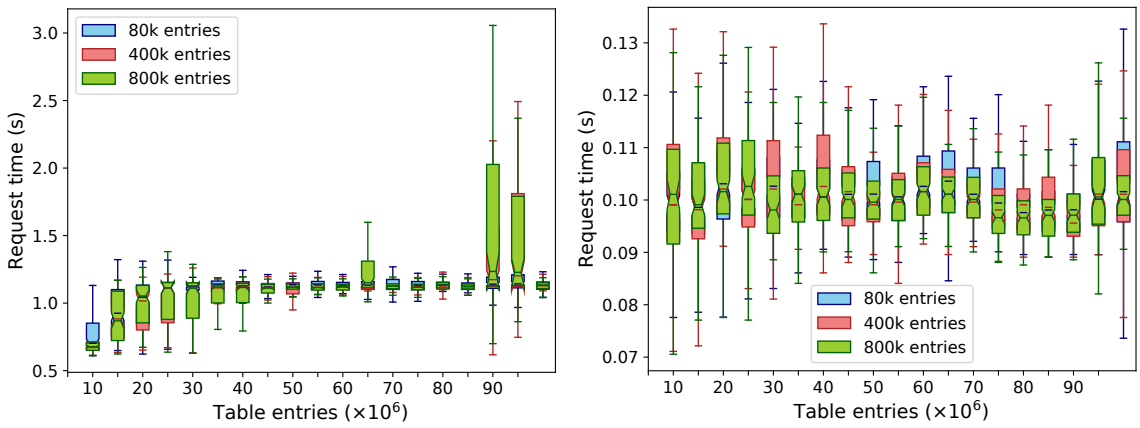



*Figure 3.12: Request time versus DAQBroker container size. The different colors depict different request sizes.The left plot characterizes the LP machine performance and the right plot the HP machine performance.*

# Chapter 4

# DAQBroker and the CLOUD Experiment

DAQBroker was first implemented on the CLOUD experiment. In this chapter the CLOUD experiment is presented, along with the data acquisition challenges that lead to the conceptualization of DAQBroker. First, a broad description of the experiment is made discussing the objectives, types of measurement and performed experiments. Second, the implementation of DAQBroker in CLOUD is described, focusing on the existing challenges at CLOUD that range from instrument data collection to data visualization and manipulation and how DAQBroker tackles those challenges. Finally, some examples of experiment results are introduced, highlighting the benefits of using this framework for online and offline data analysis.

## 4.1 The CLOUD Experiment

The CERN CLOUD experiment is an ideal example of one situation where instrument data monitoring, manipulation and storing approaches such as DAQBroker are required.

### 4.1.1 Description

The CLOUD experiment[329] consists of a collaboration of over 20 institutes around the world. The aim of this experiment is to study different processes in atmospheric particle creation and cloud generation in order to better understand and quantify the influence of the said processes. Since currently cloud processes contribute with the largest uncertainty to future climate predictions, the ultimate aim is to reduce the current uncertainty in climate forecasting models[330, 331]. The experiment set-up is shown in figure 4.1 and consists of a 27 $m^3$ stainless steel cylinder inside which a pristine atmosphere of $O_2$ and $N_2$ is created. Using high precision controls, a wide range of trace gases can be inserted with concentrations as low as 10 ppb [332] into the pristine atmosphere in order to kick-start the intended atmospheric processes. Also, different gases (Sulphuric acid (SA), $\alpha$-pinene and limonene)

can be injected into the chamber at controlled intervals in precise quantities to test the influence of individual and/or groups of gases in the different atmospheric processes[333].

Apart from precise control of chemical species, an accurate control of the temperature of the chamber is also expected. The left image of figure 4.1 shows the chamber's thermal control system. Around the chamber, surrounded by an insulated surface, an air flow with controlled temperature is constantly blown through the surface of the chamber walls, ensuring its interior is maintained at a pre-determined constant temperature [334].

The influence of cosmic rays in the development of atmospheric processes is also studied in the CLOUD experiment [329]. The CERN *P*roton *S*ynchrotron (PS) particle beam is used to bombard the chamber atmosphere with charged $\pi^+$ particles which ionize the particles inside the chamber, mimicking cosmic ray interaction with the atmosphere. A high voltage electric field can also be applied to produce an electric potential difference of up to 60 kV between the top and bottom of the chamber. This electrical field is used to remove charged particles from the chamber.
Two fans are also installed at the top and bottom ends of the chamber to ensure proper circulation and mixing of compounds. Many other quantities are also controlled, such as pressure and lighting as they can also influence the atmospheric processes [335, 336].

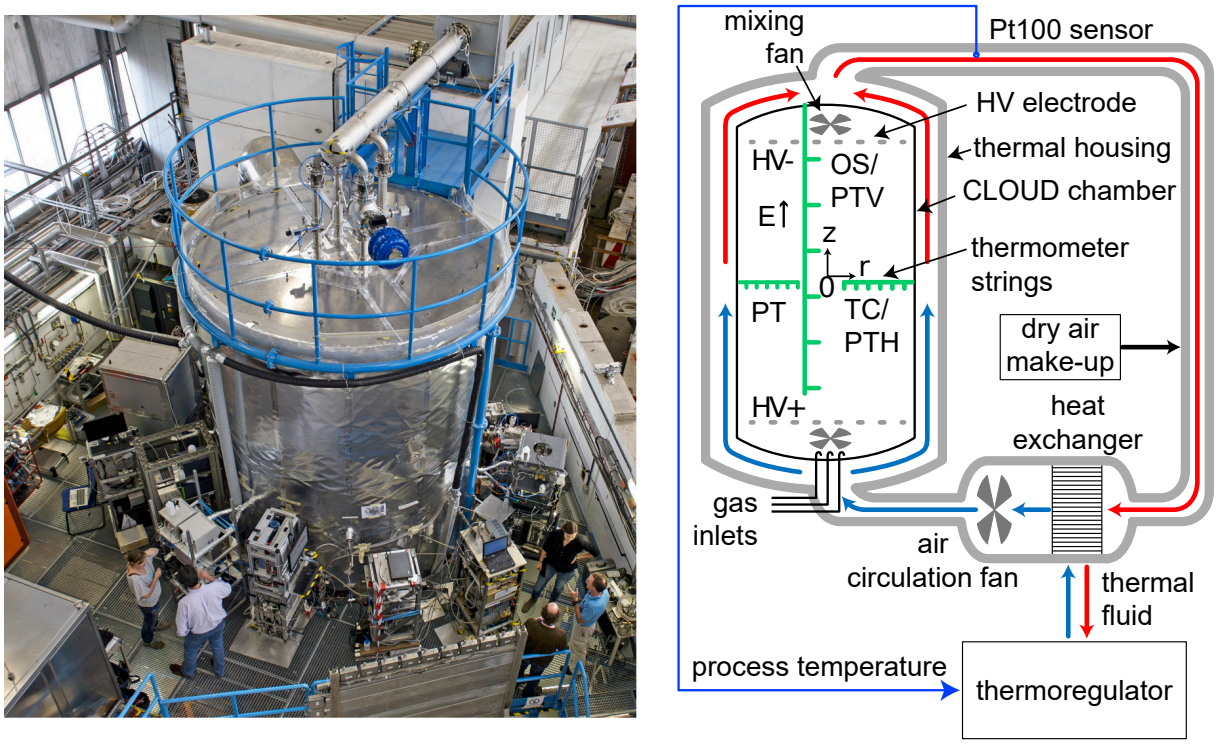



*Figure 4.1: (Left) Picture of the CLOUD chamber in operation at CERN. The large structure in the middle is the chamber itself and the measuring instruments are seen sampling at different angles around the chamber. (Right) Diagram of the thermal circulation control system with the positions of control mechanisms (gas inlets, circulation fans and high voltage cage), thermocouple (TC), optical (OS) and platinum resistance (PT) are placed at key points in the chamber to measure the temperature change over the whole surface*

### 4.1.2 Types of Experiments

The chamber operates in two main experimental modes. The first is particle formation mode, where trace gases are inserted at a controlled rate into the chamber to start particle formation processes while keeping the pressure in the chamber at 5 mbar above atmospheric pressure to prevent unwanted contaminants to enter the chamber through the instrument inlet ports. These experiments aim to precisely study the influence of a single or a set of chemical species on the rate of formation of new particles. The focus is on the generation of new particles from the successive clustering of smaller particles. These are called nucleation events and can be detected with the distribution particle counters as shown in figure 4.2. For these experiments, the different particle counter instrument data will need to be processed to produce an accurate value for the rate of change of particle generation at different sizes[337]. At the same time, data from the different mass spectrometers requires a heavy amount of processing to accurately calculate the different concentration of chemical species in the chamber.

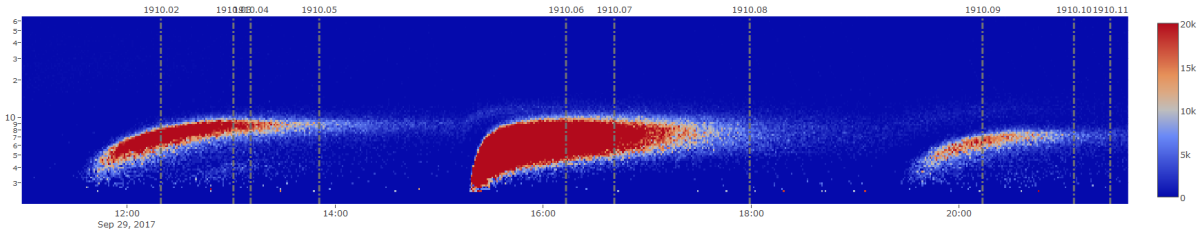



*Figure 4.2: Three distinct nucleation events produced inside the CLOUD chamber. This plot shows the size distribution of particles (particle size in y axis and number of particles in color) as a function of time. As time increases, smaller and smaller clusters gather to create larger and larger size of particles, depleting the lower sizes in the process.*

The second mode of operation is the cloud formation mode. The chamber is placed at high relative humidity and is pressurized up to 220 mbar with a distribution of particles inside. When equilibrium is reached, a controlled adiabatic expansion is preformed over a period ranging from 10 seconds to 10 minutes. This expansion cools the air and drives the relative humidity above 100%, thus allowing for the formation of clouds (liquid or ice, depending on the initial temperature)[338]. For these experiments fast readout of all instruments is required to monitor the experiment during cloud formation. It is often the case that a pre-existing particle distribution is injected into the chamber and the expansion allows the particles to grow by condensing supersaturated water onto their surfaces. The result of these abrupt expansions is an upward *shift* of the size distribution and a subsequent decrease in counts as smaller particles cluster together to form larger particles as shown in figure 4.3 which exhibits several of these events captured during one specific experimental run.

### 4.1.3 Types of Measurements

The CLOUD experiment, regardless of operation mode, requires a wide range of measurements to be performed, focusing on measuring the physical properties and chemical makeup of the atmosphere generated inside the chamber. Figure 4.4 shows a diagram of the different instruments to be used in the data campaign of 2017.

To create a more manageable description of the set of instruments, the instruments used in CLOUD are divided into three main types according to the measured quantities.

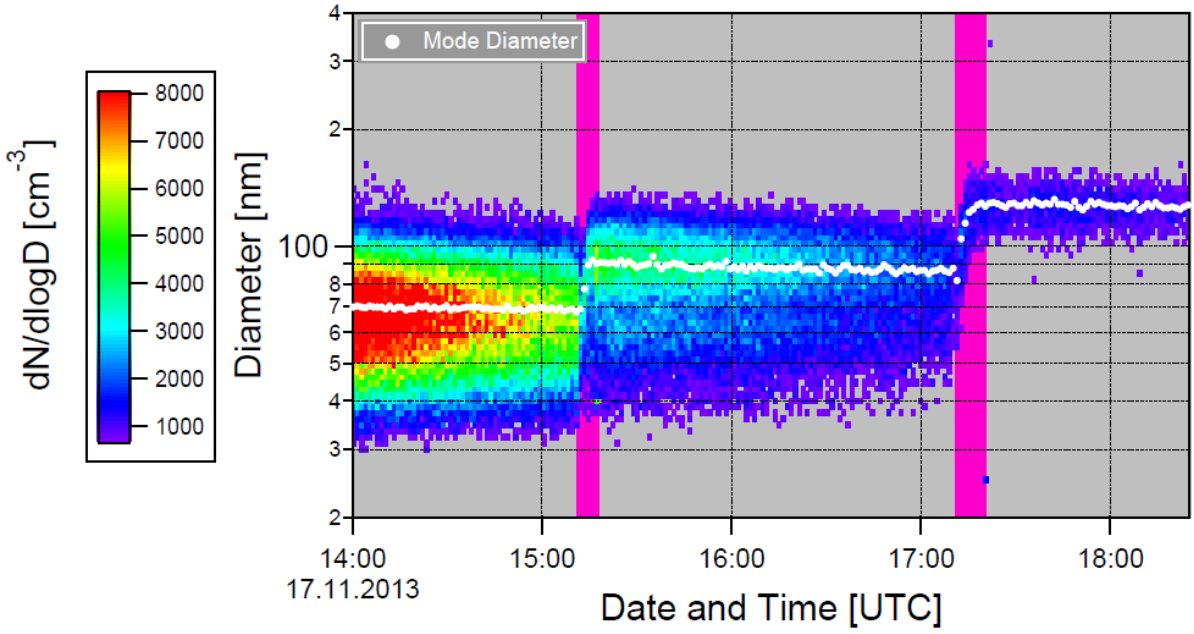



*Figure 4.3: Three distinct cloud formation events measured in the CLOUD chamber. The purple areas correspond to periods where an adiabatic expansion is performed. Note the different particle sizes resulting from the nucleation events in figure 4.2*

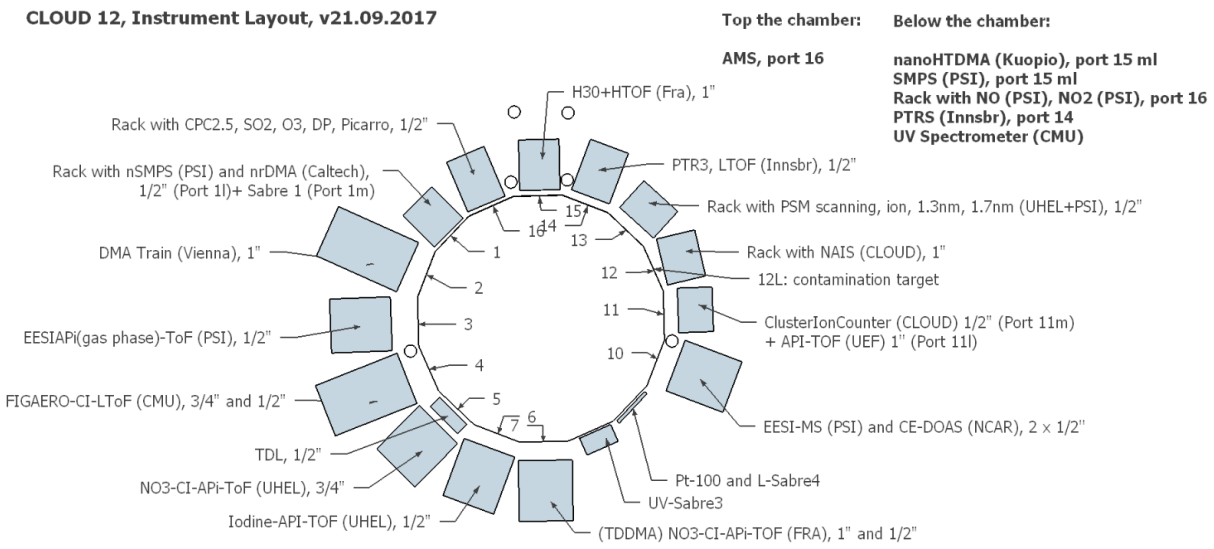



*Figure 4.4: Instrument layout for the 2017 data taking campaign. Apart from the central layer of instruments a smaller set of instruments are also placed on the bottom of the chamber.*

#### 4.1.3.1 Instruments for Physical Measurements

The physical measurements focus on characterizing large scale behaviors in the CLOUD chamber, such as temperature, humidity and particle size distributions. Table 4.1 lists all instruments making physical measurements, along with their time granularity and total disk space occupied during the CLOUD 2017 campaign.

*Table 4.1: Physical measurements used in CLOUD 2017. Instruments that only have post-processing data have a labeled granularity of PP.*

| Instrument | Description | $\Delta$t (s) | Disk (bytes) |
|---|---|---|---|
| CIC | Ions and naturally charged particles Mobility diameter $<$0.9 nm | 20 | 5,4G |
| CPC3776 | Cluster particle counter 2.5nm size cutoff | 2 | 1,2G |
| crossflowCIMS | Quantitative measurement of neutral cluster chemical composition | 1 | 253G |
| Dew | Dew Point Mirror | 10 | 6,0M |
| DMAtrain | Size-distribution in the range 1.6-10 nm | 1 | 2,8G |
| EESI | Online Aerosol molecular composition | PP | 125G |
| GCR | Ambient cosmic ray measurements | PP | 146M |
| LightSpectrometer | Visible light spectrometer (185-816nm) | 10 | 8,6G |
| LongSMPS | Particle size distribution (20-500 nm) | 173 | 204M |
| NAIS_ions | Ion size distribution mobility diameter 0.75-45 nm | 150 | 29G |
| NAIS_particles | Naturally charged particles size distribution mobility diameter 0.75-45 nm | 150 | 1,2G |
| Nano-HTDMA | Hygroscopic growth of 10-25 nm particles at different relative humidities (RH) | PP | 1,2G |
| nanoSMPS | Particle size distribution (2-65 nm) | 55 | 2,1G |
| nRDMA | Measure the size distributions of sub-10 nm (1.5-8.5nm) particles | 1 | 7,6G |
| PSM12 | Particle Size Magnifier Fixed mode Low cutoff | 1 | 2,6G |
| PSM15 | Particle Size Magnifier Scan mode Neutral | 1 | 2,7G |
| PSM16 | Particle Size Magnifier Scan mode Total | 1 | 2,7G |
| PSM_PSI | Particle Size Magnifier | 1 | 691M |
| PSM_Vienna | Particle Size Magnifier Fixed mode High cutoff | 1 | 2,5G |
| TDL | Hygrometer 10-14000 ppm H2O | 1 | 375M |

#### 4.1.3.2 Instruments for Chemical Measurements

The chemical measurements characterize the chemical composition of the atmosphere inside the chamber. Table 4.2 lists all instruments performing chemical measurements, along with their time granularity and total disk space occupied during the CLOUD 2017 campaign.

*Table 4.2: Chemical measurements used in CLOUD 2017. Instruments that only have post-processing data have a labeled granularity of PP*

| Instrument | Description | $\Delta$t (s) | Disk (bytes) |
|---|---|---|---|
| APITOF_neg | Naturally charged ions in neutral mode | 1 | 14,1G |
| Bromide-CI-APi-TOF | Iodinated species: I2, IO, OIO, I2O4, HOI and potentially I2O3, I2O5 | 10 | 433G |
| CAPS | Optical absorption spectrometer for NO2 | 10 | 28M |
| CE-DOAS | Measuring NO2, glyoxal, methyl glyoxal, IO, I2, OIO, HONO | 3 | 180G |
| CLD780 | NO measurement with 3 ppt detection limit | 2 | 43M |
| CMU_FIGAERO | Measure unaccounted organics in the gas and particle phase | 1 | 1,6T |
| HONO | HONO measurements | 0.002 | 4,7G |
| HOxROx | Measuring HOx(=OH, HO2) and ROx(=RO, RO2) radicals | 1 | 130G |
| HTOF_UFRA | NH3, amines measurements | 1 | 59G |
| ioniAPITOF | Positive ion composition | PP | 27G |
| LTOF_UFRA | HOMs, Sulfuric acid,DMA, DMA-SA clusters | 1 | 389G |
| NO | NO monitor | 10 | 49M |
| PICARRO | Ammonia measurements | 3 | 1,7G |
| PTR3 | Proton transfer mass spectrometer, ideal for most VOCs and SVOCs | 5 | 348G |
| STOF | PTR Sport TOF | 1 | 91G |

#### 4.1.3.3 Instruments for Diagnostic Measurements

The diagnostics measurements consist in measurements of control mechanisms to ensure their set points are being followed. Inevitably there will be some mixing between diagnostic and physical/chemical measurements. Table 4.3 lists all instruments performing diagnostic measurements during the CLOUD 2017 campaign.

As can be seen for either type of measurement, all instruments generate data in a wide range of time periods and use different amounts of disk space, up to a total 3.7 TB of used disk space. While the amount of disk space generated is generally not a problem for modern systems, the wide range of granularity presents a challenge for global data collection and storage in a centralized repository. The use of DAQBroker in these circumstances simplifies the storage as each instrument contains its own data table and storage folder, minimizing the existence of sparse data matrices.

*Table 4.3: Diagnostic measurements used in CLOUD 2017.*

| Instrument | Description | $\Delta$t (s) | Disk (bytes) |
|---|---|---|---|
| Fan | Mixing fans | 10 | 5,1G |
| Gas | Trace gas input control | 300 | 1,6G |
| GBox | Electrical field grounding box | 10 | 22M |
| HVFC | Field cage ion screening | 10 | 57M |
| Laser | UV laser | 10 | 932K |
| O3 | Ozone measurement | 1 | 36M |
| PhotoDiode12 | UV photodiode | 1 | 276M |
| PT100 | Pt100 temperature sensor string | 3 | 614M |
| Sabre | Control and monitor of UV linear light sources | 1 | 276M |
| Scalers | Hodoscope beam intensity measurement | 1 | 1,2G |
| SO2 | SO2 measurement | 10 | 30M |
| UVLamp | Controllable UV lamps | 10 | 54M |

## 4.2 DAQBroker Application at CLOUD

DAQBroker was a tool initially designed for the CLOUD experiment. Several of its mechanisms, mostly graphical interfaces, were thought out for this particular experiment. It was only later in the development that the aim of making DAQBroker a universal monitoring tool was considered. The use of DAQBroker in the CLOUD experiment brought specific challenges. Several instruments have highlighted limitations both in the back and front end of DAQBroker. These limitations could be identified and improved online and data collection was merely interrupted for a few hours. The different tools of DAQBroker applied to the CLOUD experiment are presented in this section along with the experimental limitations found and how they were tackled. The first part is on the creating/edition of instruments as well as the collection and storage of their data, accessing/creating/editing data visualizations, the different visualization needs for CLOUD and how these were met. Afterwards the focus is on the monitoring of networked machines, which allows users to remotely monitor the health of their machines via DAQBroker's agent. Finally the experimental run log considerations are discussed with methods for users to start/edit/end experimental runs as well as to highlight time intervals during a run where explanation of unexpected/problem events occurred.

### 4.2.1 Instrument Containers

Since the CLOUD experiment consists of a large set of instruments that come from different backgrounds and/or institutes, it is unreasonable to expect that for all instruments the data handling is similar. In fact, most instruments store the data in their own unique way implying that before DAQBroker, the system administrator constantly had to generate a rule for each new instrument and its underlying data generation.

As mentioned in 3.1.3, DAQBroker provides users with a tab for managing existing instruments. Each user can access this tab using any available web browser and consequently any device capable of running a web browser. The tab allows users to create a new instrument container by providing all the relevant information about the instrument via a data form. In this form users provide their instrument with a unique name, contact info and most importantly all relevant information about the instrument's data generation. Each instrument can have several data generation

rules which can come from different origins (data files and serial or network ports). In CLOUD most data comes from data files while only a small subset of permanent instruments provide their data directly via network or serial ports.

While a user fills the form, an example file can be supplied to test on the fly how subsequent data will be handled. The instrument creation procedure is one important improvement of DAQBroker over the otherwise less scalable method of requiring the system administrator to create a specific back end rule every time a new instrument appears in CLOUD or if data from an existing instrument changes.

Since the first version of DAQBroker was produced for the CLOUD 2017 data campaign, testing of the interface was limited. This resulted in difficulties found by users when generating instrument containers. One specific case of blatant limitation was highlighted for instruments that produce very large headers, spanning several lines of the file. Since the first versions of DAQBroker only considered headers up to 20 lines, the process of analyzing an example file on the user's browser, rendered the interface incapable of generating the instrument creation request. Another limitation was the initial low optimization when generating data channels: one particular instrument of this campaign generated over 2000 individual data channels and the user's computer would considerably slow down when attempting to create the instrument. Both these limitations were easily patched by accessing the specific classes in the DAQBroker interface code and in the case of the header limitation, introducing a dynamic parameter for header size and in the case of low optimization of channel display, to optimize the column display loop.

Regarding the back end behavior of DAQBroker's main instrument data collection loop, no particular issues were raised. Once an instrument request was properly sent, the data would begin to be collected automatically with very few exceptions, which were due to bad network infrastructure or limitations of the agent program not covered here (see 4.2.3). On the storage side, one particular instrument evidenced storage limitation that did not induce breaking behavior. This was the Gas control monitoring system, with over 400 individual data channels of data, connected to one single data source. That data source consisted of daily data files which updated each channel every 5 minutes. Also, whenever a single parameter is altered, a single entry for the change of that parameter is added to these data files. While this feature was promptly handled by DAQBroker the relational nature of the instrument container means a new line containing null values for all unchanged channels is added to the instrument's table. This resulted in a largely sparse data table which, while not competing with other instruments for disk space, was still a large amount of data for the granularity of the instrument. This limitation was not resolved as it is a result of the choices of data storage taken for DAQBroker, but it is important to highlight it as an inherent limitation of this framework for these types of exotic data file structure.

### 4.2.2 Data Visualizations

The data visualization component of DAQBroker also constitutes an improvement over the existing online CLOUD tools. This feature allows individual users to create dynamic and interactive visualizations of instrument data while in the past it was only the system administrator that could provide the necessary visualizations to users as well as

to edit the existing ones. This was because plot configuration data was maintained at a server level. DAQBroker provides a tool that allows users to group individual data channels and provide the information to create a visualization of the said channels.

In the context of CLOUD, three different graphical visualizations are required and are included in DAQBroker. The three types of visualization are exemplified in figure 4.5:

- **Time series charts** (figure 4.5a): the most popular charts used in CLOUD. These charts allow the visualization of one or several time series simultaneously providing users with valuable information on the behavior of the experiment. These charts are an improvement over existing ones not only for the fact that users can now create these charts but also by providing interactivity. Users can actively change the display of the chart, colors of the series, limits and scale of the y axis or show each series in their own individual y axis. Users can also access the data in the series and highlight individual timestamps of a series, store persistent comments on a time series chart as well as save the current visualization in a static format for later use in internal documents or publications.
- **Time-evolving histogram charts** (figure 4.5b): the most visually interesting charts to be created as they provide a heatmap of the behavior of several channels. While these charts lack the interactivity provided by the time series charts, they still provide vector graphics plots with access to the data generated of a distribution given by a set of individual data channels and how this distribution evolves in time. These plots were used to show how different size distributions evolved in time during experiments.
- **Bar charts** (figure 4.5c): being the least popular, these charts provide information comparing different sets of channels over a specific time period. Users are also able to decide if the bar charts were cumulative or averaged over the time. These charts were used, for example, to provide the vertical and horizontal distribution of the CERN PS beam at different times in the experiments, and monitor that the beam was centered.

All charts are rendered in the front end, changing the visualization from server side to the client side. The reason for this choice is the improvement of server performance, since resources provided by the CLOUD server infrastructure are better suited for collecting and processing instrument data rather than producing visualizations for users. The decentralization of these features also allows the front-end applications to provide users with a more interactive and persistent experience even at situations where connection to the server is not immediately guaranteed[339].

For each chart a list of data channels required for the generation of the chart is produced and a request for the data of each channel is sent to the DAQBroker API[1]. Once the data from all channels is retrieved the chart is then rendered. Data for a single time series is collected as it is stored, at this stage, no interpolation is performed to the data and the user is provided with the raw data collected from the instrument. There is a performance optimization choice sent for each chart request which provides the server with the number of horizontal pixels the chart is to occupy on the user's screen. This instructs the database query to provide the request with at most the number of

[1] see `https://www.daqbroker.com/documentation/webapi2.html#post--data-getData` for details on the request.

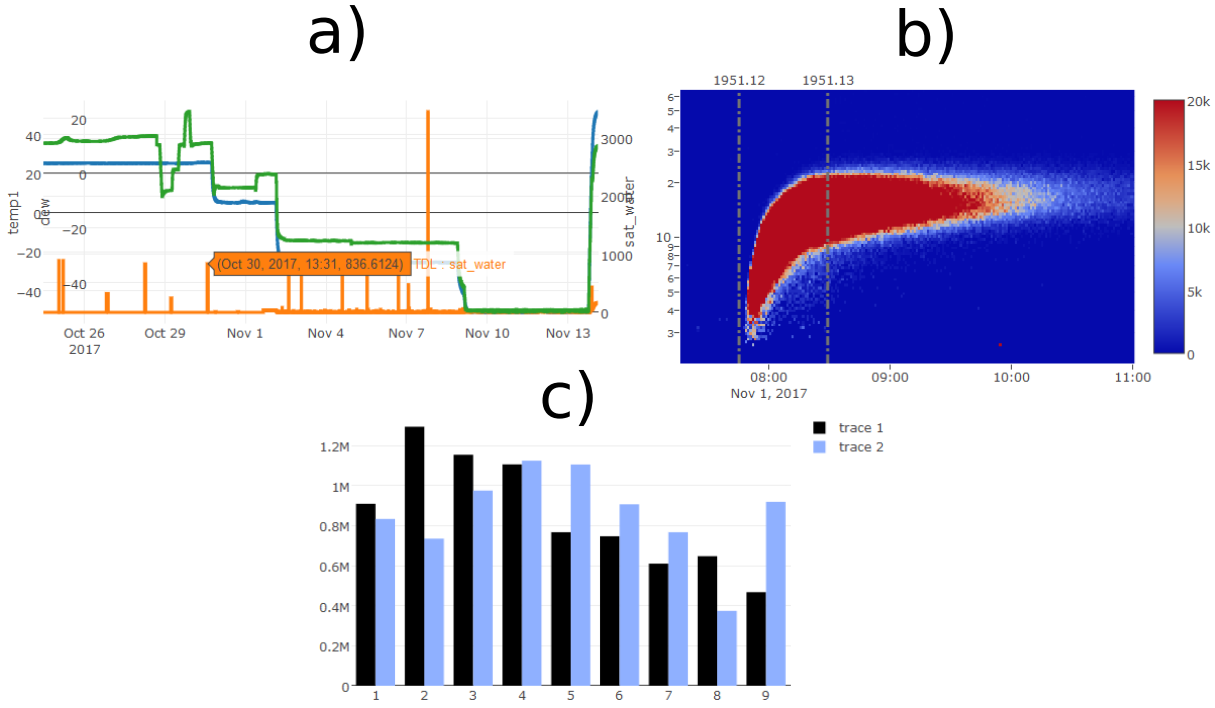



*Figure 4.5: Example of the different charts available for creation in DAQBroker. The time series chart (a) was the most popular, closely followed by the time-evolving histogram chart (b) and finally the least deployed was the bar chart (c).*

data points equal to the number of pixels in the chart, evenly spaced in time, since the screen used by the user will not be able to render more points in the plot. The user has the opportunity to zoom-in to a smaller time interval to get the data with full resolution. There is also an optional data smoothing option on the front end side with the application of a moving average filter. DAQBroker as a rule aims to provide the user with the data retrieved from each instrument as it was retrieved, allowing the user to decide whether or not this data requires processing.

The charting tool was the most used DAQBroker tool during the 2017 campaign. This experience enabled the pinpointing of several limitations, mostly due to limited testing, and were promptly patched and fixed. One interesting limitation was the server optimization choice for providing only a number of data points at most equal to the number of horizontal pixels of the plot. For the aforementioned Gas monitoring system, when data was requested from this instrument, the algorithm to gather data points equally spaced in time failed to provide any data points or miss key data points, due to the sparse data table that was created. This limitation was fixed by providing the charting tool users with a new chart option, allowing the request of the full data in the time range accompanied by the necessary warning that this would impact performance.

### 4.2.3 Network monitoring

This tool provides users with a view of the network around the connected DAQBroker server and of each individual node that is using or has used the DAQBroker network agent to connect their machine to the central DAQBroker server application. It allows different instrument operators to remotely monitor their machines health as long as the DAQBroker network agent is properly running on their machine and at least one connection is used to the server

machine.

Another feature of the network tool is that it provides users with a set of pre-compiled agent software for a variety of operating systems and CPU architectures. This was the mostly used feature of this tool as users that required to set-up a remote machine needed to access this tool to download the network agent.

The network agent tool was widely used in the CLOUD campaign of 2017 since most instruments in CLOUD arrive with their own machine that needed to be connected to the network and afterwards all data needs to be extracted from their own machine onto the central server. Of note an interesting limitation of the network agent was discovered when users tried to run the agent in a folder with no permissions. The network agent required to write files in the folder and, not being able to do it, no new data was synchronized onto the server machine and no new data was allowed to be transferred. This breaking limitation was promptly patched and was not identified sooner due to the lack of logging information for that agent.

### 4.2.4 Experimental log

Another widely used feature of DAQBroker in the CLOUD experiment was the experimental log tool. This tool allowed users to separate experimental runs in a hierarchical system. The particular case of CLOUD separated experiments into runs and each run into stages where a run contains a major change in experimental parameters while a stage only contains a single or few experimental parameters changes.

Since DAQBroker was developed with the CLOUD experiment in mind, the experimental log tool provides users with the run hierarchy followed in the CLOUD experiment: An experimental run or stage has two possible states, active and inactive. An active run is one that is currently under way and is highlighted above all other runs in the interface, while an inactive run joins the set of previously performed runs. The run list can be downloaded in different formats for use in other users' data processing or for generating internal documents.

Another requirement of the CLOUD experiment was to build a table of relevant experimental parameters provided by users. This table was implemented in DAQBroker with the ability for a properly authenticated user to change the table adding several different parameters. Each experimental parameter can be provided in three ways:

- **Text**: a free-form input that allows the user to provide textual representation of an changeable experimental parameter,
- **Choice**: a choice input that allows the user to select from a predetermined set of inputs,
- **Number**: a numerical input that allows the user to set the values of an experimental parameter (requires the user to provide the units of this parameter).

Apart from the experimental log, and the parameter table, users are also able to provide comments at specific timestamps in runs. These comments can be placed in active or inactive runs and are used to emphasize specific

events that occurred during the run, whether expected or not.

Some limitations were found for the tool, during the campaign, for example, while the REST API of DAQBroker was able to easily handle requests once properly formulated, the user interface failed when special characters were used creating runs or placing comments. These limitations were promptly patched and resolved at several stages in the campaign.

### 4.2.5 Overall analysis

This illustrates how use of DAQBroker in the CLOUD campaign was able to overcome the previous lack of thorough testing. While most issues were found in the user interface side of the framework, user experience is a key factor in determining the use and popularity of a framework. It is clear that overall testing is required to be implemented in DAQBroker at all stages of development in order to provide a properly functioning framework that can easily accommodate new modules and tools for other users.

## 4.3 Studies using DAQBroker

While the DAQBroker tests presented in 3.3 provide a good depiction of the performance of DAQBroker with varying power, it does not refer the features added for actual scientific instruments. This section will provide three separate examples of the use of DAQBroker in the context of the CERN CLOUD experiment. While it is inevitable that online data from an experimental campaign will be replaced by processed data, during the data taking effort, for the CLOUD experiment in particular, accurate values of the measured parameters are required for users to be able to follow experimental protocols and promptly decide on the next measurements that are interesting or more relevant to achieve the optimal physical meaning.

### 4.3.1 Temperature stability study

To study the temperature stability of the CLOUD chamber, a large set of temperature sensors was installed in the chamber to measure the radial and axial temperature profiles of the chamber[2]. A total of 16 sensors was installed, grouped in three rods (two radial ones and one to measure the axial profile).

The array of sensors was used during two CLOUD campaigns. The first was dedicated to the *in-situ* calibration of each sensor, during which a set of previously laboratory calibrated temperature sensors was placed in the chamber next to the sensors to calibrate, doubling the amount of sensor data. During this campaign, very specific measurements were made to provide statistically relevant data sets for calibration at a relevant range of operating temperatures and to study the stability of the chamber in a static state and without any other instruments present.

[2]This subsection contains a very abbreviated description of the work in [334]

The second campaign was during the data taking effort of the fall of 2014. During this time several experiments were preformed in CLOUD, both particle formation and cloud formation experiments, allowing to characterize the full range of CLOUD temperatures and experimental methods as well as to compare the static chamber, with no instruments connected, with the active chamber containing several instruments, which requires a continuous active refresh of the air inside the chamber.

By considering each rod of sensors a separate instrument, each sensor can be defined in the DAQBroker CLOUD environment as a data channel of their rod. This allows users to easily identify each sensor and its relative position in CLOUD, creating views that are related only to the radial, vertical position, or both. Figure 4.6 shows an example of such visualizations: a time series of the axial and vertical temperature profiles during a cloud formation event.

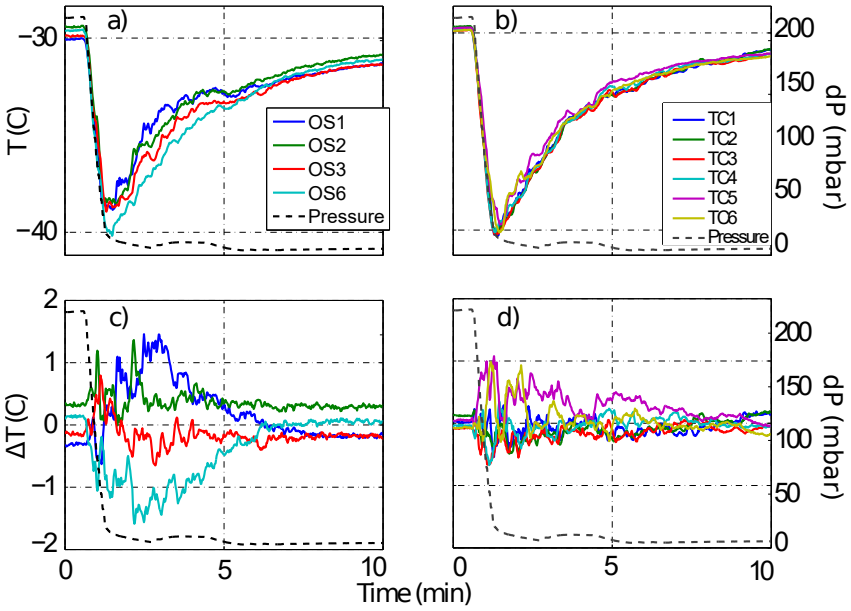



*Figure 4.6: Time series visualization of the vertical (left) and axial (right) temperature profiles during a cloud formation experiment (adiabatic expansion) in CLOUD. The top plots show the absolute temperature measured by the sensors and the bottom plots show the difference between each sensor value to the average rod sensor value at each time stamp.*

Apart from easy access and visualization of sensor data, DAQBroker's API allowed the creation of user developed scripts that would facilitate access to large amounts of sensor data, which proved particularly useful during the calibration stage of each sensor and in the selection of individual cloud formation events such as the one shown in figure 4.6.

### 4.3.2 Concentration estimation

DAQBroker's features allow for a channel's data to be manipulated. In an experimental setting such as the CLOUD experiment, often an instrument's data will require some degree of manipulation to provide meaningful data and their interpretation. Quite often the aforementioned data manipulation changes during the experiment. In the CLOUD experiment the manipulation process is particularly important when attempting to provide *online* values for compound concentrations obtained from mass spectrometry. More specifically, on the field of **C**hemical

**I**onization **M**ass **S**pectrometry (CIMS).

CIMS methods consists in the creation of ions by way of chemical reaction of a collected sample with a known substance[340]. Since this substance is known to react with specific compounds these methods allow high selectivity of the compounds to study and are also more sensitive to smaller concentrations of the said compounds[341]. In the specific case of CLOUD, a large amount of substances are used and the chamber contains different chemical behaviors at different experimental conditions (temperature, relative humidity, UV intensity, etc...), scenarios impossible to prepare in advance. The users of these instruments are thus required to make calculations on-the-fly to provide correction values and improved calculations of concentrations that must be updated centrally for other users have access.

To accommodate these types of manipulations, DAQBroker features two different ways to compute derived values from the channel data:

- **Visualization manipulation** - Users can add computed values time series to existing visualizations. The data from these manipulations is not stored but rather processed at each request from the available instrument data. This allows users to alter the computed expressions a later time if they so desire. It is ideal for representing data that is not entirely understood and require constant changing in the underlying manipulation. It is only used for viewing purposes, not enabling other users to access the data *offline* for processing needs.
- **Custom channels** - Special type of data channels that can be *programmed* according to the data in one or several channels. These channels can be created on any instrument and by any user controlling the instrument. The channel data is stored with a time granularity equal to the smallest granularity of the channels used for the custom data. These channels also offer a level of non-commitment since if the inherent manipulation of the channel is altered, the user can choose to keep the previously stored data on file.

### 4.3.3 Nucleation rate calculation

One of the most relevant parameters determined in the CLOUD experiment is the rate at which new particles of a given diameter are created inside the chamber via physical and/or chemical processes. This process is named nucleation and by knowing the rate at which nucleation occurs when specific trace gases are added to the CLOUD chamber, more knowledge is available for global environment models to estimate future and near future climate scenarios.

Several studies on nucleation rate calculation have been preformed on CLOUD and a general empirical equation is used for the experimental calculation of nucleation rates at a given particle size ($J_x$), which relies on the data collected from multiple instruments[342]:

$$J_x = \frac{dN_x}{dt}\left[k_{coag}(T, x, \bar{D})N_x^2 + k_{dill} * N_x + k_{wall}(\bar{D}) * N_x\right] \tag{4.1}$$

Where $k_{coag}$, $k_{dill}$, $k_{wall}$ are respectively coagulation, dilution and wall loss coefficients, $T$ is the temperature of the chamber, $x$ is the size of the particles that are nucleating and $\bar{D}$ is the mean size of the particle distribution function. As indicated in equation 4.1, the loss coefficients are a function of several experimental parameters. $T$ can be obtained from any of the temperature sensors discussed in 4.3.1, $x$ can be obtained from any particle counter instrument (Condensation Particle Counter, Particle Size Magnifier, etc...)[343, 344] and $\bar{D}$ can be obtained from a relevant particle distribution measurement (usually a Scanning Mobility Particle Sizer)[345]. As for the calculation of the concentrations of different chemical compounds, DAQBroker's data manipulation features allow for an easy integration of a new custom channel with equation 4.1 calculated via the aforementioned experimental parameters $x$, $\bar{D}$ and $T$.

Figure 4.7 shows processed nucleation rates calculated using 4.1 against rates calculated using a simple derivative method during a 12 hour nucleation event taken in the fall of 2017. The calculation of the different coefficients in equation 4.1 require a particle distribution measurement which only provides values every 5 minutes. The online calculation was made with only a median filter being applied to the raw data and derivative values of the collected concentration, which are implemented in DAQBroker. The ability to process online data to such close agreement with processed data is a clear advantage for the use of DAQBroker in projects that require close supervision of data and fast response to experimental data.

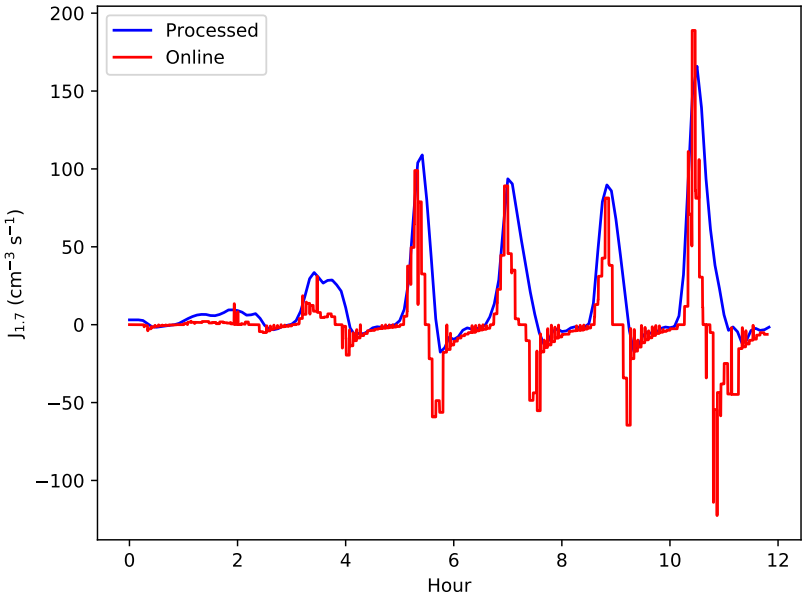



*Figure 4.7: Comparison between processed (blue)and online (red) nucleation rates.*

# Chapter 5

# Time Series Analysis with DAQBroker

In the previous chapter, the performance of DAQBroker was discussed within the framework of studies and calculation of scientific parameters. While these use cases are relevant for other experiments, they only show a limited use of the framework. Since the goal of DAQBroker is to provide general purpose monitoring of instruments, tools for statistical analysis of the monitored data should be included. These tools can provide insights into instruments' general status and of the experiment as a whole and provide users with important information apart from simple visual examination. To address this need, the purpose of this chapter is to demonstrate a series of useful methodologies ready to be implemented in DAQBroker to provide users with extended information on the behavior of an instrument's data. The corresponding software was built and tested using CLOUD experiment results.

This chapter begins by introducing selected tools based on the statistical methods described in chapter 3 and applying them to the analysis of individual and/or pairs of time series to collect meaningful information. The analysis of the single time series allows discovering statistically relevant information from a specific data such as event detection or detect similarities with other series. Later in the chapter, a larger scale analysis is introduced that aims at producing an essentially automated statistical summary of a specific data set using a large amount of data channels from a DAQBroker database. This analysis takes advantage of DAQBroker to store several data channels from a variety of instruments, while simultaneously monitoring them, to present users with an automated review of the relationships, most valuable channels (statistically) and periods during their activities.

## 5.1 Single Time Series Analysis for DAQBroker

This section describes methodologies for handling individual time series from DAQBroker. It begins by introducing a way of classifying different time series segments in terms of their stationarity. It provides methods for comparing time series in terms of their similarity and concludes by introducing a procedure for automated event detection in single time series. As a proof-of-concept, all discussed time series analysis methods are tested on data from the 2017 fall CLOUD campaign. The data used from the campaign is a specific subset of signals represent-

ing several relevant physical and chemical parameters of the experiment and the time range is days (not months) providing the reader with the most detailed data as possible, emulating available data from an on-line experiment, during which the user is interested in reviewing data from previous runs with a duration of at most a couple of days.

The specific dataset chosen to test the concepts of time series analysis consist in the following channels:

- UV intensity read out from an Hamamatsu S2281 photo-diode [346]
- Temperature measured by a Pt100 temperature sensor [334]
- Sulphuric acid concentration as measured by an LTOF mass spectrometer [347]
- Particle concentration for a fixed size PSM [343]
- $\alpha-$pinene concentration as measured by a Proton Mass Transfer (PTR) mass spectrometer [348]

A random time frame of two days is chosen between the 15th of October and the 17th of October, which lies in the middle period of the data campaign of 2017.

### 5.1.1 Time Series Classification

As discussed in 2.2.1.1, time series can be classified in terms of several properties. This section intends to gather those properties into a single concept called *activity*. It begins by introducing a method for classifying a time series, followed by proof-of-concept tests preformed on CLOUD data. It concludes with the discussion of the implementation in DAQBroker in terms of necessary user interface tools and its limitations.

#### 5.1.1.1 Classification Procedure

Stationarity of a time series was defined in 2.2.1.1 as a special case where a certain number of its properties do not vary in time. It was indicated that a stationary time series tests negative for the following properties: Trend, periodicity and unit root. If one of these properties exists, the time series is considered non-stationary.

To analyze if a time series is stationary, a battery of statistical p-value tests is applied to test if at least one of the aforementioned properties is present. If none of the properties is present, the time series can be truly classified as stationary. The 4 p-value tests for each of the aforementioned time series properties are:

- Trend - Mann-Kendall test,
- Periodicity - Fisher test,
- Unit root - ADF test,
- Stationarity - KPSS test; this test is used to reinforce the stationarity hypothesis

After applying the aforementioned tests to a certain period of the time series, it is classified stationary if all tests point to a stationary time period. The scientific advantage of a statistically stationary time segment is that it represents a phenomenon that is not affected by uncontrolled parameters. The segment is representative for controlled parameters and can be easily handled for experimental purposes. While individually, in each of the statistical tests type I and II errors can occur, the fact that these four tests are applied at the same time reduces the chance of a time period being incorrectly labeled stationary due to one single test failure.

In this section, the values used for the p-tests will vary to highlight the need for user interactivity when using real data. For instance, a user can choose to ignore periodicity tests if said user is certain that no periodicities exist in the time series of the instrument and that any existing periodicities arise from noise sources.

#### 5.1.1.2 Classification of CLOUD Time Series

To each of the CLOUD time series introduced in this section, the method of classification is applied and the results are discussed case by case. While considering each individual case, it becomes apparent that the length of time segments is a major factor in the correct evaluation of stationary data segments.

**5.1.1.2.1 UV Intensity** Figure 5.1 shows the time series for measured UV intensity and the regions where stationarity is statistically significant. For this particular time series the size of the segments chosen was 20 minutes, and the limiting p-values of the statistical tests are kept at their default values. This particular time series is quite simple to evaluate as the measurement is extremely precise the electronic noise/signal is very small and only a few number of modifications exist. While by visual analysis it is clear which segments are stationary and which aren't, further confirmation is provided by the stationarity method. This analysis shows that for this particular time period, the UV intensity time series has intervals classified as stationary.

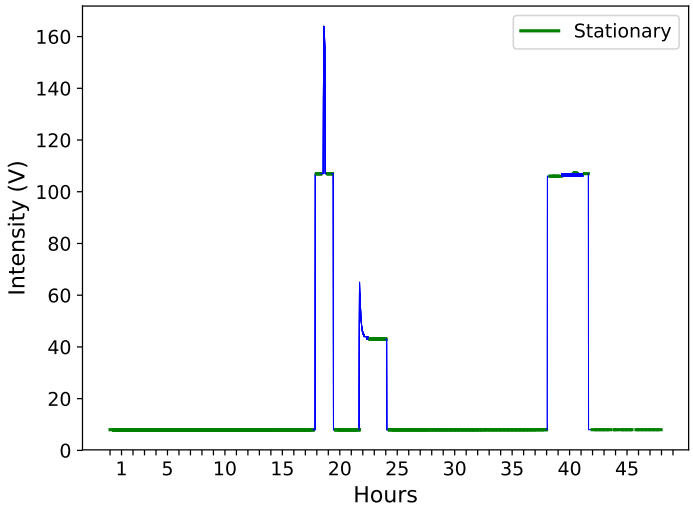



*Figure 5.1: UV intensity inside the CLOUD chamber (blue) and segments classified as stationary (green).*

**5.1.1.2.2 Temperature** Figure 5.2 shows the time series for temperature measured inside the CLOUD chamber and the segments deemed to be stationary. For this time series the size of the segments chosen was 5 minutes and the limiting p-values of the statistical tests were kept at the default value. While in this figure it seems simple visualize what the stationary periods are, the fact is that for larger segments, the statistical tests fail to find any stationary segment, which suggests that, for both high and low temperature periods, there are a significant amount of time segments that behave in a non-stationary way (convection, eddies or turbulence could be a factor). Other possible factors for non-stationarity could associated to the electronics used, as the National Instruments electronics often produce processed data giving false negatives and/or positives for the applied statistical tests, for very stable periods. This analysis shows that for this particular time period, the temperature time series has little to none intervals classified as stationary.

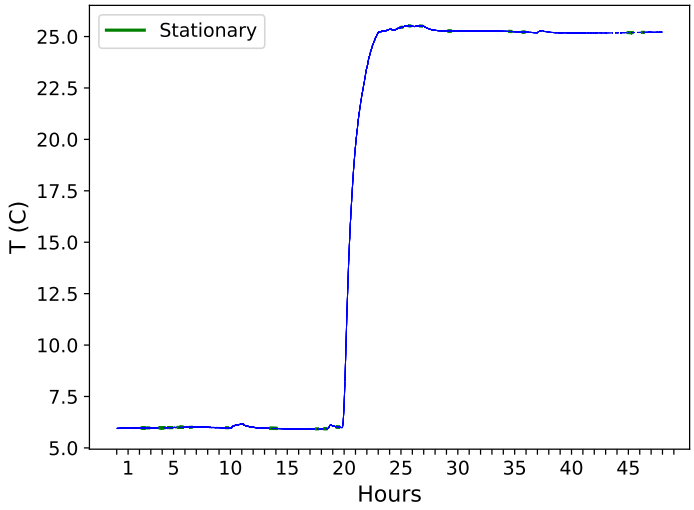



*Figure 5.2: Temperature inside the CLOUD chamber (blue) and segments classified as stationary (green).*

**5.1.1.2.3 Sulphuric Acid Concentration** Figure 5.3 shows the time series of sulphuric acid concentration inside the CLOUD chamber. For this time series the segment length was set to 1 hour and the limiting p-values for the statistical tests are kept at their default values. While being an extremely noisy time series, several stationary segments could be found even if segment sizes are significantly larger. This suggests that the segments classified as stationary are stable for periods of one hour or more. This behavior is expected for sulphuric acid concentration since the chamber is capable of producing very controlled amounts of this compound by way of photolysis using both a very stable source of $SO_2$ and the highly controlled source of UV light (figure 5.1). This analysis shows that for this particular time period, the sulphuric acid concentration time series has intervals classified as stationary.

**5.1.1.2.4 Particle Concentration** Figure 5.4 shows the concentration of small diameter particles ($d < 12$ nm) measured inside the CLOUD chamber. The segment size was chosen to be 10 minutes and the limiting p-values of the statistical tests are kept at the default value. The found stationary segments clearly show stable behavior for data at several orders of magnitude. While, in the timeseries, there are clear changes in both behavior and order of

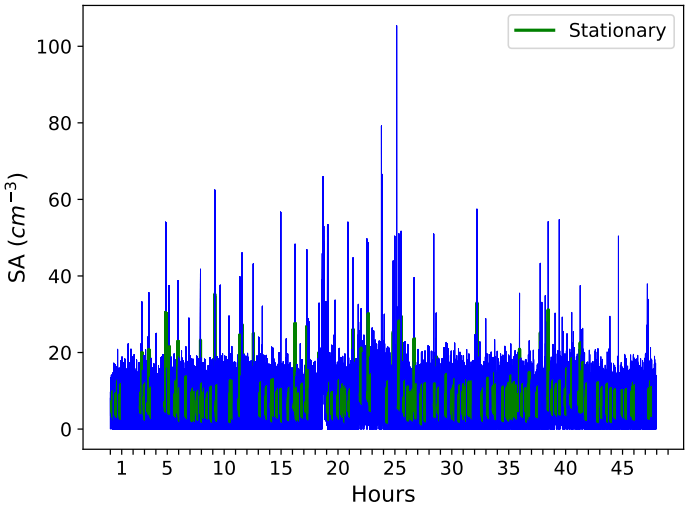



*Figure 5.3: Sulphuric acid concentration inside the CLOUD chamber (blue) and segments classified as stationary (green).*

magnitude (log y scale) due to several different experiments being performed during this 2 day period, it is clear that the stationary periods are those where the particle concentration seems stable in value. Particle concentration time series have intervals classified as stationary, and changes are usually associated to new particle formation experiments, during which particle concentration should attain a stable value for the experiment to continue to the next stage. The use of this method to either create automated control for new particle formation experiments or to signal users to change experiment mode is thus an interesting option to have on DAQBroker.

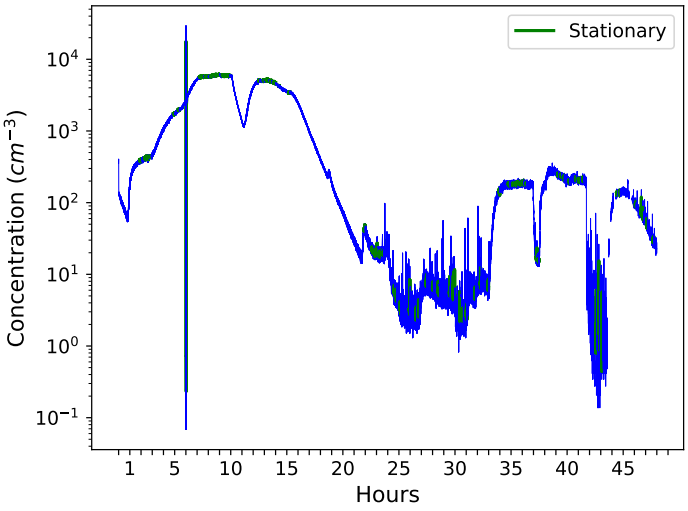



*Figure 5.4: Small particle concentration inside the CLOUD chamber (blue) and segments classified as stationary (green).*

**5.1.1.2.5 $\alpha$-pinene Concentration** Figure 5.5 shows the concentration of the $\alpha$-pinene compound inside the CLOUD chamber. The segment size was set to 10 minutes and the limiting p-values of the statistical tests were kept at their default values. Since this compound is directly related to particle creation, it is not surprising that there exists stationary segments very similar in time to those found in the particle concentration time series (figure 5.4). There are, however, fewer segments in this time series than in the former, hinting that either the measurement

or the control of the injection of the compound is not as stable as the particle creation and its measurement.

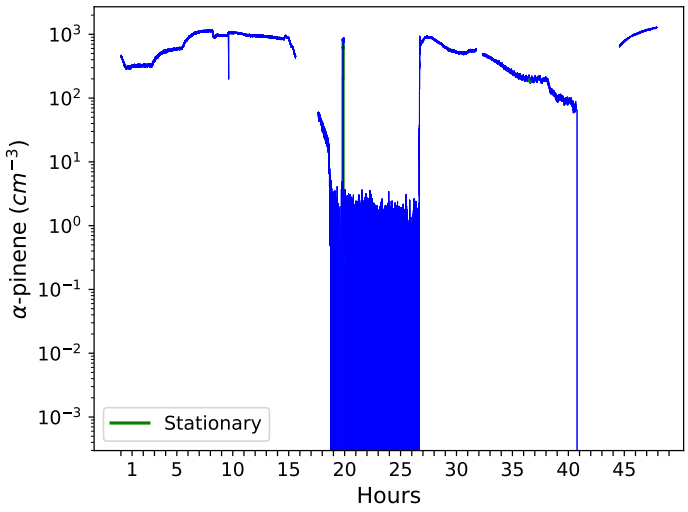



*Figure 5.5: α-pinene concentration inside the CLOUD chamber (blue) and segments classified as stationary (green).*

#### 5.1.1.3 Implementation in DAQBroker

In order for this method of classification to be implemented in DAQBroker, some constraints should be considered. First the user should be given the opportunity to filter the data in order to remove any possible outliers and improve the overall detection of stationary segments. Second the user should be able to provide the limiting factors for each individual test, or to completely suppress the evaluation of specific statistical tests, to account for particularly noisy data or overcome known sources of signal interference. Finally, a proper result of the statistical tests depends on retrieving a statistically significant amount of data points and thus in choosing the correct time interval to apply these tests. This interval varies from time series to time series (instrument to instrument) and on the phenomenon being measured. Thus the implementation of this method in DAQBroker requires strict limits on the amount of data to analyze not only to spare the server from large computational effort but most important to not provide false results due to insufficient data points. Figure 5.6 shows a suggested interface for time series classification along with an example visualization of the resulting analysis.

### 5.1.2 Time Series Comparison

As stated in chapter 2.2.1.2, the local similarity statistic (LSS) is used to assess the similarity between time series. Because of the computational limitations of this method, the comparisons are limited to specific features found in the time series. As proof-of-concept the comparison between features of time series of the CLOUD experiment are performed and a discussion on the advantages and caveats of implementing this comparison mechanism in DAQBroker is performed.

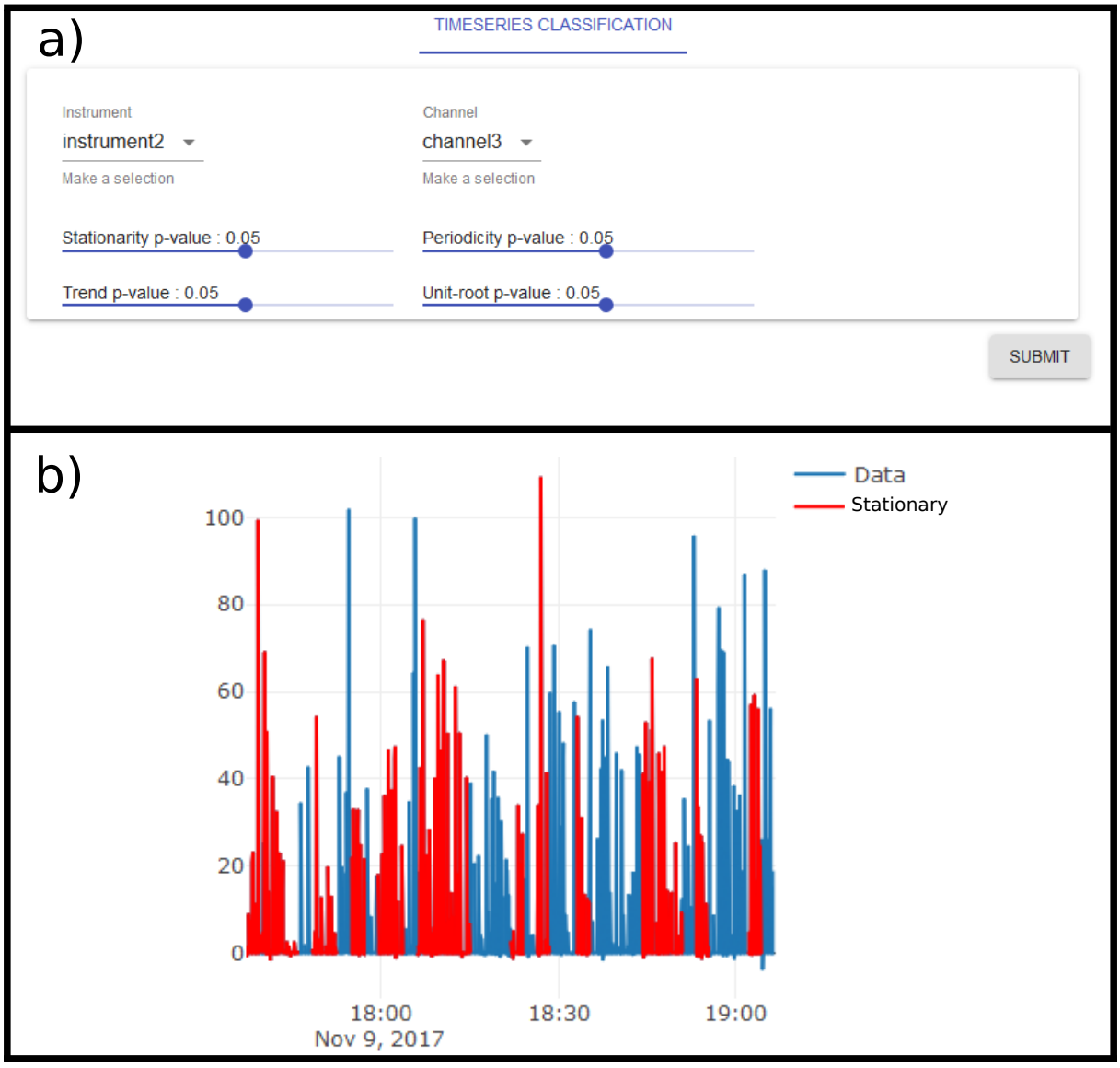



*Figure 5.6: Mock-up of the time series classification interface (a) and subsequent visualization (b)*

#### 5.1.2.1 Comparison Procedure

The method of comparison of time series intends to provide an ordered list of the compared series from most to least similar. It also decides whether any time series can be considered similar to another. This method begins with a set of pre-selected intervals identifying the features in a time series. Then, the same time length intervals are collected from the time series to be compared. For each pair of time series, the user provides a maximum time delay to be considered and then algorithm 1 is applied to both time series to calculate the LSS statistic (see chapter 2). To improve the speed of the algorithm and to limit the emergence of undesired correlations, the maximum delay is limited to a percentage (50%) of the length of the compared time series. Once all comparisons are made, the results are listed in a table for the user to analyze, the comparison with the Pearson Correlation coefficient is also listed to follow the change in the LSS with increasing delay.

#### 5.1.2.2 Comparison of CLOUD Time Series

A set of comparisons using CLOUD time series are presented in this subsection. In each main time series, three features are highlighted and comparisons are made with other time series for those features in an effort to find similar behaviors. Comparison of time series are handled feature by feature and feature selection will be manual and limited to input parameters in the CLOUD experiment. This specifically means that only temperature, UV intensity and $\alpha$-pinene concentration features will be considered, since these parameters do not result from any reaction in the chamber. The results of the comparison are presented as the LSA statistic for several different time delays (0, 10, 20 and 30 minutes) along with their execution time and the Pearson correlation coefficient to compare with traditional correlation methods.

**5.1.2.2.1 UV Intensity** Figure 5.7 presents the features of UV intensity marked for comparison. The purpose of the UV system is to increase the amount of sulphuric acid (SA) in the chamber[349]. By increasing SA, the concentration of particles is expected to increase. Also, the internal temperature of the chamber is expected to increase with the UV intensity. Thus, meaningful correlation is expected for temperature, sulphuric acid concentration and particle concentration.

Figures 5.8, 5.9 and 5.10 show the results for each feature comparison using the same period of the UV intensity time series: the local similarity statistic for a delay of 0, 10, 20 and 30 minutes ($LSS_i$) along with the execution time of the algorithm and finally the Pearson correlation coefficient. The smaller the delay is, the closer the results are to the standard correlation coefficient as expected with the definition of the local similarity algorithm. Once a meaningful delay is introduced, the values tend to change.

1. Feature 1 (figure 5.8) - for this feature one can clearly see the relevance of the local similarity statistic. For this comparison, the relationship is direct, its production reaction, from $SO_2$ by means of photolysis, is a direct consequence of the UV intensity. Since the photolysis requires time, the default correlation coefficient does not provide a value as high as the first delayed LSS statistic value. The default correlation coefficient produces a negative value since the UV intensity increases as the particle concentration decreases

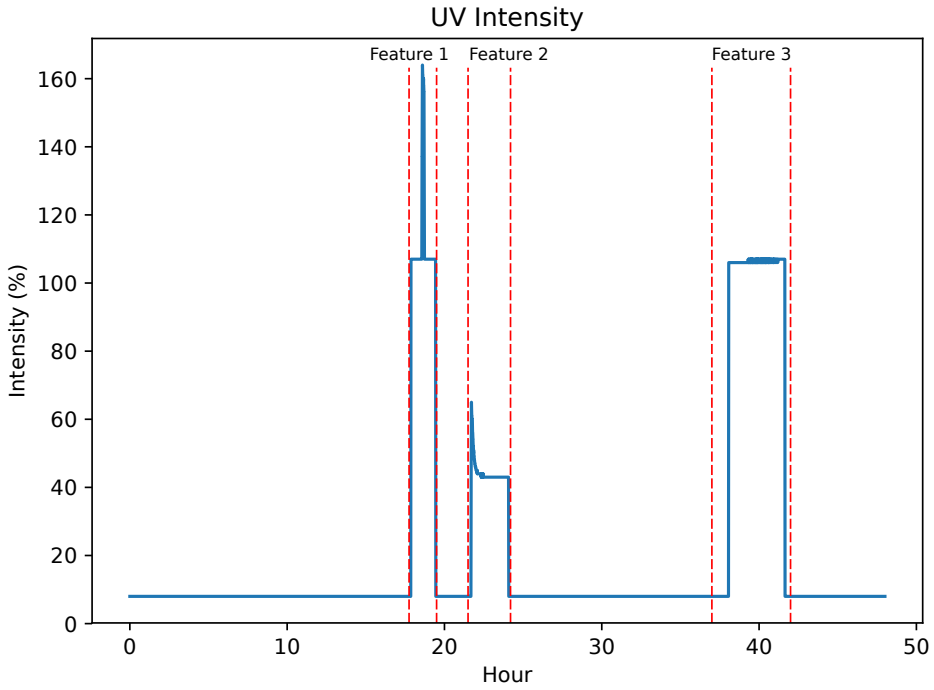



*Figure 5.7: Selected features from the UV Intensity time series.*

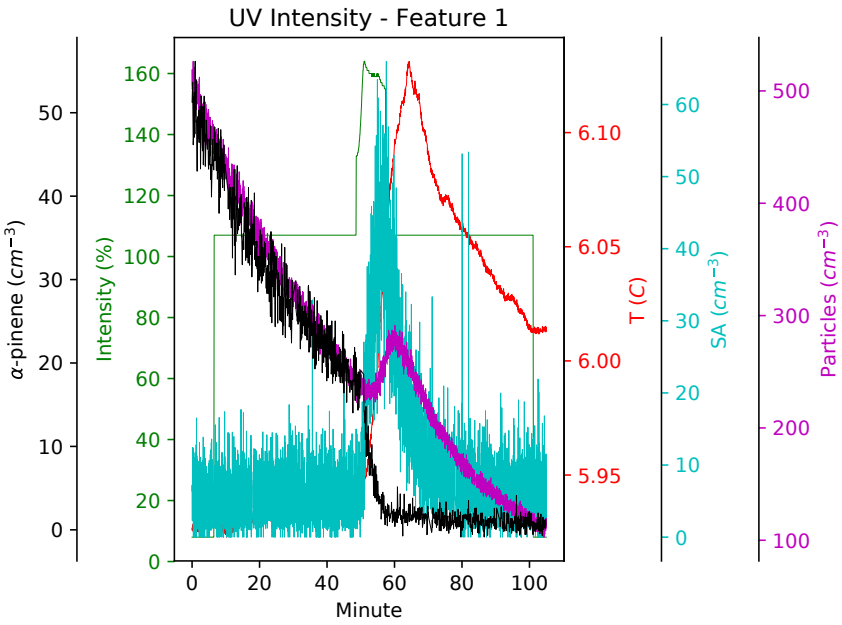
for the majority of the time series. However, there is a clear delayed intensity spike around the 50 minute mark. Knowing that the increase in particle concentration is directly related to the increase in sulphuric acid concentration, it is undeniable that the correlation between particle concentration and UV intensity should be positive. This is identified by a high delay LSS statistic for similarity value as opposed to a small globally negative correlation value provided by the default correlation coefficient (see table in figure reffig:intensityCompare1).

2. Feature 2 (figure 5.9) - for this feature, the use of the LSS statistic for is less meaningful than in feature 1, and there seems to be a smaller delay in the creation of particles. This highlights the need for correct choice of maximum time delay for feature comparison. This comparison also highlights the need for a correct choice of maximum time delay for LSS calculation. The values vary between highly correlated and highly anti-correlated, due to small fluctuations in the temperature not always associated with the UV intensity, leading to spurious correlation.

3. Feature 3 (figure 5.10) - the same behavior as feature 2 can be found for the third feature LSS producing values similar to the correlation coefficient, with a higher particle formation rate, possibly due to higher temperatures or higher $\alpha$-pinene concentration.

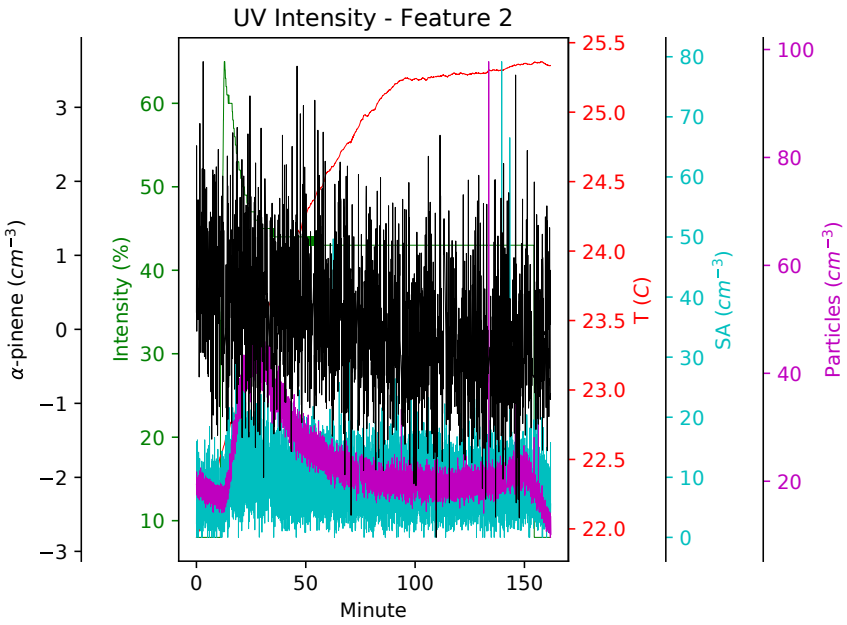
The results of this comparison effort show that LSS can be useful in situations where the behavior of the system is known, but the delay must be chosen carefully to ensure that no spurious correlation values are encountered.

**5.1.2.2.2 Temperature** Figure 5.11 shows the temperature features to be compared to the rest of the time series. Features 1 and 3 are the result of the modification of an external parameter (fan speed increase). This change



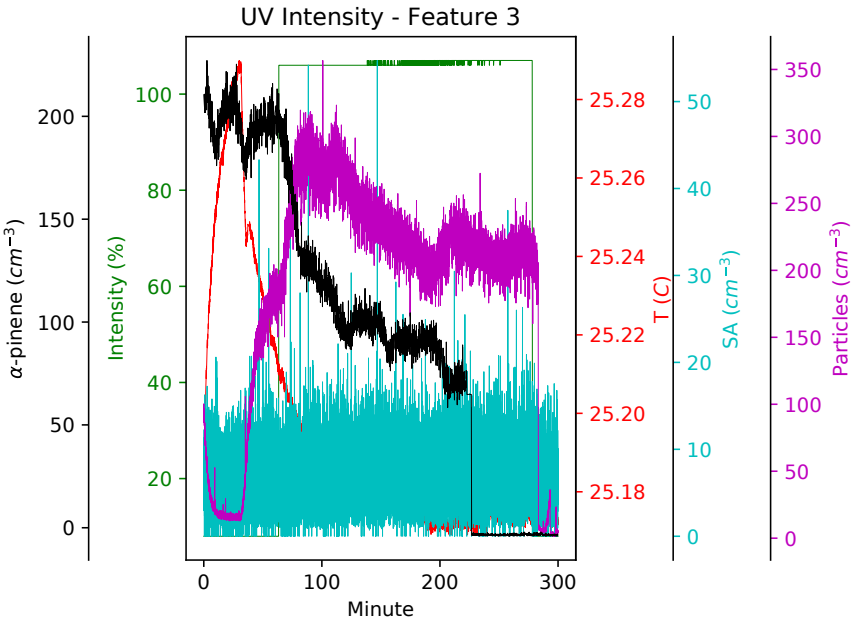

| UV Feature 1 VS. | $LSS_0$ (value,time) | $LSS_{10}$ (value,time) | $LSS_{20}$ (value,time) | $LSS_{30}$ (value,time) | Pearson Correlation |
|---|---|---|---|---|---|
| Sulphuric Acid | (0.3638,14.4) | (0.513,29.5) | (0.579,41.1) | (0.580,48.5) | 0.366 |
| α-pinene | (-0.395,13.8) | (0.396,28.8) | (0.693,40.4) | (0.986,46.9) | -0.300 |
| Particles | (-0.387,13.6) | (0.521,28.5) | (0.876,40.2) | (1.00,47.8) | -0.242 |
| Temperature | (0.168,13.9) | (0.273,30.0) | (-0.483,39.8) | (-0.806,48.0) | 0.119 |

*Figure 5.8: Behavior of all time series during the period of UV feature 1. This particular feature consists of an abrupt change at the beginning and end as well as a spike of about 10 minutes in the middle of the interval. The behavior of particle concentration is clearly influenced by this middle spike, but with a delay.*




| UV Feature 2 VS. | LSS0 (value,time) | LSS10 (value,time) | LSS20 (value,time) | LSS30 (value,time) | Pearson Correlation |
|---|---|---|---|---|---|
| Sulphuric Acid | (0.239,31.8) | (0.154,57.3) | (0.116,76.8) | (0.123,93.1) | 0.241 |
| α-pinene | (-0.135,32.7) | (-0.215,57.8) | (0.243,76.6) | (0.316,93.8) | -0.073 |
| Particles | (0.375,31.9) | (0.261,56.6) | (0.321,77.3) | (0.400,96.6) | 0.376 |
| Temperature | (0.417,32.1) | (0.704,57.5) | (-0.487,77.8) | (-0.574,91.4) | 0.193 |

*Figure 5.9: Behavior of all time series during the period of UV feature 2. This feature consists of a large spike in the beginning, followed by a slow decay of about 40 minutes. The particle plot as well as the sulphuric acid plot seems to follow the intensity plot, which is to be expected from [349].*




| UV Feature 3 VS. | $LSS_0$ (value,time) | $LSS_{10}$ (value,time) | $LSS_{20}$ (value,time) | $LSS_{30}$ (value,time) | Pearson Correlation |
|---|---|---|---|---|---|
| Sulphuric Acid | (0.142,111.0) | (0.109,156.7) | (0.097,199.7) | (0.097,243.8) | 0.142 |
| α-pinene | (-0.594,111.6) | (-0.533,157.7) | (-0.571,202.3) | (-0.594,243.8) | -0.431 |
| Particles | (0.846,111.2) | (0.813,156.9) | (0.763,201.2) | (0.657,241.7) | 0.846 |
| Temperature | (-0.780,112.6) | (-0.735,159.2) | (-0.657,200.4) | (-0.612,236.5) | -0.706 |

*Figure 5.10: Behavior of all time series during the period of UV feature 3. This feature consists of an abrupt change at the beginning and end of the period. However, in this figure α-pinene is also present, which also is affected by the introduction of UV light into the chamber but also has an external source.*

of temperature is not expected to impact any of the other chosen time series. Feature 2 is the major change in value of temperature. This feature changes some of the other time series, as temperature has an effect on all chemical reactions inside the CLOUD chamber. No meaningful correlation should be found with temperature and UV intensity, since UV intensity does not depend on the chamber temperature.

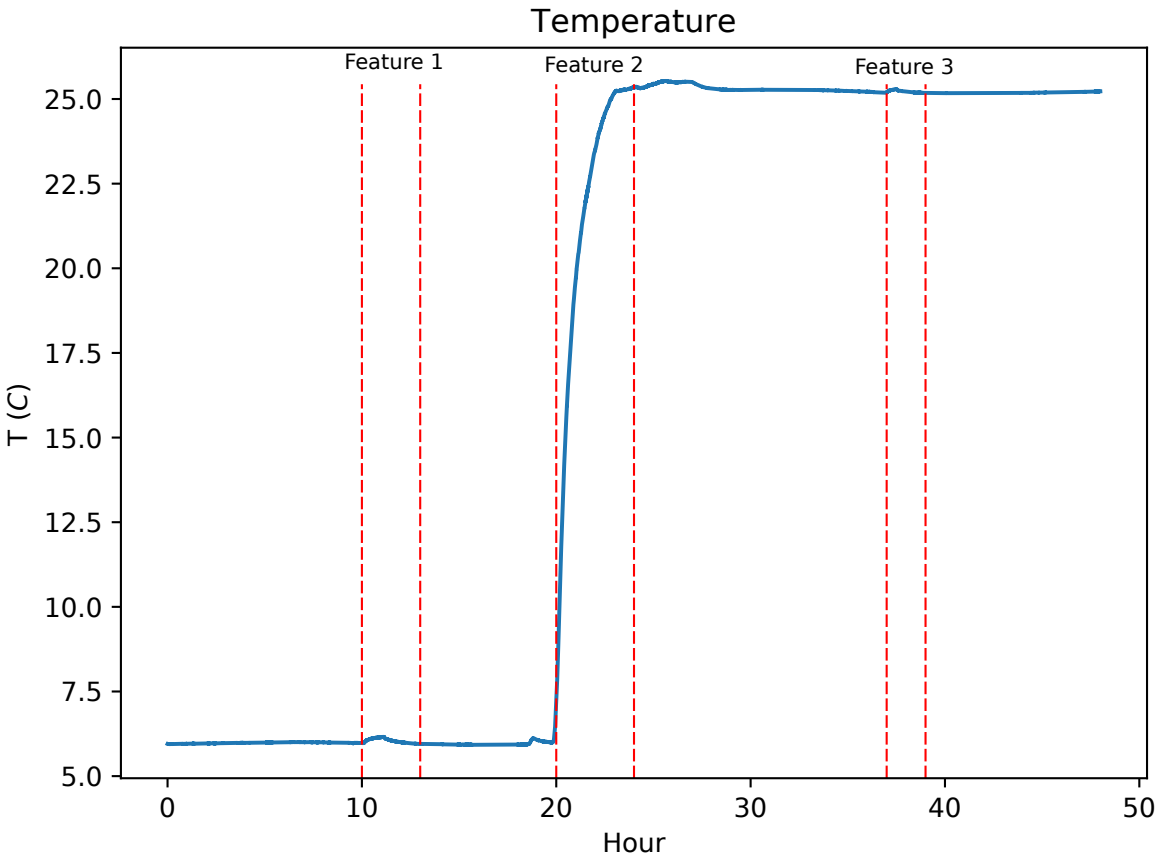



*Figure 5.11: Selected features from the temperature time series.*

Figures 5.12, 5.13 and 5.14 show the behavior of each chosen time series during the period of each selected feature as well as the result of the application of the various similarity measurements.

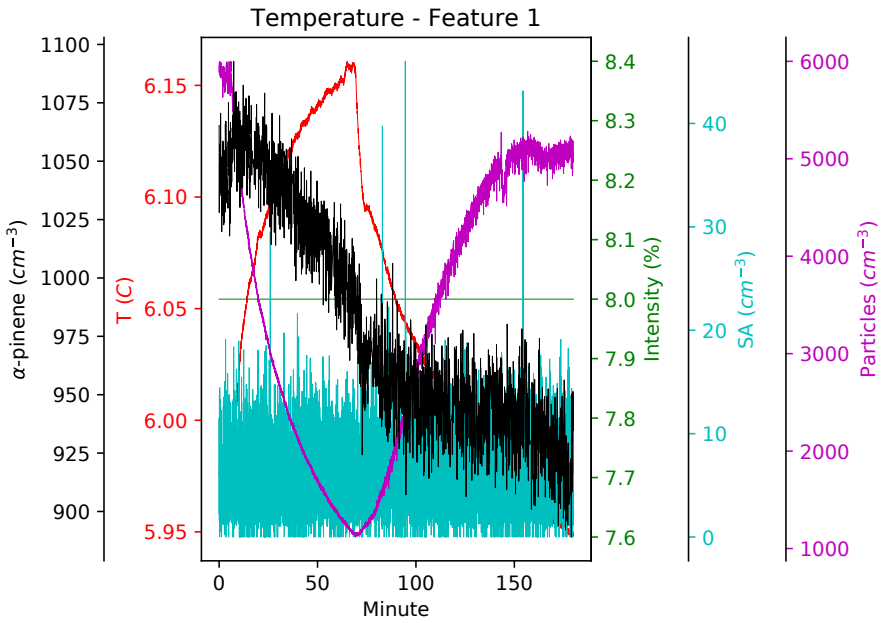
1. Feature 1 (figure 5.12) - in this feature the increase of mixing fan speed in the CLOUD experiment is related to a slight increase in the temperature and to a sharp decrease in particle concentration. Thus it's not surprising that a high anti-correlation is obtained by low LSS and default correlation coefficient. A delayed positive correlation with $\alpha$-pinene is found by way of LSS. This correlation can be more closely related to temperature variation since the $\alpha$-pinene change is not as sharp and its intake is not temperature controlled. Correlation between UV intensity and temperature failed because the algorithm is not prepared to handle channels with one single value in the whole time period.

2. Feature 2 (figure 5.13) - this feature contains a slow increase of temperature over 4 hours. Though this significant increase of temperature should produce changes in the chemical species in the chamber, both sulphuric acid and $\alpha$-pinene are at extremely low levels and no significant change is visibe in the data or

results via LSS or correlation statistics. A spurious high correlation is found with UV intensity because during this period (during which no active experiment is performed) a UV source was being used for particle cleaning. Similarly a high anti-correlation is recorded with the default correlation coefficient (but negated with larger delay LSS) since during this period the particles in the chamber were slowly being cleared.

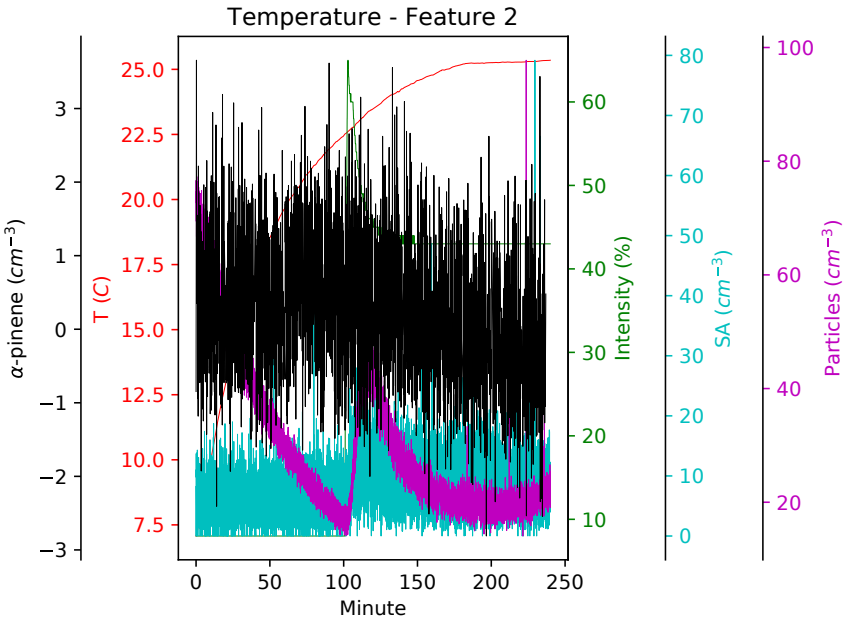
3. Feature 3 (figure 5.14) - in this feature the same increase in mixing fan speed is made, which results in a feature similar to feature 1, however, in this period, the particle and $\alpha$-pinene concentration are much lower. When the fan speed is decreased at around 30 minutes, the temperature returns to normal levels and the particles start to slowly increase in number. This results in a very high unrelated anti-correlation between temperature and particle concentration which is not dampened by LSS since a new stage of UV intensity is started during this period, which increases particle formation further while temperature decreases. A high correlation between $\alpha$-pinene is also found during this period, as the compound is necessary for particle formation and is thus decreases while lowering temperature (at the time of UV increase).

While no specific meaningful correlation was found for the chosen temperature features, considering the small number of 5 channels, meaningful values are found by correlation method and LSS for smaller delay values, that are very close..



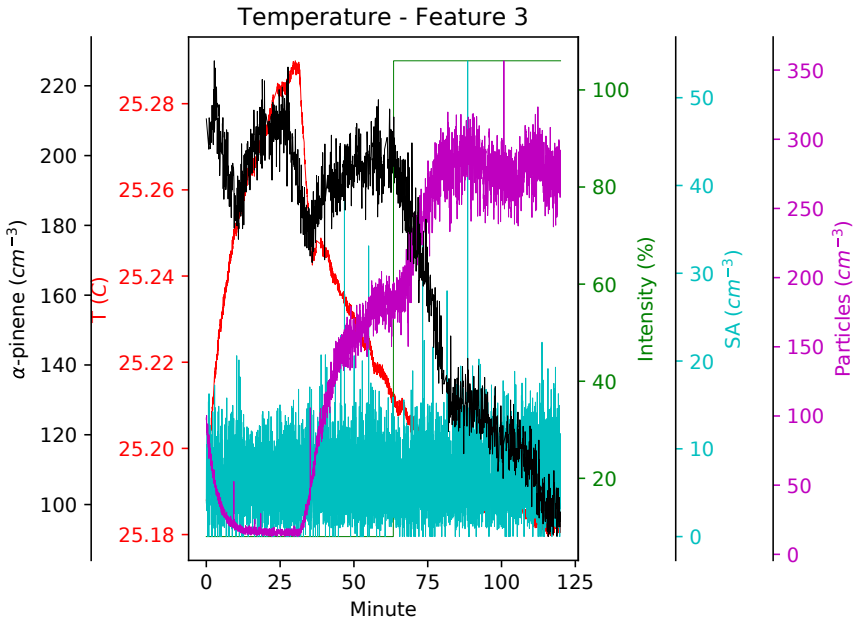

| T Feature 1 VS. | LSS$_0$ (value,time) | LSS$_{10}$ (value,time) | LSS$_{20}$ (value,time) | LSS$_{30}$ (value,time) | Pearson Correlation |
|---|---|---|---|---|---|
| Sulphuric Acid | (-0.019,39.6) | (-0.016,66.8) | (-0.017,90.7) | (-0.019,110.4) | -0.018 |
| α-pinene | (0.662,40.1) | (0.850,71.7) | (0.983,93.2) | (1.00,110.8) | 0.583 |
| Particles | (-0.902,39.2) | (-0.758,66.5) | (-0.675,87.8) | (-0.756,106.6) | -0.903 |
| UV Intensity | (0.0,40.9) | (0.0,67.8) | (0.0,89.0) | (0.0,103.7) | NaN |

*Figure 5.12: Behavior of all time series during the period of temperature feature 1. This particular feature consists of an abrupt increase of 0.2 C followed by a similarly abrupt decrease of 0.2 C. This is a small temperature change taking into account the large volume of the CLOUD chamber and should not affect the overall processes in the CLOUD chamber.*




| T Feature 2 VS. | $LSS_0$ (value,time) | $LSS_{10}$ (value,time) | $LSS_{20}$ (value,time) | $LSS_{30}$ (value,time) | Pearson Correlation |
|---|---|---|---|---|---|
| Sulphuric Acid | (0.274,70.3) | (0.229,106.8) | (0.191,140.6) | (0.161,168.5) | 0.274 |
| α-pinene | (0.282,73.1) | (0.229,106.8) | (0.172,140.2) | (0.143,165.8) | 0.237 |
| Particles | (-0.878,72.2) | (-0.717,105.7) | (-0.585,139.0) | (-0.483,168.7) | -0.880 |
| UV Intensity | (0.754,70.1) | (0.628,105.6) | (0.524,137.0) | (0.439,165.1) | 0.755 |

*Figure 5.13: Behavior of all time series during the period of temperature feature 2. This feature consists of a slow increase of temperature over several hours from 7.5 to 25 C. This feature is expected to change the chemical reactions in the chamber, however, this period is not an active experimental environment so external changes and low overall values of the other parameters fail to illustrate the dependence*




| T Feature 3 VS. | $LSS_0$ (value,time) | $LSS_{10}$ (value,time) | $LSS_{20}$ (value,time) | $LSS_{30}$ (value,time) | Pearson Correlation |
|---|---|---|---|---|---|
| Sulphuric Acid | (-0.096,17.3) | (-0.096,34.5) | (-0.078,47.9) | (-0.091,57.2) | -0.094 |
| α-pinene | (0.797,18.1) | (0.806,37.5) | (0.691,50.1) | (0.828,57.0) | 0.782 |
| Particles | (-0.935,17.6) | (-1.00,34.0) | (-0.897,46.2) | (-0.920,55.8) | -0.923 |
| UV Intensity | (-0.835,17.1) | (-0.851,34.8) | (-0.816,46.5) | (-0.996,54.9) | -0.821 |

*Figure 5.14: Behavior of all time series during the period of temperature feature 3. This feature consists of an abrupt change of 0.1 C followed by a slow decay of that temperature over 100 minutes. This is a small change that should not affect any time series. However, considering only these channels and time series, some relationships can be statistically found between particle and α-pinene concentration.*

**5.1.2.2.3** **$\alpha$-pinene** Figure 5.15 shows the selected $\alpha$-pinene features. The first two consist in consecutive increases in the concentration with the intent to map particle formation rates. The third feature consists on settling a desired concentration. $\alpha$-pinene concentration is strongly connected to new particle formation, thus that a positive correlation with small delay is expected to exist between both concentrations.

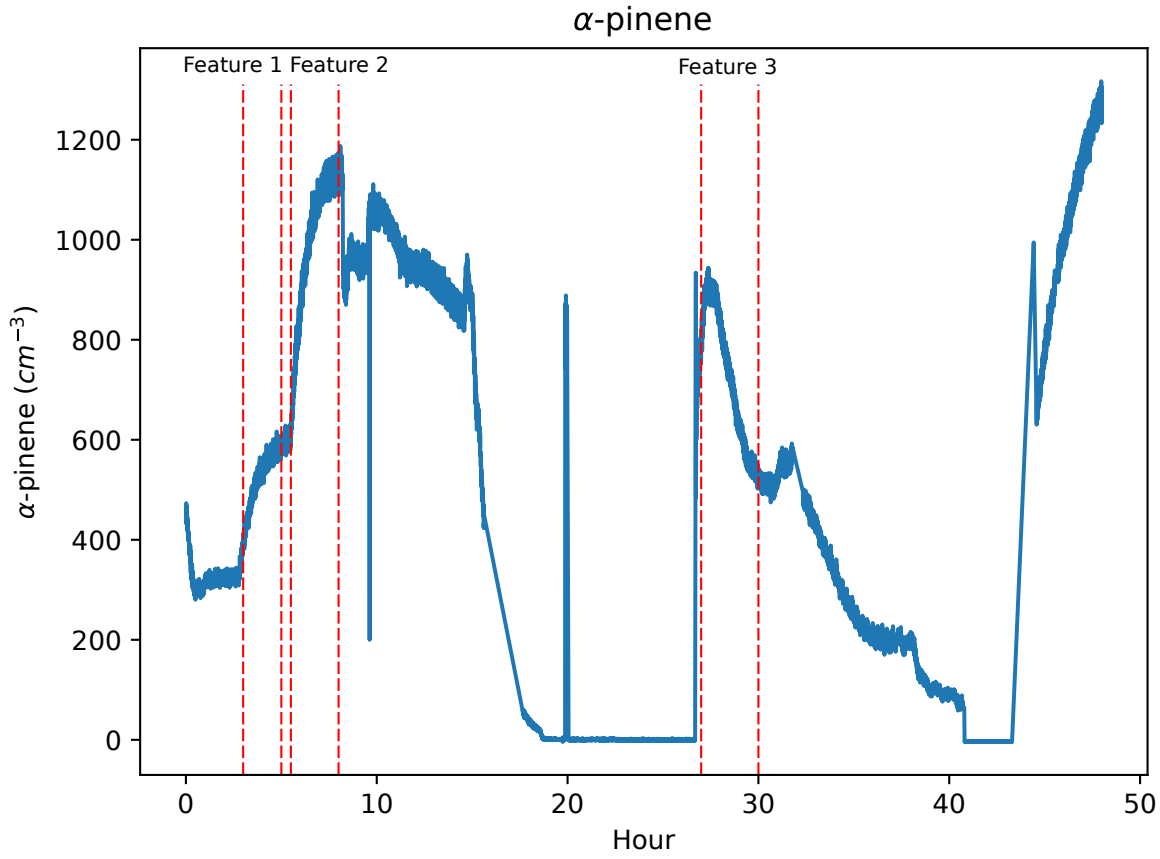



*Figure 5.15: Selected features from the $\alpha$-pinene time series.*

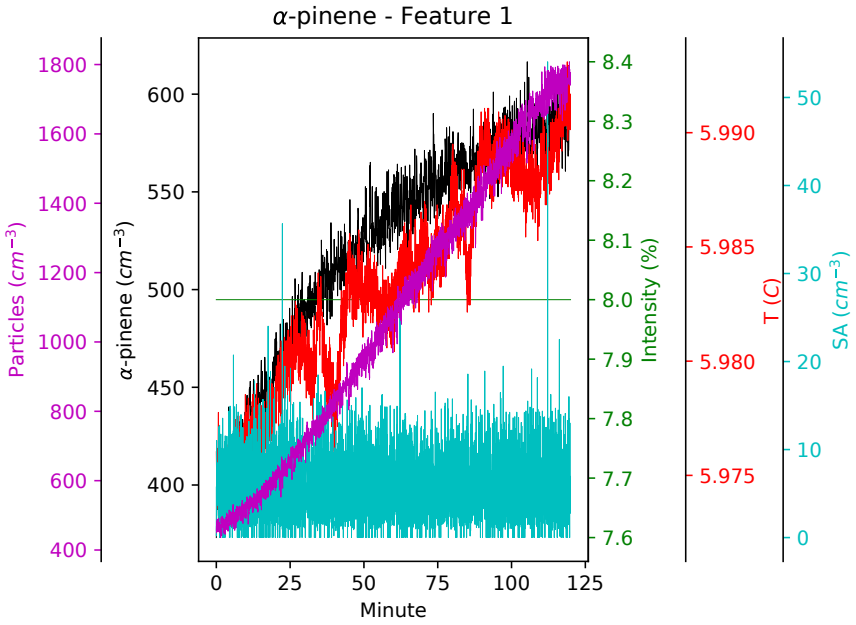
Figures 5.16, 5.17 and 5.18 show the individual behavior of each time series during the period of each event as well as the result of the comparison between time series, using LSS and correlation coefficient.

1. Feature 1 and 2 (figures 5.16 and 5.17) - these features show two nucleation events dominated by $\alpha$-pinene concentration, without any particular delay as it is evidenced by the higher numbers of the LSS statistic with smaller delay. A high correlation value is also attributed to the temperature during these intervals, which could be explained by the less efficient temperature control of the inlet gases when compared to the CLOUD chamber. However, these temperature changes are so small that they could be explained by electrical noise of the sensor's acquisition system and may not have physical meaning.

2. Feature 3 (figure 5.18) - shows an initial increase of the $\alpha$-pinene concentration up to a stable value (plateau at around 600 $cm^{-3}$). A corresponding larger similarity value is attributed to the temperature, which seems to decrease meaningfully, albeit with small values. This could be the consequence of the injection of compounds into the chamber, since these values were taken in mid-October at room temperature that was lower

than the set point of 25 °C. Thus an inefficient temperature control of inlet compounds would promote these temperature variations. Particle concentration appears to have a positive response to the change in $\alpha$-pinene, but not as meaningful as temperature and with no visible delay as illustrated by the mainly similar LSS values. During all comparisons, UV Intensity was kept constant, single valued, and the comparison algorithm fails. Sulphuric acid does not seem to have significant change with the injection of $\alpha$-pinene, which would make sense as no UV was introduced to promote photolysis.

These features illustrate the similarity of LSS and Pearson correlation coefficient, when no delays are visible in the data.

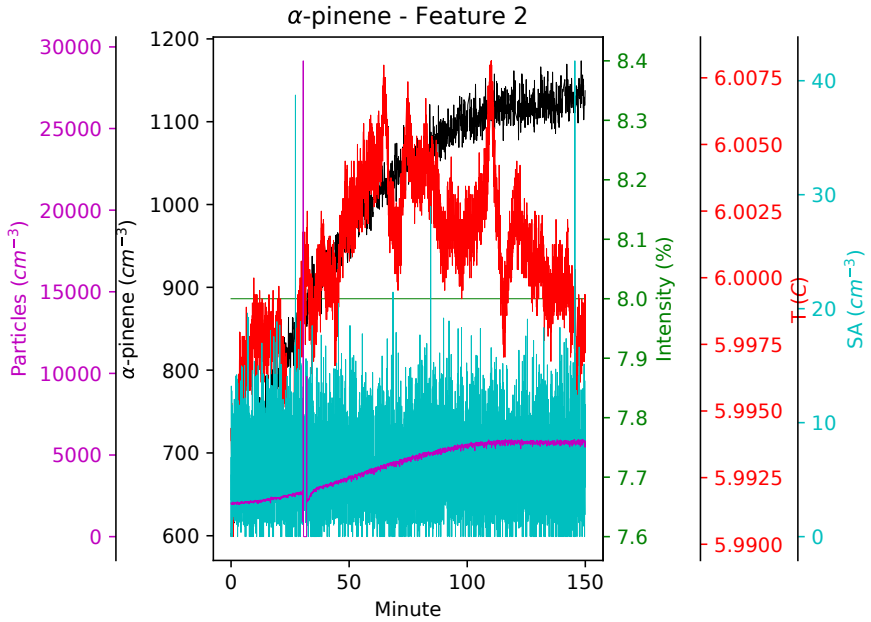



| AP Feature 1 VS. | LSS$_0$ (value,time) | LSS$_{10}$ (value,time) | LSS$_{20}$ (value,time) | LSS$_{30}$ (value,time) | Pearson Correlation |
|---|---|---|---|---|---|
| Sulphuric Acid | (0.023,17.6) | (0.025,34.3) | (-0.022,47.8) | (0.021,56.0) | 0.016 |
| UV Intensity | (0.0,17.7) | (0.0,34.1) | (0.0,46.7) | (0.0,54.2) | NaN |
| Particles | (0.933,18.1) | (0.829,36.4) | (0.811,47.3) | (0.718,58.1) | 0.935 |
| Temperature | (0.928,17.2) | (0.814,34.5) | (0.796,46.8) | (0.697,56.1) | 0.929 |

*Figure 5.16: Behavior of all time series during the period of $\alpha$-pinene feature 1. This feature consists of a slow increase in concentration from around 400 to 600 $cm^{-3}$ over the course of 2 hours. During this time particle formation is occurring, as evidenced by the increase in particle concentration, and a very slight increase in temperature is observed.*

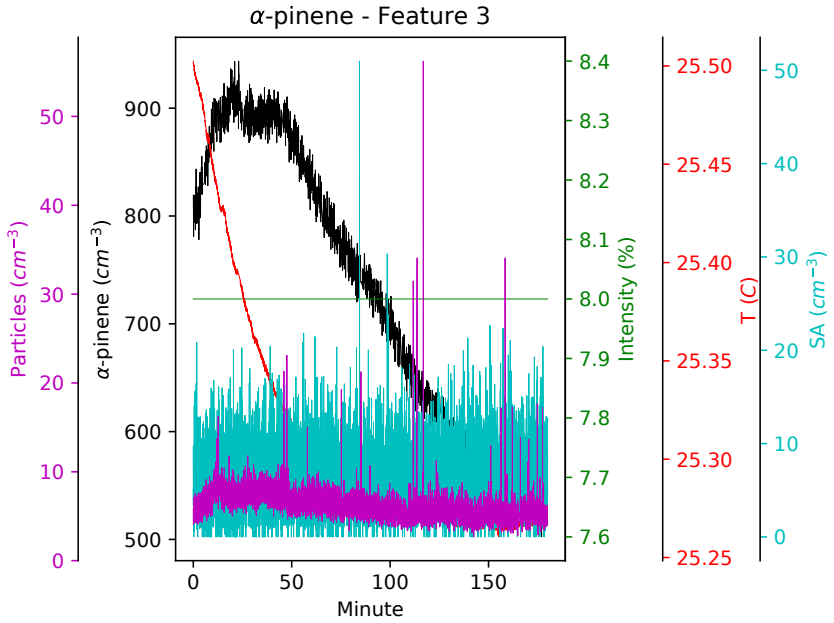
#### 5.1.2.3 Implementation in DAQBroker

As can be seen in the tables of the previous subsection, the use of the local similarity statistic with the selected features length can use delays between 20 to 150 seconds. This makes LSS a statistic of limited scope for use in large scale comparisons. One way to implement LSS is to limit the delay to small values and the feature size to a certain number of data points, however, this will also limit the amount of features that can be compared. Assuming that a database's channels and feature size are small, an interface can be created for the user to provide the maximum delay between correlated features and the time interval of the main feature to compare. This would allow DAQBroker to apply LSS to each pair of channels, producing an association network[253] that shows the




| AP Feature 2 VS. | $LSS_0$ (value,time) | $LSS_{10}$ (value,time) | $LSS_{20}$ (value,time) | $LSS_{30}$ (value,time) | Pearson Correlation |
|---|---|---|---|---|---|
| Sulphuric Acid | (0.026,27.4) | (0.022,49.0) | (0.018,66.9) | (0.017,80.0) | 0.019 |
| UV Intensity | (0.0,27.1) | (0.0,47.9) | (0.0,64.1) | (0.0,80.4) | NaN |
| Particles | (0.887,26.7) | (0.748,49.5) | (0.637,66.6) | (0.535,80.8) | 0.888 |
| Temperature | (0.664,29.6) | (0.576,50.1) | (0.498,67.8) | (0.436,80.7) | 0.603 |

*Figure 5.17: Behavior of all time series during the period of α-pinene feature 2. Similarly to feature 1 this feature consists in the increase in α-pinene concentration from 600 to 1100 $cm^{-3}$ over the course of 2.5 hours. During this time, particle formation is also observed despite the abrupt high value at around 25 minutes. Temperature seems to be stable but there is a slight increase during introduction of α-pinene.*




| AP Feature 3 VS. | $LSS_0$ (value,time) | $LSS_{10}$ (value,time) | $LSS_{20}$ (value,time) | $LSS_{30}$ (value,time) | Pearson Correlation |
|---|---|---|---|---|---|
| Sulphuric Acid | (-0.033,38.5) | (-0.035,65.0) | (-0.036,87.9) | (-0.036,106.5) | -0.033 |
| UV Intensity | (0.0,39.1) | (0.0,64.3) | (0.0,85.4) | (0.0,102.5) | NaN |
| Particles | (0.510,38.5) | (0.531,64.4) | (0.519,86.3) | (0.496,104.1) | 0.507 |
| Temperature | (0.721,39.5) | (0.855,66.3) | (0.896,90.0) | (0.899,104.3) | 0.720 |

*Figure 5.18: Behavior of all time series during the period of α-pinene feature 3. This feature consists of a fast increase of α-pinene concentration from 800 to 900 $cm^{-3}$ followed by a slow decay to around 600 $cm^{-3}$ over the course of 2.5 hours. The temperature seems to decrease during the change in compound injection. Concentration of particles seems to follow the change in behavior of α-pinene.*

most and least correlated channels. LSS can also be used to infer feature causality since the time series pair with highest LSS can be visually inspected within the delay. When the feature size and/or number of channels are too large, an alternative feature analysis can be offered by means of a correlation matrix. A correlation matrix maps pairs of channels as points in a $P \times P$ matrix ($P$ being the number of channels) via the Pearson correlation coefficient. This method does not account for delays and does not infer causality but is more efficient in terms of processing time and can be used on a multi-step analysis to filter out channels that do not present meaningful correlation for a further LSS step with less channels. Figure 5.19 shows a suggested interface necessary for the time series comparison as well as an example of the resulting visualization.


*Figure 5.19: Mock-up of the times eries comparison interface (a) and subsequent visualization (b)*

### 5.1.3 Time Series Event Detection

To complement the existing DAQBroker tools for manual input of experimental runs and events, a method of automated event detection is considered. Due to its portability and low computational effort, the symbolic wavelet partitioning method described in 2.2.1.3 is the one selected. This subsection introduces the method applied in CLOUD data followed by the corresponding results. The implementation of this method in the DAQBroker API is also discussed from a user interface point-of-view.

#### 5.1.3.1 Event Detection Implementation

The symbolic wavelet partitioning method consists in dividing a predetermined time segment into slices, the first slice is considered to be the test slice from which a wavelet transform is preformed. The transformed slice is then partitioned such that each partition of the slice contains the same number of wavelet coefficients. The number of slices is determined by an entropy minimization method using the Shannon Entropy expression[350]. The change in entropy ($\Delta E$) associated with the increase of the number of partitions of the data is calculated by:

$$\Delta E(N) = -\sum_{n=1}^{n=N} p_n \log_2 p_n \tag{5.1}$$

Where $N$ is the number of partitions chosen and $p_i$ is the probability of a data point being in a specific partition (in this partitioning method $p_i = \frac{N_p}{N}$ where $N_p$ is the number of points in the slice). Equation 5.1 is a decreasing function with increasing $N$ meaning that there is less information gained as the number of partitions increases. The user must select the limiting value for entropy variation that chooses the number of partitions created. Once a suitable wavelet partition is created for the first slice of the data, the slice will be represented by a probability vector $\mathbf{P}$, which represents the slice in the chosen partitioning scheme. The vector can be compared to other probability vectors in the same partitioning scheme by vector operations such as calculating the *angle* ($\alpha$) between these $N$-dimensional vectors:

$$\alpha_{i,j} = \text{acos}\left(\frac{\langle \mathbf{P_i}, \mathbf{P_j} \rangle}{||\mathbf{P_i}||||\mathbf{P_j}||}\right) \tag{5.2}$$

Where $\langle \bullet, \bullet \rangle$ is the inner product between two vectors and $||\bullet||$ represents the Eucledian norm of a vector. Equation 5.2 is limited to the range of $[0, \frac{\pi}{2}]$ since the elements of the probability vector are strictly positive. When the angle between vectors is sufficiently large, the segment analyzed is considered significantly altered and is classified as anomalous. This *sufficiently large* limit can be supplied by the user to select the sensitivity of event detection.

#### 5.1.3.2 CLOUD time series event detection

The described event detection algorithm is applied to the channels and time interval introduced in the beginning of the chapter. While a proper application of the method would be for recent online data, its flexibility allows it to be used on a larger time series with relatively fast execution time. The results can be used as proof-of-concept for the performance of the method.

**5.1.3.2.1 UV Intensity** Figure 5.20 shows the time series of UV intensity and highlights the time slices where an event was flagged. Each tested time segment had the duration of 30 minutes and each slice was 3 minutes long. The limit of event detection was set to 0.75, which corresponds to roughly 50% of the range of limits of equation 5.2. Since the sensor has almost no noise and very sharp changes in value exist for UV intensity modification all

existing features were flagged as possible events. Also during hour 39 and 41 a possible event is flagged associated with noise.

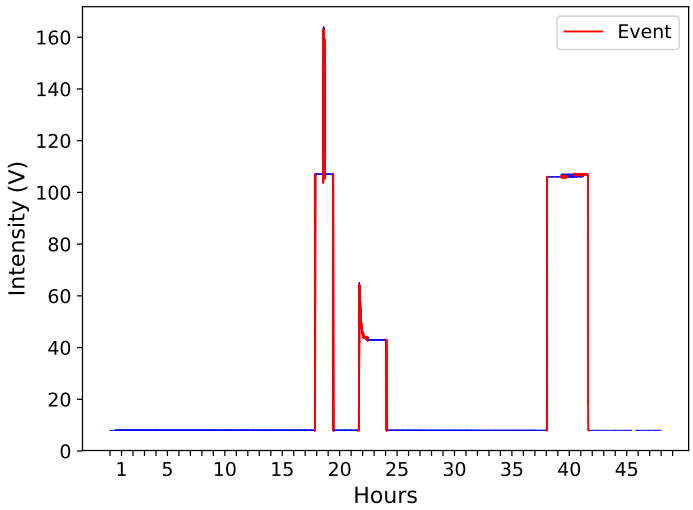



*Figure 5.20: UV intensity time series (blue) and slices flagged as possible events (red).*

**5.1.3.2.2 Temperature** Figure 5.21 shows the time series of temperature as well as the time slices where an event was flagged. Each tested time segment was 60 minutes long and 6 minute slices were considered. The limit of detection was set to 0.81, which corresponds to around 52% of the available range. In figure 5.21 one main feature exists but some smaller ones are also present, at hours 11, 19, 24 and 37. All of the aforementioned features were properly flagged as events, as well as some additional smaller length flags that would require finer analysis to decide whether they correspond to an event or not.

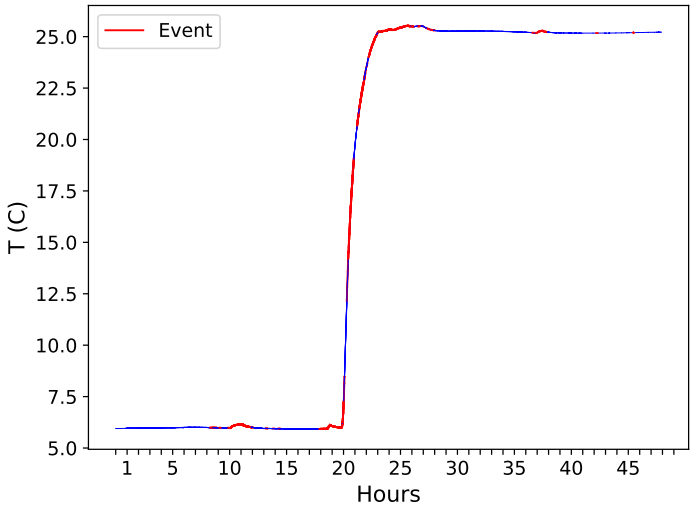



*Figure 5.21: Temperature time series (blue) and slices flagged as possible events (red).*

**5.1.3.2.3 Sulphuric Acid Concentration** Figure 5.22 shows the time series for sulphuric acid inside the CLOUD chamber. Slices that were flagged as possible events are highlighted in red. Time segments are 60 minutes long

and each tested slice is 6 minutes long. The limit of detection was set to 0.8, again slightly above 50% of the available detection range. Since this series is close to the detection limit of the instrument and very little features can be discerned by eye, a look into the list of manually recorded runs is necessary. The black dashed vertical lines represent experimental runs that took place within 6 minutes of a tested slice. From the 24 runs recorded during this period, 15 runs took place within 6 minutes of a possible event detection. The other runs were not detected by the algorithm meaning that either the limit of event signaling was not achieved or the event did not produce a difference in the sulphuric acid concentration. It can also be seen that there are several false positives according to the run list for this period[1], most notably for the period between hours 13 and 18 and the period between hours 40 and 46. These false positives could be removed by increasing the detection limit for possible events, but one must be careful not to increase the amount of false negatives as well.

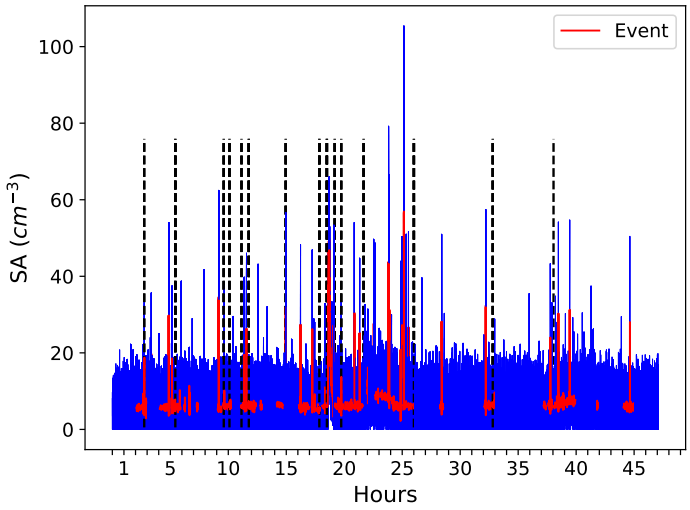



*Figure 5.22: Sulphuric acid concentration time series (blue) and slices flagged as possible events (red). Vertical black dashed lines represent experimental runs that took place in a 6 minute time window from the detection of anomalous behavior.*

**5.1.3.2.4 Particle Concentration** Figure 5.23 shows the time series of particle concentration in the CLOUD chamber as well as time intervals flagged as events. Time segments were 30 minutes long and tested slices were 3 minutes long. The limit for event flagging was set to 0.65, which corresponds to around 40% of the available range of values. It is clear that for this time series, it becomes more complicated to detect events particularly due to the existing noise which increases for smaller values. While events are correctly flagged from hours 1 to 21, when noise was larger, the false positives appear. Increasing the limit range would allow for less falsely detected events, but it would also result in more false negatives. One possible solution to overcome noise is to apply a smoothing algorithm to the data before the event detection method is applied.

**5.1.3.2.5 $\alpha$-pinene Concentration** Figure 5.24 shows the time series for concentration of $\alpha$-pinene and the slices of time flagged as events. Time segments were set to be 90 minutes long and the time slices 9 minutes long. The limit for detection was set to 0.7 which is roughly 45% of the range of the limit. This time series, much like

[1] It is also possible that some changes made to the instrument or the experimental environment were not logged but could still be detected by the algorithm.

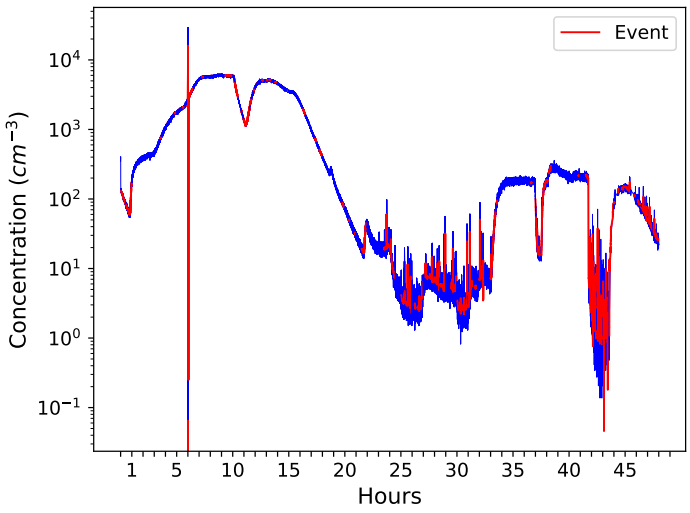



*Figure 5.23: Particle concentration time series (blue) and slices flagged as possible events (red).*

the particle concentration, has significant noise/signal ratio at small values. However, the time series stabilizes at those small values. This allows for the event detection algorithm to adapt to the larger noise and to not flag the larger noise slice as a possible event. At larger values with lower noise/signal ratio, most visible excursions of the data are correctly flagged as events, although some are missed at hour 38 and hour 45. Again a smoothing algorithm should be an option to apply to the data before this method is used.

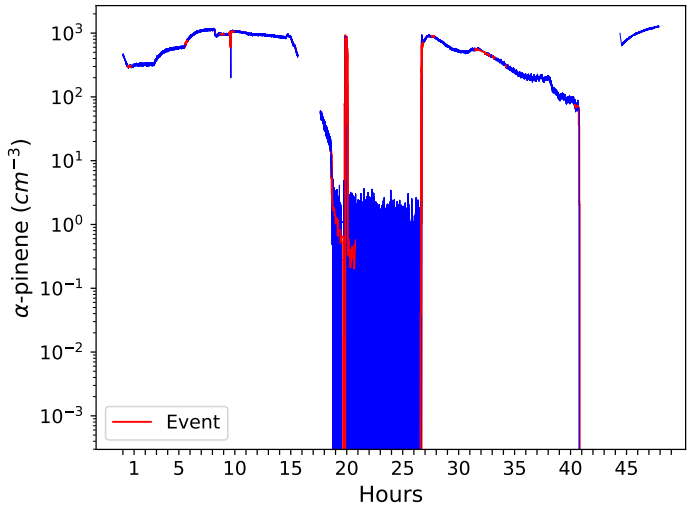



*Figure 5.24: α-pinene concentration time series (blue) and slices flagged as possible events (red).*

#### 5.1.3.3 Implementation in DAQBroker

As introduced earlier in this section, this method of event detection is ideal to complement the existing methods of manual event recording, offering users possible time periods where specific behaviors should be at least highlighted or even recorded alongside experimental runs. Fully automated event detection would need to be ensured by a supervised algorithm and not the unsupervised algorithm introduced in this session and also training would be

required for each channel. To implement this method in DAQBroker requires it to be applied to the most recent data at a time period defined by the user as well as with a specified entropy limit. Alternatively to the user specified limit, a color-coded scheme can be used to classify the results of equation 5.2, such as green for low values, yellow for intermediate values and red for large values, signifying respectively, low, medium and high confidence of the occurrence of an event. From the point of view of interface implementation, users will be able to mark a specific channel or visualization for event detection, and choose the length of time to test, as well as the limit for event detection. When a visualization of that channel is presented, slices flagged as possible events are highlighted and users can take action by either ignoring the flags or creating a new comment. One possible option that should be investigated is to offer a denoising algorithm to minimize false flags, as evidenced from the sulphuric acid and $\alpha$-pinene time series analysis. Figure 5.25 contains a suggested interface provided to the user for requesting the detection as well as a visualization to be provided with the detecetd events highlighted.

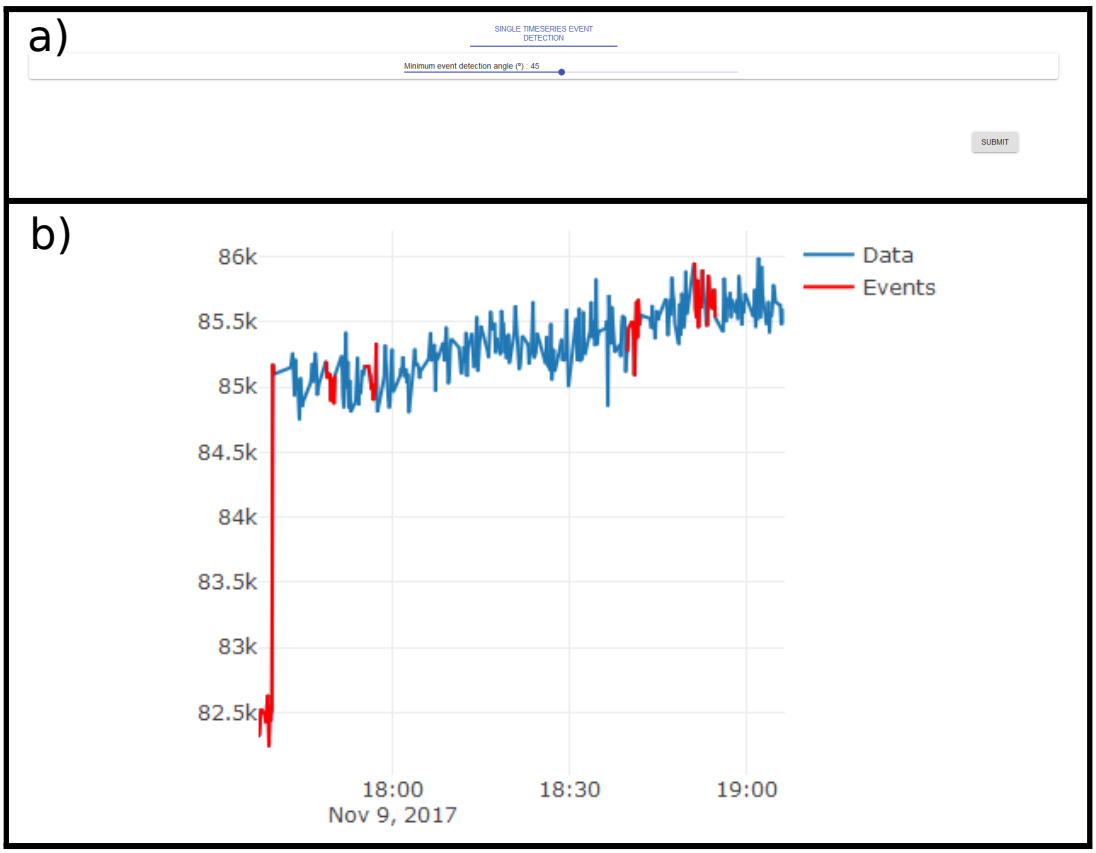



*Figure 5.25: Mockup of the event detection event interface (a) and subsequent visualization (b)*

## 5.2 Large Scale Time Series Analysis for DAQBroker

With the capability of DAQBroker to store large amounts of information from a wide range of instruments, a single user may become flustered with the breadth of information available. For this reason, a strategy for condensing information in DAQBroker is required. As stated in 3.1.3, users can divide relevant periods of experimental analysis into runs and further divide those experimental runs into stages. These structured experimental analysis provide

the ideal separation in analysis periods that allow the production of simple, statistically-based reviews of the available data, highlighting the most relevant data channels during each period, their relationships with other channels, channels that influence the flow of information and periods of more activity during the experimental run. In order for this analysis to be user-friendly, it should provide a wide array of user options, while having default values that properly convey relevant information. This section will focus on the individual analysis that provides a detailed summary of an experimental run to the users.

The methods used in this section are based on multivariate time series analysis. It is therefore required that a set of channels be interpreted as a multivariate time series. For that reason, this section begins by discussing pre-processing methods required to be first applied to the data. It continues by presenting a method of dimensionality reduction that focuses on selecting the most important time series in terms of data variation. A method of providing a visualization of relationships between all data channels and a method for signaling the most active time periods by means of event detection are introduced. This method of experimental run summary is also applied to CLOUD measurements using a specific set of data channels. Finally, a discussion on the implementation of this method in DAQBroker is discussed, highlighting it's strengths and weaknesses.

### 5.2.1 Data pre-processing

In this section, considerations are taken into account for the pre-processing of instrument data so that different data channels can be properly compared and/or placed in a hierarchical order among each other.

Considering the universal scope of DAQBroker, the data provided from different channels in DAQBroker is assumed to come from multiple sources, measure different quantities and use completely different measurement methods. However, given the nature of the experimental runs and stages, it is also fair to assume that during an experimental stage in DAQBroker, all instruments are measuring the influence of the same phenomenon, which by the scientific experiments' nature, should remain stable for a statistically significant time period[2]. This means that during a single stage, it is assumed that all measurements correspond to a stationary process with the purpose of studying the influence of one specific input variable.

The pre-processing step will focus first on time denoising, then on time regularization and last on data normalization.

#### 5.2.1.1 Noise Reduction

In order to properly study a set of time series, a first stage consists in removing unwanted components of the signal (e.g: noise) that may severely compromise the results of the analysis. However, noise reduction methods must be used with moderation, to avoid removal of significant elements of the signal. If the signal is known beforehand,

[2]This may not always be the case for a widely geographically distributed set of instruments, however, these are fringe cases that warrant other means of statistical summaries.

usually a detailed study of the signal is available and a noise model is selected to be subtracted from the spectrum of the experimental data signal[351]. It would be useful to provide the use of specific noise models in the DAQBroker framework. In this thesis, a two-step noise reduction process is implemented. First, a median filter is applied to the data to remove possible outliers. Second, a low-order Savitzky–Golay filter ($\leq 2$) is applied to the data to further decrease noise.

This processing produces a set of data channels that ultimately contain less information, but in most cases, removes the contamination associated with instrument noise for future statistical analysis.

#### 5.2.1.2 Time Regularization

As mentioned in 3.1.2, the one common trait of all data channels in DAQBroker is the time of measurements, but there is no intrinsic requirement that data is collected at the same time intervals or even starts at the same time. A multivariate time series from a set of instruments has to be processed in order to create a set of time series that contain the same number of measurements, to match the one introduced in equation 2.26. While the choice of start and end times is already solved with the use of experimental runs or stages, there is still the matter of providing the same number of data points for each time series[3]. In choosing the method to equal the number of points for all time series there is a trade off between information retained and computational complexity[352], the latter being a sensitive topic when working with continuous data updates and large amounts of data channels. Using the assumption that all instruments are measuring the same process, most time ranges are adjusted to the time scale of the measured process and there is no significant difference between the instrument sampling frequencies. With this assumption, and in order to retain all of the available information (at the expense of future computational speed) for the studied experimental run, all time series are converted to interpolated versions of themselves using the smallest time increment found in the time series collection. Unless stated otherwise, all interpolations in this section are made using the *scipy* package in Python.

#### 5.2.1.3 Data Normalization

Since most time series are produced from different instruments, different data channels will measure different physical and/or chemical quantities. It is of extreme importance to compare all time series values on equal footing and avoid the dominance of the larger values or large excursions[353]. For this reason, a normalization procedure is implemented to each time series to have its data between the same boundary numbers (chosen as 0 and 1). Thus all time series to be compared are mean subtracted and subsequently normalized by its maximum amplitude width, creating a new set of normalized-mean-reduced time series values:

[3] It could be the case that a very limited number of points exist during this time period either as a result of poor data acquisition or of low sampling frequency from the underlying instrument.

$$\overline{\mathbf{X}} = \begin{bmatrix} \overline{X_1} \\ \vdots \\ \overline{X_m} \end{bmatrix}, \quad \overline{X_i} = \left[ \hat{X_i}^1 - \widetilde{\hat{X}_i} \quad \dots \quad \hat{X_i}^n - \widetilde{\hat{X}_i} \right], \quad \hat{X}_i = \left[ \frac{X_i^1 - min(X_i)}{max(X_i) - min(X_i)} \quad \dots \quad \frac{X_i^n - min(X_i)}{max(X_i) - min(X_i)} \right], \tag{5.3}$$

Where $\widetilde{\hat{X}_i} = \frac{1}{n} \sum_{k=1}^{n} \hat{X_i}^k$ is the mean value of the normalized time series and $\overline{X_i}$ is the mean reduced time series matrix. With this new set of time series, all values are between 0 and 1 as explained. Time series which are constant will be set to 0 via mean reduction, to have no influence from constant values, and consider only the variance of each signal.

By applying the above pre-processing steps to all time series it is important to highlight that all time series data were tampered with, there was information suppression by noise reduction and data normalization steps and a added information with the time regularization step. Thus, it is important that in the future implementation of this approach in DAQBroker, these processes are optional[4].

### 5.2.2 Relevant Channel Highlighting

After the pre-processing steps, all time series contain the same number of samples and the same scale, becoming possible to analyze the set as a multivariate time series. The first step in the analysis is to find the *most relevant* time series in the set. It is important to define what *relevance* means. For a scaled set of time series, no particular measurement is more important than others assuming that noise has been reduced to small levels or even completely removed. It will thus be defined that the most relevant time series will be the one contributing most to the power spectrum. Since all time series were obtained during one single experimental stage, it is assumed that all excursions of data are due to the behavior of the studied phenomena. Since for each stage only one phenomena is studied, the assumption of stationarity should be reasonable for most timeseries. The multivariate time series can be converted to the frequency domain by applying the FFT ($\mathscr{F}$) to each row of the previously introduced normalized-mean-reduced $\overline{\mathbf{X}}$ time series matrix:

$$\mathscr{F}_{\overline{\mathbf{X}}}(\omega) = \begin{bmatrix} \mathscr{F}_{\overline{X_1}}(\omega) \\ \vdots \\ \mathscr{F}_{\overline{X_m}}(\omega) \end{bmatrix} \tag{5.4}$$

With the set of time series in the frequency domain, there is no limitation for successive time periods and PCA can be applied to $\mathscr{F}_{\overline{\mathbf{X}}}$. Choosing the relevant percentage of variance to be explained (user defined parameter $\sigma_L$) a simplified version of the previous matrix ($\overline{\mathscr{F}_{\overline{\mathbf{X}}}}$), containing only the most relevant frequencies in the most relevant time series is calculated as the sum of principal components defined in 2.2.2.1:

[4] Apart from some form of time regularization.

$$\overline{\mathscr{F}_{\overline{\mathbf{X}}}} = \sum_{i=0}^{r} \lambda_{\mathbf{i}} \mathbf{S}_i \quad , \quad \sum_{i=0}^{r} \frac{\lambda_i^2}{\sum_n \lambda_n^2} <= \sigma_L \tag{5.5}$$

The LU decomposition, however, introduces some caveats to this method. This decomposition does not take into account the relative importance of specific time series. For example, if one relatively unimportant value of a time series influences the behavior of another more relevant time series, there is no absolute certainty that the most relevant time series will be chosen over the less important one. One solution to this issue is to introduce an auxiliary step in the permutation matrix of the LU decomposition that disallows the permutation of important rows with unimportant one, the importance of each row to be defined by the user.

### 5.2.3 Channel Relationship Visualization

Apart from relevant channel highlighting, the $\overline{\mathbf{X}}$ matrix can also be used for similarity measurements. However, an appropriate similarity measurement should be selected to relate each pair of time series. For this study 4 distinct similarity measurements will be considered:

1. The Pearson correlation coefficient will be used (defined in equation 2.16) to evaluate the direct linear relationship between series,
2. The maximum of the cross correlation function (defined by equation 2.17) used as similarity measurement. This can be seen as a delay-independent generalization of the Pearson coefficient,
3. The dynamic time warping algorithm will be used to provide a different alternative to the previous calculations. The fastDTW algorithm will be used to calculate a distance measurement between series and then said distance measurement is used in equation 2.13 to convert to a similarity measurement,
4. Finally the coherence between time series is calculated (via equation 2.20) to evaluate the similarity of each time series in the frequency domain.

From each of these measurements a similarity matrix can be calculated (as in 2.32) and a network visualization of the relationships between time series can be made, as discussed in 2.2.2.2. A force directed network graph was chosen to visualize the relationships, since it more immediately provides a user with a qualitative state of the relationships via the relative distances between nodes. Since the visualized network is not directed and there is no new information obtained from making a similarity matrix of a time series with itself, there is no expected self-interaction force between nodes, thus the main diagonal of the similarity matrix will not be considered in the creation of this network graph.

#### 5.2.3.1 Expected Results

Since 4 different types of graphs are created, it is important to consider what are the expected results and the reason for using each similarity measurement. Firstly, LSS similarity was not used due to the algorithm's $O(n^2)$ complexity, which was defined in 2.2.1.2 and resulted in relatively high computing time for comparing pairs of time series in 5.1.2.2. The following describes the reasons for choosing each similarity method:

- The Pearson Correlation Coefficient is used to provide a classical comparison of time series, without considering the existence of delays in time series and provide a basis for interpretation of other results using other similarity measurements.

- The maximum value of the Correlation Function to provide a measure which improves on the Pearson Coefficient, the maximum of the correlation function is used, this provides a maximum value for comparing pairs of time series, regardless of what possible lag value exists between them. Predictably, using this maximum value will necessarily yield larger values of Correlation over the Pearson Coefficient, when a significant delay exists between series.

- The DTW similarity is used to attempt a different measurement method over correlation, the DTW distance is used as a similarity measurement. As defined in 2.2.1.2, DTW allows the distortion of the time axis to choose a shape for one series which better describes the other. While a popular method for finding similarities between time series, even when the sampling frequency of time series is different, the pre-processing step defined in 5.2.1 handles the up-sampling of time series, thus the advantage of using DTW is expected to be somewhat mitigated.

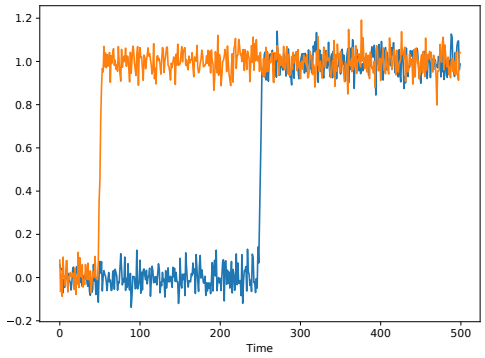
- Coherence is used much like the Pearson Correlation Coefficient in frequency space, comparing how one frequency of one signal is responsible for changing the same frequency of another signal. Since this method focuses purely on the real part of the frequency spectrum, the concept of delay is not even defined for coherence, thus a signal can only be further compared in frequencies to another. While this can come as an advantage for some signals, where linear relationships dominate, more complex relationships might get ignored when using this method.

To illustrate the result of comparing time series using a each different method, a sigmoidal function with two different time delays was compared using each method. The sigmoidal function is a time series defined as:

$$f(t) = \frac{1}{1 + e^{-t}} \tag{5.6}$$

and is one of many examples of time series that aim to be analyzed using DAQBroker. Figure REF shows two sigmoidal time series over a period of 500 seconds and both series are delayed by a period of 200 seconds as well as a table with the resulting similarity value for each method.

As expected a certain small correlation coefficient is found. When moving on to the maximum of the correlation function, a larger value is found at a lag very close to the period of lag of both series. One surprising factor is the value of the DTW similarity is quite small when compared to the previous values. This is hopefully due to a missing normalization factor related to the size of the time series and hopefully some sort of order is restored when analysing similarity matrix. The coherence value as expected is one of the largest values, since most of the frequencies that define one function are present on the other and as such it is simple to attribute similarity.




| | |
|---|---|
| Pearson | 0.3242 |
| Max. Cross-corrleation | (0.46223, [195]) |
| DTW | 0.03934 |
| Coherence | 0.70223 |

*Figure 5.26: Sigmoidal time series with different delays (left) and result of similarity measurements using the methods described (right). For the maximum cross correlation similarity, the delay for which the correlation is maximum is also shown.*

### 5.2.4 Relevant period detection

With a relevant subset of time series and with all relevant relationships highlighted in a graphical visualization, a look at the time series itself is warranted to provide users with the most relevant periods of activity in the time series.

Since most time series originate from different instruments, no particular model can be applied to all the time series to efficiently evaluate their behavior. Similarly, finding a unique model for all single time series for each experimental stage would be too cumbersome to provide as a feature to users. This section introduces a simple extension of the univariate time series event finding algorithm presented in 5.1.3.

As stated earlier in this section, all time series are regularized in time to contain the same amount of points as the most granular time series, in order to retain the maximum information possible. Thus it is possible to treat time as a single series as it is the same for all series.

1. The time series is divided into a number of user-defined evenly sized sub intervals. These are the time periods to be studied in terms of *relevance*.
2. In order to define the relevance, the symbolic wavelet partitioning method is applied to each time series in $\overline{\mathbf{X}}$ at each period.
3. Each interval is further divided into 10 smaller subintervals, the first subinterval is used as the test example for each time series and the rest of the subintervals are tested for events detection.
4. For each interval the number of subintervals that deviate from the test subinterval by an angle greater than a user-supplied angle are considered events.

For each interval the total number of events over all time series is calculated and the period with most events found is considered the most relevant.

It is simple to show users which period is the most relevant by shading the areas of the time series plot as a function of the number of events found, as shown in figure 5.27.

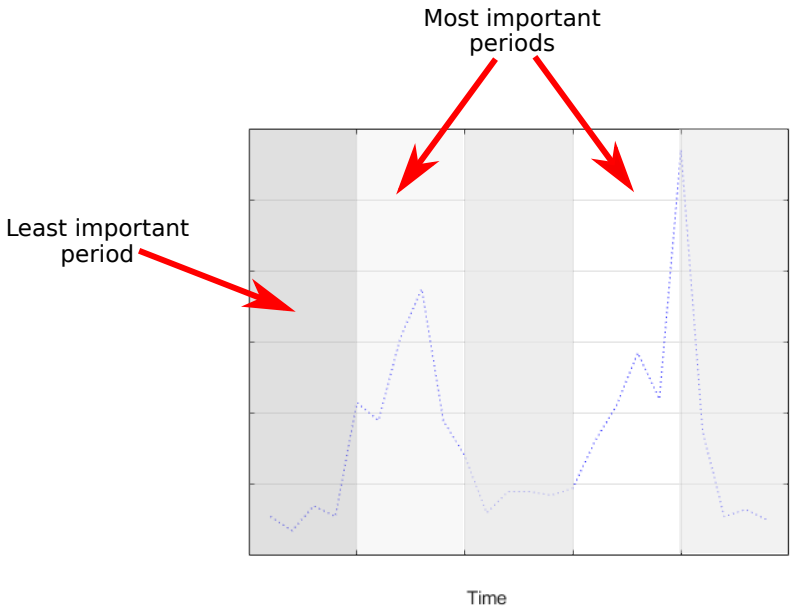



*Figure 5.27: Example of visual ranking of periods in time series plot. More shaded areas are of less interest and areas with excursions are the areas of larger interest. This is the expected behavior of the multivariate time series ranking of relevant periods.*

### 5.2.5 Application to CLOUD data

The validity of the methods presented in the previous section were tested with data from the CLOUD experiment. A full list of all channels used for this analysis can be found in tables B.1 and B.2. A total of 88 channels are screened for data and, in the case there is no data for a specific time period, the corresponding channels are discarded from the analysis. 52% of the total number of CLOUD instruments integrate in this analysis but only around 1% of the total number of channels is represented. This small representation of channels is placated with the knowledge that some channels are composite channels with large amounts of channels for a single time series. If one takes into account the individual channels used in the composite channels, the channel representations increases to around 30%. It must also be considered that most of the data channels in this specific CLOUD campaign originate from the Light Spectrometer, which samples the light spectrum at a sub-nanometer resolution (from 185 to 876 nm) resulting in over 44% of the total number of channels. The non-represented channels in the whole data set are either channels used to monitor instrument status or for monitoring parameters that do not play a part in the selected time periods. A reduced set of time series is also considered to test whether reducing the main data set provides more scientifically relevant insights. This reduced time series set is shown in tables B.3 and B.4.

Similarly to the bulk of experimental stages in the 2017 CLOUD campaign, one specific particle generation run (1962) was chosen to apply this method. The analysis of this method is performed on 5 different stages of the aforementioned run, which illustrate a typical set of experimental stages for these types of runs. This particular run tested the influence of organic components in the formation of atmospheric particles under low sulphuric acid concentrations, each stage is now described and summarized in table 5.1:

- In a preparation stage, the precursor gases' concentrations are set to predetermined values, specifically,

sulphuric acid and $SO_2$ are set to low values and $\alpha$-pinene and ozone concentrations are set an initial value.

- The first stage (1962.01) starts by turning on the CLOUD chamber's ultraviolet lights, which induce ozonolysis in the chamber's atmosphere and leads to the formation of sulphuric acid. This stage is preformed under the application of an electrical field over the chamber to screen out charged particles, thus studying only neutral chemical pathways.

- The second stage (1962.02) consists in the removal of the applied electrical field, which introduces charged particles to the chamber's atmosphere and should result in an increased rate of particle formation since new chemical pathways for particle formation are introduced.

- The third stage (1962.03) is a transition stage, where the concentration of $\alpha$-pinene is increased to prepare for a new particle generation event. During this time, the electrical field is re-applied in preparation for another neutral stage and a set of mixing fans is set to maximum value to remove already created particles from the chamber.

- The fourth stage (1962.04) is the same as the first stage, except a larger concentration of $\alpha$-pinene is available. This stage starts with turning down the mixing fan speed to allow particles generated to be read by the instruments.

- The fifth stage (1962.05) is the same as the second stage at a higher concentration of $\alpha$-pinene. This stage starts similarly to the second stage, by turning off the electrical field, introducing charged particle generation pathways.

.

In order not to burden the reader with large amounts of results, only the analysis of one stage is described in this section. The analysis of other stages can be accessed in appendix B.

For all runs, the user-defined parameters are kept constant to produce comparable results. For the highlighting of important time series, the PCA explained variance is kept at 97%. Finally the limiting angle for event detection is kept at 45° ($\pi/4$) and the time period is divided into 4 sub-intervals to test for event detection relevance.

*Table 5.1: Stages of particle generation run studied via multivariate analysis and respective summaries*

| Stage | Description | Expected behavior |
|---|---|---|
| 1962.01 | Neutral AP 150 | Organic particles dependent on $\alpha$-pinene (AP) are created, this gives rise to eventual particle creation |
| 1962.02 | Charged AP 150 | Neutral fields switched off, larger particle growth should be observed |
| 1962.03 | Increase AP to 300 | Period for increasing $\alpha$-pinene and cleaning of the chamber |
| 1962.04 | Neutral AP 300 | Larger values of $\alpha$-pinene should increase particle generation, even at a neutral stage |
| 1962.05 | Charged AP 300 | Neutral fields switched off again, even larger particle growth should be observed |

#### 5.2.5.1 Run 1962.01 - Relevant Time Series Highlighting

Figure 5.28 shows the most relevant time series highlighted with the method described in 5.2.2.

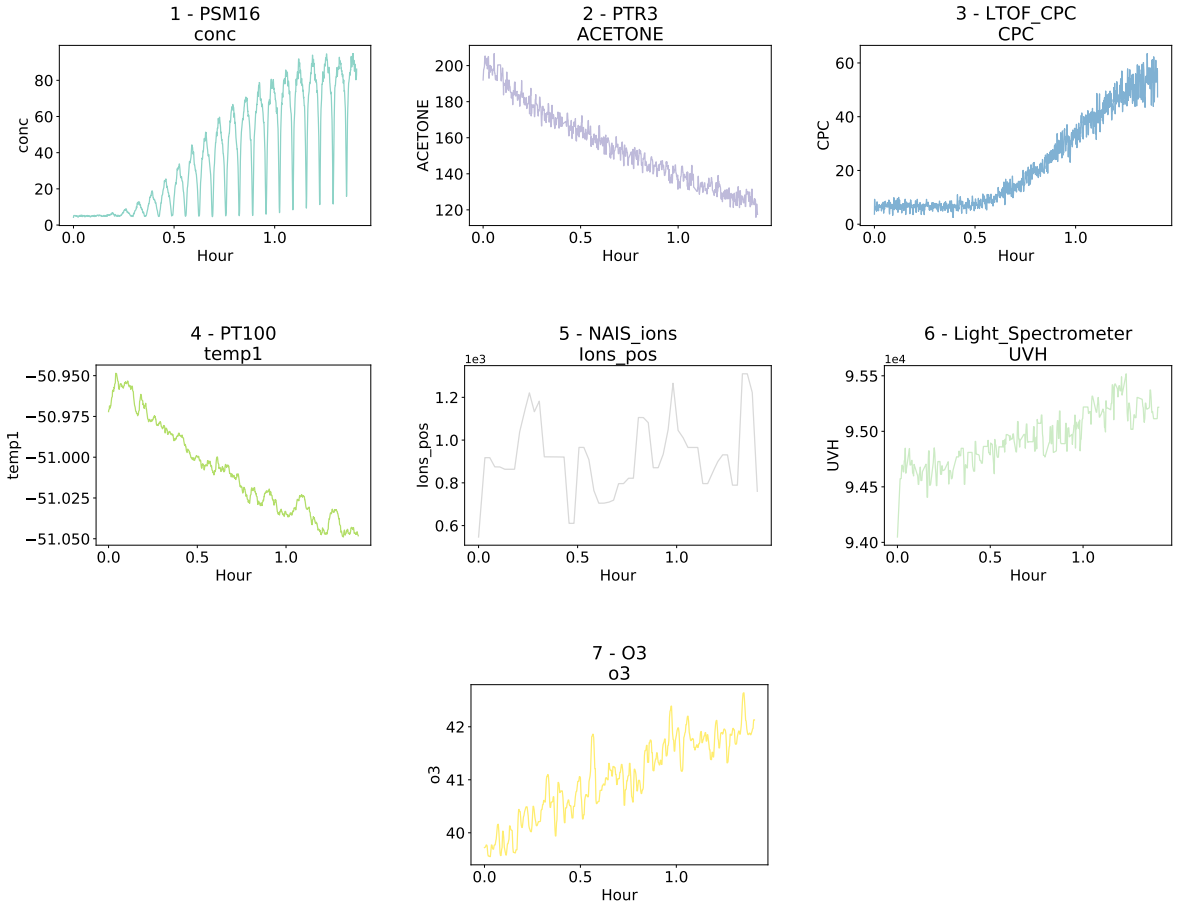



*Figure 5.28: Most relevant time series for CLOUD experimental run 1962.01. The compared time series are shown in their original sampled values, before normalization to give the reader an idea of the absolute change of values. This visualization can be shown to users to allow for a more detailed study of the results.*

The main advantage of this method is to reduce the time series set, in this case resulting in a 90% reduction from the initial set of time series while retaining most (97%) of the frequency variance of the whole set. Also of note is the fact that 2 time series highlighted for this experimental run, corresponding to instruments for different particle size measurements. This is in line with the focus of the run in creating particles. One can go further with this analysis by noting that the particle measuring channels (*LTOF CPC* and *PSM16 conc*) measure only small particles and the fact that this particular run generated uncharged particles with low concentrations of precursors, meaning that most clusters would not have grown to detectable sizes. Also of note the highlighting of two chemical species' (Acetone and $O_3$) concentration, with an increase in the first and a slight decrease in the second, hinting that these chemical species or their precursors are relevant in particle generation.

Highlighting of the *NAIS_ions - Ions_pos* channel, measuring the concentration of positive values, is unexpected.

This time series is clearly an important channel to consider but since this run is neutral, the highlighting of charged particle channels is not expected. The reason for the selection of this channel is due to the noise at low values (compare with the charged run in figure B.1, which features values at a higher order of magnitude) that is not fully removed during the pre-processing state. This makes the noise frequencies be highlighted and is thus considered relevant. This issue can be resolved by another auxiliary step that would allow users to apply a specific model for noise removal on their time series, in the pre-processing stage of the method.

Figure 5.29 shows the most relevant time series highlighted when using the reduced time series data set. What is most immediately apparent is the number of highlighted time series, which reduced by 3. This means that a significant amount of information was removed from the main set, since less time series are required to explain the variance of the reduced set. While less channels are required, it is unclear whether the new plots show new information, since two of the highlighted time series (*Fan speed1* and *PTR3 Acetic Acid*) seem to be measuring noisy measurements.

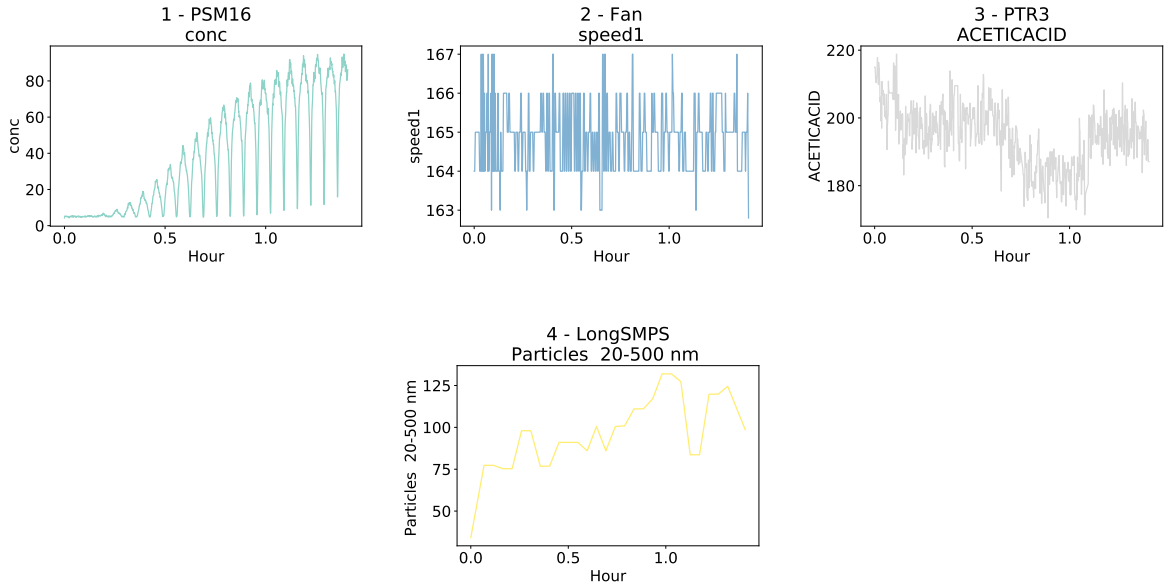



*Figure 5.29: Most relevant time series for CLOUD experimental run 1962.01 using the reduced set of time series. The compared time series are shown in their original sampled values, before normalization to give the reader an idea of the absolute change of values. This visualization can be shown to users to allow for a more detailed study of the results.*

#### 5.2.5.2 Run 1962.01 - Time Series Relationships

To study the relationships between time series, two sets of channels were used for this particular part of the process. Firstly we will consider the initial set of channels defined in tables B.1 and B.2. However, there is a possibility that some channels chosen from the base set, while not relevant to the scientific analysis of the stage being preformed and due to the nature of their statistics, be it improper pre-processing or sampling below the instrument's detection limit, can cause the resulting network graph to become skewed toward unexpected relationships. Thus, the initial set of channels was pruned to consider only channels relevant to the experimental run preformed. While this should not be the norm when using of DAQBroker, since the whole focus of these methods is to preform exploratory analysis on the collected data, relying solely on statistical methods to provide results for users is also not practical in a real world scenario and thus users should be allowed to choose subsets of data channels on which to apply these exploratory methods. The resulting subset of channels was chosen to reflect the fact that this is a low sulphuric acid particle generation experiment, where the levels of $\alpha$-pinene are gradually increased at the run's half point and several stages of neutral and charged particle generation attempts are made, thus eliminating several channels, due to a combination of non-relevant physics/chemistry, measurements below limit of detection and equivalent methods of detection, all of which could cause spurious relationships to occur. It is also worth noting that while a change in the set of channels used does indeed change the results of the previously made PCA, it merely affects the number of ranks found, the underlying conclusions and highlighted channels for both sets remain fundamentally the same except that less channels are highlighted.

##### 5.2.5.2.1 Full channel set

**5.2.5.2.1.1 Pearson Correlation Similarity** Figure 5.30 shows the Pearson correlation network graph of the channels during the experimental run. The strength of each pair of channels is visualized by a reversed grey scale (0% is light grey and 100% black). Node sizes are related to the degree of eigenvector centrality, thus highlighting the most influential nodes in terms of the respective network. To ease the identification of nodes in figure 5.30, table 5.2 ranks each node's eigenvector centrality in descending order, thus the top ranked nodes will correspond to the largest nodes in figure 5.30. The color of each node places said node in one of the several possible communities found after applying the Louvain algorithm, described in section 2.2.2.2.

It is clear from table 5.2 that the most influential nodes are those of particle concentration measurements, followed by light spectrometry measurements with a small contribution of chemical measurements in the ozone times series. These highlighted relationships are all expected, since the goal of this stage was to use specific wavelengths to generate ozonolysis in the chamber, thus driving up the particle generation rate. It can also be shown in that for other runs, this method can also generate predictable results (see B.2)

Regarding the three communities found in either graph, no direct explanation for the separation is immediately found as each set of nodes for each network graph consists of both chemical and physical measurements. It ap-

**Stage 1962.01**
3 communities found

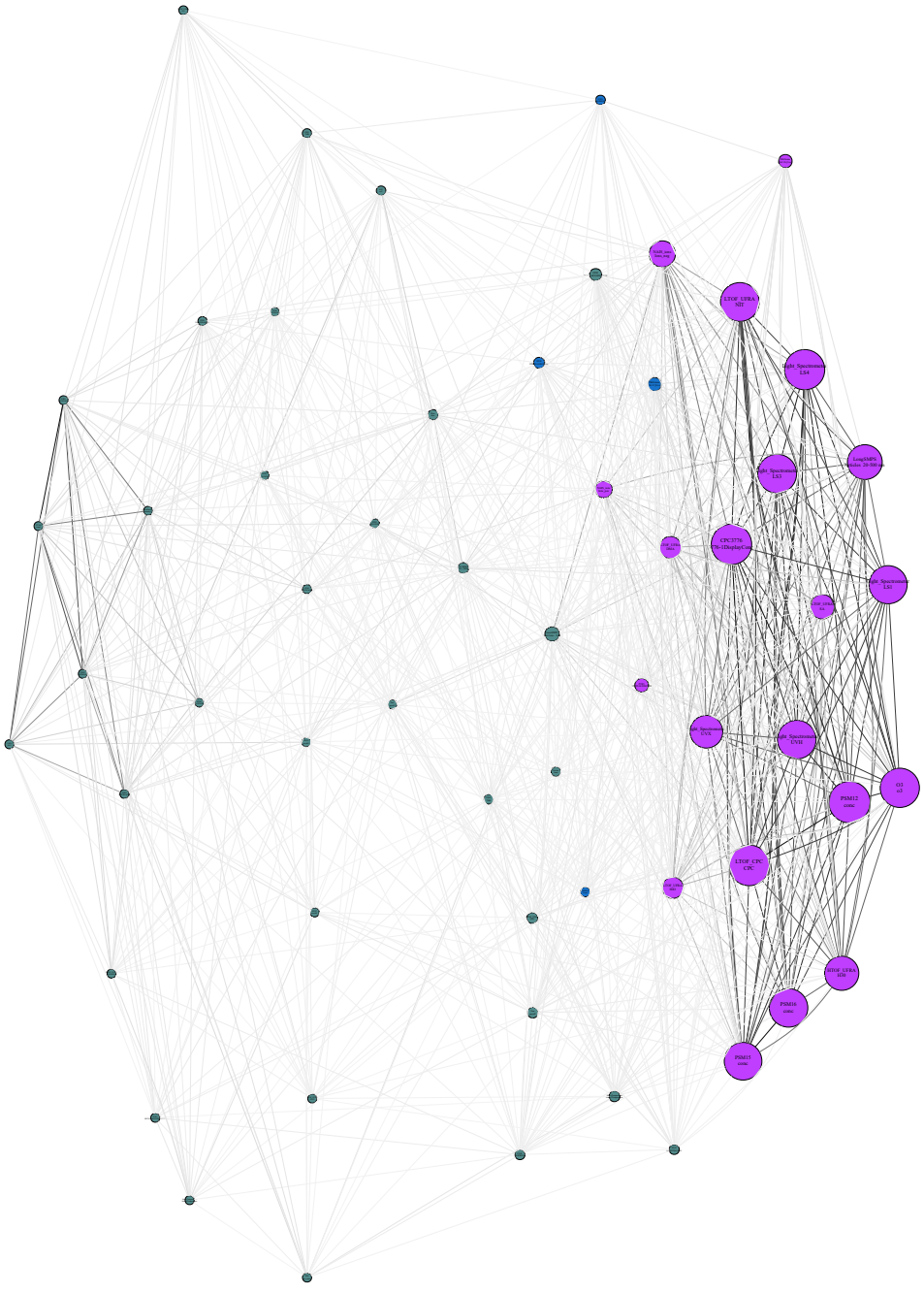


*Figure 5.30: Pearson correlation network graph for the CLOUD experimental run 1962.01. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table 5.2: Eigenvector ($\epsilon$) centrality node ranking for run 1962.01 using the Pearson correlation as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | PSM12 | conc | 0.2841 |
| 2 | LTOF_CPC | CPC | 0.277 |
| 3 | CPC3776 | 3776-1DisplayConc. | 0.2754 |
| 4 | Light_Spectrometer | LS4 | 0.2731 |
| 5 | O3 | o3 | 0.2645 |
| 6 | Light_Spectrometer | LS3 | 0.2614 |
| 7 | LTOF_UFRA | NIT | 0.2603 |
| 8 | Light_Spectrometer | UVH | 0.2596 |
| 9 | PSM16 | conc | 0.2583 |
| 10 | Light_Spectrometer | LS1 | 0.2548 |
| 11 | PSM15 | conc | 0.2507 |
| 12 | LongSMPS | Particles 20-500 nm | 0.2258 |
| 13 | HTOF_UFRA | H30 | 0.2204 |
| 14 | Light_Spectrometer | UVX | 0.2078 |
| 15 | NAIS_ions | Ions_neg | 0.1513 |
| 16 | LTOF_UFRA | SA | 0.1362 |
| 17 | LTOF_UFRA | DMA | 0.1149 |
| 18 | LTOF_UFRA | HIO | 0.1076 |
| 19 | NAIS_ions | Ions_pos | 0.0701 |
| 20 | nanoSMPS | Total Conc. #/cm | 0.048 |
| 21 | DMAtrain | Ch1 C [#/cc] | 0.0415 |
| 22 | PTR3 | PINONALDEHYDE | 0.0407 |
| 23 | DMAtrain | Ch0 Ch [#/cc] | 0.0381 |
| 24 | STOF | BETACARYOPHYLENE | 0.0278 |
| 25 | STOF | PINONALDEHYDE | 0.0182 |
| 26 | PICARRO | NH3_2MIN | 0.0177 |
| 27 | HTOF_UFRA | NH3 | 0.0176 |
| 28 | SabreTemperature | Sabre3Temperature | 0.0159 |
| 29 | Fan | speed2 | 0.0157 |
| 30 | Dew | dew | 0.0122 |
| 31 | STOF | ISOPRENE | 0.0099 |
| 32 | STOF | NAPHTHALENE | 0.009 |
| 33 | nRDMA | Count | 0.0079 |
| 34 | STOF | TMB | 0.0076 |
| 35 | PTR3 | ISOPRENE | 0.0063 |
| 36 | Scalers | brust | 0.0058 |
| 37 | SO2 | so2 | 0.0044 |
| 38 | PTR3 | TOLUENE | 0.0043 |
| 39 | Fan | speed1 | 0.0025 |
| 40 | PTR3 | TMB | 0.0022 |
| 41 | HVFC | mvolt1 | 0.0021 |
| 42 | STOF | ACETICACID | 0.0019 |
| 43 | HVFC | mvolt2 | 0.0017 |
| 44 | PTR3 | NAPHTHALENE | 0.0014 |
| 45 | HTOF_UFRA | H7O3 | 0.0012 |
| 46 | TDL | sat_water | 0.0011 |
| 47 | PTR3 | ACETICACID | 0.0011 |
| 48 | PhotoDiode12 | intensity2 | 0.0007 |
| 49 | STOF | APINENE | 0.0007 |
| 50 | STOF | TOLUENE | 0.0006 |
| 51 | PT100 | temp1 | 0.0006 |
| 52 | PTR3 | ACETONE | 0.0005 |
| 53 | PTR3 | APINENE | 0.0005 |
| 54 | HTOF_UFRA | O2 | 0.0004 |
| 55 | HTOF_UFRA | H3OH2O | 0.0003 |
| 56 | PTR3 | BETACARYOPHYLENE | 0.0003 |
| 57 | CAPS | NO2 | 0.0003 |
| 58 | STOF | ACETONE | 0.0002 |
| 59 | SabreTemperature | Sabre4Temperature | 0.0002 |

pears that each existing community seems to group time series with similar values of eigenvalue centrality and thus the coloring of each node according to its community further inform the user of the importance of each time series. When compared with other graphs, this graph shows the most spread in nodes, easing the identification of communities of nodes.

While important information can be retrieved from the network graph, that can shed light on the behavior of the studied multivariate time series, it also becomes clear that a static representation of the relationships between each pair of time series becomes not very useful for the larger number of variables included in the multivariate time series. Introducing a strength threshold for a statistically significant relationship would provide more manageable static visualizations. Alternatively, since the end goal of this method is to be implemented in DAQBroker, a more interesting solution to this problem is to consider this method of visualization in a more dynamic sense. This second approach is developed in 5.2.6.

**5.2.5.2.1.2 Cross-correlation Similarity** Figure 5.30 shows the maximum cross-correlation network graph of the channels during the experimental run. To calculate the pairwise relationship between time series ($s^{XY}_{max\_corr}$), the following expression is used:

$$s^{XY}_{max\_corr} = |\max_k \mathscr{R}_{XY}(k)| \tag{5.7}$$

Where $\mathscr{R}_{XY}(k)$ is the cross-correlation function between time series $X$ and $Y$, given by equation 2.17, which is a function of the delay between time series. Compared to the Pearson correlation, this method should find differences when there is significant delay between time series that otherwise have similar linear relationships. Table 5.3 again is used to help identify the nodes in the map in the same way as in the previous section.

As can be seen from analysing table 5.3, very similar results are found, with one single exception of the highlighting of a chemical channel measuring mass in range of nitrate compounds, which was already prominent in table 5.2 (up by only 2 ranks). Inspecting the individual channel, there is indeed a change in the values which cannot not be immediately attributed to noise in the instrument, but can be related to a change in a measurement state of the instrument, the instrument operator should be contacted to highlight the source of this change. It can be seen how, considering any possible delay, some channels can become related to others in a spurious way, as is the case here, however, this method should not be discarded as relevant information can be retrieved from introducing the possibility of delay analysis. It is also worth noting that the graph produced in figure 5.31 is now much more compact, as the values of maximum cross-correlation are overall higher. Again, three communities are found, all of which seem once again to group the nodes in terms of the eigenvector centrality, once again in aiding the grouping of nodes, which is more complex in this case.

**5.2.5.2.1.3 DTW Similarity** Figure 5.32 shows the DTW network graph of the channels during the experimental run. The similarity between timeseries in this case $s^{XY}_{DTW}$ is found by calculating the DTW distance between the series ($d^{XY}_{DTW}$) and applying equation 2.13:

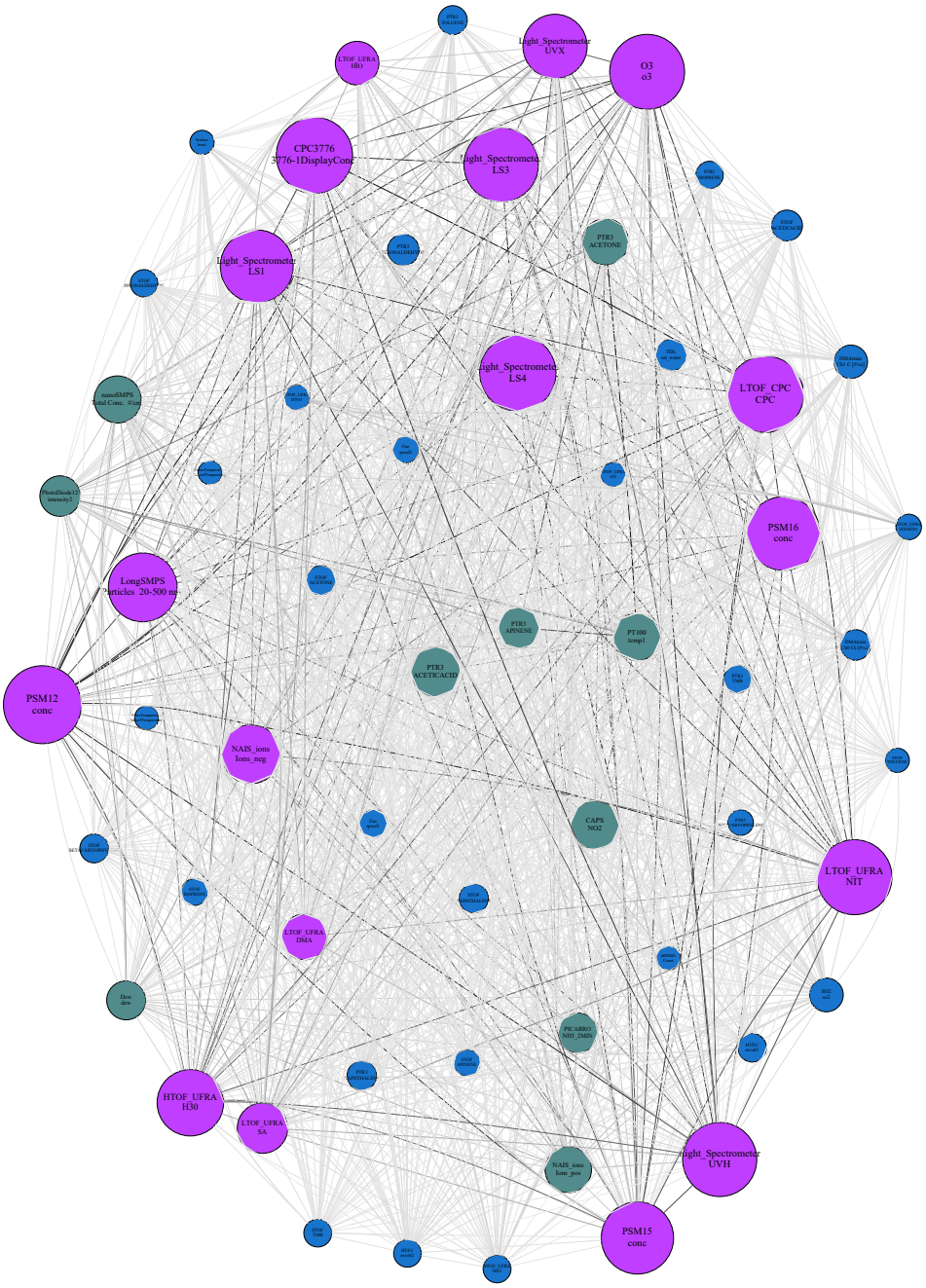



*Figure 5.31: Maximum cross-correlation network graph for the CLOUD experimental run 1962.01. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table 5.3: Eigenvector ($\epsilon$) centrality node ranking for run 1962.01 using the maximum cross-correlation as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | PSM12 | conc | 0.2431 |
| 2 | LTOF_CPC | CPC | 0.2376 |
| 3 | Light_Spectrometer | LS4 | 0.2371 |
| 4 | CPC3776 | 3776-1DisplayConc. | 0.2368 |
| 5 | LTOF_UFRA | NIT | 0.2318 |
| 6 | O3 | o3 | 0.2314 |
| 7 | Light_Spectrometer | LS3 | 0.2299 |
| 8 | Light_Spectrometer | UVH | 0.2286 |
| 9 | PSM16 | conc | 0.2273 |
| 10 | Light_Spectrometer | LS1 | 0.2238 |
| 11 | PSM15 | conc | 0.2214 |
| 12 | LongSMPS | Particles 20-500 nm | 0.2082 |
| 13 | HTOF_UFRA | H30 | 0.1989 |
| 14 | Light_Spectrometer | UVX | 0.19 |
| 15 | NAIS_ions | Ions_neg | 0.1708 |
| 16 | LTOF_UFRA | SA | 0.1372 |
| 17 | PTR3 | ACETICACID | 0.1346 |
| 18 | CAPS | NO2 | 0.1302 |
| 19 | PT100 | temp1 | 0.1254 |
| 20 | nanoSMPS | Total Conc. #/cm | 0.1244 |
| 21 | PTR3 | ACETONE | 0.1216 |
| 22 | NAIS_ions | Ions_pos | 0.1215 |
| 23 | LTOF_UFRA | DMA | 0.1197 |
| 24 | LTOF_UFRA | HIO | 0.111 |
| 25 | PTR3 | APINENE | 0.1017 |
| 26 | PhotoDiode12 | intensity2 | 0.099 |
| 27 | PICARRO | NH3_2MIN | 0.0969 |
| 28 | Dew | dew | 0.093 |
| 29 | SO2 | so2 | 0.0726 |
| 30 | DMAtrain | Ch1 C [#/cc] | 0.0715 |
| 31 | TDL | sat_water | 0.0645 |
| 32 | PTR3 | PINONALDEHYDE | 0.0641 |
| 33 | STOF | ACETICACID | 0.0615 |
| 34 | DMAtrain | Ch0 Ch [#/cc] | 0.0604 |
| 35 | STOF | ACETONE | 0.0573 |
| 36 | PTR3 | NAPHTHALENE | 0.0565 |
| 37 | STOF | NAPHTHALENE | 0.0557 |
| 38 | PTR3 | TOLUENE | 0.0546 |
| 39 | STOF | BETACARYOPHYLENE | 0.0539 |
| 40 | HVFC | mvolt1 | 0.053 |
| 41 | Fan | speed1 | 0.0506 |
| 42 | PTR3 | TMB | 0.0504 |
| 43 | STOF | TMB | 0.05 |
| 44 | HTOF_UFRA | NH3 | 0.0499 |
| 45 | HVFC | mvolt2 | 0.0491 |
| 46 | STOF | PINONALDEHYDE | 0.0483 |
| 47 | PTR3 | ISOPRENE | 0.0482 |
| 48 | STOF | ISOPRENE | 0.0461 |
| 49 | Fan | speed2 | 0.0461 |
| 50 | STOF | APINENE | 0.0456 |
| 51 | PTR3 | BETACARYOPHYLENE | 0.0417 |
| 52 | HTOF_UFRA | H3OH2O | 0.0411 |
| 53 | HTOF_UFRA | H7O3 | 0.041 |
| 54 | HTOF_UFRA | O2 | 0.0383 |
| 55 | nRDMA | Count | 0.0363 |
| 56 | STOF | TOLUENE | 0.0361 |
| 57 | Scalers | brust | 0.0359 |
| 58 | SabreTemperature | Sabre3Temperature | 0.0351 |
| 59 | SabreTemperature | Sabre4Temperature | 0.0348 |

$$s_{DTW}^{XY} = \frac{1}{1 + d_{DTW}^{XY}} \tag{5.8}$$

Compared to the previous methods, this method allows more freedom in distorting the time axis to find a better relationship and is thus suitable when instruments have very different sampling frequencies. Due to the pre-processing stage, it is not expected for this method to show any improvement. Table 5.4 again is used to help identify the nodes in the map in the same way as in the previous section.

From the analysis of table 5.4, a striking difference is found from the previous methods. While some particle concentration measurements are highlighted, they are far from being the most popular and unexpectedly, several chemical channels measuring non-relevant chemical species (specifically $NH_3$) are highlighted, which belong to the same instrument. From analysing the individual time series, it is obvious that these time series are measuring only noise and the increased freedom to distort the time axis to find relationships works against finding any relevant information, and highlights only noise which unsurprisingly comes from the same instrument. This result highlights the need for a user to review the channels being analysed and iterate through the set of channels to be analysed and find problematic channels in the analysis. Also of note is the strengths of the relationships between channels using this method, which are highlighted by the colors of the connections between nodes in figure 5.32, which are very loosely connected, this results in a more fairly homogeneous graph where it becomes harder to distinguish between relevant nodes, hinting that the DTW similarity is possibly not suited for use as a similarity measurement for the current set of channels. This limitation is probably contributing to the unfavorable results found in table 5.4. The Louvain algorithm, however is able to identify 3 distinct communities, which once again are related to their respective eigenvector centrality.

**5.2.5.2.1.4 Coherence Similarity** Figure 5.33 shows the coherence network graph of the channels during the experimental run. The similarity between timeseries in this case $s_{coh}^{XY}$ is found by calculating the average coherence between the series:

$$s_{coh}^{XY} = \overline{\mathscr{C}_{XY}(\omega)} \tag{5.9}$$

Where $\overline{Z}$ represents an average over all available frequencies and $\mathscr{C}_{XY}$ represents the coherence between time series $X$ and $Y$ and is given by equation 2.20, where each element of the calculation is estimated by the result derived in equation 2.24.

Compared to the previously used methods, this one compares time series in a completely different domain, that of frequency and thus the concept of delays between time series does not exist. Table 5.5 again is used to help identify the nodes in the map in the same way as in the previous section.

Analysis of table 5.5 shows that, much like the DTW similarity, several channels are highlighted which do not possess any relevant features. The same chemical species highlighted in the DTW graph are once again highlighted. The signals from the high voltage field cage are also highlighted which upon closer inspection show very small oscillations, which could be a result of improper pre-processing of the data. Unlike all other graphs,

# Stage 1962.01

3 communities found

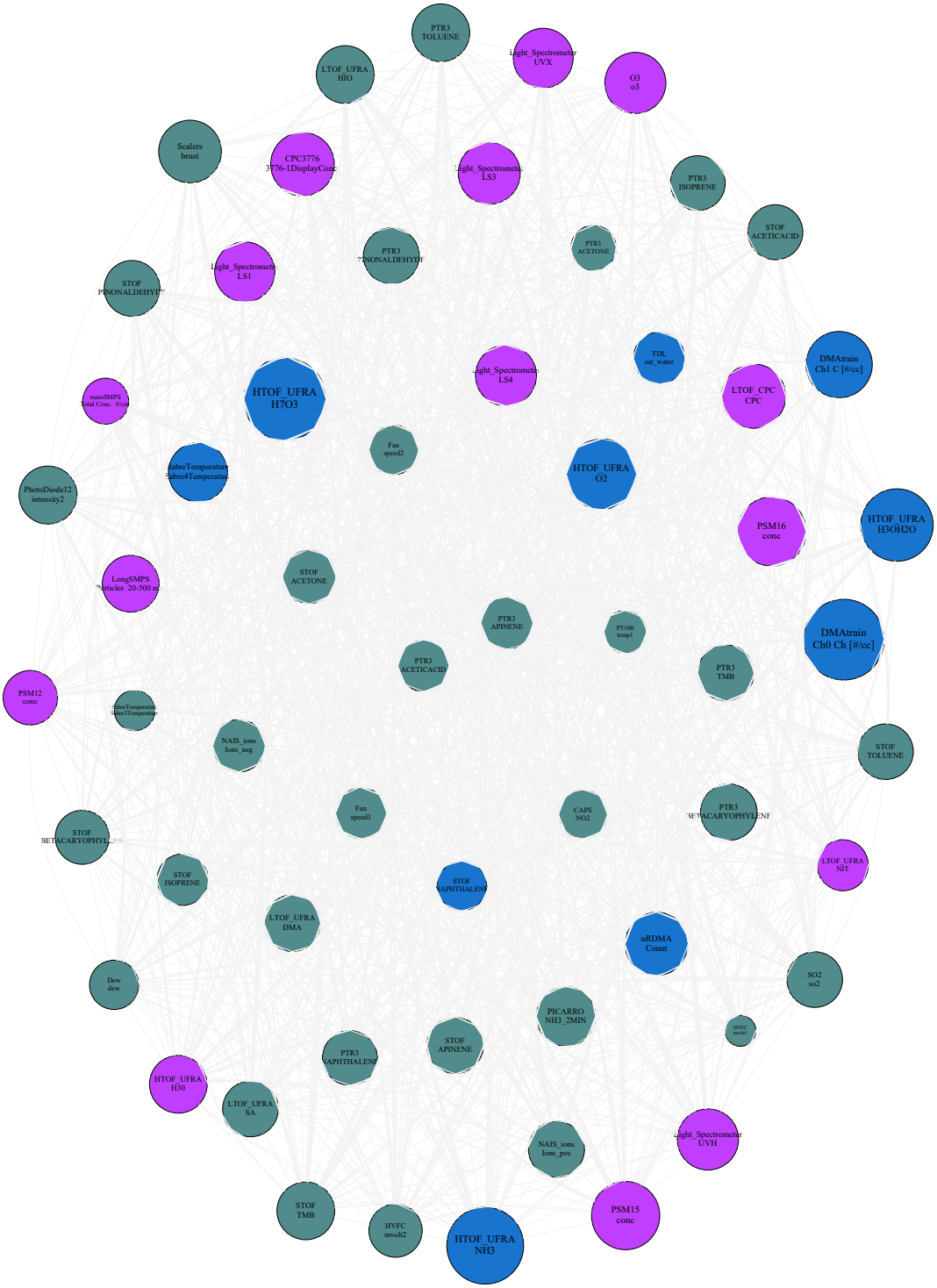



*Figure 5.32: DTW network graph for the CLOUD experimental run 1962.01. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table 5.4: Eigenvector ($\epsilon$) centrality node ranking for run 1962.01 using DTW as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | HTOF_UFRA | H7O3 | 0.2029 |
| 2 | DMAtrain | Ch0 Ch [#/cc] | 0.1969 |
| 3 | HTOF_UFRA | NH3 | 0.1854 |
| 4 | HTOF_UFRA | H3OH2O | 0.1723 |
| 5 | HTOF_UFRA | O2 | 0.1671 |
| 6 | PSM16 | conc | 0.1627 |
| 7 | PSM15 | conc | 0.1601 |
| 8 | DMAtrain | Ch1 C [#/cc] | 0.1542 |
| 9 | LTOF_CPC | CPC | 0.1482 |
| 10 | CPC3776 | 3776-1DisplayConc. | 0.1465 |
| 11 | nRDMA | Count | 0.1457 |
| 12 | Scalers | brust | 0.1436 |
| 13 | Light_Spectrometer | LS3 | 0.1416 |
| 14 | Light_Spectrometer | UVH | 0.1391 |
| 15 | O3 | o3 | 0.1389 |
| 16 | Light_Spectrometer | LS4 | 0.1388 |
| 17 | Light_Spectrometer | LS1 | 0.1361 |
| 18 | Light_Spectrometer | UVX | 0.1351 |
| 19 | PICARRO | NH3_2MIN | 0.1343 |
| 20 | SabreTemperature | Sabre4Temperature | 0.1331 |
| 21 | PhotoDiode12 | intensity2 | 0.1297 |
| 22 | PTR3 | TOLUENE | 0.129 |
| 23 | LTOF_UFRA | HIO | 0.129 |
| 24 | STOF | TMB | 0.1289 |
| 25 | HTOF_UFRA | H30 | 0.1287 |
| 26 | LongSMPS | Particles 20-500 nm | 0.1261 |
| 27 | LTOF_UFRA | DMA | 0.1257 |
| 28 | STOF | APINENE | 0.1256 |
| 29 | PTR3 | BETACARYOPHYLENE | 0.1254 |
| 30 | PTR3 | TMB | 0.1248 |
| 31 | NAIS_ions | Ions_pos | 0.1239 |
| 32 | PTR3 | PINONALDEHYDE | 0.1237 |
| 33 | STOF | PINONALDEHYDE | 0.1231 |
| 34 | LTOF_UFRA | SA | 0.123 |
| 35 | SO2 | so2 | 0.1227 |
| 36 | STOF | TOLUENE | 0.1223 |
| 37 | STOF | ACETICACID | 0.1208 |
| 38 | PTR3 | NAPHTHALENE | 0.1206 |
| 39 | PTR3 | ISOPRENE | 0.1204 |
| 40 | PSM12 | conc | 0.1191 |
| 41 | STOF | BETACARYOPHYLENE | 0.1178 |
| 42 | STOF | ACETONE | 0.1174 |
| 43 | HVFC | mvolt2 | 0.1157 |
| 44 | NAIS_ions | Ions_neg | 0.1139 |
| 45 | PTR3 | APINENE | 0.1129 |
| 46 | TDL | sat_water | 0.1124 |
| 47 | PTR3 | ACETICACID | 0.1108 |
| 48 | STOF | ISOPRENE | 0.1105 |
| 49 | Fan | speed1 | 0.1102 |
| 50 | LTOF_UFRA | NIT | 0.1092 |
| 51 | Fan | speed2 | 0.1058 |
| 52 | STOF | NAPHTHALENE | 0.1053 |
| 53 | Dew | dew | 0.1039 |
| 54 | CAPS | NO2 | 0.1032 |
| 55 | nanoSMPS | Total Conc. #/cm | 0.0941 |
| 56 | PTR3 | ACETONE | 0.0924 |
| 57 | PT100 | temp1 | 0.0847 |
| 58 | SabreTemperature | Sabre3Temperature | 0.0782 |
| 59 | HVFC | mvolt1 | 0.0495 |

**Stage 1962.01**

2 communities found

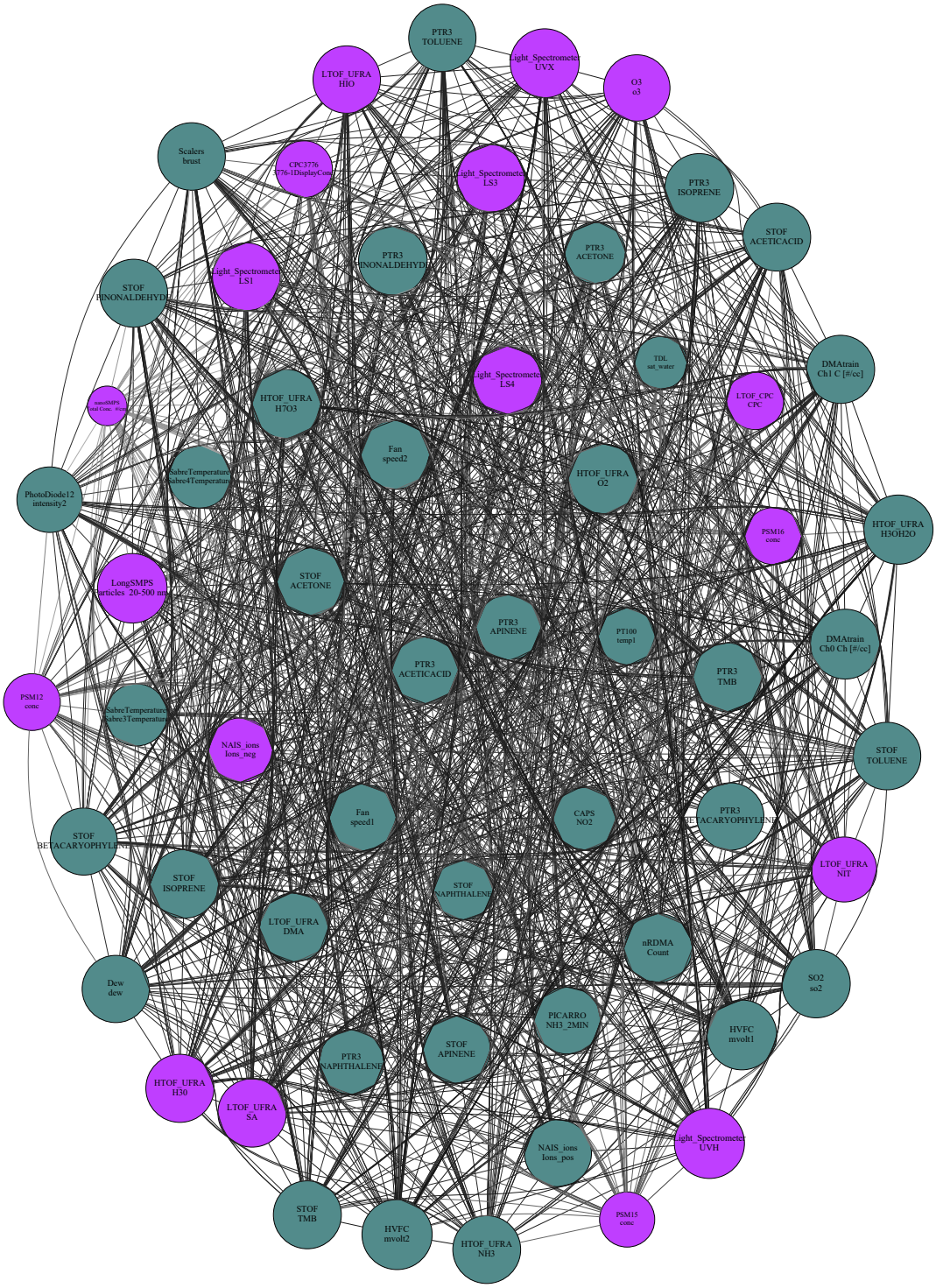



*Figure 5.33: Coherence network graph for the CLOUD experimental run 1962.01. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table 5.5: Eigenvector ($\epsilon$) centrality node ranking for run 1962.01 using the coherence as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | HVFC | mvolt2 | 0.1403 |
| 2 | Light_Spectrometer | UVH | 0.1401 |
| 3 | HVFC | mvolt1 | 0.1394 |
| 4 | DMAtrain | Ch0 Ch [#/cc] | 0.1393 |
| 5 | HTOF_UFRA | O2 | 0.1393 |
| 6 | LongSMPS | Particles 20-500 nm | 0.1387 |
| 7 | HTOF_UFRA | H3OH2O | 0.138 |
| 8 | HTOF_UFRA | H7O3 | 0.1379 |
| 9 | Fan | speed2 | 0.1375 |
| 10 | PTR3 | ISOPRENE | 0.1374 |
| 11 | LTOF_UFRA | DMA | 0.1373 |
| 12 | Light_Spectrometer | UVX | 0.1373 |
| 13 | nRDMA | Count | 0.137 |
| 14 | PTR3 | TMB | 0.1369 |
| 15 | STOF | ISOPRENE | 0.1369 |
| 16 | PTR3 | PINONALDEHYDE | 0.1367 |
| 17 | Light_Spectrometer | LS4 | 0.1363 |
| 18 | LTOF_UFRA | SA | 0.1362 |
| 19 | SO2 | so2 | 0.1358 |
| 20 | DMAtrain | Ch1 C [#/cc] | 0.1355 |
| 21 | HTOF_UFRA | H30 | 0.1355 |
| 22 | STOF | PINONALDEHYDE | 0.1352 |
| 23 | STOF | TMB | 0.1352 |
| 24 | HTOF_UFRA | NH3 | 0.1349 |
| 25 | STOF | ACETICACID | 0.1349 |
| 26 | Light_Spectrometer | LS1 | 0.1349 |
| 27 | STOF | ACETONE | 0.1345 |
| 28 | STOF | TOLUENE | 0.1343 |
| 29 | Scalers | brust | 0.1343 |
| 30 | LTOF_UFRA | HIO | 0.134 |
| 31 | Light_Spectrometer | LS3 | 0.134 |
| 32 | STOF | APINENE | 0.1339 |
| 33 | PTR3 | BETACARYOPHYLENE | 0.1337 |
| 34 | PTR3 | TOLUENE | 0.1336 |
| 35 | Dew | dew | 0.1334 |
| 36 | Fan | speed1 | 0.1333 |
| 37 | STOF | BETACARYOPHYLENE | 0.1331 |
| 38 | NAIS_ions | Ions_pos | 0.1331 |
| 39 | PICARRO | NH3_2MIN | 0.1329 |
| 40 | PTR3 | ACETICACID | 0.1327 |
| 41 | O3 | o3 | 0.1314 |
| 42 | PhotoDiode12 | intensity2 | 0.1296 |
| 43 | PTR3 | APINENE | 0.1295 |
| 44 | PTR3 | NAPHTHALENE | 0.1287 |
| 45 | LTOF_UFRA | NIT | 0.1284 |
| 46 | NAIS_ions | Ions_neg | 0.1272 |
| 47 | CAPS | NO2 | 0.1235 |
| 48 | SabreTemperature | Sabre4Temperature | 0.1228 |
| 49 | SabreTemperature | Sabre3Temperature | 0.1223 |
| 50 | PTR3 | ACETONE | 0.1196 |
| 51 | STOF | NAPHTHALENE | 0.1161 |
| 52 | LTOF_CPC | CPC | 0.1116 |
| 53 | PT100 | temp1 | 0.1108 |
| 54 | PSM16 | conc | 0.1102 |
| 55 | CPC3776 | 3776-1DisplayConc. | 0.11 |
| 56 | PSM12 | conc | 0.1098 |
| 57 | PSM15 | conc | 0.1062 |
| 58 | TDL | sat_water | 0.1004 |
| 59 | nanoSMPS | Total Conc. #/cm | 0.0709 |

very strong relationships are found between most nodes and only two communities are found using the Louvain algorithm.

*Table 5.6: Eigenvector ($\epsilon$) centrality node ranking for run 1962.01 using the Pearson correlation as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | PSM12 | conc | 0.3347 |
| 2 | Light_Spectrometer | LS4 | 0.3301 |
| 3 | CPC3776 | 3776-1DisplayConc. | 0.3243 |
| 4 | O3 | o3 | 0.3194 |
| 5 | Light_Spectrometer | UVH | 0.3154 |
| 6 | Light_Spectrometer | LS3 | 0.3154 |
| 7 | Light_Spectrometer | LS1 | 0.3077 |
| 8 | PSM16 | conc | 0.3022 |
| 9 | LongSMPS | Particles 20-500 nm | 0.2709 |
| 10 | Light_Spectrometer | UVX | 0.256 |
| 11 | NAIS_ions | Ions_neg | 0.1844 |
| 12 | NAIS_ions | Ions_pos | 0.077 |
| 13 | nanoSMPS | Total Conc. #/cm | 0.0592 |
| 14 | PTR3 | PINONALDEHYDE | 0.0493 |
| 15 | PhotoDiode12 | intensity | 0.0234 |
| 16 | PICARRO | NH3_2MIN | 0.0201 |
| 17 | Dew | dew | 0.0162 |
| 18 | STOF | ISOPRENE | 0.0132 |
| 19 | Scalers | brust | 0.0076 |
| 20 | PTR3 | ISOPRENE | 0.007 |
| 21 | SO2 | so2 | 0.0068 |
| 22 | PTR3 | TOLUENE | 0.0059 |
| 23 | HVFC | mvolt1 | 0.0033 |
| 24 | Fan | speed1 | 0.0029 |
| 25 | PTR3 | TMB | 0.0025 |
| 26 | PTR3 | NAPHTHALENE | 0.0022 |
| 27 | PTR3 | ACETICACID | 0.002 |
| 28 | TDL | sat_water | 0.0013 |
| 29 | STOF | APINENE | 0.0011 |
| 30 | PT100 | temp1 | 0.0011 |
| 31 | PTR3 | ACETONE | 0.001 |
| 32 | PTR3 | APINENE | 0.0009 |
| 33 | CAPS | NO2 | 0.0005 |
| 34 | PTR3 | BETACARYOPHYLENE | 0.0004 |

**5.2.5.2.2 Reduced Channel Set** As was noted previously, be it by being deemed unnecessary for the experiment or by containing problematic statistics, there is a case for reviewing the set of channels chosen. While this compromises the exploratory analysis that a larger dataset provides, it allows the analysis to highlight more relevant relationships. Tables B.3 and B.4 contain the reduced set of channels, which focused on containing most of the physical measurements but removed channels that used similar measurements to avoid spurious correlation and removed several chemical measurements that were not considered important for the run or were measuring at levels lower than the detection limit. The similarity analysis was then repeated.

**5.2.5.2.2.1 Pearson Correlation Similarity** Figure 5.34 shows the resulting network graph using the Pearson correlation coefficient as a similarity measurement. Table 5.6 shows the ranking of nodes according to their eigenvector centrality for simpler identification.

When comparing 5.34 with figure 5.30, very little changed and the first one seems to be a less dense version of

**Stage 1962.01**

2 communities found

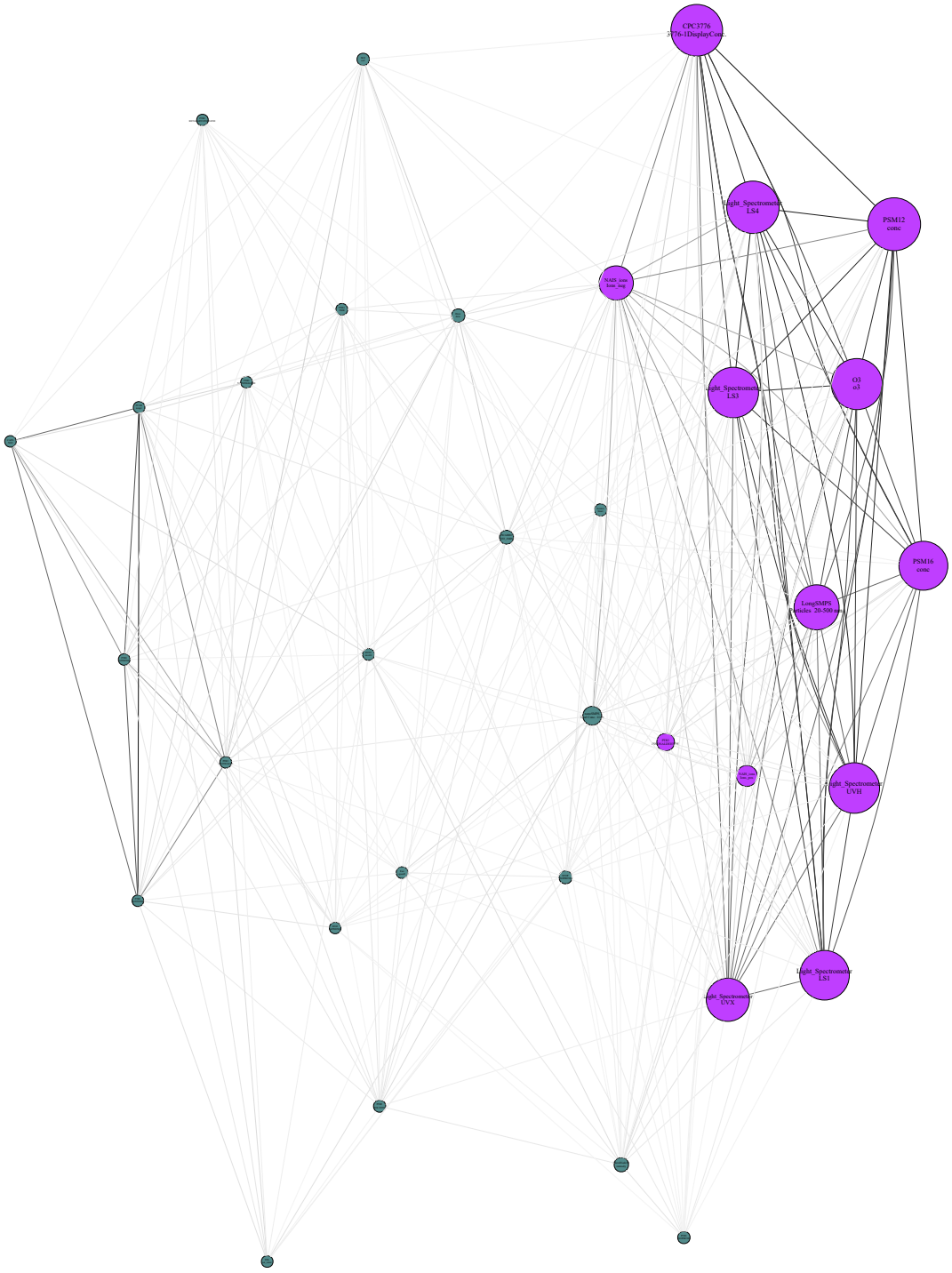



*Figure 5.34: Pearson correlation network graph for the CLOUD experimental run 1962.01 using a reduced set of channels. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table 5.7: Eigenvector ($\epsilon$) centrality node ranking for run 1962.01 using the maximum cross-correlation as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | PSM12 | conc | 0.2841 |
| 2 | Light_Spectrometer | LS4 | 0.2836 |
| 3 | O3 | o3 | 0.2759 |
| 4 | CPC3776 | 3776-1DisplayConc. | 0.2747 |
| 5 | Light_Spectrometer | LS3 | 0.2738 |
| 6 | Light_Spectrometer | UVH | 0.2728 |
| 7 | Light_Spectrometer | LS1 | 0.2646 |
| 8 | PSM16 | conc | 0.2643 |
| 9 | LongSMPS | Particles 20-500 nm | 0.2484 |
| 10 | Light_Spectrometer | UVX | 0.2262 |
| 11 | NAIS_ions | Ions_neg | 0.2042 |
| 12 | PTR3 | ACETICACID | 0.1722 |
| 13 | CAPS | NO2 | 0.165 |
| 14 | PT100 | temp1 | 0.1535 |
| 15 | nanoSMPS | Total Conc. #/cm | 0.1529 |
| 16 | NAIS_ions | Ions_pos | 0.1528 |
| 17 | PTR3 | ACETONE | 0.1475 |
| 18 | PTR3 | APINENE | 0.1225 |
| 19 | PICARRO | NH3_2MIN | 0.1222 |
| 20 | Dew | dew | 0.1182 |
| 21 | SO2 | so2 | 0.087 |
| 22 | TDL | sat_water | 0.0784 |
| 23 | PTR3 | PINONALDEHYDE | 0.0705 |
| 24 | PTR3 | NAPHTHALENE | 0.0653 |
| 25 | HVFC | mvolt1 | 0.0625 |
| 26 | PTR3 | TOLUENE | 0.0622 |
| 27 | Fan | speed1 | 0.0564 |
| 28 | PTR3 | TMB | 0.0561 |
| 29 | PTR3 | ISOPRENE | 0.0533 |
| 30 | STOF | APINENE | 0.0503 |
| 31 | STOF | ISOPRENE | 0.0495 |
| 32 | PhotoDiode12 | intensity | 0.0461 |
| 33 | PTR3 | BETACARYOPHYLENE | 0.045 |
| 34 | Scalers | brust | 0.0346 |

the second. Such a claim is confirmed when comparing table 5.6 with 5.2, the most relevant time series are the particle concentration measurements, along with light spectrometer measurements and ozone concentration. The Louvain algorithm once again highlights two different communities of nodes, with no particular practical meaning other than being clustered by their eigenvector similarity value, thus further aiding the user in determining the importance of each node in the graph.

**5.2.5.2.2.2 Cross-correlation Similarity** Figure 5.35 shows the resulting network graph using the maximum cross-correlation as similarity. Table 5.7 shows the ranking of nodes according to their eigenvector centrality to help identify each node.

Analysing figure 5.35 similar conclusions are made as the ones in the previous section. This graph seems to be a less dense version of figure 5.31, favouring once again the particle concentration, light intensity and ozone measurements over all others. One difference is that for this graph, the Louvain algorithm finds three communities,

# Stage 1962.01

3 communities found

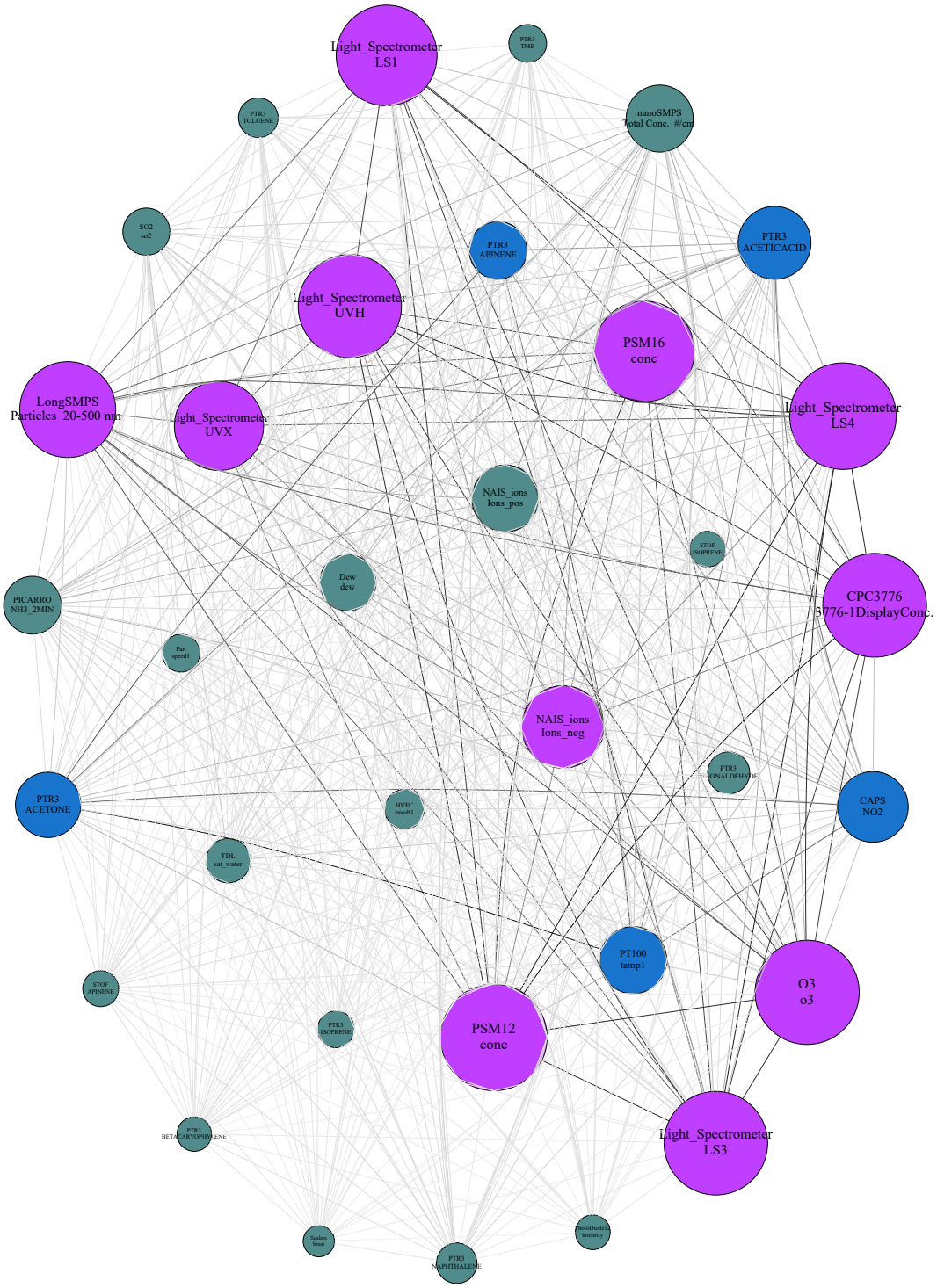



*Figure 5.35: Maximum cross-correlation network graph for the CLOUD experimental run 1962.01 using a reduced set of channels. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table 5.8: Eigenvector ($\epsilon$) centrality node ranking for run 1962.01 using DTW as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | CPC3776 | 3776-1DisplayConc. | 0.2417 |
| 2 | Light_Spectrometer | LS4 | 0.2374 |
| 3 | Light_Spectrometer | UVH | 0.2367 |
| 4 | O3 | o3 | 0.2287 |
| 5 | Light_Spectrometer | LS3 | 0.2255 |
| 6 | Light_Spectrometer | LS1 | 0.2179 |
| 7 | LongSMPS | Particles 20-500 nm | 0.2122 |
| 8 | PSM12 | conc | 0.2074 |
| 9 | Light_Spectrometer | UVX | 0.2069 |
| 10 | PICARRO | NH3_2MIN | 0.1789 |
| 11 | NAIS_ions | Ions_pos | 0.1789 |
| 12 | SO2 | so2 | 0.1714 |
| 13 | PTR3 | PINONALDEHYDE | 0.1677 |
| 14 | PSM16 | conc | 0.1674 |
| 15 | PTR3 | ISOPRENE | 0.1648 |
| 16 | PTR3 | TMB | 0.1614 |
| 17 | NAIS_ions | Ions_neg | 0.1604 |
| 18 | Scalers | brust | 0.1602 |
| 19 | PTR3 | TOLUENE | 0.1593 |
| 20 | STOF | APINENE | 0.1586 |
| 21 | STOF | ISOPRENE | 0.1569 |
| 22 | PTR3 | BETACARYOPHYLENE | 0.1564 |
| 23 | PTR3 | ACETICACID | 0.1525 |
| 24 | Dew | dew | 0.1468 |
| 25 | Fan | speed1 | 0.1453 |
| 26 | PTR3 | APINENE | 0.1443 |
| 27 | TDL | sat_water | 0.1372 |
| 28 | PTR3 | NAPHTHALENE | 0.1357 |
| 29 | CAPS | NO2 | 0.1324 |
| 30 | nanoSMPS | Total Conc. #/cm | 0.1287 |
| 31 | PTR3 | ACETONE | 0.1215 |
| 32 | PT100 | temp1 | 0.1117 |
| 33 | HVFC | mvolt1 | 0.0711 |
| 34 | PhotoDiode12 | intensity | 0.0554 |

similarly to the full set graph. Once again these communities have no immediate meaning other than to help the user highlight the importance of each node.

**5.2.5.2.2.3 DTW Similarity** Figure 5.36 shows the network graph using DTW similarity as a similarity measurement. Table 5.8 once again ranks the nodes in terms of their eigenvector centrality for easier identification.

Figure 5.36 also resembles a less dense version of figure 5.32. Since several of the removed channels comprised of the top ranking channels of table 5.4 it is not surprising that table 5.8 looks significantly different, however, the results now match those that are reported by the other methods. Analysing the similarity values that connect each node once again shows that DTW as a similarity measurement is not the best choice of similarity measurement for this CLOUD dataset, since the similarity values found are all very small.

# Stage 1962.01

2 communities found

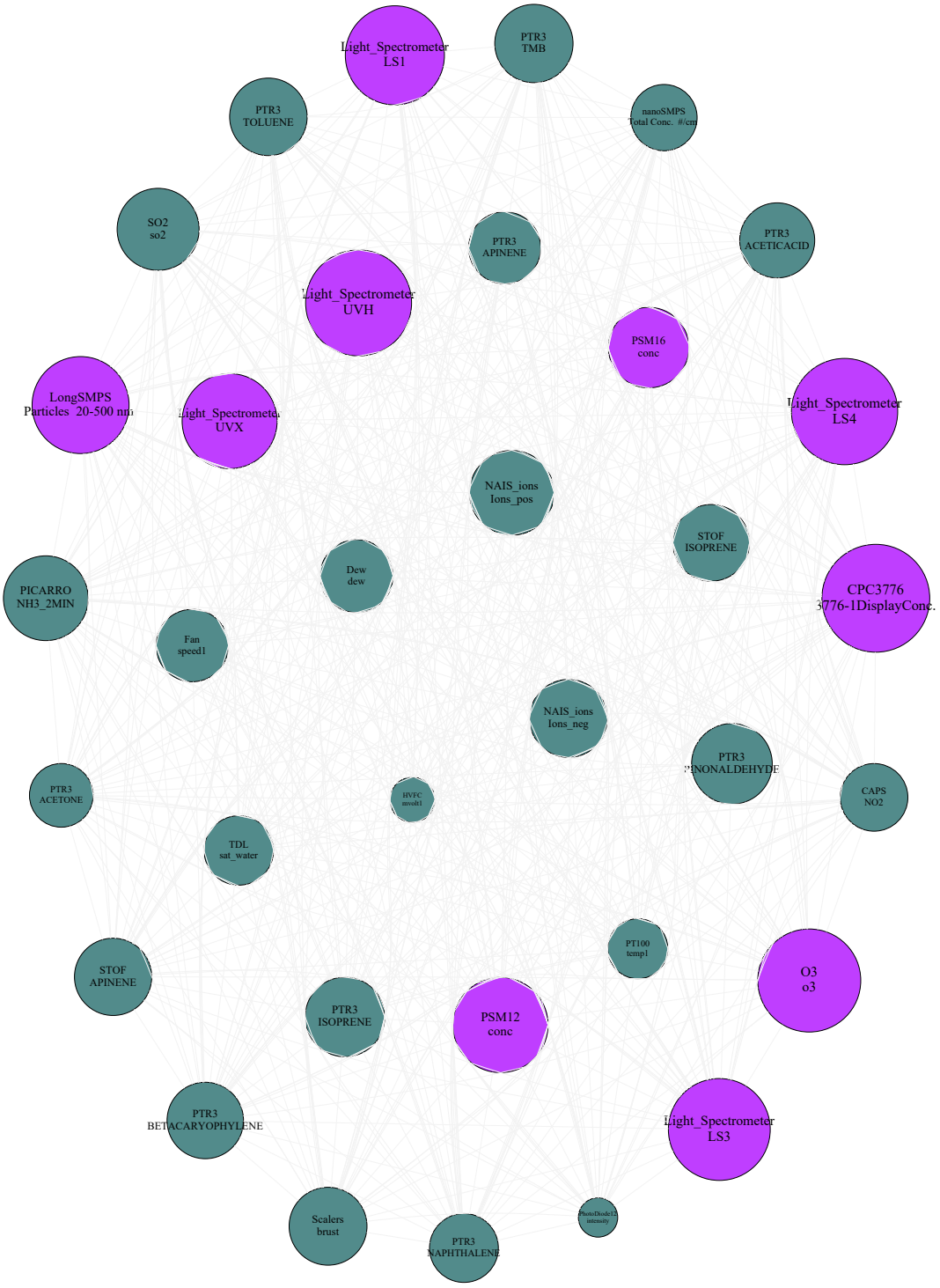



*Figure 5.36: DTW network graph for the CLOUD experimental run 1962.01 using a reduced set of channels. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table 5.9: Eigenvector ($\epsilon$) centrality node ranking for run 1962.01 using the coherence as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | Light_Spectrometer | UVH | 0.1893 |
| 2 | HVFC | mvolt1 | 0.1884 |
| 3 | LongSMPS | Particles 20-500 nm | 0.1871 |
| 4 | Light_Spectrometer | UVX | 0.1856 |
| 5 | PTR3 | ISOPRENE | 0.1855 |
| 6 | STOF | ISOPRENE | 0.1851 |
| 7 | PTR3 | TMB | 0.1851 |
| 8 | PTR3 | PINONALDEHYDE | 0.1847 |
| 9 | Light_Spectrometer | LS4 | 0.1844 |
| 10 | SO2 | so2 | 0.1837 |
| 11 | Light_Spectrometer | LS1 | 0.1826 |
| 12 | Scalers | brust | 0.1815 |
| 13 | Light_Spectrometer | LS3 | 0.1814 |
| 14 | STOF | APINENE | 0.1811 |
| 15 | PTR3 | TOLUENE | 0.1808 |
| 16 | PTR3 | BETACARYOPHYLENE | 0.1808 |
| 17 | Dew | dew | 0.1807 |
| 18 | Fan | speed1 | 0.1803 |
| 19 | PTR3 | ACETICACID | 0.18 |
| 20 | PICARRO | NH3_2MIN | 0.1799 |
| 21 | NAIS_ions | Ions_pos | 0.1795 |
| 22 | O3 | o3 | 0.1779 |
| 23 | PTR3 | APINENE | 0.1749 |
| 24 | PTR3 | NAPHTHALENE | 0.1742 |
| 25 | NAIS_ions | Ions_neg | 0.1733 |
| 26 | CAPS | NO2 | 0.1668 |
| 27 | PTR3 | ACETONE | 0.1616 |
| 28 | PT100 | temp1 | 0.15 |
| 29 | CPC3776 | 3776-1DisplayConc. | 0.1498 |
| 30 | PSM16 | conc | 0.1491 |
| 31 | PSM12 | conc | 0.149 |
| 32 | TDL | sat_water | 0.1366 |
| 33 | nanoSMPS | Total Conc. #/cm | 0.0985 |

**5.2.5.2.2.4 Coherence Similarity** Figure 5.37 shows the resulting network graph using the coherence as a similarity measurement. Table 5.9 helps identify each node by ranking them in terms of their eigenvector centrality.

Once again, figure 5.37 is a less dense version of figure 5.33. When analysing table 5.9, the same channels from the full set still seem to dominate. An emphasis is still given to the high voltage channel since, which is still an unexpected product of the existing noise that was not properly removed from the data as well as some chemical channels measuring similar species and of the same instrument are also highlighted. While highlighting several relevant channels, it is not clear whether analysing this particular set of time series from the CLOUD experiment in the frequency domain is useful.

**5.2.5.2.3 Explaining the Results** To explain the results found not only in this section but also in B.2, a deeper look into the data is required. This section will consider the previously analysed reduced time series set to interpret the results of similarity measurements applied to pairs of time series. For this set, the most relevant time series are

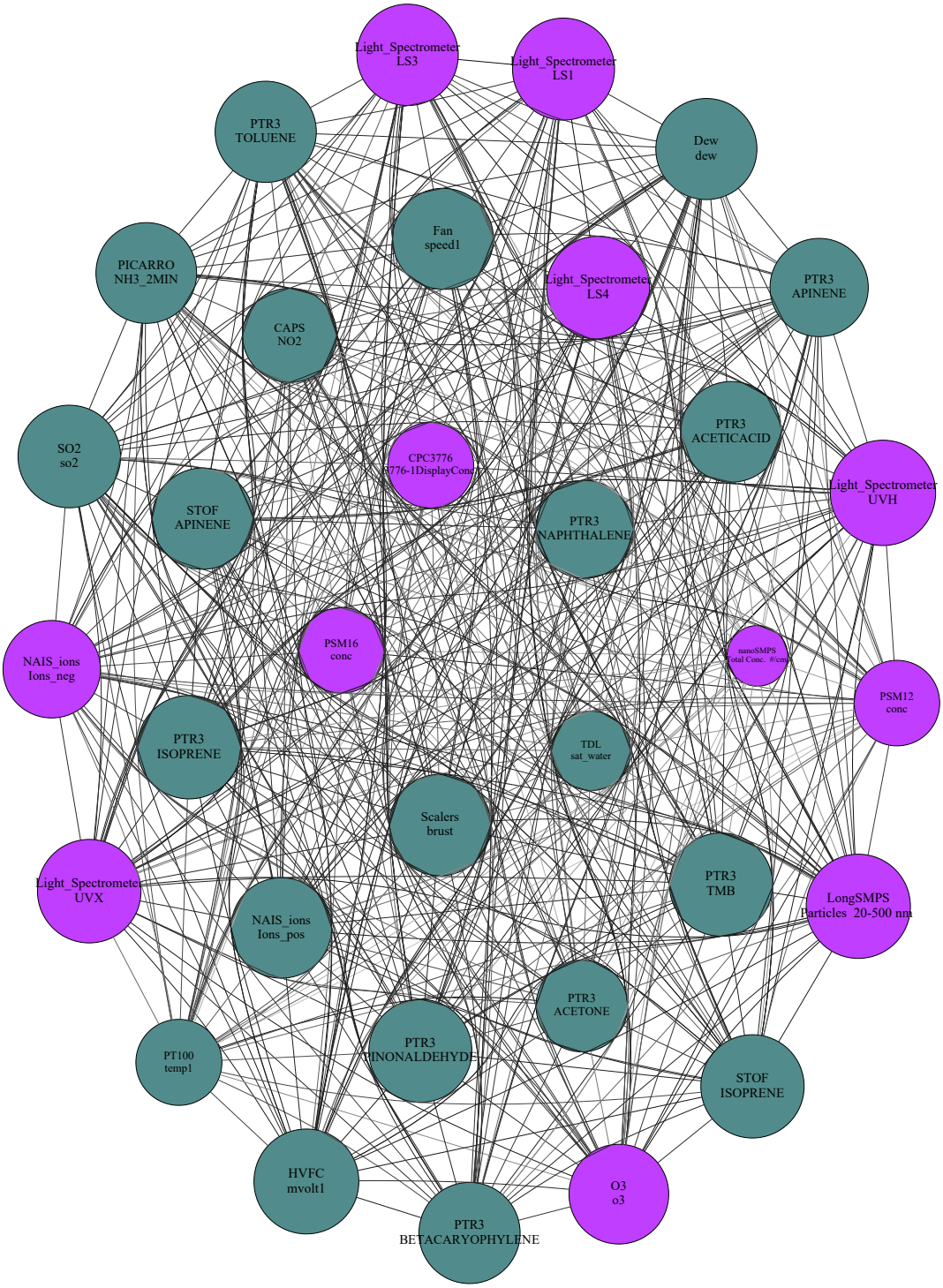



*Figure 5.37: Coherence network graph for the CLOUD experimental run 1962.01 using a reduced set of channels. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

chosen according to one eigenvector centrality rank, in this case the rank chosen was the DTW similarity rank. For each time series, the most relevant relationships are found using each comparison method described and the results are analysed. Tables 5.10, 5.11 and 5.12 shows the top ranks for the DTW similarity method, Pearson correlation coefficient and Maximum Cross-Correlation, respectively:

*Table 5.10: Top ranks of DTW similarity*

| Channel, Rank | Rank 1 | Rank 2 | Rank 3 | Rank 4 |
|---|---|---|---|---|
| CPC3776-1Conc., 1 | 6.73e-03, PSM12 conc | 6.23e-03, L_Spect LS4 | 5.93e-03, L_Spect UVH | 4.86e-03, O3 o3 |
| L_Spect LS4, 2 | 6.23e-03, CPC3776-1Conc. | 6.16e-03, PSM12 conc | 4.70e-03, L_Spect UVH | 3.97e-03, LSMPS 20-500 nm |
| L_Spect UVH, 3 | 5.93e-03, CPC3776-1Conc. | 4.70e-03, L_Spect LS4 | 4.43e-03, LSMPS 20-500 nm | 4.34e-03, PSM12 conc |
| O3 o3, 4 | 4.86e-03, CPC3776-1Conc. | 4.34e-03, L_Spect LS3 | 3.73e-03, L_Spect LS1 | 3.63e-03, PSM12 conc |
| L_Spect LS3, 5 | 4.34e-03, O3 o3 | 3.72e-03, LSMPS 20-500 nm | 3.66e-03, L_Spect LS1 | 3.44e-03, CPC3776-1Conc. |
| L_Spect LS1, 6 | 3.73e-03, O3 o3 | 3.66e-03, L_Spect LS3 | 3.60e-03, L_Spect UVX | 3.59e-03, CPC3776-1Conc. |
| LSMPS 20-500 nm , 7 | 4.43e-03, L_Spect UVH | 3.97e-03, L_Spect LS4 | 3.95e-03, CPC3776-1Conc. | 3.74e-03, PSM12 conc |
| PSM12 conc, 8 | 6.73e-03, CPC3776-1Conc. | 6.16e-03, L_Spect LS4 | 4.34e-03, L_Spect UVH | 3.74e-03, LSMPS 20-500 nm |

*Table 5.11: Top ranks of Pearson correlation*

| Channel, Rank | Rank 1 | Rank 2 | Rank 3 | Rank 4 |
|---|---|---|---|---|
| CPC3776-1Conc., 3 | 9.69e-01, PSM12 conc | 8.76e-01, L_Spect LS4 | 8.36e-01, L_Spect UVH | 8.32e-01, L_Spect LS3 |
| L_Spect LS4, 2 | 9.18e-01, PSM12 conc | 9.13e-01, L_Spect LS3 | 9.05e-01, L_Spect UVH | 8.76e-01, O3 o3 |
| L_Spect UVH, 5 | 9.05e-01, L_Spect LS4 | 8.76e-01, PSM12 conc | 8.43e-01, L_Spect LS3 | 8.36e-01, CPC3776-1Conc. |
| O3 o3, 4 | 8.76e-01, L_Spect LS4 | 8.74e-01, PSM12 conc | 8.40e-01, L_Spect LS3 | 8.34e-01, L_Spect UVH |
| L_Spect LS3, 6 | 9.13e-01, L_Spect LS4 | 8.72e-01, PSM12 conc | 8.43e-01, L_Spect UVH | 8.40e-01, O3 o3 |
| L_Spect LS1, 7 | 8.61e-01, L_Spect LS4 | 8.38e-01, PSM12 conc | 8.14e-01, L_Spect LS3 | 8.11e-01, L_Spect UVH |
| LSMPS 20-500 nm , 9 | 7.73e-01, O3 o3 | 7.32e-01, PSM12 conc | 6.99e-01, L_Spect LS4 | 6.69e-01, L_Spect UVH |
| PSM12 conc, 1 | 9.69e-01, CPC3776-1Conc. | 9.18e-01, L_Spect LS4 | 8.76e-01, L_Spect UVH | 8.74e-01, O3 o3 |

*Table 5.12: Top ranks of Maximum cross correlation*

| Channel, Rank | Rank 1 | Rank 2 | Rank 3 | Rank 4 |
|---|---|---|---|---|
| CPC3776-1Conc., 3 | 9.69e-01, PSM12 conc, -1 | 8.78e-01, L_Spect LS4, 20 | 8.38e-01, L_Spect UVH, 109 | 8.37e-01, L_Spect LS3, -31 |
| L_Spect LS4, 1 | 9.18e-01, PSM12 conc, 7 | 9.13e-01, L_Spect LS3, -1 | 9.05e-01, L_Spect UVH, -1 | 8.78e-01, CPC3776-1Conc., 20 |
| L_Spect UVH, 6 | 9.05e-01, L_Spect LS4, -1 | 8.76e-01, PSM12 conc, -1 | 8.43e-01, L_Spect LS3, -3 | 8.38e-01, CPC3776-1Conc., 109 |
| O3 o3, 5 | 8.76e-01, L_Spect LS4, -1 | 8.74e-01, PSM12 conc, -1 | 8.56e-01, L_Spect LS3, 55 | 8.34e-01, L_Spect UVH, -1 |
| L_Spect LS3, 4 | 9.13e-01, L_Spect LS4, -1 | 8.72e-01, PSM12 conc, -2 | 8.56e-01, O3 o3, 55 | 8.43e-01, L_Spect UVH, -3 |
| L_Spect LS1, 7 | 8.61e-01, L_Spect LS4, -1 | 8.40e-01, PSM12 conc, -13 | 8.15e-01, L_Spect LS3, 14 | 8.13e-01, CPC3776-1Conc., -28 |
| LSMPS 20-500 nm , 9 | 7.78e-01, O3 o3, -52 | 7.44e-01, PSM12 conc, -128 | 6.99e-01, L_Spect LS4, -1 | 6.88e-01, CPC3776-1Conc., -189 |
| PSM12 conc, 2 | 9.69e-01, CPC3776-1Conc., -1 | 9.18e-01, L_Spect LS4, 7 | 8.76e-01, L_Spect UVH, -1 | 8.74e-01, O3 o3, -1 |

From analysing tables 2 and 3 it is clear that for this stage there is little difference found in either set due to the very small time delay found for the maximum correlation. This is further emphasised by comparing with the DTW similarity results found in table 1, while the values themselves are low, for most channels, the relative intensity of the relationship is kept at a very similar value (dividing the Pearson correlation by the DTW similarity yields similar results for most pairs of channels).

#### 5.2.5.3 Run 1962.01 - Relevant Period Selection

Table 5.13 shows the channels where at least one sub-period statistically deviates more than the provided threshold value, the number of times for each period it happens as well as the totals for each channel and period. Table 5.14 shows the same detection using a classical rolling z-score window of threshold $5\sigma$ for comparison. For this method the full data set of time series was used since it was decided in 5.2.5.1 that the reduced does not provide any further insights that the full set provides.

*Table 5.13: Events in each period for run 1962.01 using wavelet ME partitioning.*

| Channel | Period 1 | Period 2 | Period 3 | Period 4 | Total |
|---|---|---|---|---|---|
| Scalers brust | 7 | 7 | 5 | 8 | 27 |
| NAIS_ions Ions_neg | 5 | 7 | 7 | 7 | 26 |
| Dew dew | 6 | 0 | 6 | 8 | 20 |
| nRDMA Count | 0 | 0 | 9 | 6 | 15 |
| Fan speed2 | 5 | 0 | 4 | 3 | 12 |
| Fan speed1 | 1 | 1 | 4 | 5 | 11 |
| LongSMPS Particles 20-500 nm | 6 | 0 | 3 | 2 | 11 |
| Light_Spectrometer LS1 | 6 | 1 | 0 | 2 | 9 |
| HVFC mvolt1 | 0 | 0 | 0 | 9 | 9 |
| Light_Spectrometer LS3 | 2 | 3 | 2 | 1 | 8 |
| Light_Spectrometer UVX | 1 | 2 | 0 | 4 | 7 |
| NAIS_ions Ions_pos | 6 | 0 | 0 | 0 | 6 |
| Light_Spectrometer LS4 | 3 | 3 | 0 | 0 | 6 |
| Light_Spectrometer UVH | 1 | 3 | 0 | 1 | 5 |
| TDL sat_water | 0 | 2 | 0 | 3 | 5 |
| HTOF_UFRA H3OH2O | 5 | 0 | 0 | 0 | 5 |
| CAPS NO2 | 1 | 1 | 0 | 2 | 4 |
| DMAtrain Ch0 Ch [#/cc] | 0 | 0 | 0 | 4 | 4 |
| PTR3 PINONALDEHYDE | 1 | 1 | 1 | 0 | 3 |
| PTR3 APINENE | 1 | 0 | 1 | 0 | 2 |
| STOF NAPHTHALENE | 0 | 2 | 0 | 0 | 2 |
| DMAtrain Ch1 C [#/cc] | 1 | 0 | 0 | 1 | 2 |
| PTR3 ACETONE | 0 | 1 | 0 | 0 | 1 |
| PTR3 TOLUENE | 0 | 0 | 0 | 1 | 1 |
| PTR3 NAPHTHALENE | 0 | 1 | 0 | 0 | 1 |
| PTR3 ACETICACID | 1 | 0 | 0 | 0 | 1 |
| SO2 so2 | 1 | 0 | 0 | 0 | 1 |
| PTR3 ISOPRENE | 0 | 0 | 1 | 0 | 1 |
| PTR3 BETACARYOPHYLENE | 0 | 1 | 0 | 0 | 1 |
| PTR3 TMB | 1 | 0 | 0 | 0 | 1 |
| Total | 61 | 36 | 43 | 67 | 207 |

Comparing tables 5.13 and 5.14 it is clear that the more classical method captures more events per channel, but fails to capture events on most channels. However, the more classical method seems to highlight the more important part of the experimental run, which are the two middle sub periods, with very similar amounts of detected events for each, while the wavelet ME partitioning method fails to capture the importance of the second sub-period. This could be due to a poor choice of separation of periods due to the interface of the two first sub-periods being in a transition state and thus the choice of template period against which to check for anomalies should be reviewed for this detection method. Also of note that for either method, the channels that most contribute to event detection are not those that are highlighted nor are they expected to be related to particle generation (for instance *Dew dew*,

*Table 5.14: Events in each period for run 1962.01 using a rolling z-score*

| Channel | Period 1 | Period 2 | Period 3 | Period 4 | Total |
|---|---|---|---|---|---|
| HTOF_UFRA NH3 | 16 | 16 | 12 | 16 | 60 |
| HTOF_UFRA H7O3 | 10 | 18 | 18 | 11 | 57 |
| nRDMA Count | 10 | 18 | 9 | 16 | 53 |
| HTOF_UFRA O2 | 9 | 17 | 11 | 16 | 53 |
| HTOF_UFRA H3OH2O | 16 | 12 | 22 | 0 | 50 |
| DMAtrain Ch0 Ch [#/cc] | 10 | 10 | 10 | 5 | 35 |
| Scalers brust | 6 | 10 | 0 | 6 | 22 |
| TDL sat_water | 0 | 0 | 18 | 0 | 18 |
| DMAtrain Ch1 C [#/cc] | 7 | 4 | 0 | 6 | 17 |
| Light_Spectrometer UVH | 4 | 0 | 0 | 0 | 4 |
| Total | 88 | 105 | 100 | 76 | 369 |

measuring dew point and *Scalers brust* measuring the intensity of cosmic rays bombarding the chamber - which should not influence a neutral run). Analyzing the other particle generation experiments (tables B.9, B.19 and B.24) shows that this is not always the case.

Using these statistically significant events, one can effectively transform figure 5.28 to reflect the amount of events observed by scaling the brightness of one time period to the total number of events observed during that time period. Figure 5.38 shows said alteration to the original figure. Analyzing the ranked relevant sub-periods shows that the most important part of the stage is the first sub-period, followed by the last, third and finally second sub-periods. This result is unexpected since one would assume that the first two sub-periods would be the most relevant as it is during these that particles begin to form. However, it is clear that some signals start to show change before the second sub-period while others do it during the second sub-period (see *PSM16*, *PT100* and *O3* plots versus *LTOF_CPC* and *Light_Spectrometer_UVH* plots).

While the caveats discussed in this section do not necessarily imply a problem with the event detection method, it is clear that a degree of fine-tuning of said method should be available for users to handle detection at a signal-by-signal case should it be necessary. This limitation can be dampened by providing the user with the opportunity to change the detection threshold for each used channel. Despite this being a cumbersome job for the user, it is the only way of providing each channel with similar ranking power. Also the proposed fixes to the pre-processing method suggested in previous subsections would also improve the performance of this event detection algorithm.

### 5.2.6 Implementation in DAQBroker

As mentioned previously, this method of statistical analysis of the collected data can very easily be implemented into DAQBroker as a method of providing a statistics-based summary of an experimental run. The summary generation can be presented to the user as an option, after the experimental run or stage is finished, as a background process that can be reviewed when completed from the run list interface (see 3.1.3). Implementing this method requires simple additions to the main DAQBroker code, both in the back-end and the front-end due to the modularity of DAQBroker's code, In this section the required changes to each part of the code to provide users with this functionality are presented.

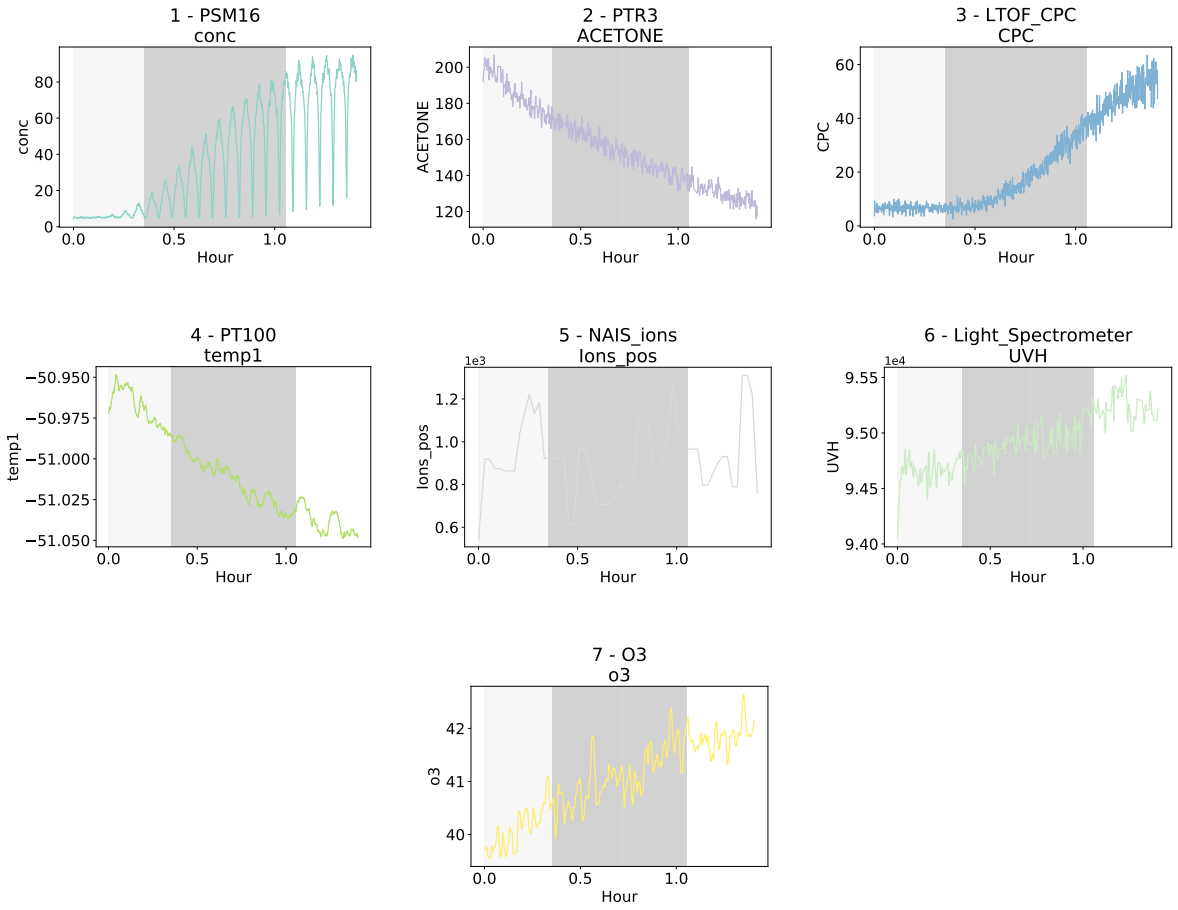



*Figure 5.38: Most relevant time series for CLOUD experimental run 1962.01 coupled with ranking of specific time periods to the number of events observed in each event. The time period was divided into 4 distinct sub-periods, which were then tested for the existence of events. The ranking of each sub-period shades each sub-period more the less events are found. In this case, as time increases, more events were found in each sub-period.*

#### 5.2.6.1 Back-end Considerations

Before implementing this facility in DAQBroker, one must first consider whether the method should be performed in a sequential or parallel fashion. The application of the fairly efficient version described previously yields execution times of around 150 seconds for 93 data channels with time ranges from 1.4 to 2.7 hours. The execution speeds are very different for sequential execution when in the context of a web interface since a high amount of waiting time for the end result is associated. It would thus be more acceptable to allow the user to navigate to other interfaces while waiting for the different methods to be executed, thus a background execution of the methods should be preferred.

Two features must be explored for the back-end implementation of this method in DAQBroker: The first consists in the implementation of the different parts of this method onto self-contained functions that handle a set of provided time series. In this case a new class (the MVSG class) is introduced which contains all necessary methods for providing the time series pre-processing, relevance highlighting, relationships and time period ranking steps described in the previous sections. The second front focuses on the addition of these sets of functions to the DAQBroker methods set. A new call made by the user and all required actions are performed in the background providing all necessary information in a timely fashion, for the front-end to produce meaningful visualizations for the user. To this end a new DAQBroker REST API is introduced to handle the background generation of the necessary tasks.

**5.2.6.1.1 MVSG Class** The **M**ulti-**V**ariate **S**ummary **G**enerator class is responsible for collecting and processing the information contained in a set of time series. What follows is a list of the class' attributes and methods to handle all the necessary steps of time series processing.

- Attributes :
  - *time_series*: An array containing the set of provided time series. Each time series is an array of arrays, where sub-array contains the time and the respective value of the time series at that time.
  - *names*: An array containing the names of the data channels to which each time series corresponds
  - *time series_processed*: An array that will contain arrays of values of the pre-processed time series, after the pre-processing steps have been performed. No time information is required as time as been regularized and only one array of time is required to provide users with time information,
  - *time*: An array containing the regularized time after the pre-processing steps have been executed,
  - *fft*: An array containing the arrays of the Fourier transform of the pre-processed multivariate time series. This array will be used for the PCA step and to calculate the power of each pre-processed series,
  - *pca_limit*: A float value between 0 and 1 to illustrate the amount of explained variance to be extracted from the PCA analysis step,

- *fft_reconstructed*: A reconstructed version of the *fft* attribute after the PCA step has been preformed,
- *time_series_sorted*: An array of sorted time series to be created at the time of the LU decomposition,
- *names_sorted*: An array of sorted channel names to be created at the time of the LU decomposition,
- *cross_spectrum*: A matrix where each element consists of the cross spectral density of two time series, this matrix will be used to calculate the different coherence matrices which will be used to create the respective network graphs;
- *G and G_part*: Network graphs for full and partial coherence (respectively),
- *limit_angle*: An array of angles that serve as a threshold value for event detection in each time series,
- *series_angles*: An array of arrays containing, for each time series, the angles between the first and subsequent periods inside each sub-period of the multivariate time series;

• Methods :

- *add_time_series*: Method to add a new time series to the multivariate set. Must include as an argument the time series as defined in the *time series* attribute and should contain an optional argument to supply the name of the data channel,
- *regularize_time*: Method to collect all time series granularity perform all necessary interpolation functions,
- *normalize_timeseries*: Method to normalize each time-regularized time series, creating the *time_series_processed* attribute,
- *spectral_calculations*: Method to calculate the Fourier transform of the multivariate time series. Generates the *fft* attribute and calculates the power of each time series, which allows an internal sorting of the *time_series_processed* and *names* attributes to account for the power of each time series,
- *pca_analysis*: Method to implement the PCA analysis step of the multivariate time series. Generates the *fft_reconstructed* attribute and makes use of the *pca_limit* attribute,
- *csd_calculation*: Method to calculate the cross spectral density matrix. This method can take advantage of a high degree of parallelization to generate the *cross_spectrum* attribute
- *graph_generation*: Method to create the full and partial coherence network graphs,
- *event_detection*: Method to divide the *time* attribute into a user-defined number of sub-periods and perform event detection for each time series in these sub-intervals. Generates the *series_angles* attribute and uses the *limit_angle* attribute.

This is a simplified description of the class, highlighting the most important methods and attributes. Other auxiliary methods are required in order for the class to work efficiently and also many of them for error handling to account for unexpected errors.

In order to use the MVSG class, it must be provided a set of time series by means of multiple uses of the *add_timeseries* method. More code is required to collect the time series and supply them to the MVSG class.

**5.2.6.1.2 REST Endpoint** To allow for implementation in a web-based user interface, a new REST API endpoint should be added to DAQBroker. This endpoint should receive all the relevant information from the user to be able to instantiate a MVSG object and correctly run all its different methods. The following is a description of the aforementioned endpoint containing all the required inputs from the user and a short description of the steps to perform and where to store the produced data.

- Inputs:
  - *exp_run*: A unique descriptor of the experimental run to generate a summary. This value will be used to define the start and end times of the summary,
  - *channels*: A list of unique channel identifiers containing the channels to be included in the summary generation,
  - *normalize*: A boolean that tells the method to perform normalize time series data,
  - *mfilter*: A boolean that tells the method to perform a median filter on the data,
  - *sgfilter*: A boolean that tells the method to perform a Savitzky–Golay filter on the data,
  - *mfilter_window*: An integer containing the window of the median filter. Must be odd,
  - *sgfilter_degree*: An integer containing the degree of the Savitzky–Golay filter,
  - *pca_limit*: A float from 0 to 1 containing the percentage of explained variance from the PCA step,
  - *strength_limit*: A float from 0 to 1 containing minimum relationship strength to consider when making the coherence network graphs,
  - *partitions*: An integer containing the number of equal parts the experiment time period is to be divided into,
  - *limit_angle*: An array of floats from 0 to $\pi/2$ with the limiting angles for statistically relevant event detection of each channel
- Outputs:
  - *process_id*: A unique identifier of the process executing the summary routine. Information about this process should be stored in a job queue for easy access (see appendix A *jobs* table). Once completed, the result of this method should be permanently stored in a specific table that contains an entry for each performed summary execution mapped to the run it was attempted on. The columns of this table must include:
    - Unique experimental run identifier,
    - Date of task start/end,

* Results of the PCA analysis, most relevant channels,
* The cross-spectral matrix,
* The full and partial coherence matrices,
* Event detection results.

With this endpoint in place, it is possible to send the server a request for a summary execution either via a pre-designed user interface or by using the REST API over another interface. This is simply a template of what the inputs and outputs of the endpoint must contain, several other security, rate limiting and error handling considerations must be considered.

#### 5.2.6.2 Front-end Considerations

In order to build the user interface that enables the summarized analysis of experimental runs, two major blocks are considered.

The first block consists of preparing the request to be sent in the context of the aforementioned REST endpoint.

The second block should consider the visualization options for the different results of the run summary. Considerations for PCA, coherence graphs and event detection must be considered.

**5.2.6.2.1 Request Preparation** As was previously stated, the request for a run summary should be sent at the time the users requests to end said run. This block should take into account that several parameters must be sent in a structured way, it should be able to handle errors and as best as possible prevent abuse of the interface. It must also take into account the background nature of the run summary task and provide users with the appropriate waiting response if a task is still under way for a specific run, or an optional run summary prompt when ending an experimental run. To prepare this summary a user-interface for the choice of all parameters in the simplest possible manner is presented. The choice of parameters must be divided according to the different parts of the method:

1. Data channels to include,
2. Pre-processing values (filter windows and degrees),
3. PCA explained variance,
4. Limiting relationship strength,
5. Event detection threshold angle.

All these sections contain clearly defined prompts for the user to consider when performing a run summary request. The interface fields should be previously filled with default values either from predefined values or from values inputted by the user during a previous run summary request (this last choice requires an extra server request to

collect the previous defaults). For validation must be considered in this stage to avoid incorrect inputs by the user. Once the request is sent, the user is given a message that the handling of the request is underway. Ideally the user will be presented with a progress bar showing the step in which the request is currently at (pre-processing, PCA, coherence calculation, event detection). After starting, the interface changes, informing users that a summary request is currently underway and preventing users from performing more requests (this locking should occur first at the server-side). Figure 5.39 shows a mock-up of what the interface mentioned look like. A separator tab would allow a clean division between each step of the analysis showing the user only the important information for each step.

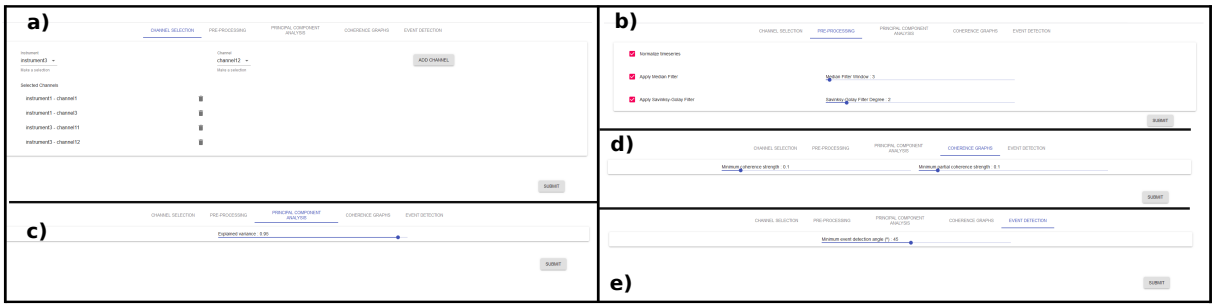



*Figure 5.39: Suggested implementation of the request preparation user interface. The user is presented with a multi-tab interface to prepare all necessary parameters to be sent to the REST API. An initial tab (a) allows users to choose what channels to user for the multivariate time series, another tab (b) provides users with the possibility to change important pre-processing parameters and the last 3 tabs provide important parameters for each of the steps of the run summary: PCA (c), times eries relationships (d) and event detection (a).*

**5.2.6.2.2 Summary Visualization** The second and arguably most important part of the front-end considerations refers to the presentation of the results to the user. To this end a completely new section of the DAQBroker front-end interface is suggested. This section can be reached via the user interface when browsing the run list; on runs where at least one finished run summary exists, a button allowing the user to view the existing summaries is provided. The user is then redirected to the new section where he can easily share this visualization with other users by way of the unique URL of the visualization, which is then rendered for each user.

The new section should be able to create meaningful visualizations of the most relevant channels, the most important periods of the experimental run as well as to highlight the network structure of each channel and its relationships, providing in depth tools to explore at detail each one. The suggested tool should thus include a network graph for all available similarity measurements (not necessarily just the ones used in 5.2.5.2), as well as a list of the most important time series, with the most important periods highlighted for example using the strategy implemented in figure 5.38.

Since most of these features require several interactive elements, a mock-up of the desired application is available at `https://www.daqbroker.com/mvrs_mockup/`. This mock-up is not optimized for mobile displays, however, a fully functioning implementation into DAQBroker's web interface would require mobile considerations of the application.

# Chapter 6

# Results and Conclusions

This thesis presents DAQBroker, a proposal for an open and flexible framework that allows monitoring and control of a set of scientific instruments in a way that is simple for the end user but also provides the automation tools that are often available for the experienced programmer. The creation and implementation of DAQBroker in the CLOUD experiment answers the need for such a framework in this experiment, and is also an opportunity to cultivate its use and dissemination. This chapter summarizes all the current capabilities of DAQBroker and in-use learning of required improvements, as well as of near future implementations of newer capabilities. This chapter will also provide a description of long term efforts to further improve the framework and aid in its dissemination.

## 6.1 Achievements

The main achievement of this PhD is the creation of the instrument monitoring framework DAQBroker that allows users to monitor and collect instrument data by providing a universal data storing format for all instruments. This framework also allows for easy relocation of instruments between sites, provides users with an API for third-party applications to be run and allows users to gather different information for one definite experimental run by providing tools to flag the time periods belonging to the different experiments.

The current version of DAQBroker provides users with two main applications:

- A server application responsible for providing the web server as well as ensuring connection and monitoring of instruments associated with different database servers.
- An agent application to be installed in remote machines and ensures connection to the server application, allowing said server to request simple machine metrics and the most recent data from instruments connected to the remote machine

While the agent application is used to connect remote instruments to one or several running server applications and

receive orders from them, the server application provides several functionalities:

- A main server process that continuously queries a database of existing instruments and networked computers and sends orders to agents to collect new data,
- A logging process that stores relevant information about the data collection of each instrument (this process also exists for each agent application),
- A REST API that provide methods for third party users to access the data collected by the server,
- A web application that allows users to create and edit new virtual representations of instruments, access the real time data provided provided by those representations and control all the features of the DAQBroker framework, given the proper login credentials.

DAQBroker's performance was tested using both low and mid/high performance machines (both commercially available). These tests were aimed at achieving quantifiable limits of the framework with specific hardware limitations.

- Generating a predefined number of copies of the same instrument, the gauging server performance was tested as instrument number increased by measuring the collection time for each instrument compared to the set collection period for that instrument. For an increasing set of instruments and regardless of machine, it was clear that the collection time degraded and became more unreliable. Another conclusion was that the software behind the framework is I/O bound, as expected for the strong application focus on reading and writing. In quantifiable terms, DAQBroker was proven to handle up to 20 separate instruments, with a total of 200 channels for a low performance machine and 2000 channels on a high performance machine generating data at 1 Hz, uploading data to the DAQBroker database every 10 seconds without exceeding more than 50% of the aforementioned period.
- For testing the performance of the framework with data generation rate, a single instrument with varying data generation rate was used. It was concluded that DAQBroker is capable of handling data generation speeds up to 100 Hz (although this is taxing on a low performance machine).
- The performance in data request operations, to perform tasks such as data visualization was also tested. The optimized queries in DAQBroker are able to provide a response that spans up to 800000 data points in around 1 second for the low performance machine and around 0.1 seconds in the high performance machine whether that instrument contains 0 entries or $10^6$ entries.

The suite of tools provided by DAQBroker for both online and offline analysis is described and applied to several examples. A study of the temperature stability of the CLOUD experiment leveraged both DAQBroker's data warehouse model for instruments and its API to create an easily accessible database for further analysis of the stability of the experiment. Two other online examples were provided to showcase the ability for DAQBroker to generate custom data channels with semi-processed data, one creating volatile visualization channels with temporary calculations of mass spectrometry temperatures and another generating more permanent channels using data from many instruments to obtain comprehensive particle generation rates.

Apart from the specific uses of DAQBroker in the CLOUD experiment the possibility of creating of visualizations of monitored or saved data, selecting some channels or semi-processed data, DAQBroker aims at providing users with optional general methods of data analysis, applicable to any time series. Three methods of univariate time series analysis were introduced and tested on a limited set of data from 5 data channels of different instruments used during the CLOUD experimental campaign of the fall of 2017:

- The first method allows the evaluation of stationarity intervals of a time series by applying statistical tests to said series that test the time series for specific features, stationarity, trend, unit roots and seasonality. The implementation of this method allows users to control the limit of the validity of the statistical tests and select the specific tests required.
- The second method allows for time series comparison using the local similarity statistic. It was shown that this statistic agrees with the standard Pearson correlation coefficient for direct comparison but it extends the comparison by providing the possibility of introducing a time delay to emphasize delayed similarities. The evaluation of this statistic is computationally heavy and thus the method is considered only for comparison of pre-selected features of input parameters of the CLOUD data. The use of this method becomes prohibitive in an online environment for numerous and large features (in terms of data points) and sizable delays. It was concluded that the useful implementation of this method in DAQBroker implies several limitations on the algorithm to allow for smooth feature comparison.
- The third and final method is event detection. It consists of introducing an unsupervised event detection algorithm based on maximum entropy partitioning of the wavelet transform of a selected time period of the data channel. A dissimilarity measure is used to compare how the probability vector of one time period varies from another. If this dissimilarity measure surpasses a user-defined value, the method flags the time period as being sufficiently different to mark an event. With CLOUD data, it was shown that when a proper time interval is selected, this method accurately flags time periods associated with abrupt changes in the general behavior of the time series. The implementation of this method in DAQBroker requests users to supply the limit above which an event is flagged as well as the size of the features to be analyzed for the algorithm then flags events highlighting the corresponding time periods.

To complement the referred analysis, a set of statistical analysis tools is also considered for implementation in DAQBroker, dubbed the *multivariate run summary*. This analysis aimed at producing automated experimental run summaries using DAQBroker's ability to generate experimental logs, defining certain time periods as relevant for a specific scientific effort. The generation of these automated summaries includes three distinct time series analysis processes to the user-defined set of various data channels collected by DAQBroker from several machines. The first process is a principal component analysis of the frequency spectrum of the selected multivariate time series in order to retrieve the most relevant time series of the set according to a certain user-defined percentage of explained frequency variance. This application to CLOUD data revealed that in all cases, several channels that were expected to be relevant were properly highlighted. However, some limitations were discovered mostly related to improper pre-processing of data channels. The second process consists of the generation of network graphs based on the coherence and partial coherence of pairs of time series, which allows for visual inspection of the relationships that exist between the different time series in the set. Application of this process to CLOUD

data produced insights into the structure of the data in CLOUD, such that a complex relation exists between all instruments and particle measurement instruments and that all time series in the particle generation experiments can be considered in one of two communities which indicates that the CLOUD experiment is dominated by two different main sets of frequencies. The third process consists in a generalized version of the aforementioned event detection method applied to a multivariate time series. The focus of this method is to generate a ranked subset of periods from the main time period of experimental run ordered by the number of events found for each time series. Application to CLOUD data showed that a common limiting value for event detection for all time series results in an incorrect ranking of events, at least according to visual inspection of the data. Although more refinement is required for these processes to be utilized as an automated run summary tool, it is an important first step for providing the users with an automated and dynamic tool to generate a comprehensible analysis of very large set of time series and its implementation in DAQBroker is definitely an important added value.

DAQBroker aims to be a constantly evolving process and as such several other possibilities for tools to leverage time series statistics to aid users. Two possibilities that are currently being considered are the following:

1. *Multivariate time series modelling* - this effort aims to provide users with a tool that allows the definition of a theoretical model for the behavior of a set of monitored time series. With the model and the time series being collected in real time, provide either metrics on how the model is performing against the collected values and/ estimate future values based on current time series values. These results can be used, among others, to automate scientific efforts, schedule automated alerts for users or improve the underlying model. Several methods exist to model multivariate time series using methods such as Kalman filters and Multivariate Autoregressive (MAR) mooels [354, 355]

2. *Multivariate stage clustering* - this effort aims to create tools for users to cluster sets of time series leveraging the time series partitioning already defined in 3.1.2.1. These methods would allow users to group experimental efforts according to statistics extracted from a user-defined set of time series. This allows users to easily group experimental efforts to more easily extract meaningful information from their data. Several methods exist for clustering time series most of which focus on calculating similarity measurements between multivariate time series [293, 356]

DAQBroker is thus introduced as an open source framework, aimed for use in scientific environments where fast and modular access to a set of instruments is required within a team of one or more scientists spanning a wide spectrum of knowledge about the instruments used. The framework is available for download at in many formats, from simple executables to the python module, following an open source code policy (the repository can be found at ).

## 6.2 Future Work

Apart from the time series analysis methods presented, several other features are scheduled to be implemented in DAQBroker with the intent to make this framework an open and accessible way of not only monitoring and manipulate instrument data but also providing tools to control instruments as well. These features vary significantly

in scope, ambition and overall knowledge of the challenges thus this section will be divided into short-term goals and long-term goals.

In terms of short-term goals, several small optimization steps are required to increase individual instrument, database and database engine scalability. Currently the application is optimized for the MySQL database engine and several queries should be updated to be compatible with all SQL engines. Updates to the custom channel scripting language must also be implemented, such as including basic derivation and histogram operations. Another important improvement to apply is a new data gathering method that allows the parsing of data in binary files (for example, the HDF5 format) since more specialized instruments will often create these files. The OPeNDAP project contains several modules for handling popular binary files, these modules should be able to be leveraged to be used in DAQBroker to allow for many more instruments to be monitored without the need for a previous external file processing step. However a simple interface must be designed to allow users to provide the proper information to be stored in the database. As mentioned previously, several statistical methods are also planned to be implemented to increase automated analysis of data, providing users and their peers with more information about their time series. Several improvements to the web application are also expected in order to improve both client and server side performance and provide the user with more information when exceptions occur. Other improvements include automatic contacts to the instrument operator when a machine is reaching resource capacity, to allow actions to be taken to prevent potential loss of data. In order to increase the dissemination of the framework, attempts to generate partnerships with universities and institutes are planned. This could come in the form of personalized distributions of DAQBroker and regular instructional seminars.

On the side of the development of DAQBroker, it was noted in chapter 4 that there were limitations found when implementing DAQBroker in the CLOUD experiment due to the lack of testing and quality assessment mainly of the front-end user interface but also on some of the back-end processes. One goal to strive for in DAQBroker is to introduce the concept of test-driven-development to the application with the creation of unit tests that must successfully pass before a new feature pushed to production.

In terms of long-term goals, one of the major milestones aimed by DAQBroker is providing users the ability to control their instruments via the framework itself. Whether by working with the instrument manufacturers to create custom built controls for the web application or to provide general purpose user interface tools and server side requests for setting instrument parameters. The former would require first the large dissemination of the framework itself to convince instrument manufacturers to work with DAQBroker. This is already possible for specific cases by altering the source code of DAQBroker to include specialized modes for each instrument. The latter option requires changing the instrument model introduced in chapter 3 to include a control module parallel to data source modules to independently handle control and data acquisition. Another large long-term feature to be added is the possibility of using a non relational database model for data storage, which would allow DAQBroker to run without the need to connect to a relational database. In order for this change to be done, the methods for data fetching must switch from database queries to reading files from the disk, this change is subject to unpredictable errors coming from various sources such as the file syncing algorithm and other read requests. While the existing

model is appropriate for storing data from most instruments, there are still instruments that are not compatible with this format without some sort of formatting of the output format. One way of dealing with this issue is to create non-SQL methods of data storage that would rely only on the sources of instrument data, without having to parse said data until a data query is performed. This would require a large overhaul of the current instrument model and the whole framework itself, but would make DAQBroker a truly universally compatible tool for instrument data monitoring. A possible long-term attempts to disseminate the framework would come through associating with known brands of hardware manufacturers or cloud service providers to supply the framework as a supported service.

# Bibliography


[1] P. Agnes, I. Albuquerque, T. Alexander, A. Alton, K. Arisaka, D. M. Asner, M. Ave, H. O. Back, B. Baldin, K. Biery, et al. The electronics, trigger and data acquisition system for the liquid argon time projection chamber of the darkside-50 search for dark matter. *Journal of Instrumentation*, 12(12):P12011, 2017.

[2] M. Ergeneci, K. Gokcesu, E. Ertan, and P. Kosmas. An embedded, eight channel, noise canceling, wireless, wearable semg data acquisition system with adaptive muscle contraction detection. *IEEE transactions on biomedical circuits and systems*, 12(1):68–79, 2018.

[3] B. Li, J. Li, X. Lan, Y. An, W. Gao, and Y. Jiang. Experiences of building a medical data acquisition system based on two-level modeling. *International journal of medical informatics*, 112:114–122, 2018.

[4] H. Bhin, Y. Lim, S. Park, and J. Choi. Automated psychophysical personality data acquisition system for human-robot interaction. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2017 14th International Conference on*, pages 159–160. IEEE, 2017.

[5] J. B. Roeber, S. K. Pitla, R. M. Hoy, J. D. Luck, and M. F. Kocher. Tractor power take-off torque measurement and data acquisition system. *Applied Engineering in Agriculture*, 33(5):679–686, 2017.

[6] D. Svirida, D. Collaboration, et al. Electronics of the data acquisition system of the danss detector based on silicon photomultipliers. *Physics of Particles and Nuclei*, 49(1):84–85, 2018.

[7] R. Abbasi, M. Ackermann, J. Adams, M. Ahlers, J. Ahrens, K. Andeen, J. Auffenberg, X. Bai, M. Baker, S. Barwick, et al. The icecube data acquisition system: Signal capture, digitization, and timestamping. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 601(3):294–316, 2009.

[8] P. Hari, E. Nikinmaa, T. Pohja, E. Siivola, J. Bäck, T. Vesala, and M. Kulmala. Station for measuring ecosystem-atmosphere relations: Smear. In *Physical and Physiological Forest Ecology*, pages 471–487. Springer, 2013.

[9] J. P. Grotzinger, J. Crisp, A. R. Vasavada, R. C. Anderson, C. J. Baker, R. Barry, D. F. Blake, P. Conrad, K. S. Edgett, B. Ferdowski, et al. Mars science laboratory mission and science investigation. *Space science reviews*, 170(1-4):5–56, 2012.

[10] X. Liu, L. G. Huey, R. J. Yokelson, V. Selimovic, I. J. Simpson, M. Müller, J. L. Jimenez, P. Campuzano-Jost, A. J. Beyersdorf, D. R. Blake, et al. Airborne measurements of western us wildfire emissions: Comparison

with prescribed burning and air quality implications. *Journal of Geophysical Research: Atmospheres*, 122 (11):6108–6129, 2017.

[11] J. D. Adams. Scientific teams and institution collaborations: Evidence from u.s. universities, 1981-1999. Working Paper 10640, National Bureau of Economic Research, July 2004. URL http://www.nber.org/papers/w10640.

[12] C. S. Wagner, T. A. Whetsell, and L. Leydesdorff. Growth of international collaboration in science: revisiting six specialties. *Scientometrics*, 110(3):1633–1652, 2017. URL https://EconPapers.repec.org/RePEc:spr:scient:v:110:y:2017:i:3:d:10.1007_s11192-016-2230-9.

[13] J. Oliveira e Sá, J. C. Sá, J. L. Pereira, F. Pimenta, and M. Monteiro. Internet of things: An evolution of development and research area topics. 2017.

[14] D. Hortelano, T. Olivares, M. C. Ruiz, C. Garrido-Hidalgo, and V. López. From sensor networks to internet of things. bluetooth low energy, a standard for this evolution. *Sensors*, 17(2):372, 2017.

[15] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber. Scientific data management in the coming decade. *Acm Sigmod Record*, 34(4):34–41, 2005.

[16] T. Hey, S. Tansley, K. M. Tolle, et al. *The fourth paradigm: data-intensive scientific discovery*, volume 1. Microsoft research Redmond, WA, 2009.

[17] H. Junninen, A. Lauri, P. Keronen, P. Aalto, V. Hiltunen, P. Hari, and M. Kulmala. Smart-smear: on-line data exploration and visualization tool for smear stations. *Boreal Env. Res*, 14:447–457, 2009.

[18] C. Currey, A. Bartle, C. Lukashin, C. Roithmayr, and J. Gallagher. Multi-instrument inter-calibration (MIIC) system. *Remote Sensing*, 8(11):902, nov 2016. doi: 10.3390/rs8110902. URL https://doi.org/10.3390%2Frs8110902.

[19] D. Tescaro, A. López-Oramas, A. Moralejo, D. Mazin, et al. The magic telescopes daq software and the on-the-fly online analysis client. *arXiv preprint arXiv:1310.1565*, 2013.

[20] N. Instruments. What is labview?, Mar. 2018. URL http://www.ni.com/en-us/shop/labview.html.

[21] ADLink. Daq software & utility - adlink technology, Mar. 2018. URL https://emb.adlinktech.com/en/Data_Acquisition_DAQ_Software_Utility.aspx.

[22] Z. Hons. A versatile daq, monitoring and data processing system for nuclear experiments in camac and vme standards. *arXiv preprint arXiv:1508.01379*, 2015.

[23] T. Kos, T. Kosar, and M. Mernik. Development of data acquisition systems by using a domain-specific modeling language. *Computers in Industry*, 63(3):181 – 192, 2012. ISSN 0166-3615. doi: https://doi.org/10.1016/j.compind.2011.09.004. URL http://www.sciencedirect.com/science/article/pii/S0166361511001059.

[24] C. Hennig, K. Kneupner, and D. Kinna. Connecting programmable logic controllers (plc) to control and data acquisition a comparison of the jet and wendelstein 7-x approach. *Fusion Engineering and Design*,

87(12):1972 – 1976, 2012. ISSN 0920-3796. doi: https://doi.org/10.1016/j.fusengdes.2012.05.009. URL http://www.sciencedirect.com/science/article/pii/S0920379612002967. Proceedings of the 8th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research.

[25] K. Stouffer, J. Falco, and K. Scarfone. Guide to industrial control systems (ics) security. *NIST special publication*, 800(82):16–16, 2011.

[26] C. Markman, A. Wool, and A. A. Cardenas. A new burst-dfa model for scada anomaly detection. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*, pages 1–12. ACM, 2017.

[27] Z. Lü, Y. Lü, M. Yuan, and Z. Wang. A heterogeneous large-scale parallel scada/dcs architecture in 5g ogce. In *Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2017 10th International Congress on*, pages 1–7. IEEE, 2017.

[28] Zabbix. Zabbix :: The enterprise-class open source network monitoring solution, Mar. 2018. URL https://www.zabbix.com/.

[29] A. Dias, J. Almeida, A. Amorim, A. David, A. Tomé, C. Collaboration, et al. The cloud data acquisition system and online derivation of nucleation rates. In *AIP Conference Proceedings*, volume 1527, pages 393–396. AIP, 2013.

[30] D. Elia, G. Vino, S. Bagnasco, A. Crescente, G. Donvito, A. Franco, S. Lusso, D. Mura, S. Piano, G. Platania, et al. A dashboard for the italian computing in alice. In *Journal of Physics: Conference Series*, volume 898, page 092054. IOP Publishing, 2017.

[31] J.-H. Um, S. Han, H. Kim, and K. Park. Massive oceancolor data processing and analysis system: Tupix-oc. In *e-Science (e-Science), 2017 IEEE 13th International Conference on*, pages 450–451. IEEE, 2017.

[32] J. Liu, P. Zhang, and F. Wang. Real-time dc servo motor position control by pid controller using labview. In *Intelligent Human-Machine Systems and Cybernetics, 2009. IHMSC'09. International Conference on*, volume 1, pages 206–209. IEEE, 2009.

[33] S. N. Karlsdottir and J. W. Halloran. Rapid oxidation characterization of ultra-high temperature ceramics. *Journal of the American Ceramic Society*, 90(10):3233–3238, 2007.

[34] S. Laha, S. Mahindar, S. Deb, K. Hati, and D. Sur. Mind controlled automation system with eeg. *International Journal of Scientific & Engineering Research*, 8(3):17–18, 2017.

[35] MathWorks. Matlab - mathworks, Mar. 2018. URL https://www.mathworks.com/products/matlab.html.

[36] I. Colak, S. Demirbas, I. Sefa, E. Irmak, and H. T. Kahraman. Remote controlling and monitoring of hvac system over internet. 2008.

[37] Y.-S. Lee, J.-H. Yang, S.-Y. Kim, W.-S. Kim, and O.-K. Kwon. Development of a rapid control prototyping system based on matlab and usb daq boards. *Journal of Institute of Control, Robotics and Systems*, 18(10): 912–920, 2012.

[38] B. Englitz, S. V. David, M. D. Sorenson, and S. A. Shamma. Manta—an open-source, high density electrophysiology recording suite for matlab. *Frontiers in neural circuits*, 7:69, 2013.

[39] GitHub. Pypl popularity of programming language, Mar. 2018. URL `https://pypl.github.io/PYPL.html`.

[40] G. Labs. Grafana - the open platform for analytics and monitoring, Mar. 2018. URL `https://grafana.com/`.

[41] S. Eisele, A. Dubey, G. Karsai, and S. Lukic. Wip abstract: Transactive energy demo with riaps platform. In *Cyber-Physical Systems (ICCPS), 2017 ACM/IEEE 8th International Conference on*, pages 91–92. IEEE, 2017.

[42] R. Fay, J. Bland, and S. Jones. Next generation monitoring: Tier 2 experience. In *Journal of Physics: Conference Series*, volume 898, page 092035. IOP Publishing, 2017.

[43] E. Casalicchio and V. Perciballi. Measuring docker performance: What a mess!!! In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*, pages 11–16. ACM, 2017.

[44] OPeNDAP. Opendap - advanced software for remote data retrieval, Mar. 2018. URL `https://www.opendap.org/`.

[45] M. M. Rienecker, M. J. Suarez, R. Gelaro, R. Todling, J. Bacmeister, E. Liu, M. G. Bosilovich, S. D. Schubert, L. Takacs, G.-K. Kim, et al. Merra: Nasa's modern-era retrospective analysis for research and applications. *Journal of climate*, 24(14):3624–3648, 2011.

[46] S. C. Hankin, J. D. Blower, T. Carval, K. S. Casey, C. Donlon, O. Lauret, T. Loubrieu, A. Srinivasan, J. Trinanes, O. Godoy, et al. Netcdf-cf-opendap: Standards for ocean data interoperability and object lessons for community data standards processes. In *Oceanobs 2009, Venice Convention Centre, 21-25 septembre 2009, Venise*, 2010.

[47] F. Baart, G. de Boer, W. de Haas, G. Donchyts, M. Philippart, M. van Koningsveld, and M. Plieger. A comparison between wcs and opendap for making model results and data products available through the internet. *Transactions in GIS*, 16(2):249–265, 2012.

[48] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of internet content delivery systems. *ACM SIGOPS Operating Systems Review*, 36(SI):315–327, 2002.

[49] G. Pallis and A. Vakali. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1):101–106, 2006.

[50] A. Vakali and G. Pallis. Content delivery networks: Status and trends. *IEEE Internet Computing*, 7(6): 68–74, 2003.

[51] D. S. Simbeye, J. Zhao, and S. Yang. Design and deployment of wireless sensor networks for aquaculture monitoring and control based on virtual instruments. *Computers and Electronics in Agriculture*, 102:31–42, 2014.

[52] J. E. McNamara. *Technical aspects of data communication*. Digital Press, 2014.

[53] D. M. Cornwell. Nasa's optical communications program for 2015 and beyond. In *Free-Space Laser Communication and Atmospheric Propagation XXVII*, volume 9354, page 93540E. International Society for Optics and Photonics, 2015.

[54] F. Heine, G. Mühlnikel, H. Zech, D. Tröndle, S. Seel, M. Motzigemba, R. Meyer, S. Philipp-May, and E. Benzi. Lct for the european data relay system: in orbit commissioning of the alphasat and sentinel 1a lcts. In *Free-Space Laser Communication and Atmospheric Propagation XXVII*, volume 9354, page 93540G. International Society for Optics and Photonics, 2015.

[55] M. M. U. Rathore, A. Paul, A. Ahmad, B.-W. Chen, B. Huang, and W. Ji. Real-time big data analytical architecture for remote sensing application. *IEEE journal of selected topics in applied earth observations and remote sensing*, 8(10):4610–4621, 2015.

[56] I. Stojmenovic. Machine-to-machine communications with in-network data aggregation, processing, and actuation for large-scale cyber-physical systems. *IEEE Internet of Things Journal*, 1(2):122–128, 2014.

[57] H. Dhola, A. Patel, A. Thakar, N. Singh, R. Dave, D. Parmar, K. Mehta, N. Goswami, S. Gajjar, and U. Baruah. Data transfer methods in real-time controller of ion cyclotron high-voltage power supply. *IEEE Transactions on Nuclear Science*, 65(2):828–835, 2018.

[58] Y.-y. Fang and X.-j. Chen. Design and simulation of uart serial communication module based on vhdl. In *Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on*, pages 1–4. IEEE, 2011.

[59] J. Axelson. *Serial Port Complete: The Developer's Guide*. Lakeview Research LLC, 2007.

[60] I. J. Wickelgren. The facts about firewire [serial communication bus]. *IEEE spectrum*, 34(4):19–25, 1997.

[61] A. Amelia, B. V. Sundawa, M. Pardede, W. Sutrisno, M. Rusdi, et al. Implementation of the rs232 communication trainer using computers and the atmega microcontroller for interface engineering courses. In *Journal of Physics: Conference Series*, volume 890, page 012095. IOP Publishing, 2017.

[62] R. Przesmycki. Rs232 interface in the process of electromagnetic infiltration. In *Progress in Electromagnetics Research Symposium-Fall (PIERS-FALL), 2017*, pages 350–356. IEEE, 2017.

[63] P. J. Dickhudt and K. K. Hathaway. Custom data logger for real-time remote field data collections. Technical report, Army Engineer Research and Development Center Vicksburg United States, 2017.

[64] J. Iamsamang, N. Bijaphala, P. Boonperm, R. Lordthong, and P. Naiyanetr. A modular vital sign monitoring based-on usb communication. In *Biomedical Engineering (BioMed), 2017 13th IASTED International Conference on*, pages 241–246. IEEE, 2017.

[65] P. A. Pattanaik, N. Sahoo, and S. Mishra. Implementation of demand side management using microcontroller and wireless communication. In *Electrical, Computer and Communication Technologies (ICECCT), 2017 Second International Conference on*, pages 1–6. IEEE, 2017.

[66] E. Mastinu, B. Håkansson, and M. Ortiz-Catalan. Low-cost, open source bioelectric signal acquisition sys-

tem. In *Wearable and Implantable Body Sensor Networks (BSN), 2017 IEEE 14th International Conference on*, pages 19–22. IEEE, 2017.

[67] Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 15.1a: Wireless medium access control (mac) and physical layer (phy) specifications for wireless personal area networks (wpan). *IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002)*, pages 1–700, June 2005. doi: 10.1109/IEEESTD.2005.96290.

[68] R. Piyare and M. Tazil. Bluetooth based home automation system using cell phone. In *Consumer Electronics (ISCE), 2011 IEEE 15th International Symposium on*, pages 192–195. IEEE, 2011.

[69] J. Machacek and J. Drapela. Control of serial port (rs-232) communication in labview. In *Modern Technique and Technologies, 2008. MTT 2008. International Conference*, pages 36–40. IEEE, 2008.

[70] L. Wang, Z. Liu, G. Zhang, and L. Zhao. A multi-channel distributed monitoring platform based on ethernet and virtual instrument technology. In *Electronic Measurement & Instruments, 2009. ICEMI'09. 9th International Conference on*, pages 4–579. IEEE, 2009.

[71] G. C. Walsh, H. Ye, and L. G. Bushnell. Stability analysis of networked control systems. *IEEE transactions on control systems technology*, 10(3):438–446, 2002.

[72] D. G. Leeper. A long-term view of short-range wireless. *Computer*, 34(6):39–44, 2001.

[73] Information technology - open systems interconnection - basic reference model: The basic model, 1994.

[74] V. Cerf. Specification of internet transmission control program. Rfc, RFC Editor, Dec. 1974. URL `https://www.ietf.org/rfc/rfc0675.txt`.

[75] J. Postel. User datagram protocol. Rfc, RFC Editor, Aug. 1980. URL `https://www.ietf.org/rfc/rfc768.txt`.

[76] N. Instruments. Connecting instruments via ethernet/lan, Mar. 2018. URL `http://www.ni.com/getting-started/set-up-hardware/instrument-control/ethernet-connect`.

[77] F. D'Alessio, A. Di Cianno, A. Di Paola, C. Giuliani, D. Guidubaldi, D. Lorenzetti, E. Micolucci, F. Pedichini, R. Speziali, G. Valentini, et al. Swircam: a nir imager-spectrometer to search for extragalactic supernovae. In *Optical and IR Telescope Instrumentation and Detectors*, volume 4008, pages 748–759. International Society for Optics and Photonics, 2000.

[78] P. Nikiel, S. Schlenker, V. Filimonov, B. Farnham, and H. Boterenbrood. Opc unified architecture within the control system of the atlas experiment. 2014.

[79] V. I. T. Association et al. *The VMEbus specification*. VMEbus International Trade Association, 1987.

[80] P. Semiconductors. Canbus specification, 1996.

[81] V. Kostakos and D. Ferreira. The rise of ubiquitous instrumentation. *Frontiers in ICT*, 2:3, 2015. ISSN 2297-198X. doi: 10.3389/fict.2015.00003. URL `https://www.frontiersin.org/article/10.3389/fict.2015.00003`.

[82] G. Caesar. Integrating pxi with vxi, gpib, usb, and lxi instrumentation. In *Autotestcon, 2005. IEEE*, pages 857–861. IEEE, 2005.

[83] J.-H. Won, T. Pany, and G. W. Hein. Gnss software defined radio. *Inside GNSS*, 1(5):48–56, 2006.

[84] J. P. Grinias, J. T. Whitfield, E. D. Guetschow, and R. T. Kennedy. An inexpensive, open-source USB arduino data acquisition device for chemical instrumentation. *Journal of Chemical Education*, 93(7):1316–1319, June 2016. doi: 10.1021/acs.jchemed.6b00262. URL `https://doi.org/10.1021/acs.jchemed.6b00262`.

[85] G. Jung, S. Kim, and J. G. Kim. Skin condition measurement by using multispectral imaging system (conference presentation). In *Photonics in Dermatology and Plastic Surgery*, volume 10037, page 100370Q. International Society for Optics and Photonics, 2017.

[86] Y. H. Santana, D. M. Betancourt, A. I. M. Valdes, Y. H. Sanchez, G. A. G. Nieto, and R. M. Alonso. Dtmb monitoring tool based on a commercial set-top box. In *Broadband Multimedia Systems and Broadcasting (BMSB), 2017 IEEE International Symposium on*, pages 1–6. IEEE, 2017.

[87] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of network and computer applications*, 23(3):187–200, 2000.

[88] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3, 2016.

[89] B. W. Kernighan and D. M. Ritchie. *The C programming language*. Englewood Cliffs: Prentice Hall, 1988.

[90] N. B. Dale and J. Lewis. *Computer science illuminated*. Jones & Bartlett Learning, 2007.

[91] G. M. Cartwright, C. T. Friedrichs, P. J. Dickhudt, T. Gass, and F. H. Farmer. Using the acoustic doppler velocimeter (adv) in the mudbed real-time observing system. In *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*, pages 1–9. IEEE, 2009.

[92] L. Sokoloff. Gpib instrument control. In *ASEE Annual Conference Proceedings*, 2002.

[93] C. Tilmes, J. Arnfield, and A. Fleig. Data formats: Using self-describing data formats. 2011.

[94] R. Rew and G. Davis. Netcdf: an interface for scientific data access. *IEEE computer graphics and applications*, 10(4):76–82, 1990.

[95] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson. An overview of the hdf5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, pages 36–47. ACM, 2011.

[96] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408. ACM, 2003.

[97] P. A. Harris, R. Taylor, R. Thielke, J. Payne, N. Gonzalez, and J. G. Conde. Research electronic data capture (redcap)—a metadata-driven methodology and workflow process for providing translational research informatics support. *Journal of biomedical informatics*, 42(2):377–381, 2009.

[98] C. Bisdikian. An overview of the bluetooth wireless technology. *IEEE Communications magazine*, 39(12): 86–94, 2001.

[99] G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin. The stanford testbed of autonomous rotorcraft for multi agent control (starmac). In *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*, volume 2, pages 12–E. IEEE, 2004.

[100] M. Mathis, J. Heffner, and R. Reddy. Web100: extended tcp instrumentation for research, education and diagnosis. *ACM SIGCOMM Computer Communication Review*, 33(3):69–79, 2003.

[101] T. A. Lutz. Using tcp/ip as an instrument interface. *Sensors-the Journal of Applied Sensing Technology*, 15 (7):43–47, 1998.

[102] A. Tura, P. Santini, D. Longo, and L. Quareni. A telemedicine instrument for home monitoring of patients with chronic respiratory diseases. *Annali dell'Istituto superiore di sanità*, 43(1):101–109, 2007.

[103] S. Tilak, P. Hubbard, M. Miller, and T. Fountain. The ring buffer network bus (rbnb) dataturbine streaming data middleware for environmental observing systems. In *e-Science and Grid Computing, IEEE International Conference on*, pages 125–133. IEEE, 2007.

[104] S. Parkin and S.-H. Yang. Memory on the racetrack. *Nature nanotechnology*, 10(3):195–198, 2015.

[105] E. M. Hart, P. Barmby, D. LeBauer, F. Michonneau, S. Mount, P. Mulrooney, T. Poisot, K. H. Woo, N. B. Zimmerman, and J. W. Hollister. Ten simple rules for digital data storage. *PLoS computational biology*, 12 (10):e1005097, 2016.

[106] G. Lu, L. Ho, R. Danilak, R. N. Mullendore, J. Jones, and A. J. Tomlin. Data reliability schemes for data storage systems, Apr. 28 2015. US Patent 9,021,339.

[107] N. Marz and J. Warren. *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015.

[108] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47:98–115, 2015.

[109] A. Achpal, V. B. Kumar, and K. Mahesh. Modeling ontology semantic constraints in relational database management system. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 2016.

[110] D. Van Aken, A. Pavlo, G. J. Gordon, and B. Zhang. Automatic database management system tuning through large-scale machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1009–1024. ACM, 2017.

[111] R. Winkelmann, K. Jaensch, S. Cassidy, and J. Harrington. emur: Main package of the emu speech database management system. *Version 0.2*, 1, 2016.

[112] C. Győrödi, R. Győrödi, G. Pecherle, and A. Olah. A comparative study: Mongodb vs. mysql. In *Engineering of Modern Electric Systems (EMES), 2015 13th International Conference on*, pages 1–6. IEEE, 2015.

[113] L. Ashdown and T. Kyte. Oracle database concepts. *E40540-04*, 2015.

[114] M. Wittig and A. Wittig. *Amazon web services in action*. Manning, 2016.

[115] K. L. Berg, T. Seymour, and R. Goel. History of databases. *International Journal of Management & Information Systems (Online)*, 17(1):29, 2013.

[116] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6): 377–387, jun 1970. doi: 10.1145/362384.362685. URL `https://doi.org/10.1145/362384.362685`.

[117] R. Sen, J. Chen, and N. Jimsheleishvilli. Query optimization time: The new bottleneck in real-time analytics. In *Proceedings of the 3rd VLDB Workshop on In-Memory Data Mangement and Analytics*, page 8. ACM, 2015.

[118] P. Buneman, R. E. Frankel, and R. Nikhil. An implementation technique for database query languages. *ACM Transactions on Database Systems (TODS)*, 7(2):164–186, 1982.

[119] R. Elmasri and S. Navathe. *Fundamentals of database systems*. Addison-Wesley Publishing Company, 2010.

[120] N. Leavitt. Will nosql databases live up to their promise? *Computer*, 43(2), 2010.

[121] J. Han, E. Haihong, G. Le, and J. Du. Survey on nosql database. In *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, pages 363–366. IEEE, 2011.

[122] V. Kuznetsov, D. Evans, and S. Metson. Life in extra dimensions of database world or penetration of nosql in hep community. In *Journal of Physics: Conference Series*, volume 396, page 052043. IOP Publishing, 2012.

[123] J. Becla. Lessons learned from managing a petabyte. Technical report, SLAC National Accelerator Lab., Menlo Park, CA (United States), 2018.

[124] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, and D. Gosain. A survey and comparison of relational and non-relational database. *International Journal of Engineering Research & Technology*, 1(6), 2012.

[125] I. Konstantinou, E. Angelou, C. Boumpouka, D. Tsoumakos, and N. Koziris. On the elasticity of nosql databases over cloud management platforms. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2385–2388. ACM, 2011.

[126] J. Pokorny. Nosql databases: a step to database scalability in web environment. *International Journal of Web Information Systems*, 9(1):69–82, 2013.

[127] L. M. Vaquero, L. Rodero-Merino, and R. Buyya. Dynamically scaling applications in the cloud. *ACM SIGCOMM Computer Communication Review*, 41(1):45–52, 2011.

[128] Neo4j. The neo4j graph platform – the no.1 platform for connected data, Mar. 2018. URL `https://neo4j.com/`.

[129] MongoDB. Mongodb for giant ideas — mongodb, Mar. 2018. URL `https://www.mongodb.com/`.

[130] Microsoft. Microsoft azure cloud computing platform & services, Mar. 2018. URL `https://azure.microsoft.com/en-us/`.

[131] Google. Cloud storage - online data storage — google cloud, Mar. 2018. URL `https://cloud.google.com/storage/`.

[132] J. Dittrich and J.-A. Quiané-Ruiz. Efficient big data processing in hadoop mapreduce. *Proceedings of the VLDB Endowment*, 5(12):2014–2015, 2012.

[133] R. Bourret et al. Xml and databases. 1999.

[134] D. J. Abadi, P. A. Boncz, and S. Harizopoulos. Column-oriented database systems. *Proceedings of the VLDB Endowment*, 2(2):1664–1665, 2009.

[135] I. Robinson, J. Webber, and E. Eifrem. *Graph databases*. " O'Reilly Media, Inc.", 2013.

[136] R. Hecht and S. Jablonski. Nosql evaluation: A use case oriented survey. In *Cloud and Service Computing (CSC), 2011 International Conference on*, pages 336–341. IEEE, 2011.

[137] A. Moniruzzaman and S. A. Hossain. Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. *arXiv preprint arXiv:1307.0191*, 2013.

[138] A. Nayak, A. Poriya, and D. Poojary. Type of nosql databases and its comparison with relational databases. *International Journal of Applied Information Systems*, 5(4):16–19, 2013.

[139] R. Hankins, T. Diep, M. Annavaram, B. Hirano, H. Eri, H. Nueckel, and J. P. Shen. Scaling and characterizing database workloads: Bridging the gap between research and practice. In *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pages 151–162. IEEE, 2003.

[140] S. Spinner, S. Kounev, X. Zhu, L. Lu, M. Uysal, A. Holler, and R. Griffith. Runtime vertical scaling of virtualized applications via online model estimation. In *Self-Adaptive and Self-Organizing Systems (SASO), 2014 IEEE Eighth International Conference on*, pages 157–166. IEEE, 2014.

[141] S. Dutta, S. Gera, A. Verma, and B. Viswanathan. Smartscale: Automatic application scaling in enterprise clouds. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 221–228. IEEE, 2012.

[142] F. Chong, G. Carraro, and R. Wolter. Multi-tenant data architecture. *MSDN Library, Microsoft Corporation*, pages 14–30, 2006.

[143] D. Singh and C. K. Reddy. A survey on platforms for big data analytics. *Journal of Big Data*, 2(1):8, 2015.

[144] S. Gupta and B. Kuntal Saroha. Fundamental research in distributed database. *IJCSMS*, 11(2), 2011.

[145] K. Akkaya, F. Senel, A. Thimmapuram, and S. Uludag. Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility. *IEEE Transactions on Computers*, 59(2):258–271, 2010.

[146] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News*, 33(2):51–59, 2002.

[147] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci, et al. Windows azure storage: a highly available cloud storage service with strong consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 143–157. ACM, 2011.

[148] B. A. Coan, B. M. Oki, and E. K. Kolodner. Limitations on database availability when networks partition. In *Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, pages 187–194. ACM, 1986.

[149] Apache. Distributed database - apache ignite, Mar. 2018. URL `https://ignite.apache.org/use-cases/database/distributed-database.html`.

[150] MySQL. Mysql :: Mysql cluster cge, Mar. 2018. URL `https://www.mysql.com/products/cluster/`.

[151] M. K. Sandhu, A. Kaur, and R. Kaur. Data warehouse schemas. *Int. J. of Innovative Research in Advance Engineering (IJIIRAE)*, 2:47–51, 2015.

[152] N. Tryfona, F. Busborg, and J. G. Borch Christiansen. starer: a conceptual model for data warehouse design. In *Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP*, pages 3–8. ACM, 1999.

[153] S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. *ACM Sigmod record*, 26(1):65–74, 1997.

[154] C. Adamson. *Mastering data warehouse aggregates: solutions for star schema performance*. John Wiley & Sons, 2012.

[155] A. G. Büchner and M. D. Mulvenna. Discovering internet marketing intelligence through online analytical web usage mining. *ACM Sigmod Record*, 27(4):54–61, 1998.

[156] A. Sen and A. P. Sinha. A comparison of data warehousing methodologies. *Communications of the ACM*, 48(3):79–84, 2005.

[157] D. Merrill. Mashups: The new breed of web app. *IBM Web Architecture Technical Library*, pages 1–13, 2006.

[158] A. Charland and B. Leroux. Mobile application development: web vs. native. *Communications of the ACM*, 54(5):49–53, 2011.

[159] M. Schoeberl. *Jop: A java optimized processor for embedded real-time systems*. VDM Publishing, 2008.

[160] S. Xanthopoulos and S. Xinogalos. Mobile app development in html5. In *AIP Conference Proceedings*, volume 1648, page 310009. AIP Publishing, 2015.

[161] P. Fraternali. Tools and approaches for developing data-intensive web applications: a survey. *ACM Computing Surveys (CSUR)*, 31(3):227–263, 1999.

[162] Z. Mahmood. The promise and limitations of service oriented architecture. *International journal of Computers*, 1(3):74–78, 2007.

[163] S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, 34(1):1–11, 2011.

[164] D. Scott and R. Sharp. Abstracting application-level web security. In *Proceedings of the 11th international conference on World Wide Web*, pages 396–407. ACM, 2002.

[165] J. Li, Y. Xiong, X. Liu, and L. Zhang. How does web service api evolution affect clients? In *2013 IEEE 20th International Conference on Web Services*, pages 300–307. IEEE, 2013.

[166] R. Battle and E. Benson. Bridging the semantic web and web 2.0 with representational state transfer (rest). *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):61–69, 2008.

[167] M. Lanthaler and C. Gütl. On using json-ld to create evolvable restful services. In *Proceedings of the Third International Workshop on RESTful Design*, pages 25–32. ACM, 2012.

[168] M. Lanthaler and C. Gütl. Model your application domain, not your json structures. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1415–1420. ACM, 2013.

[169] R. T. Fielding and R. N. Taylor. *Architectural styles and the design of network-based software architectures*, volume 7. University of California, Irvine Doctoral dissertation, 2000.

[170] K. Matkovic, H. Hauser, R. Sainitzer, and M. E. Groller. Process visualization with levels of detail. In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.*, pages 67–70. IEEE, 2002.

[171] S. Chien, A. Davies, D. Tran, B. Cichy, G. Rabideau, R. Castano, R. Sherwood, J. Jones, S. Grosvenor, D. Mandl, et al. Using automated planning for sensorweb response. 2004.

[172] S. Pathak and S. Pathak. Data visualization techniques, model and taxonomy. In *Data Visualization and Knowledge Engineering*, pages 249–271. Springer, 2020.

[173] M. O. Ward, G. Grinstein, and D. Keim. *Interactive data visualization: foundations, techniques, and applications*. AK Peters/CRC Press, 2015.

[174] D. Ebert, B. Fisher, and K. Gaither. Introduction to the minitrack on interactive visual analytics and visualization for decision making–making sense of a growing digital world. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[175] M. D. Ryan. Cloud computing security: The scientific challenge, and a survey of solutions. *Journal of Systems and Software*, 86(9):2263–2268, 2013.

[176] L. T. Honarvar, B. R. Witte, S. C. Fatigante, and G. L. Harless. User authentication system and methods thereof, June 12 2007. US Patent 7,231,657.

[177] E. Bertino, G. Ghinita, A. Kamra, et al. Access control for databases: concepts and systems. *Foundations and Trends® in Databases*, 3(1–2):1–148, 2011.

[178] A. Ron, A. Shulman-Peleg, and E. Bronshtein. No sql, no injection? examining nosql security. *arXiv preprint arXiv:1506.04082*, 2015.

[179] S. Som, S. Sinha, and R. Kataria. Study on sql injection attacks: Mode detection and prevention. *International Journal of Engineering Applied Sciences and Technology, Indexed in Google Scholar, ISI etc., Impact Factor: 1.494*, 1(8):23–29, 2016.

[180] M. Lawal, A. B. M. Sultan, and A. O. Shakiru. Systematic literature review on sql injection attack. *International Journal of Soft Computing*, 11(1):26–35, 2016.

[181] B. Razavi. *Principles of data conversion system design*, volume 126. IEEE press New York, 1995.

[182] C. J. De Luca, L. D. Gilmore, M. Kuznetsov, and S. H. Roy. Filtering the surface emg signal: Movement artifact and baseline noise contamination. *Journal of biomechanics*, 43(8):1573–1579, 2010.

[183] J. L. Doob and J. L. Doob. *Stochastic processes*, volume 7. Wiley New York, 1953.

[184] A. Melanson and A. Longtin. Data-driven inference for jump-diffusion models of neuroscientific data. *Bulletin of the American Physical Society*, 2018.

[185] T. M. Earnest, J. A. Cole, and Z. Luthey-Schulten. Simulating biological processes: stochastic physics from whole cells to colonies. *Reports on Progress in Physics*, 81(5):052601, 2018.

[186] J. Fan and Q. Yao. *Nonlinear time series: nonparametric and parametric methods*. Springer Science & Business Media, 2008.

[187] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[188] P. Billingsley. *Probability And Measure*. WI, 1995. ISBN 8126517719. URL `https://www.amazon.com/Probability-Measure-Patrick-Billingsley/dp/8126517719?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=8126517719`.

[189] C. Chatfield. *The analysis of time series: an introduction*. CRC press, 2016.

[190] J. D. Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[191] W. A. Fuller. *Introduction to statistical time series*, volume 428. John Wiley & Sons, 2009.

[192] J. Geweke. Measurement of linear dependence and feedback between multiple time series. *Journal of the American statistical association*, 77(378):304–313, 1982.

[193] M. B. Priestley. Spectral analysis and time series. 1981.

[194] U. K. Müller. Size and power of tests of stationarity in highly autocorrelated time series. *Journal of Econometrics*, 128(2):195–213, 2005.

[195] E. Marshall and E. Boggis. The statistics tutor's quick guide to commonly used statistical tests. *University of Sheffield. Available online: http://www. statstutor. ac. uk/resources/uploaded/tutorsquickguidetostatistics. pdf*, 2016.

[196] G. M. Sullivan and R. Feinn. Using effect size—or why the p value is not enough. *Journal of graduate medical education*, 4(3):279–282, 2012.

[197] M. D. Lieberman and W. A. Cunningham. Type i and type ii error concerns in fmri research: re-balancing the scale. *Social cognitive and affective neuroscience*, 4(4):423–428, 2009.

[198] M. Lanne and P. Saikkonen. Reducing size distortions of parametric stationarity tests. *Journal of Time Series Analysis*, 24(4):423–439, 2003.

[199] D. Kwiatkowski, P. C. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of econometrics*, 54(1-3):159–178, 1992.

[200] M. B. Priestley. Non-linear and non-stationary time series analysis. 1988.

[201] A. Banerjee, J. J. Dolado, J. W. Galbraith, D. Hendry, et al. Co-integration, error correction, and the econometric analysis of non-stationary data. *OUP Catalogue*, 1993.

[202] C. R. Nelson and C. R. Plosser. Trends and random walks in macroeconmic time series: some evidence and implications. *Journal of monetary economics*, 10(2):139–162, 1982.

[203] H. Akaike. Fitting autoregressive models for prediction. *Annals of the institute of Statistical Mathematics*, 21(1):243–247, 1969.

[204] P. C. Phillips and P. Perron. Testing for a unit root in time series regression. *Biometrika*, 75(2):335–346, 1988.

[205] G. M. Goerg. Testing for white noise against locally stationary alternatives. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(6):478–492, 2012. doi: 10.1002/sam.11157. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11157`.

[206] N. Yuan, Z. Fu, and S. Liu. Extracting climate memory using fractional integrated statistical model: A new perspective on climate prediction. *Scientific reports*, 4:6577, 2014.

[207] J. H. Cochrane. [pitfalls and opportunities: What macroeconomists should know about unit roots]: Comment. *NBER macroeconomics annual*, 6:201–210, 1991.

[208] Y.-W. Cheung and K. S. Lai. Lag order and critical values of the augmented dickey–fuller test. *Journal of Business & Economic Statistics*, 13(3):277–280, 1995.

[209] S. N. Durlauf and P. C. Phillips. Trends versus random walks in time series analysis. *Econometrica: Journal of the Econometric Society*, pages 1333–1354, 1988.

[210] A. Atkinson, S. Koopman, and N. Shephard. Detecting shocks: Outliers and breaks in time series. *Journal of Econometrics*, 80(2):387 – 422, 1997. ISSN 0304-4076. doi: https://doi.org/10.1016/S0304-4076(97)00050-X. URL http://www.sciencedirect.com/science/article/pii/S030440769700050X.

[211] H. Andersson. Are commodity prices mean reverting? *Applied Financial Economics*, 17(10):769–783, 2007.

[212] S. Fatichi, S. Barbosa, E. Caporali, and M. Silva. Deterministic versus stochastic trends: Detection and challenges. *Journal of Geophysical Research: Atmospheres*, 114(D18), 2009.

[213] A. K. Wagner, S. B. Soumerai, F. Zhang, and D. Ross-Degnan. Segmented regression analysis of interrupted time series studies in medication use research. *Journal of clinical pharmacy and therapeutics*, 27(4):299–309, 2002.

[214] P. B. Kenny and J. Durbin. Local trend estimation and seasonal adjustment of economic and social time series. *Journal of the Royal Statistical Society. Series A (General)*, pages 1–41, 1982.

[215] H. B. Mann. Nonparametric tests against trend. *Econometrica: Journal of the Econometric Society*, pages 245–259, 1945.

[216] M. G. Kendall. Rank correlation methods. 1955.

[217] S. Hylleberg, C. Jørgensen, and N. K. Sørensen. Seasonality in macroeconomic time series. *Empirical Economics*, 18(2):321–335, 1993.

[218] J. H. F. Flores, P. M. Engel, and R. C. Pinto. Autocorrelation and partial autocorrelation functions to improve neural networks models on univariate time series forecasting. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE, 2012.

[219] F. L. Ramsey. Characterization of the partial autocorrelation function. *The Annals of Statistics*, pages 1296–1301, 1974.

[220] P. A. Beedlow, E. H. Lee, D. T. Tingey, R. S. Waschmann, and C. A. Burdick. The importance of seasonal temperature and moisture patterns on growth of douglas-fir in western oregon, usa. *Agricultural and forest meteorology*, 169:174–185, 2013.

[221] R. Farley and A. Fitter. Temporal and spatial variation in soil resources in a deciduous woodland. *Journal of Ecology*, 87(4):688–696, 1999.

[222] L. F. Olsen and W. M. Schaffer. Chaos versus noisy periodicity: alternative hypotheses for childhood epidemics. *Science*, 249(4968):499–504, 1990.

[223] S. Wichert, K. Fokianos, and K. Strimmer. Identifying periodically expressed transcripts in microarray time series data. *Bioinformatics*, 20(1):5–20, 2004.

[224] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. *Fast subsequence matching in time-series databases*, volume 23. ACM, 1994.

[225] T. W. Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[226] T. A. Matyas and K. M. Greenwood. Visual analysis of single-case time series: Effects of variability, serial dependence, and magnitude of intervention effects. *Journal of Applied Behavior Analysis*, 23(3):341–351, 1990.

[227] M. C. Hao, U. Dayal, D. A. Keim, and T. Schreck. Multi-resolution techniques for visual exploration of large time-series data. In *EUROVIS 2007*, pages 27–34, 2007.

[228] S.-H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

[229] M. Radovanović. *High-dimensional data representations and metrics for machine learning and data mining*. PhD thesis, Citeseer, 2011.

[230] G. Zoltan. Role of similarity measures in time series analysis. 2015.

[231] P. Bloomfield. *Fourier analysis of time series: an introduction*. John Wiley & Sons, 2004.

[232] W. M. Gentleman and G. Sande. Fast fourier transforms: For fun and profit. In *Proceedings of the November 7-10, 1966, Fall Joint Computer Conference*, pages 563–578, New York, NY, USA, 1966. ACM. doi: 10.1145/1464291.1464352. URL `http://doi.acm.org/10.1145/1464291.1464352`.

[233] S. W. Porges, R. E. Bohrer, M. N. Cheung, F. Drasgow, P. M. McCabe, and G. Keren. New time-series statistic for detecting rhythmic co-occurrence in the frequency domain: the weighted coherence and its application to psychophysiological research. *Psychological bulletin*, 88(3):580, 1980.

[234] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *International conference on foundations of data organization and algorithms*, pages 69–84. Springer, 1993.

[235] D. Rafiei and A. Mendelzon. Similarity-based queries for time series data. In *ACM SIGMOD Record*, volume 26, pages 13–25. ACM, 1997.

[236] T. Yamane. Statistics: An introductory analysis. 1973.

[237] W.-C. Wang, K.-W. Chau, C.-T. Cheng, and L. Qiu. A comparison of performance of several artificial intelligence methods for forecasting monthly discharge time series. *Journal of hydrology*, 374(3-4):294–306, 2009.

[238] J. Lee Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

[239] E. J. G. Pitman. Significance tests which may be applied to samples from any populations. ii. the correlation coefficient test. *Supplement to the Journal of the Royal Statistical Society*, 4(2):225–232, 1937.

[240] J. P. Pluim, J. A. Maintz, and M. A. Viergever. Interpolation artefacts in mutual information-based image registration. *Computer vision and image understanding*, 77(2):211–232, 2000.

[241] J. P. Lewis. Fast normalized cross-correlation. In *Vision interface*, volume 10, pages 120–123, 1995.

[242] T. Helleseth. Some results about the cross-correlation function between two maximal linear sequences. *Discrete Mathematics*, 16(3):209–232, 1976.

[243] K. Briechle and U. D. Hanebeck. Template matching using fast normalized cross correlation. In *Optical Pattern Recognition XII*, volume 4387, pages 95–103. International Society for Optics and Photonics, 2001.

[244] M. Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[245] P. Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855:1–23, 2008.

[246] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

[247] C. A. Ratanamahatana and E. Keogh. Everything you know about dynamic time warping is wrong. In *Third workshop on mining temporal and sequential data*. Citeseer, 2004.

[248] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[249] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):67–72, 1975.

[250] A. Rilliard, A. Allauzen, and P. Boula_de_Mareüil. Using dynamic time warping to compute prosodic similarity measures. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[251] C. Cassisi, P. Montalto, M. Aliotta, A. Cannata, and A. Pulvirenti. Similarity measures and dimensionality reduction techniques for time series data mining. In *Advances in data mining knowledge discovery and applications*. InTech, 2012.

[252] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[253] L. C. Xia, J. A. Steele, J. A. Cram, Z. G. Cardon, S. L. Simmons, J. J. Vallino, J. A. Fuhrman, and F. Sun. Extended local similarity analysis (elsa) of microbial community and other time series data with replicates. In *BMC systems biology*, volume 5, page S15. BioMed Central, 2011.

[254] L. C. Xia, D. Ai, J. Cram, J. A. Fuhrman, and F. Sun. Efficient statistical significance approximation for local similarity analysis of high-throughput time series data. *Bioinformatics*, 29(2):230–237, 2012.

[255] X. Huang, W. Miller, S. Schwartz, and R. C. Hardison. Parallelization of a local similarity algorithm. *Bioinformatics*, 8(2):155–165, 1992.

[256] Q. Ruan, D. Dutta, M. S. Schwalbach, J. A. Steele, J. A. Fuhrman, and F. Sun. Local similarity analysis reveals unique associations among marine bacterioplankton species and environmental factors. *Bioinformatics*, 22(20):2532–2538, 2006.

[257] W. E. Durno, N. W. Hanson, K. M. Konwar, and S. J. Hallam. Expanding the boundaries of local similarity analysis. In *BMC genomics*, volume 14, page S3. BioMed Central, 2013.

[258] H. Gish and D. Cochran. Generalized coherence (signal detection). In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pages 2745–2748. IEEE, 1988.

[259] E. Wolf. *Introduction to the Theory of Coherence and Polarization of Light*. Cambridge University Press, 2007.

[260] Y. Katznelson. *An introduction to harmonic analysis*. Cambridge University Press, 2004.

[261] Y. Wu, Y. Lin, Z. Zhou, D. C. Bolton, J. Liu, and P. Johnson. Cascaded contextual region-based convolutional neural network for event detection from time series signals: A seismic application. *arXiv preprint arXiv:1709.07943*, 2017.

[262] D.-I. Kim, T. Y. Chun, S.-H. Yoon, G. Lee, and Y.-J. Shin. Wavelet-based event detection method using pmu data. *IEEE Transactions on Smart grid*, 8(3):1154–1162, 2017.

[263] B. J. Goode, J. I. M. Reyes, D. R. Pardo-Yepez, G. L. Canale, R. M. Tong, D. Mares, M. Roan, and N. Ramakrishnan. Time-series analysis of blog and metaphor dynamics for event detection. In *Advances in Cross-Cultural Decision Making*, pages 17–27. Springer, 2017.

[264] J. C. Nation, A. Aboulian, D. Green, P. Lindahl, J. Donnal, S. B. Leeb, G. Bredariol, and K. Stevens. Nonintrusive monitoring for shipboard fault detection. In *Sensors Applications Symposium (SAS), 2017 IEEE*, pages 1–5. IEEE, 2017.

[265] S. Aminikhanghahi and D. J. Cook. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367, 2017.

[266] M. I. Jordan and D. E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive science*, 16(3):307–354, 1992.

[267] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 8(3):275–300, 2004.

[268] M. Riedmiller. Advanced supervised learning in multi-layer perceptrons—from backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3):265–278, 1994.

[269] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[270] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[271] H. T. Zhao, J. Liang, Y. Cai, and D. Brunnett. Adaptive threshold for detecting touchdown or contamination, Mar. 4 2014. US Patent 8,665,546.

[272] R. D. Fricker, B. L. Hegler, and D. A. Dunfee. Comparing syndromic surveillance detection methods: Ears'versus a cusum-based methodology. *Statistics in Medicine*, 27(17):3407–3429, 2008.

[273] V. Guralnik and J. Srivastava. Event detection from time series data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 33–42. ACM, 1999.

[274] I. Lapteacru. On the consistency of the Z-score to measure the bank risk. working paper or preprint, Apr. 2016. URL `https://hal.archives-ouvertes.fr/hal-01301846`.

[275] F. Recknagel and I. Ostrovsky. Inferential modelling of time series by evolutionary computation. in obrador, b., jones, id and jennings, e.(eds.) netlake toolbox for the analysis of high-frequency data from lakes (fact-sheet 11). 2017.

[276] J. U. Hjorth. *Computer intensive statistical methods: Validation, model selection, and bootstrap*. Routledge, 2017.

[277] K. Kanchymalay, N. Salim, A. Sukprasert, R. Krishnan, and U. R. Hashim. Multivariate time series forecasting of crude palm oil price using machine learning techniques. In *IOP Conference Series: Materials Science and Engineering*, volume 226, page 012117. IOP Publishing, 2017.

[278] B. Y. Reis and K. D. Mandl. Time series modeling for syndromic surveillance. *BMC Medical Informatics and Decision Making*, 3(1):2, 2003.

[279] R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2014.

[280] P. Omenzetter and J. M. W. Brownjohn. Application of time series analysis for bridge monitoring. *Smart Materials and Structures*, 15(1):129, 2006.

[281] A. D. Jassby and T. M. Powell. Detecting changes in ecological time series. *Ecology*, 71(6):2044–2052, 1990.

[282] A. Ray. Symbolic dynamic analysis of complex systems for anomaly detection. *Signal Processing*, 84(7): 1115–1130, 2004.

[283] V. Rajagopalan and A. Ray. Symbolic time series analysis via wavelet-based partitioning. *Signal Processing*, 86(11):3309–3320, 2006.

[284] M. Yasar and A. Ray. Trend detection and data mining via wavelet and hilbert-huang transforms. In *American Control Conference, 2008*, pages 4292–4297. IEEE, 2008.

[285] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transactions on image processing*, 1(2):205–220, 1992.

[286] E. Spiegel and C. O'Donnell. *Incidence Algebras (Chapman & Hall/CRC Pure and Applied Mathematics)*. CRC Press, 1997. ISBN 0824700368. URL `https://www.amazon.com/Incidence-Algebras-Chapman-Applied-Mathematics/dp/0824700368?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0824700368`.

[287] M. Sifuzzaman, M. Islam, and M. Ali. Application of wavelet transform and its advantages compared to fourier transform. 2009.

[288] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, 11(7):674–693, 1989.

[289] M. M. Hassoun and P. Lin. A new network approach to symbolic simulation of large-scale networks. In *Circuits and Systems, 1989., IEEE International Symposium on*, pages 806–809. IEEE, 1989.

[290] S. C. Chin, A. Ray, and V. Rajagopalan. Symbolic time series analysis for anomaly detection: a comparative evaluation. *Signal Processing*, 85(9):1859–1868, 2005.

[291] M. L. Chin and J. J. Burred. Audio event detection based on layered symbolic sequence representations. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 1953–1956. IEEE, 2012.

[292] L. Cao, A. Mees, and K. Judd. Dynamics from multivariate time series. *Physica D: Nonlinear Phenomena*, 121(1-2):75–88, 1998.

[293] A. Singhal and D. E. Seborg. Clustering multivariate time-series data. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 19(8):427–438, 2005.

[294] S. T. Roweis. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, dec 2000. doi: 10.1126/science.290.5500.2323. URL `https://doi.org/10.1126/science.290.5500.2323`.

[295] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001.

[296] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999.

[297] T.-c. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[298] J. Zubova, O. Kurasova, and M. Liutvinavičius. Dimensionality reduction methods: The comparison of speed and accuracy. *Information Technology And Control*, 47(1):151–160, 2018.

[299] K. Pearson. LIII. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, nov 1901. doi: 10.1080/14786440109462720. URL `https://doi.org/10.1080/14786440109462720`.

[300] J. D. Horel. Complex principal component analysis: Theory and examples. *Journal of climate and Applied Meteorology*, 23(12):1660–1673, 1984.

[301] V. Klema and A. Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control*, 25(2):164–176, 1980.

[302] C. Ding and X. He. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*, page 29. ACM, 2004.

[303] Z.-K. Gao, Y.-X. Yang, P.-C. Fang, N.-D. Jin, C.-Y. Xia, and L.-D. Hu. Multi-frequency complex network from time series for uncovering oil-water flow structure. *Scientific reports*, 5:8222, 2015.

[304] H. Senaratne, M. Mueller, M. Behrisch, F. Lalanne, J. Bustos-Jiménez, J. Schneidewind, D. Keim, and T. Schreck. Urban mobility analysis with mobile network data: A visual analytics approach. *IEEE Transactions on Intelligent Transportation Systems*, 19(5):1537–1546, 2018.

[305] A. P. Schotter, O. Buchel, and T. Vashchilko. Interactive visualization for research contextualization in international business. *Journal of World Business*, 53(3):356–372, 2018.

[306] M. Rubinov and O. Sporns. Complex network measures of brain connectivity: uses and interpretations. *Neuroimage*, 52(3):1059–1069, 2010.

[307] E. Estrada. Graph and network theory in physics. *arXiv preprint arXiv:1302.4378*, 2013.

[308] S. G. Kobourov. Spring embedders and force directed graph drawing algorithms. *arXiv preprint arXiv:1201.3011*, 2012.

[309] Y. Xiang, H. Li, S. Wang, and W. Xu. Debugging openstack problems using a state graph approach. *CoRR*, abs/1606.05963, 2016. URL `http://arxiv.org/abs/1606.05963`.

[310] M. Baur and U. Brandes. Crossing reduction in circular layouts. In *Graph-Theoretic Concepts in Computer Science*, pages 332–343. Springer Berlin Heidelberg, 2004. doi: 10.1007/978-3-540-30559-0_28. URL `https://doi.org/10.1007/978-3-540-30559-0_28`.

[311] N. Gehlenborg and B. Wong. Networks. *Nature Methods*, 9(2):115–115, feb 2012. doi: 10.1038/nmeth.1862. URL `https://doi.org/10.1038/nmeth.1862`.

[312] A. Landherr, B. Friedl, and J. Heidemann. A critical review of centrality measures in social networks. *Business & Information Systems Engineering*, 2(6):371–385, 2010.

[313] M. M. Makagon, B. McCowan, and J. A. Mench. How can social network analysis contribute to social behavior research in applied ethology? *Applied animal behaviour science*, 138(3-4):152–161, 2012.

[314] M. Barthelemy. Betweenness centrality in large complex networks. *The European physical journal B*, 38 (2):163–168, 2004.

[315] K. Okamoto, W. Chen, and X.-Y. Li. Ranking of closeness centrality for large-scale social networks. In *International Workshop on Frontiers in Algorithmics*, pages 186–195. Springer, 2008.

[316] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[317] Z. Yang, R. Algesheimer, and C. J. Tessone. A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, 6:30750, 2016.

[318] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008. doi: 10.1088/1742-5468/2008/10/p10008. URL `https://doi.org/10.1088/1742-5468/2008/10/p10008`.

[319] E. D. M. Thomson and B. Raymor. Generic event delivery using http push, May 2018. URL https://tools.ietf.org/html/draft-ietf-webpush-protocol-12.

[320] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali. On scalability of software-defined networking. *IEEE Communications Magazine*, 51(2):136–141, 2013.

[321] Ł. Kufel. Tools for distributed systems monitoring. *Foundations of Computing and Decision Sciences*, 41 (4):237–260, 2016.

[322] P. Hintjens. *ZeroMQ: messaging for many applications*. " O'Reilly Media, Inc.", 2013.

[323] T. Haerder and A. Reuter. Principles of transaction-oriented database recovery. *ACM Computing Surveys (CSUR)*, 15(4):287–317, 1983.

[324] L. D. Paulson. Building rich web applications with ajax. *Computer*, 38(10):14–17, 2005.

[325] A. Ronacher. Jinja2 documentation. *Welcome to Jinja2—Jinja2 Documentation (2.8-dev)*, 2008.

[326] M. Bayer. Sqlalchemy—the database toolkit for python. *URL http://www. sqlalchemy. org/. Accessed on the 13th of November*, 2012.

[327] Amazon. Amazon rds free tier - amazon web services (aws), May 2018. URL https://aws.amazon.com/rds/free/.

[328] TP-Link. Archer c5 — ac1200 wireless dual band gigabit router, May 2018. URL https://www.tp-link.com/us/products/details/cat-9_Archer-C5.html.

[329] J. Kirkby, J. Curtius, J. Almeida, E. Dunne, J. Duplissy, S. Ehrhart, A. Franchin, S. Gagné, L. Ickes, A. Kürten, A. Kupc, A. Metzger, F. Riccobono, L. Rondo, S. Schobesberger, G. Tsagkogeorgas, D. Wimmer, A. Amorim, F. Bianchi, M. Breitenlechner, A. David, J. Dommen, A. Downard, M. Ehn, R. C. Flagan, S. Haider, A. Hansel, D. Hauser, W. Jud, H. Junninen, F. Kreissl, A. Kvashin, A. Laaksonen, K. Lehtipalo, J. Lima, E. R. Lovejoy, V. Makhmutov, S. Mathot, J. Mikkilä, P. Minginette, S. Mogo, T. Nieminen, A. Onnela, P. Pereira, T. Petäjä, R. Schnitzhofer, J. H. Seinfeld, M. Sipilä, Y. Stozhkov, F. Stratmann, A. Tomé, J. Vanhanen, Y. Viisanen, A. Vrtala, P. E. Wagner, H. Walther, E. Weingartner, H. Wex, P. M. Winkler, K. S. Carslaw, D. R. Worsnop, U. Baltensperger, and M. Kulmala. Role of sulphuric acid, ammonia and galactic cosmic rays in atmospheric aerosol nucleation. *Nature*, 476(7361):429–433, aug 2011. doi: 10.1038/nature10343. URL https://doi.org/10.1038/nature10343.

[330] E. M. Dunne, H. Gordon, A. Kürten, J. Almeida, J. Duplissy, C. Williamson, I. K. Ortega, K. J. Pringle, A. Adamov, U. Baltensperger, et al. Global atmospheric particle formation from cern cloud measurements. *Science*, 354(6316):1119–1124, 2016.

[331] H. Gordon, J. Kirkby, U. Baltensperger, F. Bianchi, M. Breitenlechner, J. Curtius, A. Dias, J. Dommen, N. M. Donahue, E. M. Dunne, et al. Causes and importance of new particle formation in the present-day and pre-industrial atmospheres. *Journal of Geophysical Research: Atmospheres*, 2017.

[332] J. Duplissy, M. B. Enghoff, K. L. Aplin, F. Arnold, H. Aufmhoff, M. Avngaard, U. Baltensperger, T. Bondo, R. Bingham, K. Carslaw, J. Curtius, A. David, B. Fastrup, S. Gagné, F. Hahn, R. G. Harrison, B. Kellett,

J. Kirkby, M. Kulmala, L. Laakso, A. Laaksonen, E. Lillestol, M. Lockwood, J. Mäkelä, V. Makhmutov, N. D. Marsh, T. Nieminen, A. Onnela, E. Pedersen, J. O. P. Pedersen, J. Polny, U. Reichl, J. H. Seinfeld, M. Sipilä, Y. Stozhkov, F. Stratmann, H. Svensmark, J. Svensmark, R. Veenhof, B. Verheggen, Y. Viisanen, P. E. Wagner, G. Wehrle, E. Weingartner, H. Wex, M. Wilhelmsson, and P. M. Winkler. Results from the cern pilot cloud experiment. *Atmospheric Chemistry and Physics*, 10(4):1635–1647, 2010. doi: 10.5194/acp-10-1635-2010. URL `https://www.atmos-chem-phys.net/10/1635/2010/`.

[333] M. J. Lawler, P. M. Winkler, J. Kim, L. Ahlm, J. Tröstl, A. P. Praplan, S. Schobesberger, A. Kürten, J. Kirkby, F. Bianchi, et al. Unexpectedly acidic nanoparticles formed in dimethylamine–ammonia–sulfuric-acid nucleation experiments at cloud. *Atmospheric Chemistry and Physics*, 16(21):13601–13618, 2016.

[334] A. Dias, S. Ehrhart, A. Vogel, C. Williamson, J. Almeida, J. Kirkby, S. Mathot, S. Mumford, and A. Onnela. Temperature uniformity in the cern cloud chamber. *Atmospheric Measurement Techniques*, 10(12):5075, 2017.

[335] A. Kupc, A. Amorim, J. Curtius, A. Danielczok, J. Duplissy, S. Ehrhart, H. Walther, L. Ickes, J. Kirkby, A. Kürten, J. Lima, S. Mathot, P. Minginette, A. Onnela, L. Rondo, and P. Wagner. A fibre-optic uv system for h2so4 production in aerosol chambers causing minimal thermal effects. *Journal of Aerosol Science*, 42(8):532 – 543, 2011. ISSN 0021-8502. doi: https://doi.org/10.1016/j.jaerosci.2011.05.001. URL `http://www.sciencedirect.com/science/article/pii/S0021850211000632`.

[336] C. R. Hoyle, C. Fuchs, E. Järvinen, H. Saathoff, A. Dias, I. El Haddad, M. Gysel, S. C. Coburn, J. Tröstl, A.-K. Bernhammer, F. Bianchi, M. Breitenlechner, J. C. Corbin, J. Craven, N. M. Donahue, J. Duplissy, S. Ehrhart, C. Frege, H. Gordon, N. Höppel, M. Heinritzi, T. B. Kristensen, U. Molteni, L. Nichman, T. Pinterich, A. S. H. Prévôt, M. Simon, J. G. Slowik, G. Steiner, A. Tomé, A. L. Vogel, R. Volkamer, A. C. Wagner, R. Wagner, A. S. Wexler, C. Williamson, P. M. Winkler, C. Yan, A. Amorim, J. Dommen, J. Curtius, M. W. Gallagher, R. C. Flagan, A. Hansel, J. Kirkby, M. Kulmala, O. Möhler, F. Stratmann, D. R. Worsnop, and U. Baltensperger. Aqueous phase oxidation of sulphur dioxide by ozone in cloud droplets. *Atmospheric Chemistry and Physics*, 16(3):1693–1712, 2016. doi: 10.5194/acp-16-1693-2016. URL `https://www.atmos-chem-phys.net/16/1693/2016/`.

[337] J. Tröstl, W. K. Chuang, H. Gordon, M. Heinritzi, C. Yan, U. Molteni, L. Ahlm, C. Frege, F. Bianchi, R. Wagner, et al. The role of low-volatility organic compounds in initial particle growth in the atmosphere. *Nature*, 533(7604):527, 2016.

[338] E. Järvinen, K. Ignatius, L. Nichman, T. B. Kristensen, C. Fuchs, C. R. Hoyle, N. Höppel, J. C. Corbin, J. Craven, J. Duplissy, S. Ehrhart, I. El Haddad, C. Frege, H. Gordon, T. Jokinen, P. Kallinger, J. Kirkby, A. Kiselev, K.-H. Naumann, T. Petäjä, T. Pinterich, A. S. H. Prevot, H. Saathoff, T. Schiebel, K. Sengupta, M. Simon, J. G. Slowik, J. Tröstl, A. Virtanen, P. Vochezer, S. Vogt, A. C. Wagner, R. Wagner, C. Williamson, P. M. Winkler, C. Yan, U. Baltensperger, N. M. Donahue, R. C. Flagan, M. Gallagher, A. Hansel, M. Kulmala, F. Stratmann, D. R. Worsnop, O. Möhler, T. Leisner, and M. Schnaiter. Observation of viscosity transition in $alpha$-pinene secondary organic aerosol. *Atmospheric Chemistry and Physics*, 16

(7):4423–4438, 2016. doi: 10.5194/acp-16-4423-2016. URL https://www.atmos-chem-phys.net/16/4423/2016/.

[339] A. Biørn-Hansen, T. A. Majchrzak, and T.-M. Grønli. Progressive web apps for the unified development of mobile applications. In *International Conference on Web Information Systems and Technologies*, pages 64–86. Springer, 2017.

[340] A. G. Harrison. *Chemical ionization mass spectrometry*. CRC press, 1992.

[341] A. Kurten, L. Rondo, S. Ehrhart, and J. Curtius. Calibration of a chemical ionization mass spectrometer for the measurement of gaseous sulfuric acid. *The Journal of Physical Chemistry A*, 116(24):6375–6386, 2012.

[342] A. Kurten, C. Williamson, J. Almeida, J. Kirkby, and J. Curtius. On the derivation of particle nucleation rates from experimental formation rates. *Atmospheric Chemistry and Physics*, 15(8):4063–4075, apr 2015. doi: 10.5194/acp-15-4063-2015. URL https://doi.org/10.5194/acp-15-4063-2015.

[343] J. Vanhanen, J. Mikkilä, K. Lehtipalo, M. Sipilä, H. Manninen, E. Siivola, T. Petäjä, and M. Kulmala. Particle size magnifier for nano-cn detection. *Aerosol Science and Technology*, 45(4):533–542, 2011.

[344] D. Wimmer, K. Lehtipalo, A. Franchin, J. Kangasluoma, F. Kreissl, A. Kürten, A. Kupc, A. Metzger, J. Mikkilä, T. Petäjä, et al. Performance of diethylene glycol-based particle counters in the sub-3 nm size range. *Atmospheric Measurement Techniques*, 6(7):1793, 2013.

[345] S. C. Wang and R. C. Flagan. Scanning electrical mobility spectrometer. *Aerosol Science and Technology*, 13(2):230–240, 1990.

[346] Hamamatsu. Si photodiode s2281-01 — hamamatsu photonics, May 2018. URL https://www.hamamatsu.com/eu/en/product/alpha/S/4103/S2281-01/index.html.

[347] Tofwerk. Custom tof - mass spectrometer for ion source and data analysis -tofwerk, May 2018. URL https://www.tofwerk.com/products/custom/.

[348] A. Jordan, S. Haidacher, G. Hanel, E. Hartungen, L. Märk, H. Seehauser, R. Schottkowsky, P. Sulzer, and T. Märk. A high resolution and high sensitivity proton-transfer-reaction time-of-flight mass spectrometer (ptr-tof-ms). *International Journal of Mass Spectrometry*, 286(2-3):122–128, 2009.

[349] A. Kupc, A. Amorim, J. Curtius, A. Danielczok, J. Duplissy, S. Ehrhart, H. Walther, L. Ickes, J. Kirkby, A. Kürten, et al. A fibre-optic uv system for h2so4 production in aerosol chambers causing minimal thermal effects. *Journal of Aerosol Science*, 42(8):532–543, 2011.

[350] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[351] A. D. Martin, A. W. R. Soundy, B. J. Panckhurst, C. P. Brown, D. Schumayer, T. C. A. Molteno, and M. Parry. Real-time uncertainty quantification using correlated noise models for gnss positioning. In *2017 IEEE SENSORS*, pages 1–3, Oct 2017. doi: 10.1109/ICSENS.2017.8233899.

[352] K. Zoumpatianos, S. Idreos, and T. Palpanas. Indexing for interactive exploration of big data series. In

*Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 1555–1566, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2376-5. doi: 10.1145/2588555.2610498. URL http://doi.acm.org/10.1145/2588555.2610498.

[353] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270. ACM, 2012.

[354] M. Arnold, X. Milner, H. Witte, R. Bauer, and C. Braun. Adaptive ar modeling of nonstationary time series by means of kalman filtering. *IEEE transactions on biomedical engineering*, 45(5):553–562, 1998.

[355] S. K. Ahn and G. C. Reinsel. Nested reduced-rank autoregressive models for multiple time series. *Journal of the American Statistical Association*, 83(403):849–856, 1988.

[356] Y. Kakizawa, R. H. Shumway, and M. Taniguchi. Discrimination and clustering for multivariate time series. *Journal of the American Statistical Association*, 93(441):328–340, 1998.

[357] S. Hasan. snowflake - pypi, Mar. 2018. URL https://pypi.org/project/snowflake/.

[358] C.-F. Natali. ntplib - pypi, Mar. 2018. URL https://pypi.org/project/ntplib/.

[359] M. Khan, M. Cooke, S. Utembe, A. Archibald, P. Maxwell, W. Morris, P. Xiao, R. Derwent, M. Jenkin, C. Percival, et al. A study of global atmospheric budget and distribution of acetone using global atmospheric model stochem-cri. *Atmospheric Environment*, 112:269–277, 2015.

# Appendix A

# DAQBroker database schema

In section 3.1.2.4, a subset of the full database schema of DAQBroker is presented. The focus of that section was to highlight the instrument data warehouse relational model. As can be seen in figure 3.3, several tables were not described. While not relevant for the instrument model, these tables are crucial for the correct performance of DAQBroker. This appendix intends to fully document the database schema of DAQBroker, highlighting the purpose of each table and their columns.

The DAQBroker environment is replicated in each database engine used and on each active server application. On a database engine a DAQBroker environment will contain one global database for settings and several user created data storage databases. Each server application creates a local database file that contains information regarding the existing database engines, local folder structure and available nodes. This appendix will be divided into three parts each describing the mentioned databases.

## A.1 Global engine database

The global engine database is created the first time an engine uses the DAQBroker framework and contains 3 tables with information specific to the engine being used. The table structure is the following:

- **Databases** - Contains information on all data storage databases created using the engine and information on whether or not the databases are active for data storage. Accordingly it has two columns
  - *dbname* - A finite length string column containing the name of each individual data storage database. These will have the prefix '*daqbro_*' added to it to minimize conflicts with existing databases,
  - *active* - A boolean column that indicates whether a database is active or not for data storage. Inactive databases will not automatically retrieve the instrument's most recent data and will only allow administrator users to make changes to the information in the database.
- **Users** - This table contains information on all created users using DAQBroker in the database engine. It also

contains information on the type of user, thus defining what interfaces and system calls are available for that user. Similar to the previous table, it is organized in two columns:

  - *username* - A finite length string column containing the name of an individual user. To avoid conflicts with existing users in the database the user name should be a carefully planned task for the system administrator,
  - *type* - An integer column that defines the type of user. As defined in 3.1.4, DAQBroker supports 4 types of users : Administrator, Operator, User and Guest.

- **Links** - Contains information on all shareable web links created by each user. Shareable links point to a specific endpoint of the DAQBroker web application where some manipulation has already been made by the user. Currently shareable links are limited to data visualizations but could be extended in future versions of DAQBroker. It is organized in four columns:
  - *clock* - An integer containing the time of creation of the shareable link. The current type of links are one-time shareable model with limited life. 7 day old links are removed from the database,
  - *linkid* - A finite length string that contains a unique identifier for the shareable link,
  - *site* - A finite length string that contains the endpoint of the DAQBroker web application pointed to by the link,
  - *variable* - A text column that contains the user-made data manipulations to the endpoint that will be transferred to the user that receives and follows the shareable link.

## A.2 Data storage database

The data storage databases contain the bulk information of the DAQBroker framework and also the specific information. Each of these databases contain information about the specific data effort requirements, short or long term and as such information that is only usable in the context and timespan during which the database is active. While inactive databases can be revisited and its data queried, it is not advised that the database be changed when inactive, reason why this task can only be made by an administrator user. The description of the data storage database schema tables (represented in figure 3.3) follow:

- **Instruments** - Contains the information of the main instrument block. The attributes of each instrument are as follows:
  1. *Name* - String with the name of the instrument. The name is unique and cannot be repeated.
  2. *instid* - Integer with the unique identification of the instrument.
  3. *active* - Boolean defining whether an instrument is in active data gathering or not,
  4. *description* - Blob that contains a description of what the instrument is supposed to do,
  5. *username* - String relating the instrument to the DAQBroker user that created it,

6. *email* - Blob containing contact information for the instrument operator (does not have to be just email),
7. *insttype* - Integer that defines the type of instrument storage format,
8. *log* - JSON encoded blob that contains a public electronic instrument log that users can fill out to help keep track of changes made to the instrument.

- **Instmeta** - Contains the information of each instrument's data source blocks. The attributes of said blocks are as follows:
  1. *clock* - Integer defining the last edit timestamp of the data source,
  2. *name* - String identifying the data source,
  3. *metaid* - Integer uniquely identifying the data source,
  4. *instid* - Integer indicating the instrument that is the data source (see Instruments - instid),
  5. *type* - Integer defining the type of data source (allows specific data gathering methods to be chosen),
  6. *node* - String defining the network node associated with this data source,
  7. *remarks* - JSON encoded blob containing relevant information for data gathering methods for the type of data source chosen,
  8. *sentRequest* - Boolean that tests whether a remote data gathering routine is under way (for time out purposes),
  9. *lastAction* - Integer defining the timestamp of the last automated action preformed on this data source,
  10. *lasterrortime* - Integer defining the timestamp of the last error encountered when preforming data gathering for this data source,
  11. *lastError* - Blob containing information of the last error encountered when preforming data gathering for this data source.
  12. *lockSync* - Boolean to test whether there is a data gathering action currently taking place for this data source (used with *sentRequest* for timeout purposes).
- **Channels** - Contains the information of each instrument's data channels. The attributes each channel are:
  1. *Name* - String containing the name of data channel,
  2. *channelid* - Integer uniquely identifying the data channel,
  3. *channeltype* - Integer that defines the type of data channel (used for data manipulation),
  4. *valuetype* - Integer that defines the type of data returned by this channel (i.e: string, numbers),
  5. *units* - String containing the physical units of the data provided by this data channel,
  6. *instid* - Integer relating the data channel the instrument source (see Instruments - instid),
  7. *description* - Blob containing a description of the data provided by the data channel,

8. *active* - A Boolean that tests whether the data channel is actively providing data,
9. *remarks* - JSON encoded blob containing relevant information for automated handling of the specific data channel,
10. *metaid* - Integer relating the data channel with its data source (see Instmeta - metaid),
11. *lastclock* - Integer defining the timestamp of the last gathered data value,
12. *lastValue* - String containing the last gathered value on the channel,
13. *fileorder* - Integer defining an order for the data channel to be gathered (only used on specific data gathering methods),
14. *alias* - String defining an alias of the channel to be used in specific data gathering methods,
15. *firstClock* - Integer defining the timestamp of the first gathered data value (can be updated if newer values are gathered).

- **parsing** - Contains information on all the parsing operations and files on which they were executed for a single data source. This is used to prevent multiple parsing of the same data to occur, which compromises performance. Each parsing entry is defined by the following columns:
  1. *clock* - Integer containing the timestamp creation of this entry,
  2. *lastAction* - Integer containing the timestamp of last parsing activity done on this entry,
  3. *metaid* - Integer uniquely identifying the parsing entry,
  4. *instid* - Integer identifying the instrument to which this entry belongs to,
  5. *type* - Integer defining the type of data parsing to be preformed, same as **instmeta** - *type*,
  6. *locked* - Boolean that defines whether parsing for this entry is currently taking place or not,
  7. *forcelock* - Boolean that defines whether or not parsing this entry has been stopped by user request,
  8. *remarks* - Text entry containing type specific information regarding parsing operations executed.
- **collections** - Contains the information of user defined data channel collections. These collections can be used to choose large amounts of channels with a single request. A collection is defined by the following columns:
  1. *Name* - Finite length string containing a user defined name for the collection,
  2. *channels* - Text entry containing a JSON encoded string of a list of data channels,
  3. *remarks* - Text entry containing supplementary information about the collection. Current version of DAQBroker does not use this entry,
- **plots** - Contains information about the user defined and saved data visualization. These visualizations are available for all other users to see, each visualization is defined by the folowing columns:
  1. *plotname* - Finite length string containing a user defined name for the visualization,

2. *plotid* - Integer containing a unique identifier for that visualization,

3. *channels* - JSON encoded string containing a list of data channels relevant to the creation of the visualization,

4. *plottype* - Integer defining the type of visualization to be created. Currently DAQBroker can create 4 types of visualizations: Simple time traces; Time evolved histograms; Single histograms; External plots;

5. *adminPlot* - Boolean used to define whether or not a visualization should only be altered byb an administrator. Currently not used and to be removed in the future,

6. *active* - Boolean that defines whether or not this visualization is to be listed to other users. Currently not used in the API,

7. *remarks* - Text entry containing supporting type dependant information for the visualization, such as axis limits, logarithmic axis, colours and more,

- **plotcomments** - Contains information on user supplied comments to specific visualizations. Each user comment is defined by the following columns:

  1. *clock* - Integer containing the timestamp of the comment,

  2. *plotid* - Integer containing a unique identifier for that visualization into which the comment was inserted,

  3. *channelid* - Channel identifier of the comment. To be used for comments on specific visualizations that allow specific channels to receive a comment from a user,

  4. *comment* - Text entry containing the user comment

  5. *author* - Text entry containing the username of the user that supplied the comment,

  6. *remarks* - Text entry containing supporting information for the comment. Currently not being used,

- **layouts** - Contains information regarding user defined saved multiple data visualizations. Each layout is made out of plots defined in the **plots** table and is defined by the following columns:

  1. *Name* - Limited length string containing a user defined name for the layout

  2. *layoutid* - Integer containing a unique identifier for the layout,

  3. *plots* - JSON encoded string containing a list of visualizations to be added to the layout,

  4. *format* - Text string containing the definition of the format of the layout, such as number of vertical and horizontal plot spots.

- **runs** - Contains information on the previous and current additions to the experimental run list parameters. Each entry contains changes made to the list and is defined by the following columns:

  1. *clock* - Integer containing the timestamp of the edit to the experiment list,

2. *lastUpdate* - Integer containing the timestamp of the previous edit to the experiment list,
3. *isLinked* - Boolean containing a legacy check to link the run list to a google spreadsheet,
4. *linkRemarks* - Legacy JSON encoded string containing information on how to link the google spreadsheet,
5. *linkType* - Legacy integer containing the type of link to be made,
6. *runlistRemarks* - JSON encoded string containing the information on the parameters to represent the run list, which will be columns in the **runlist** table. Each run list parameter can be of 3 types: Number, text, drop-down menu with user defined choices

- **runlist** - Contains the list of experimental runs logged by users. Changes to the run list parameters will result in adding, editing, deleting columns in the run list. A run list without any user defined parameters consists of the following columns:
  1. *start* - Integer containing the timestamp of the start of the experimental run, is automatically created on run/stage start, can be changed when a new run/stage is started,
  2. *end* - Integer containing the timestamp of the end of the experimental run, is automatically created on run/stage end, can be changed when a new run/stage is started,
  3. *run* - Finite length string containing the unique identifier for a run and its stage (ex: '1900.10' will be run '1900', stage '10' and uniquely identifies a run and its stage),
  4. *summary* - Text string containing a summary of the run/stage,
  5. *comments* - JSON encoded string of a list of user supplied comments to a run/stage, identifying specific sections of the run/stage where a meaningful event occurred,
  6. *active* - Boolean defining whether or not a run/stage is currently in an active state.
- **jobs** - Contains information on the job queue implemented in the DAQBroker. This list is to be moved with high priority to the local server application database as job queues from multiple servers can conflict with each other. Each job consists of the following columns:
  1. *clock* - Integer containing the timestamp of the job creation time. This is used to time out an existing job if its type is set to be short running,
  2. *jobid* - A finite length string defining a unique identifier for the job,
  3. *type* - Type of job being requests, the current version of DAQBroker only expects one type of job, that being a data query,
  4. *username* - Username requesting the job,
  5. *status* - integer defining the current status of the job: 1 being completed without errors; 0 being under way and -1 being completed with errors,
  6. *data* - Text entry pointing at the memory object containing the result of the requested job,

7. *error* - Text entry containing the error message of a possible error. To be used in conjunction with a -1 on the *status* column,

8. *reqid* - A text entry uniquely defining the source of the request of the job.

- **subscribers** - Contains a list of emails of users who request to be updated with information regarding this database:

## A.3 Local storage database

The server local storage database contains certain machine-specific settings that are required for the proper behaviour of the DAQBroker server application. The current iteration of the DAQBroker server application stores 5 tables:

- **folder** - Contains information on all available and/or previously used folders on the server machine. The DAQBroker API allows the application to scan and if possible create new folders for various purposes. The following columns define each folder:
  - *clock* - An integer containing the creation time of the folder,
  - *path* - A finite length string that contains the full path of the folder, this string will have different compositions depending on the operating system the server application is being used in,
  - *type* - An integer that defines the type of usage for the folder, currently 3 types of folder usage are supported: backup, uploads and temporary files,
  - *remarks* - A text field that contains secondary information about the folder. The current version of DAQBroker does not use this field.
- **global** - A single row table that contains the current local settings of the server application. The settings are defined by each column:
  - *clock* - An integer containing the time of last settings change,
  - *version* - A finite string containing the version of the server application,
  - *backupfolder* - Finite length string containing the currently used backup folder,
  - *importfolder* - Finite length string containing the currently used import folder,
  - *tempfolder* - Finite length string containing the currently used temporary folder,
  - *ntp* - A text field that contains the address of the NTP server to use when software time synchronization is required by DAQBroker,
  - *logport* - An integer field that contains the port that handles logging requests,
  - *commport* - An integer field that contains the port that handles communication requests,

  - *remarks* - A text field that contains secondary information about the machine. This field currently keeps relevant information about the general operation of the machine.

- **nodes** - Contains information about all the nodes running the DAQBroker client application that are connected to this specific machine. Each node is defined by the following columns:
  - *node* - A unique identifier generated by the python snowflake persistent unique ID generation[357],
  - *name* - A user provided name for the machine to be identified as in the DAQBroker network. Defaults to the machine name as viewed by Python, so it is possible for conflicts to occur when a new machine is being inserted to the network,
  - *address* - String containing the current address of the machine as seen from the server machine,
  - *local* - String containing the current address of the server machine, as seen from the client machine. Arguably unnecessary, but currently used to ensure client-server communication,
  - *port* - Integer containing the current communications port of the client application,
  - *active* - A boolean used to test if the node is actively available for instrument monitoring,
  - *lastActive* - An integer containing the timestamp of the last communication made between the node and the server application. This time is used to check if a node is to be automatically set to inactive or not,
  - *tsyncauto* - A boolean to check if the node is to use the current server application NTP address to synchronize its software clock,
  - *remarks* - A text field that contains secondary information about the machine. This field currently keeps relevant information about the general operation of the machine such as available RAM and ROM, CPU utilization and other meaningful parameters.

- **ntp** - Contains information about all the NTP servers used on this server machine. NTP synchronization is achieved via python's native NTP library[358] and a simple custom-made update algorithm. Each NTP server is identified by the following columns:
  - *clock* - Integer containing a timestamp of the creation time of the NTP record on the DAQBroker server application,
  - *server* - A finite length string containing a unique identifier of the NTP server address,
  - *port* - An integer containing the port of the NTP server, in case a non-standard port is used.

- **servers** - Contains information on all the database engines used by the owning machine. Each engine is identified by the following:
  - *server* - Finite length string containing the full address of the database engine ('localhost' in the case of a local engine)

– *engine* - The engine that is being run on that address. This information is required for login purposes and to allow for the same servers to run different engines under the DAQBroker framework which, while unusual is not impossible to find in certain environments.

# Appendix B

# Multivariate Time Series Analysis

This appendix provides analysis of all CLOUD experimental runs selected and not shown in 5.2.5. It begins by listing the number of channels integrated into the multivariate time series to be analyzed.

## B.1 Chosen Data Channels

This section provides a list of channels that are used on to create a multivariate time series, as described in 5.2. This set of channels contains their underlying instrument and whether the measurement is that of a physical quantity or a chemical quantity as well as specific comments when the channels are a composite of several channels. This large list of channels could be used in an exploratory analysis of the data from an experimental run to attempt to find hitherto unknown relationships between time series.

*Table B.1: Full list of channels to be used for multivariate time series analysis*

| # | Channel | Instrument | Type | Comment |
|---|---|---|---|---|
| 1 | intensity | PhotoDiode12 | Physical | N/A |
| 2 | intensity2 | PhotoDiode12 | Physical | N/A |
| 3 | mvolt1 | HVFC | Physical | N/A |
| 4 | mvolt2 | HVFC | Physical | N/A |
| 5 | shutter1 | UVLamp | Physical | N/A |
| 6 | intensity1 | UVLamp | Physical | N/A |
| 7 | shutter2 | UVLamp | Physical | N/A |
| 8 | intensity2 | UVLamp | Physical | N/A |
| 9 | shutter3 | UVLamp | Physical | N/A |
| 10 | intensity3 | UVLamp | Physical | N/A |
| 11 | shutter4 | UVLamp | Physical | N/A |
| 12 | intensity4 | UVLamp | Physical | N/A |
| 13 | speed1 | Fan | Physical | N/A |
| 14 | speed2 | Fan | Physical | N/A |
| 15 | temp1 | PT100 | Physical | N/A |
| 16 | brust | Scalers | Physical | N/A |
| 17 | sat_water | TDL | Physical | N/A |
| 18 | so2 | SO2 | Chemical | N/A |
| 19 | o3 | O3 | Chemical | N/A |
| 20 | Ch0: Ch [#/cc] | DMAtrain | Physical | N/A |
| 21 | Ch1: C [#/cc] | DMAtrain | Physical | N/A |
| 22 | Ch2: C [#/cc] | DMAtrain | Physical | N/A |
| 23 | Ch3: C [#/cc] | DMAtrain | Physical | N/A |
| 24 | Ch4: C [#/cc] | DMAtrain | Physical | N/A |
| 25 | Ch5: C [#/cc] | DMAtrain | Physical | N/A |
| 26 | Sabre3Temperature | SabreTemperature | Physical | N/A |
| 27 | Sabre4Temperature | SabreTemperature | Physical | N/A |
| 28 | 3776-1DisplayConc. | CPC3776 | Physical | N/A |
| 29 | 3010-1DisplayConc. | CPC3776 | Physical | N/A |
| 30 | 3010-2DisplayConc. | CPC3776 | Physical | N/A |
| 31 | 3785DisplayConc. | CPC3776 | Physical | N/A |
| 32 | NH3 | HTOF_UFRA | Chemical | N/A |
| 33 | H30 | HTOF_UFRA | Chemical | N/A |
| 34 | H3OH2O | HTOF_UFRA | Chemical | N/A |
| 35 | H7O3 | HTOF_UFRA | Chemical | N/A |
| 36 | O2 | HTOF_UFRA | Chemical | N/A |
| 37 | conc | PSM15 | Physical | N/A |
| 38 | conc | PSM16 | Physical | N/A |
| 39 | conc | PSM_Vienna | Physical | N/A |
| 40 | conc | PSM12 | Physical | N/A |
| 41 | Count | nRDMA | Physical | N/A |
| 42 | Total Conc. (#/cm) | nanoSMPS | Physical | N/A |
| 43 | I2Br | CMU_FIGAERO | Chemical | N/A |
| 44 | BrH2O | CMU_FIGAERO | Chemical | N/A |

In order to study relationships that can be obscured through spurious similarities, a reduced set of channels is chosen. The focus of this set is to prune the initial set, considering only the channels relevant to the stages being studied, similar to what would be done during the CLOUD experiment to emphasise certain results. The chosen channels are the following:

*Table B.2: Full list of channels to be used for multivariate time series analysis (continued)*

| # | Channel | Instrument | Type | Comment |
|---|---|---|---|---|
| 45 | NO2 | CAPS | Chemical | N/A |
| 46 | I2_CEDOAS | CE-DOAS | Chemical | N/A |
| 47 | HONO_CEDOAS | CE-DOAS | Chemical | N/A |
| 48 | NO2_CEDOAS | CE-DOAS | Chemical | N/A |
| 49 | Glyoxal_CEDOAS | CE-DOAS | Chemical | N/A |
| 50 | Megly_CEDOAS | CE-DOAS | Chemical | N/A |
| 51 | APINENE | STOF | Chemical | N/A |
| 52 | ISOPRENE | STOF | Chemical | N/A |
| 53 | ACETONE | STOF | Chemical | N/A |
| 54 | ACETICACID | STOF | Chemical | N/A |
| 55 | TMB | STOF | Chemical | N/A |
| 56 | NAPHTHALENE | STOF | Chemical | N/A |
| 57 | TOLUENE | STOF | Chemical | N/A |
| 58 | BETACARYOPHYLENE | STOF | Chemical | N/A |
| 59 | PINONALDEHYDE | STOF | Chemical | N/A |
| 60 | dew | Dew | Physical | N/A |
| 61 | NIT | LTOF_UFRA | Chemical | N/A |
| 62 | SA | LTOF_UFRA | Chemical | N/A |
| 63 | HIO | LTOF_UFRA | Chemical | N/A |
| 64 | DMA | LTOF_UFRA | Chemical | N/A |
| 65 | CPC | LTOF_CPC | Physical | N/A |
| 66 | conc | PSM_PSI | Physical | N/A |
| 67 | HV-Actual | Laser | Physical | N/A |
| 68 | Egy-Actual | Laser | Physical | N/A |
| 69 | APINENE | PTR3 | Chemical | N/A |
| 70 | ISOPRENE | PTR3 | Chemical | N/A |
| 71 | ACETONE | PTR3 | Chemical | N/A |
| 72 | ACETICACID | PTR3 | Chemical | N/A |
| 73 | TMB | PTR3 | Chemical | N/A |
| 74 | NAPHTHALENE | PTR3 | Chemical | N/A |
| 75 | TOLUENE | PTR3 | Chemical | N/A |
| 76 | BETACARYOPHYLENE | PTR3 | Chemical | N/A |
| 77 | PINONALDEHYDE | PTR3 | Chemical | N/A |
| 78 | NH3_2MIN | PICARRO | Chemical | N/A |
| 79 | UVX | Light_Spectrometer | Physical | Sum over wavelengths of UVX |
| 80 | LS3 | Light_Spectrometer | Physical | Sum over wavelengths of LS3 |
| 81 | LS4 | Light_Spectrometer | Physical | Sum over wavelengths of LS4 |
| 82 | LS1 | Light_Spectrometer | Physical | Sum over wavelengths of LS1 |
| 83 | UVH | Light_Spectrometer | Physical | Sum over wavelengths of UVH |
| 84 | Ions_neg | NAIS_ions | Physical | Sum over negative ion channels of NAIS_ions |
| 85 | Ions_pos | NAIS_ions | Physical | Sum over positive ion channels of NAIS_ions |
| 86 | Particles_neg | NAIS_particles | Physical | Sum over positive ion channels of NAIS_particles |
| 87 | Particles_pos | NAIS_particles | Physical | Sum over positive ion channels of NAIS_particles |
| 88 | Particles (20-500 nm) | LongSMPS | Physical | Sum over all particle sizes of Long SMPS |

*Table B.3: Reduced list of channels to be used for multivariate time series analysis*

| # | Channel | Instrument | Type | Comment |
|---|---|---|---|---|
| 1 | intensity | PhotoDiode12 | Physical | N/A |
| 2 | mvolt1 | HVFC | Physical | N/A |
| 3 | intensity1 | UVLamp | Physical | N/A |
| 4 | speed1 | Fan | Physical | N/A |
| 5 | brust | Scalers | Physical | N/A |
| 6 | sat_water | TDL | Physical | N/A |
| 7 | so2 | SO2 | Chemical | N/A |
| 8 | o3 | O3 | Chemical | N/A |
| 9 | 3776-1DisplayConc. | CPC3776 | Physical | N/A |
| 10 | 3010-1DisplayConc. | CPC3776 | Physical | N/A |
| 11 | 3010-2DisplayConc. | CPC3776 | Physical | N/A |
| 12 | 3785DisplayConc. | CPC3776 | Physical | N/A |
| 13 | conc | PSM15 | Physical | N/A |
| 14 | conc | PSM16 | Physical | N/A |
| 15 | conc | PSM_Vienna | Physical | N/A |
| 16 | conc | PSM12 | Physical | N/A |
| 17 | Total Conc. (#/cm) | nanoSMPS | Physical | N/A |
| 18 | I2Br | CMU_FIGAERO | Chemical | N/A |
| 19 | BrH2O | CMU_FIGAERO | Chemical | N/A |
| 20 | NO2 | CAPS | Chemical | N/A |
| 21 | I2_CEDOAS | CE-DOAS | Chemical | N/A |
| 22 | HONO_CEDOAS | CE-DOAS | Chemical | N/A |
| 23 | NO2_CEDOAS | CE-DOAS | Chemical | N/A |
| 24 | Glyoxal_CEDOAS | CE-DOAS | Chemical | N/A |
| 25 | Megly_CEDOAS | CE-DOAS | Chemical | N/A |

*Table B.4: Reduced list of channels to be used for multivariate time series analysis (continued)*

| # | Channel | Instrument | Type | Comment |
|---|---|---|---|---|
| 26 | APINENE | STOF | Chemical | N/A |
| 27 | ISOPRENE | STOF | Chemical | N/A |
| 28 | dew | Dew | Physical | N/A |
| 29 | conc | PSM_PSI | Physical | N/A |
| 30 | HV-Actual | Laser | Physical | N/A |
| 31 | Egy-Actual | Laser | Physical | N/A |
| 32 | APINENE | PTR3 | Chemical | N/A |
| 33 | ISOPRENE | PTR3 | Chemical | N/A |
| 34 | ACETONE | PTR3 | Chemical | N/A |
| 35 | ACETICACID | PTR3 | Chemical | N/A |
| 36 | TMB | PTR3 | Chemical | N/A |
| 37 | NAPHTHALENE | PTR3 | Chemical | N/A |
| 38 | TOLUENE | PTR3 | Chemical | N/A |
| 39 | BETACARYOPHYLENE | PTR3 | Chemical | N/A |
| 40 | PINONALDEHYDE | PTR3 | Chemical | N/A |
| 41 | NH3_2MIN | PICARRO | Chemical | N/A |
| 42 | UVX | Light_Spectrometer | Physical | Sum over wavelengths of UVX |
| 43 | LS3 | Light_Spectrometer | Physical | Sum over wavelengths of LS3 |
| 44 | LS4 | Light_Spectrometer | Physical | Sum over wavelengths of LS4 |
| 45 | LS1 | Light_Spectrometer | Physical | Sum over wavelengths of LS1 |
| 46 | UVH | Light_Spectrometer | Physical | Sum over wavelengths of UVH |
| 47 | Ions_neg | NAIS_ions | Physical | Sum over negative ion channels of NAIS_ions |
| 48 | Ions_pos | NAIS_ions | Physical | Sum over positive ion channels of NAIS_ions |
| 49 | Particles_neg | NAIS_particles | Physical | Sum over positive ion channels of NAIS_particles |
| 50 | Particles_pos | NAIS_particles | Physical | Sum over positive ion channels of NAIS_particles |
| 51 | Particles (20-500 nm) | LongSMPS | Physical | Sum over all particle sizes of Long SMPS |

## B.2 Runs Analysis

This section contains the analysis of each run presented in table 5.1 and which were not shown in 5.2.5. The user is reminded that the results of this analysis with many are available in an interactive web application to be used as a mockup of what would be the use in DAQBroker, accessible via `https://www.daqbroker.com/mvrs_`

mockup/.

### B.2.1 Run 1962.02

This was the second run in a series of particle generation runs, designed to test the contribution of certain compounds to the rate of new particle generation. This particular run introduced charged particles to an already existing background of particle generation, and consequently a relative increase of particle generation is expected. In terms of defining characteristics, it is expected that the particle counters should be highlighted, as in the previous run, and the charged particle counters are also expected to be highlighted.

#### B.2.1.1 Relevant Time Series Highlighting

Figure B.1 shows the most relevant time series according to the PCA method defined in 5.2 at a 98% explained variance. As expected, particle counters are represented, however, no relevance is given to the charged particle counters. This is not unexpected, since the variance contribution of the charged particle counters may be shared with most particle counters and thus ignored in the PCA analysis. No particular trend is immediately distinguishable in the highlighted chemical channels, however, the reduction of acetone concentration could be used as a proxy on the consumption of $\alpha$-pinene for new particle formation, since some studies suggest a link between acetone and oxidation of $\alpha$-pinene[359**?** ].

Figure B.2 shows the result of the series highlighting using a reduced set. The most relevant feature is the number of channels highlighted, which reduced by over than half. This means that quite a lot of information was removed by moving to the reduced set of time series. While, relevant time series are highlighted, specifically the particle concentration measurements, they do not paint the whole picture of a charged particle generation run. It does not seem that for this stage the reduced set is a proper data set to consider to produce more scientifically relevant results.

#### B.2.1.2 Time Series Relationships

As discussed in the previous section, the full set of time series was determined to be more relevant than the reduced set, it will thus be the set used to evaluate the relationships between time series using the method defined in 5.2.3.

**B.2.1.2.1 Pearson correlation similarity** Figure B.3 shows the network similarity graph using the Pearson coefficient applied to the full time series data set, with the size of its node being related to the value of its eigenvector coefficient. Table B.5 helps in identifying nodes in figure B.3 by ranking each node according to its eigenvector coefficient, meaning that higher ranked nodes in table B.5 correspond to larger nodes in figure B.5.

Analyzing figure B.3 and table B.3, there is a clear distinction between two sets of nodes, highlighted by their coloring, which is related to the assigned community of each node according to the Louvain algorithm. A first

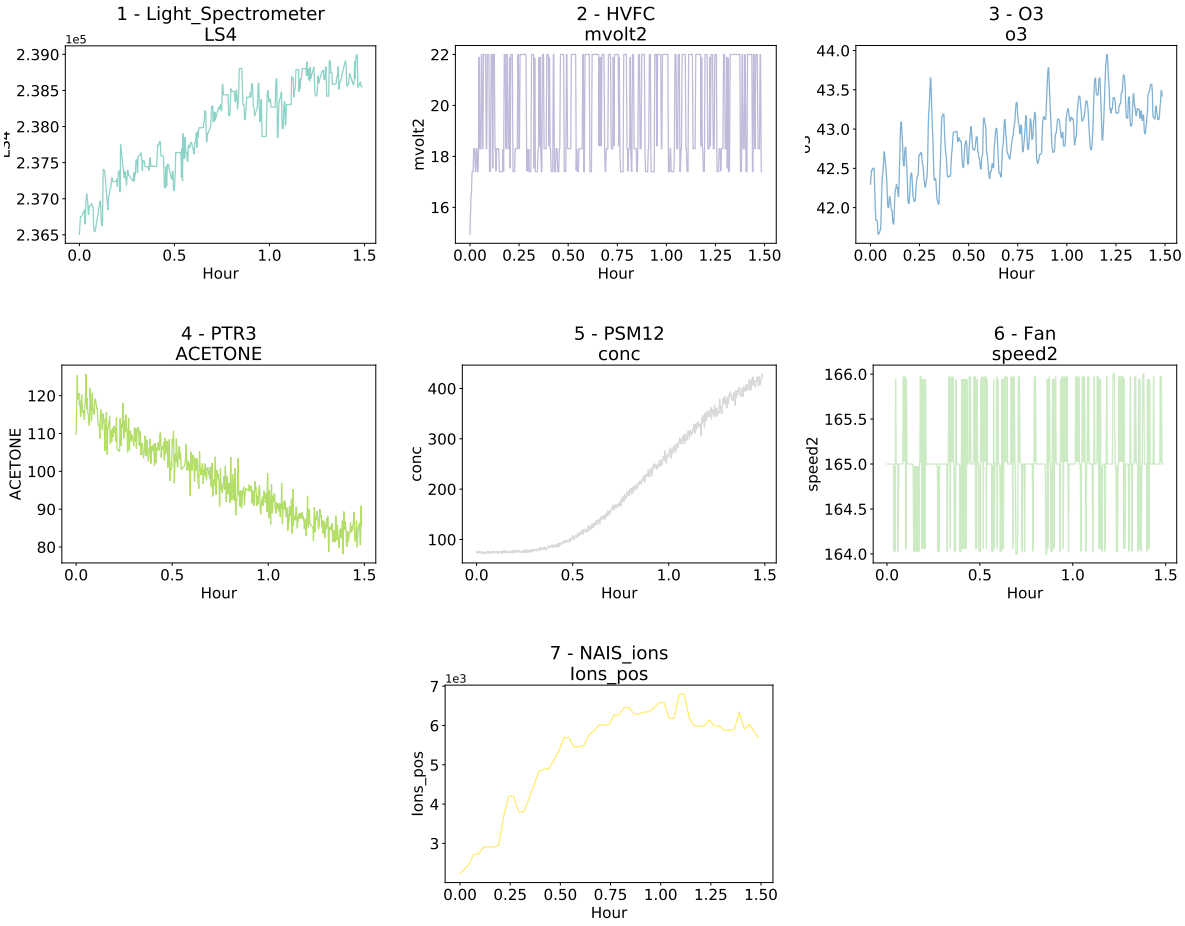



*Figure B.1: Most relevant time series for CLOUD experimental run 1962.02 using the full data set. The compared time series are shown in their original sampled values, before normalization to give the reader an idea of the absolute change of values. This visualization can be shown to users to allow for a more detailed study of the results.*

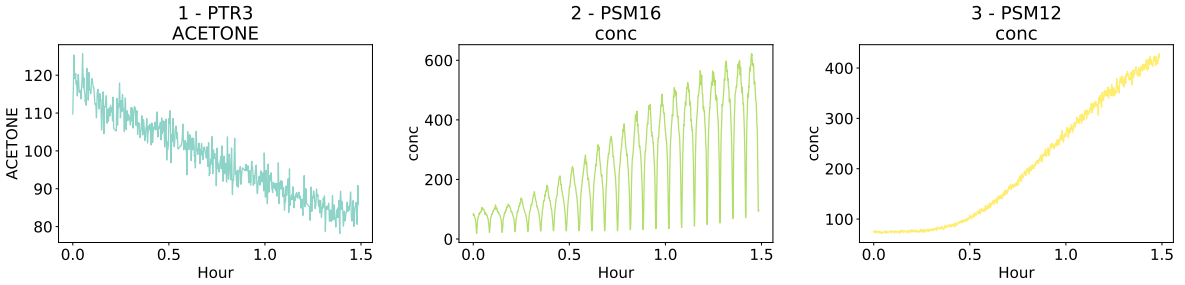



*Figure B.2: Most relevant time series for CLOUD experimental run 1962.02 using the reduced data set. The compared time series are shown in their original sampled values, before normalization to give the reader an idea of the absolute change of values. This visualization can be shown to users to allow for a more detailed study of the results.*

community consists of a more tightly couple set of nodes consisting of mostly particle concentration measurements and light measurements, indicating a strong statistical connection between those groups, which is expected, has been extensively discussed and is also found in figure 5.30, which is analogous to this figure for the previous stage. The second community consists of loosely coupled nodes which populate the rest of the network graph. These appear to be measurements that are loosely couple between each other and it is difficult to define the relevance of this community to the whole set, since it contains time series which should be relevant to the whole data set (for example, the $\alpha$-pinene measurements). Of note in this stage is the addition to the first discussed community of the ion concentration time series (*NAIS_ions Ions_pos* and *NAIS_ions Ions_neg*), which is an expected result since charged particles are generated in this stage whereas in the previous stage this was not the case.

**B.2.1.2.2 Maximum cross-correlation similarity** Figure B.4 shows the similarity network graph with the maximum cross-correlation between pairs of time series as the similarity measurement. Once again, each node is sized by its eigenvector centrality and table B.6 aids in identifying each node in the figure by ranking nodes based on their eigenvector centrality.

Much like comparing tables 5.2 and 5.3, table B.6 presents many similarities to table B.5 with the domination in the highest ranks of light intensity and particle concentration measurements. Again of note the increase in ranking of the charged particle measurements. What differs from the previous graph is the introduction of a third community, further stratifying the data set into groups of which seem to be related much like was discussed in 5.2.5.2.

**B.2.1.2.3 DTW similarity** Figure B.5 illustrates the similarity network graph using the DTW distance as a similarity measurement. Again, table B.7 is provided to aid in the identification of individual nodes.

As in the discussion made in 5.2.5.2 the relationships found between nodes using this measurement are extremely low as evidenced by the reversed grey scale used to quantify the strength of the relationships between nodes, when compared to the other network graphs shown. The same type of channels are also highlighted using this method, which relation several channels from a small number of instruments which indicate that random noise is being highlighted incorrectly, possibly due to improper pre-processing. For this particular similarity measurement it would be more useful to use a reduced set of time series to minimize the existence of spurious relationships between time series. Regardless of the data set used, the values of similarity found are extremely low, meaning that this similarity measurement is probably not suited for the CLOUD time series.

**B.2.1.2.4 Coherence similarity** Finally, figure B.5 shows the similarity network graph using coherence as a similarity measurement. Table B.7 once again is provided to aid in the identification of the nodes in the graph.

Once again as reported in 5.2.5.2, coherence similarity highlights several nodes from the same instruments most likely finding similarity in unprocessed noise. However, there is a marginal increase from the DTW measurement since the relationship values are larger. Some relevant time series are also highlighted, such as a small amount

**Stage 1962.02**

2 communities found

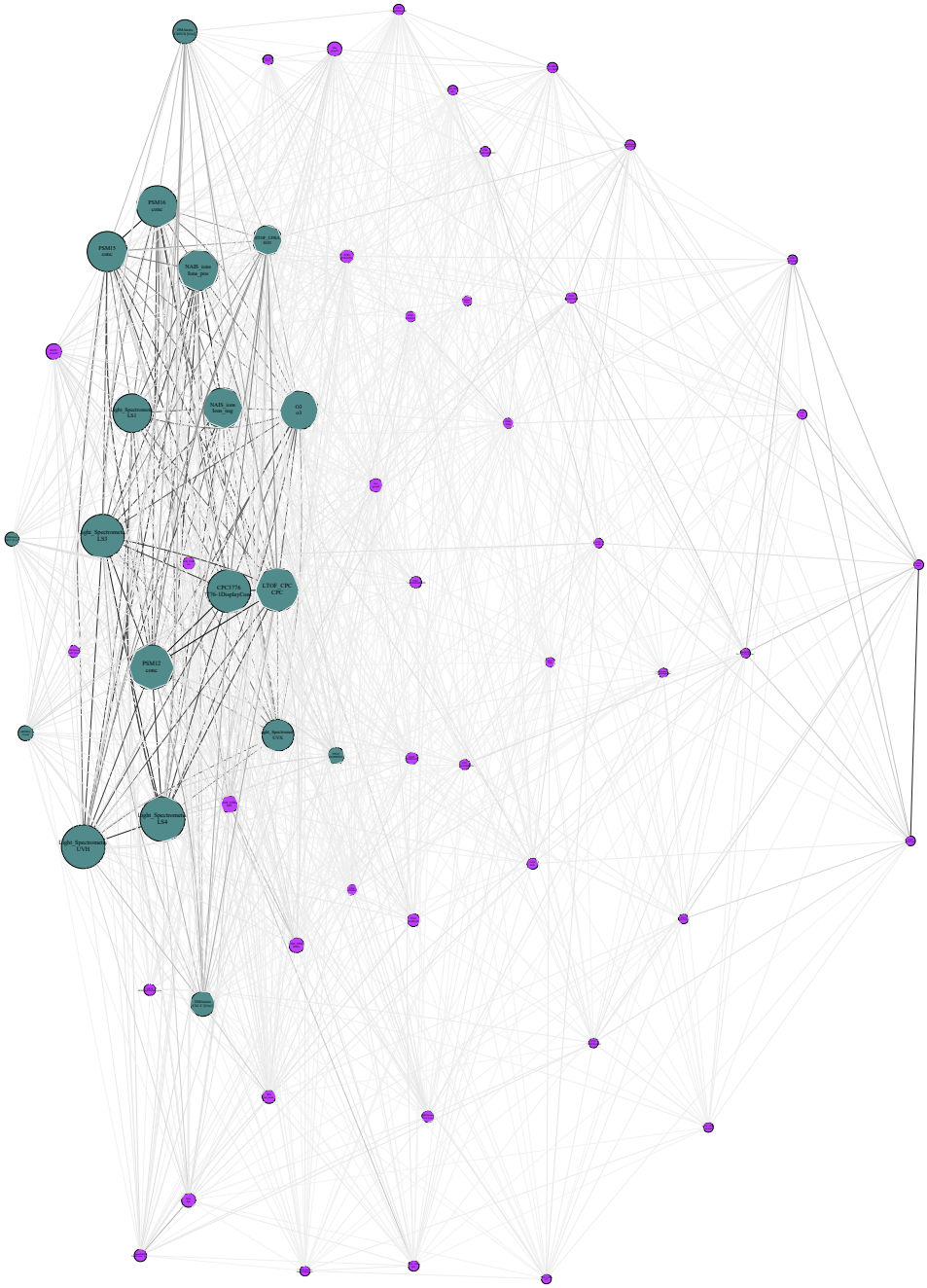


*Figure B.3: Pearson correlation network graph for the CLOUD experimental run 1962.02. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table B.5: Eigenvector ($\epsilon$) centrality node ranking for run 1962.02 using the Pearson correlation as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | Light_Spectrometer | LS4 | 0.295 |
| 2 | PSM12 | conc | 0.2922 |
| 3 | Light_Spectrometer | UVH | 0.2846 |
| 4 | Light_Spectrometer | LS3 | 0.2838 |
| 5 | LTOF_CPC | CPC | 0.2837 |
| 6 | CPC3776 | 3776-1DisplayConc. | 0.282 |
| 7 | PSM16 | conc | 0.2628 |
| 8 | NAIS_ions | Ions_pos | 0.2614 |
| 9 | PSM15 | conc | 0.2558 |
| 10 | NAIS_ions | Ions_neg | 0.2547 |
| 11 | O3 | o3 | 0.2426 |
| 12 | Light_Spectrometer | LS1 | 0.242 |
| 13 | Light_Spectrometer | UVX | 0.1852 |
| 14 | HTOF_UFRA | H30 | 0.1667 |
| 15 | DMAtrain | Ch1 C [#/cc] | 0.1267 |
| 16 | DMAtrain | Ch0 Ch [#/cc] | 0.1214 |
| 17 | LTOF_UFRA | NIT | 0.0632 |
| 18 | STOF | ISOPRENE | 0.0598 |
| 19 | HVFC | mvolt2 | 0.0508 |
| 20 | LTOF_UFRA | DMA | 0.0473 |
| 21 | nRDMA | Count | 0.046 |
| 22 | Fan | speed1 | 0.0395 |
| 23 | Dew | dew | 0.038 |
| 24 | DMAtrain | Ch2 C [#/cc] | 0.0371 |
| 25 | Fan | speed2 | 0.0349 |
| 26 | PTR3 | TOLUENE | 0.0343 |
| 27 | TDL | sat_water | 0.0327 |
| 28 | PTR3 | ISOPRENE | 0.028 |
| 29 | LTOF_UFRA | SA | 0.0265 |
| 30 | STOF | ACETICACID | 0.0232 |
| 31 | nanoSMPS | Total Conc. #/cm | 0.022 |
| 32 | DMAtrain | Ch4 C [#/cc] | 0.0218 |
| 33 | PTR3 | PINONALDEHYDE | 0.0191 |
| 34 | DMAtrain | Ch3 C [#/cc] | 0.0182 |
| 35 | STOF | BETACARYOPHYLENE | 0.0167 |
| 36 | PTR3 | ACETICACID | 0.0141 |
| 37 | CAPS | NO2 | 0.0129 |
| 38 | STOF | APINENE | 0.0117 |
| 39 | STOF | TOLUENE | 0.0114 |
| 40 | STOF | NAPHTHALENE | 0.0104 |
| 41 | PTR3 | TMB | 0.0102 |
| 42 | LTOF_UFRA | HIO | 0.009 |
| 43 | STOF | PINONALDEHYDE | 0.0089 |
| 44 | PICARRO | NH3_2MIN | 0.0083 |
| 45 | PTR3 | BETACARYOPHYLENE | 0.0075 |
| 46 | Scalers | brust | 0.0069 |
| 47 | PhotoDiode12 | intensity2 | 0.0063 |
| 48 | HTOF_UFRA | H7O3 | 0.0054 |
| 49 | HTOF_UFRA | NH3 | 0.0037 |
| 50 | SabreTemperature | Sabre3Temperature | 0.0032 |
| 51 | STOF | TMB | 0.003 |
| 52 | HVFC | mvolt1 | 0.0027 |
| 53 | SabreTemperature | Sabre4Temperature | 0.002 |
| 54 | LongSMPS | Particles 20-500 nm | 0.0016 |
| 55 | HTOF_UFRA | O2 | 0.0015 |
| 56 | PTR3 | APINENE | 0.0009 |
| 57 | PTR3 | NAPHTHALENE | 0.0009 |
| 58 | HTOF_UFRA | H3OH2O | 0.0009 |
| 59 | STOF | ACETONE | 0.0009 |
| 60 | SO2 | so2 | 0.0007 |
| 61 | PT100 | temp1 | 0.0006 |
| 62 | PTR3 | ACETONE | 0.0005 |

# Stage 1962.02

3 communities found

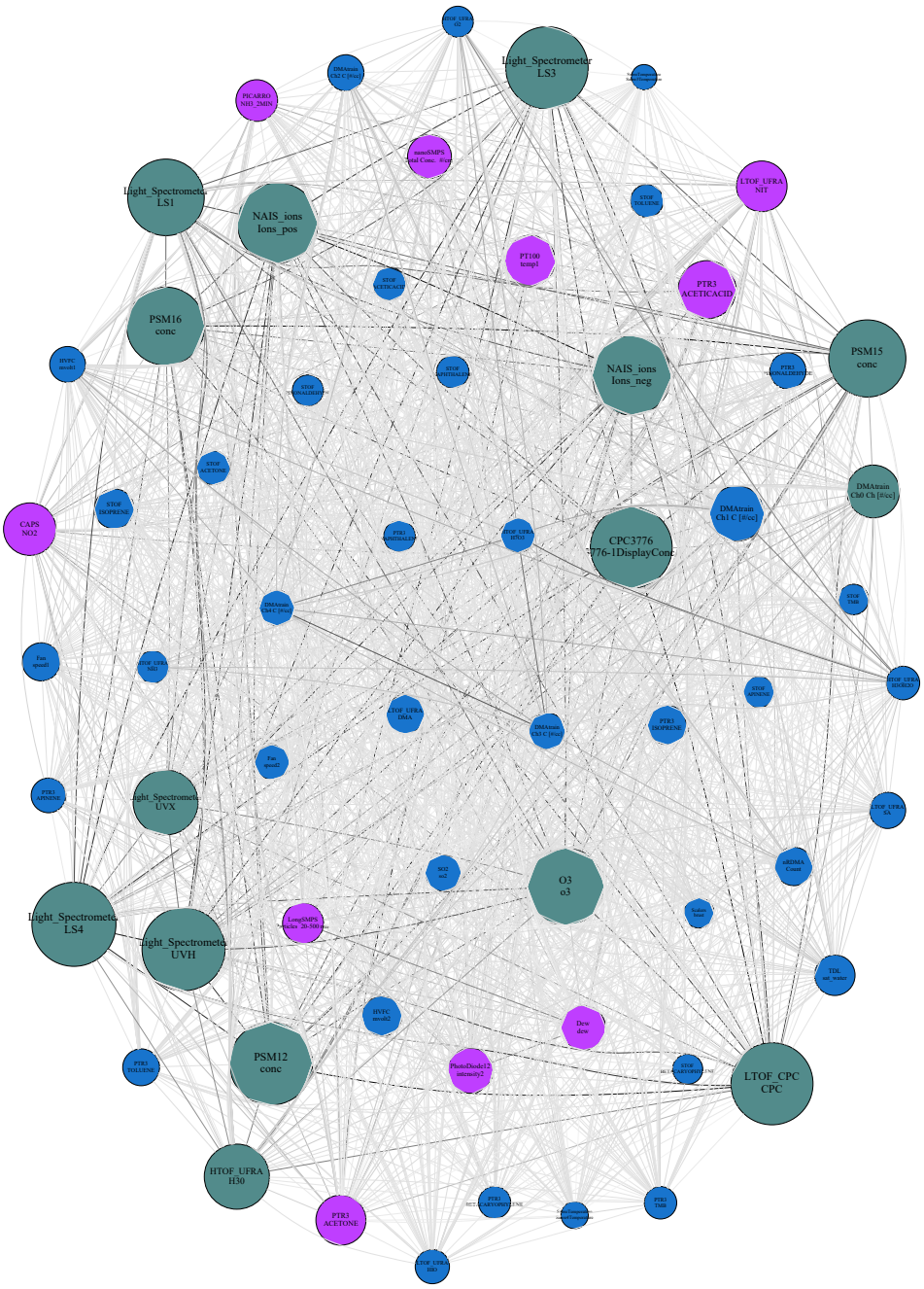



*Figure B.4: Maximum cross-correlation network graph for the CLOUD experimental run 1962.02. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table B.6: Eigenvector ($\epsilon$) centrality node ranking for run 1962.02 using the maximum cross-correlation as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | Light_Spectrometer | LS4 | 0.2399 |
| 2 | Light_Spectrometer | UVH | 0.2364 |
| 3 | PSM12 | conc | 0.2363 |
| 4 | Light_Spectrometer | LS3 | 0.2338 |
| 5 | LTOF_CPC | CPC | 0.2329 |
| 6 | CPC3776 | 3776-1DisplayConc. | 0.2321 |
| 7 | NAIS_ions | Ions_pos | 0.2278 |
| 8 | NAIS_ions | Ions_neg | 0.2238 |
| 9 | PSM16 | conc | 0.2192 |
| 10 | O3 | o3 | 0.2158 |
| 11 | PSM15 | conc | 0.2154 |
| 12 | Light_Spectrometer | LS1 | 0.2134 |
| 13 | HTOF_UFRA | H30 | 0.1728 |
| 14 | Light_Spectrometer | UVX | 0.1726 |
| 15 | PTR3 | ACETICACID | 0.1482 |
| 16 | DMAtrain | Ch1 C [#/cc] | 0.1383 |
| 17 | DMAtrain | Ch0 Ch [#/cc] | 0.1294 |
| 18 | CAPS | NO2 | 0.1283 |
| 19 | LTOF_UFRA | NIT | 0.1217 |
| 20 | PT100 | temp1 | 0.1217 |
| 21 | PTR3 | ACETONE | 0.1197 |
| 22 | PhotoDiode12 | intensity2 | 0.1032 |
| 23 | Dew | dew | 0.1014 |
| 24 | nanoSMPS | Total Conc. #/cm | 0.0978 |
| 25 | PICARRO | NH3_2MIN | 0.0899 |
| 26 | LongSMPS | Particles 20-500 nm | 0.0878 |
| 27 | TDL | sat_water | 0.0859 |
| 28 | HVFC | mvolt2 | 0.0849 |
| 29 | STOF | ISOPRENE | 0.0837 |
| 30 | PTR3 | ISOPRENE | 0.0825 |
| 31 | LTOF_UFRA | DMA | 0.0814 |
| 32 | nRDMA | Count | 0.0788 |
| 33 | Fan | speed1 | 0.0775 |
| 34 | DMAtrain | Ch3 C [#/cc] | 0.0735 |
| 35 | PTR3 | TOLUENE | 0.073 |
| 36 | SO2 | so2 | 0.072 |
| 37 | PTR3 | PINONALDEHYDE | 0.0708 |
| 38 | LTOF_UFRA | SA | 0.0708 |
| 39 | DMAtrain | Ch2 C [#/cc] | 0.0706 |
| 40 | DMAtrain | Ch4 C [#/cc] | 0.0695 |
| 41 | HVFC | mvolt1 | 0.0693 |
| 42 | Fan | speed2 | 0.0665 |
| 43 | HTOF_UFRA | H7O3 | 0.0661 |
| 44 | PTR3 | APINENE | 0.0653 |
| 45 | LTOF_UFRA | HIO | 0.0637 |
| 46 | STOF | ACETONE | 0.0627 |
| 47 | HTOF_UFRA | H3OH2O | 0.0614 |
| 48 | STOF | ACETICACID | 0.061 |
| 49 | STOF | TOLUENE | 0.0592 |
| 50 | STOF | NAPHTHALENE | 0.0578 |
| 51 | HTOF_UFRA | NH3 | 0.0576 |
| 52 | PTR3 | TMB | 0.0575 |
| 53 | PTR3 | BETACARYOPHYLENE | 0.0566 |
| 54 | PTR3 | NAPHTHALENE | 0.055 |
| 55 | STOF | APINENE | 0.0536 |
| 56 | STOF | PINONALDEHYDE | 0.0534 |
| 57 | HTOF_UFRA | O2 | 0.0525 |
| 58 | STOF | TMB | 0.0507 |
| 59 | Scalers | brust | 0.0497 |
| 60 | STOF | BETACARYOPHYLENE | 0.0493 |
| 61 | SabreTemperature | Sabre4Temperature | 0.0394 |
| 62 | SabreTemperature | Sabre3Temperature | 0.0325 |

# Stage 1962.02

3 communities found

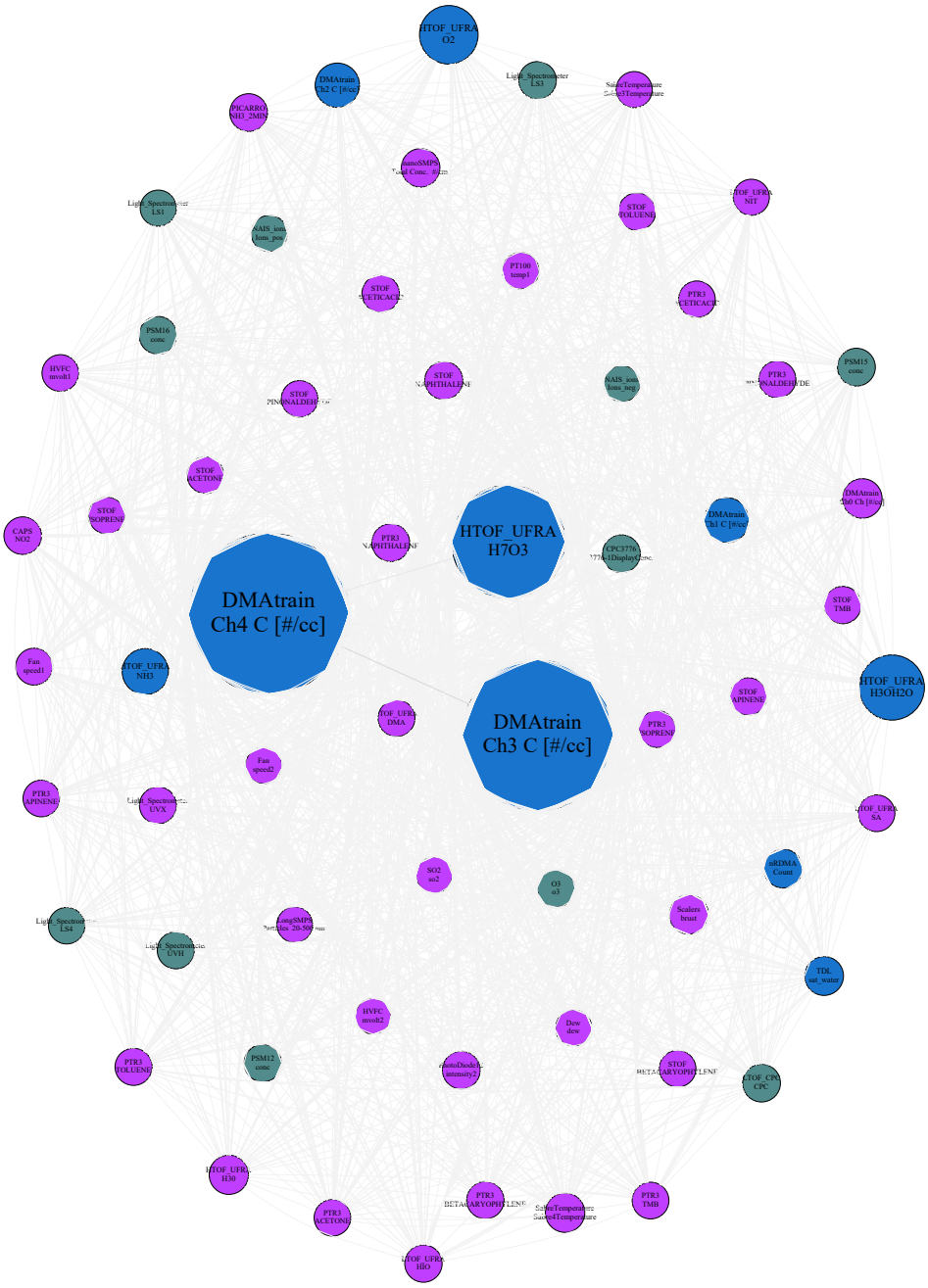



*Figure B.5: DTW similarity network graph for the CLOUD experimental run 1962.02. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table B.7: Eigenvector ($\epsilon$) centrality node ranking for run 1962.02 using the DTW distance as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | DMAtrain | Ch4 C [#/cc] | 0.6458 |
| 2 | DMAtrain | Ch3 C [#/cc] | 0.598 |
| 3 | HTOF_UFRA | H7O3 | 0.4055 |
| 4 | HTOF_UFRA | H3OH2O | 0.157 |
| 5 | HTOF_UFRA | O2 | 0.123 |
| 6 | HTOF_UFRA | NH3 | 0.0594 |
| 7 | DMAtrain | Ch1 C [#/cc] | 0.0541 |
| 8 | DMAtrain | Ch2 C [#/cc] | 0.0502 |
| 9 | DMAtrain | Ch0 Ch [#/cc] | 0.0262 |
| 10 | SabreTemperature | Sabre4Temperature | 0.0247 |
| 11 | Scalers | brust | 0.0244 |
| 12 | HTOF_UFRA | H30 | 0.0242 |
| 13 | TDL | sat_water | 0.0223 |
| 14 | nRDMA | Count | 0.0222 |
| 15 | nanoSMPS | Total Conc. #/cm | 0.0207 |
| 16 | STOF | NAPHTHALENE | 0.0185 |
| 17 | PSM15 | conc | 0.0172 |
| 18 | PSM16 | conc | 0.017 |
| 19 | PTR3 | NAPHTHALENE | 0.0167 |
| 20 | PICARRO | NH3_2MIN | 0.0166 |
| 21 | LTOF_CPC | CPC | 0.0166 |
| 22 | PhotoDiode12 | intensity2 | 0.0166 |
| 23 | CPC3776 | 3776-1DisplayConc. | 0.0166 |
| 24 | CAPS | NO2 | 0.0159 |
| 25 | STOF | TOLUENE | 0.0155 |
| 26 | Light_Spectrometer | LS3 | 0.0154 |
| 27 | PSM12 | conc | 0.0153 |
| 28 | O3 | o3 | 0.0149 |
| 29 | STOF | TMB | 0.0147 |
| 30 | STOF | APINENE | 0.0146 |
| 31 | STOF | ACETICACID | 0.0146 |
| 32 | LTOF_UFRA | HIO | 0.0145 |
| 33 | PTR3 | BETACARYOPHYLENE | 0.0143 |
| 34 | PTR3 | TOLUENE | 0.0134 |
| 35 | PTR3 | PINONALDEHYDE | 0.013 |
| 36 | Light_Spectrometer | UVH | 0.013 |
| 37 | Light_Spectrometer | UVX | 0.0128 |
| 38 | PTR3 | ISOPRENE | 0.0127 |
| 39 | Fan | speed1 | 0.0126 |
| 40 | LTOF_UFRA | DMA | 0.0119 |
| 41 | STOF | ACETONE | 0.0116 |
| 42 | PTR3 | TMB | 0.0115 |
| 43 | STOF | ISOPRENE | 0.0114 |
| 44 | HVFC | mvolt1 | 0.0114 |
| 45 | PT100 | temp1 | 0.0114 |
| 46 | LTOF_UFRA | SA | 0.0112 |
| 47 | PTR3 | APINENE | 0.0111 |
| 48 | PTR3 | ACETONE | 0.011 |
| 49 | PTR3 | ACETICACID | 0.0109 |
| 50 | STOF | BETACARYOPHYLENE | 0.0109 |
| 51 | STOF | PINONALDEHYDE | 0.0108 |
| 52 | Light_Spectrometer | LS4 | 0.0103 |
| 53 | SO2 | so2 | 0.0098 |
| 54 | Fan | speed2 | 0.0098 |
| 55 | NAIS_ions | Ions_neg | 0.0095 |
| 56 | Dew | dew | 0.0094 |
| 57 | NAIS_ions | Ions_pos | 0.0091 |
| 58 | Light_Spectrometer | LS1 | 0.0083 |
| 59 | SabreTemperature | Sabre3Temperature | 0.0083 |
| 60 | LTOF_UFRA | NIT | 0.0078 |
| 61 | LongSMPS | Particles 20-500 nm | 0.0065 |
| 62 | HVFC | mvolt2 | 0.0061 |

of light intensity time series and particle concentration measurements. Once again, this measurement should be used in a reduced set to reduce the chance of spurious similarity, making it not very proper for use in exploratory analysis.

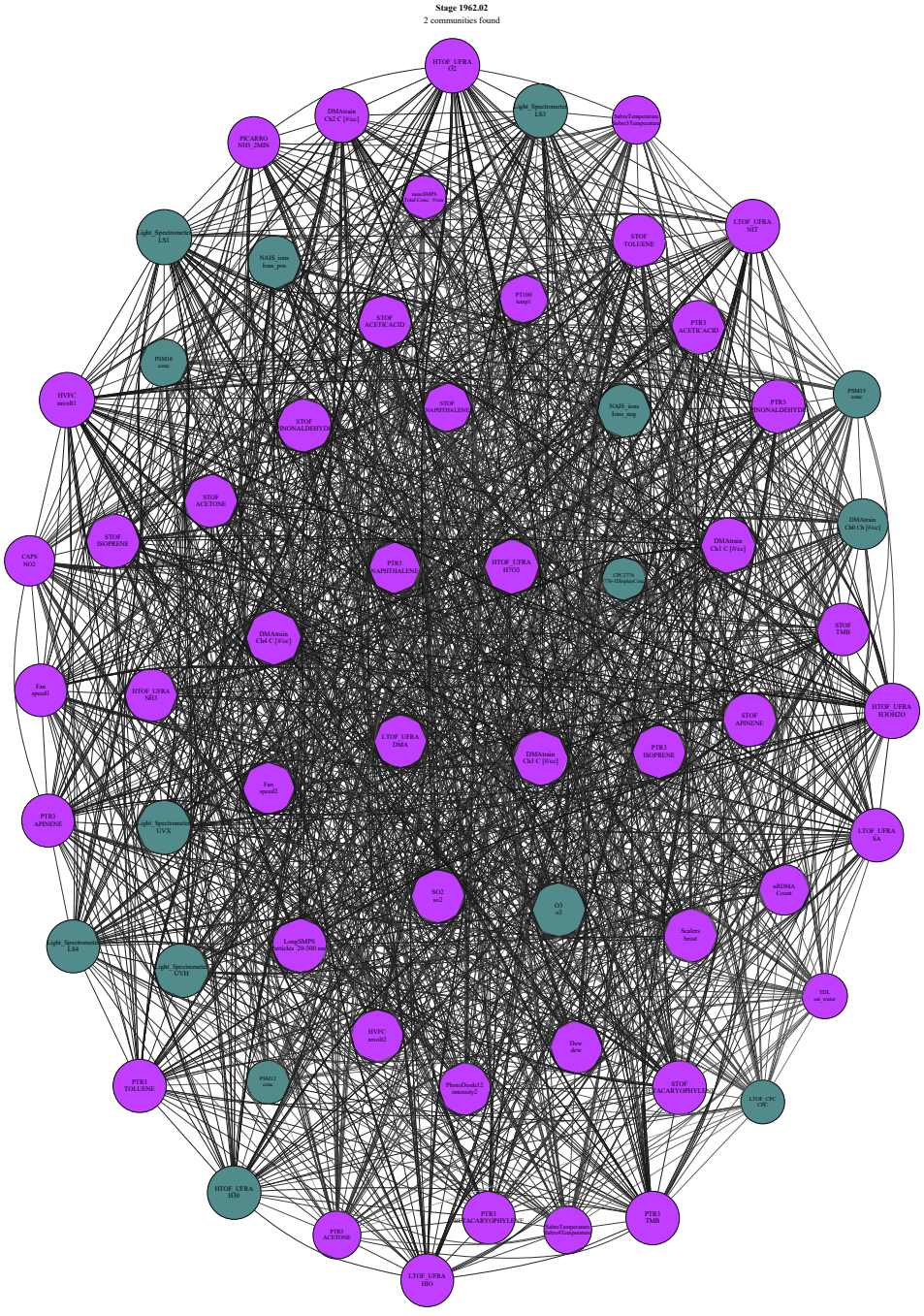



*Figure B.6: Coherence network graph for the CLOUD experimental run 1962.02. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

### B.2.1.3 Relevant Period Selection

Figure B.7 shows the shaded version of figure B.1, when accounting for event detection, as described in 5.2. The full dataset is used here because in B.2.1.1 the full data set was deemed more relevant than the reduced set.

*Table B.8: Eigenvector ($\epsilon$) centrality node ranking for run 1962.02 using the coherence as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | HVFC | mvolt1 | 0.1356 |
| 2 | HTOF_UFRA | H3OH2O | 0.1355 |
| 3 | DMAtrain | Ch3 C [#/cc] | 0.1354 |
| 4 | DMAtrain | Ch4 C [#/cc] | 0.1351 |
| 5 | Light_Spectrometer | LS1 | 0.135 |
| 6 | DMAtrain | Ch1 C [#/cc] | 0.1349 |
| 7 | HTOF_UFRA | H7O3 | 0.1345 |
| 8 | HTOF_UFRA | O2 | 0.1334 |
| 9 | Light_Spectrometer | UVX | 0.1323 |
| 10 | LongSMPS | Particles 20-500 nm | 0.1322 |
| 11 | LTOF_UFRA | NIT | 0.1322 |
| 12 | Light_Spectrometer | LS3 | 0.1321 |
| 13 | Light_Spectrometer | UVH | 0.1321 |
| 14 | STOF | BETACARYOPHYLENE | 0.1319 |
| 15 | DMAtrain | Ch2 C [#/cc] | 0.1316 |
| 16 | O3 | o3 | 0.1314 |
| 17 | STOF | ISOPRENE | 0.1313 |
| 18 | HTOF_UFRA | H30 | 0.1311 |
| 19 | PTR3 | TMB | 0.1309 |
| 20 | PTR3 | PINONALDEHYDE | 0.1308 |
| 21 | Scalers | brust | 0.1307 |
| 22 | STOF | APINENE | 0.1307 |
| 23 | PTR3 | ACETICACID | 0.1307 |
| 24 | LTOF_UFRA | SA | 0.1305 |
| 25 | STOF | PINONALDEHYDE | 0.1305 |
| 26 | SO2 | so2 | 0.1303 |
| 27 | PTR3 | ISOPRENE | 0.1302 |
| 28 | PTR3 | TOLUENE | 0.1301 |
| 29 | NAIS_ions | Ions_neg | 0.1297 |
| 30 | STOF | ACETONE | 0.1297 |
| 31 | STOF | ACETICACID | 0.1296 |
| 32 | Light_Spectrometer | LS4 | 0.1296 |
| 33 | NAIS_ions | Ions_pos | 0.1296 |
| 34 | PTR3 | APINENE | 0.1294 |
| 35 | STOF | TOLUENE | 0.1293 |
| 36 | LTOF_UFRA | HIO | 0.1291 |
| 37 | LTOF_UFRA | DMA | 0.1291 |
| 38 | PTR3 | BETACARYOPHYLENE | 0.129 |
| 39 | Fan | speed1 | 0.1286 |
| 40 | HTOF_UFRA | NH3 | 0.1283 |
| 41 | STOF | TMB | 0.1282 |
| 42 | HVFC | mvolt2 | 0.1279 |
| 43 | Dew | dew | 0.1274 |
| 44 | PICARRO | NH3_2MIN | 0.127 |
| 45 | Fan | speed2 | 0.1267 |
| 46 | PhotoDiode12 | intensity2 | 0.1266 |
| 47 | PTR3 | NAPHTHALENE | 0.1258 |
| 48 | DMAtrain | Ch0 Ch [#/cc] | 0.1233 |
| 49 | nRDMA | Count | 0.1229 |
| 50 | CAPS | NO2 | 0.1226 |
| 51 | SabreTemperature | Sabre3Temperature | 0.1175 |
| 52 | STOF | NAPHTHALENE | 0.1172 |
| 53 | PT100 | temp1 | 0.1165 |
| 54 | PTR3 | ACETONE | 0.1162 |
| 55 | SabreTemperature | Sabre4Temperature | 0.1156 |
| 56 | PSM16 | conc | 0.1155 |
| 57 | PSM15 | conc | 0.1146 |
| 58 | TDL | sat_water | 0.1096 |
| 59 | LTOF_CPC | CPC | 0.1055 |
| 60 | nanoSMPS | Total Conc. #/cm | 0.1053 |
| 61 | PSM12 | conc | 0.1049 |
| 62 | CPC3776 | 3776-1DisplayConc. | 0.1039 |

Contrary to the previous run, most events were located in the first half of the experimental run, that are associated to the increase in charged particles witnessed in both positive and negative particle counters during the first half of the experimental run. Table B.9 shows the distribution of found events over each data channel. While several of the PCA-highlighted channels are not present in this table, a total of 4 channels out of the 13 presented contribute with 75 events, around 40% of the total detected events. While this indicates better selectivity compared with run 1962.01, there is need for optimization of the event detection method, such as defining custom limit parameters to account for the specifications of each channel.

*Table B.9: Events in each period for run 1962.02 using wavelet ME partitioning.*

| Channel | Period 1 | Period 2 | Period 3 | Period 4 | Total |
|---|---|---|---|---|---|
| nanoSMPS Total Conc. #/cm | 9 | 2 | 0 | 6 | 17 |
| NAIS_ions Ions_pos | 2 | 4 | 3 | 7 | 16 |
| HTOF_UFRA O2 | 8 | 7 | 0 | 0 | 15 |
| Fan speed1 | 7 | 4 | 2 | 0 | 13 |
| HTOF_UFRA NH3 | 3 | 4 | 0 | 6 | 13 |
| LongSMPS Particles 20-500 nm | 0 | 8 | 2 | 2 | 12 |
| HVFC mvolt1 | 9 | 0 | 0 | 0 | 9 |
| Light_Spectrometer LS3 | 2 | 4 | 1 | 1 | 8 |
| STOF NAPHTHALENE | 2 | 3 | 1 | 2 | 8 |
| HVFC mvolt2 | 2 | 4 | 0 | 1 | 7 |
| NAIS_ions Ions_neg | 0 | 3 | 4 | 0 | 7 |
| TDL sat_water | 4 | 0 | 1 | 1 | 6 |
| Scalers brust | 0 | 0 | 6 | 0 | 6 |
| Light_Spectrometer LS4 | 1 | 3 | 0 | 1 | 5 |
| Light_Spectrometer UVH | 2 | 0 | 3 | 0 | 5 |
| Dew dew | 0 | 0 | 4 | 0 | 4 |
| Fan speed2 | 0 | 0 | 0 | 3 | 3 |
| DMAtrain Ch1 C [#/cc] | 3 | 0 | 0 | 0 | 3 |
| PTR3 TOLUENE | 1 | 1 | 0 | 0 | 2 |
| Light_Spectrometer LS1 | 1 | 0 | 0 | 1 | 2 |
| Light_Spectrometer UVX | 0 | 0 | 0 | 1 | 1 |
| CAPS NO2 | 0 | 0 | 1 | 0 | 1 |
| PTR3 ACETICACID | 1 | 0 | 0 | 0 | 1 |
| PTR3 NAPHTHALENE | 1 | 0 | 0 | 0 | 1 |
| Total | 58 | 47 | 28 | 32 | 165 |

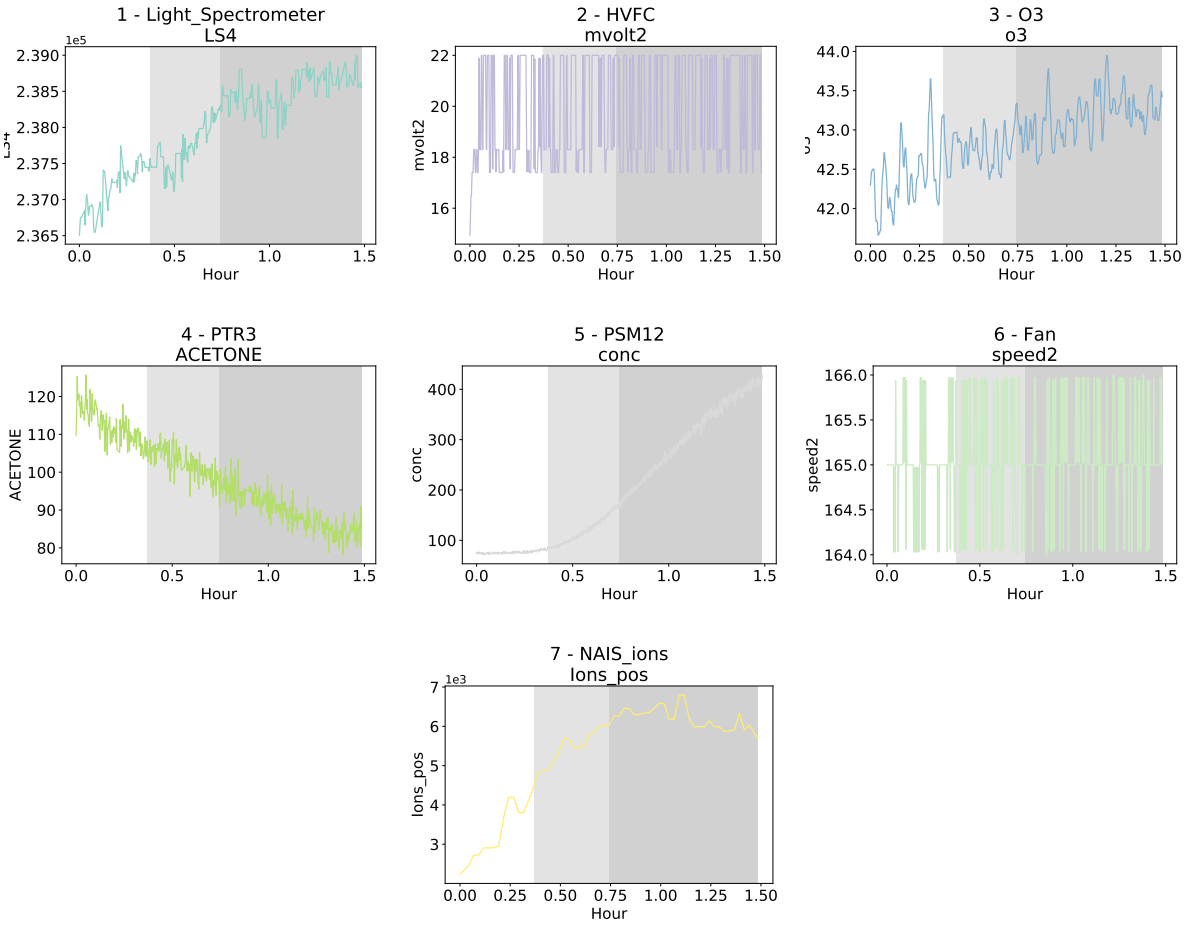



*Figure B.7: Most relevant time series for CLOUD experimental run 1962.02 coupled with ranking of specific time periods to the number of events observed in each event. The time period was divided into 4 distinct sub-periods, which were then tested for the existence of events. The ranking of each sub-period shades each sub-period more the less events are found. In this case, as time increases, more events were found in each sub-period.*

### B.2.2 Run 1962.03

This run is probably the most different in terms of objectives of all the other runs presented, as its goal is to eliminate, rather than create particles in the CLOUD chamber. This is done using an electrical field that pushes particles towards the edges of the chamber, ensuring that said particles be either purged from or sent to the walls of the chamber. This is required when a significant number of particles exist in the chamber to avoid interference with further particle generation experiments. During this experimental run, the levels of $\alpha$-pinene were also increased from 100 to 250 ppb for further particle generation experiments.

#### B.2.2.1 Relevant Time Series Highlighting

Figure B.8 shows the most relevant time series. As expected from the description of this run, the $\alpha$-pinene time series is highlighted in figure B.8, however, no particle concentration measurements are highlighted, instead showing two different measurements of other chemical species, specifically $NO_2$ and $H_3O$. It is unclear what the importance of the $H_3O$ channel is, it could be that the measurement is being made below detection limit and the noise of this specific channel is being highlighted but it is overwhelmingly clear the the $NO_2$ measurement is either being made below detection limit or is not properly calibrated. There is thus a clear need to either apply a channel-specific noise reduction model or simply remove these channels from the analysis. Also of note is the reduction in relevant time series for this run, hinting that there is a reduction of the variance in this run, thus less time series are needed to explain the same percentage of variance from other runs.

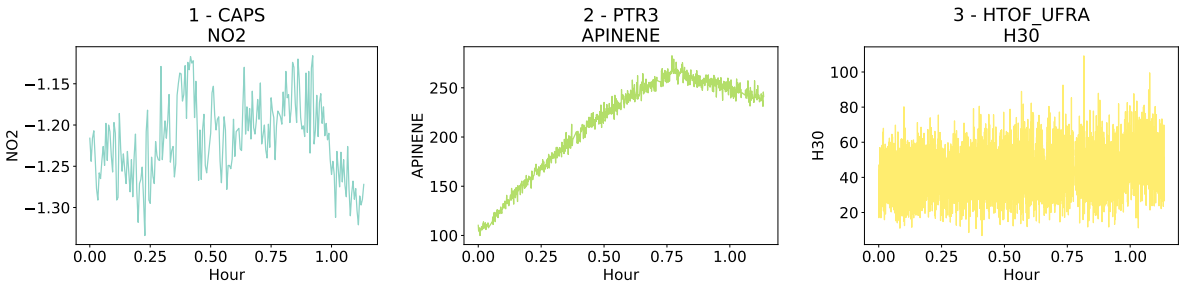



*Figure B.8: Most relevant time series for CLOUD experimental run 1962.03. The compared time series are shown in their original sampled values, before normalization to give the reader an idea of the absolute change of values. This visualization can be shown to users to allow for a more detailed study of the results.*

Using the reduced set of time series, a change is the output is observed, which is highlighted in figure B.9. While the $\alpha$-pinene measurement is still highlighted, now the $NO_2$ and $H_3O$ are replaced by a concentration measurement (*PSM12*) and a measurement of another chemical species, acetic acid. While the acetic acid measurement still seems to be a noisy measurement, there is a clear explanation for the highlighting of the particle concentration measurement as stated earlier. It is also noticeable that reducing the time series set also did not reduce the amount of information contained since it was still required that three time series explain the same amount of variance. It thus seems that for this stage, the reduced time series set is a better set to analyze.

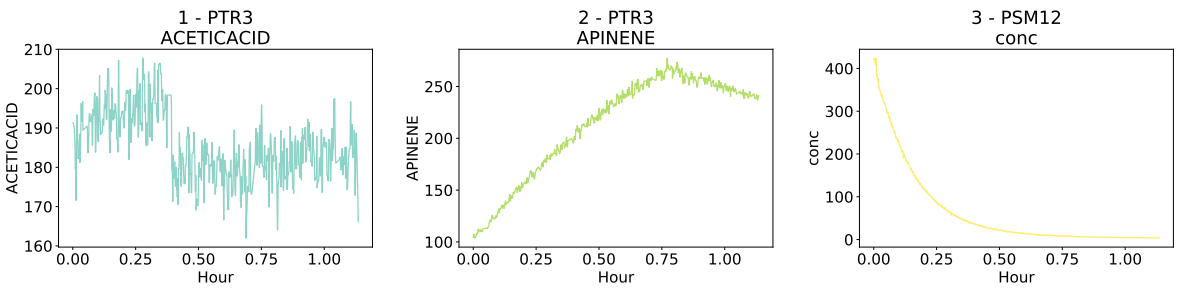



*Figure B.9: Most relevant time series for CLOUD experimental run 1962.03 using a reduced set of time series. The compared time series are shown in their original sampled values, before normalization to give the reader an idea of the absolute change of values. This visualization can be shown to users to allow for a more detailed study of the results.*

#### B.2.2.2 Time Series Relationships

As discussed in the previous section, the reduced set of time series was determined to be more scientifically relevant than the full set, unlike previous runs, this will then be the set used to evaluate the relationships between time series.

**B.2.2.2.1 Pearson correlation similarity** Figure B.10 shows the similarity network graph using the Pearson correlation as similarity measurement. Each node is sized according to its eignevector centrality and table B.10 ranks each node also according to its eigenvector centrality, thus it helps identifying the nodes in figure B.10 by relating larger nodes in the figure with higher ranking nodes in the table.

When analysing both figure and the table, a clear departure from previous stages is seen. Several different time series are highly ranked in table B.10, ranging from temperature measurements, chemical measurements and light intensity measurements. This result is in line with the description provided of this run, highlighting two different $\alpha$-pinene signals and other organic compounds (e.g: *PTR3 Toluene*), temperature measurements which can be explained by thermal load from the increase of the speed of the fans (this is a known taboo issue in CLOUD). However, very little particle concentration measurements are highlighted and all of them are relegated to the bottom half of table B.10, making them less statistically relevant for this stage. Once again, several communities are found using the Louvain algorithm, with little explanation as to the scientific relationships that exist inside each community.

**B.2.2.2.2 Maximum cross-correlation similarity** Figure B.11 shows the similarity network with maximum cross-correlation as a similarity measurement. Once again, table B.11 helps identify nodes in figure B.11 by relating the size of each node in the figure to an eigenvector centrality in the table (larger nodes are higher ranked in eigenvector centrality).

Analysing both the figure and the table shows very similar behaviour to using the Pearson correlation, much like in previous stages. However, particle concentration measurements more and higher ranked for this similarity

**Stage 1962.03**

4 communities found

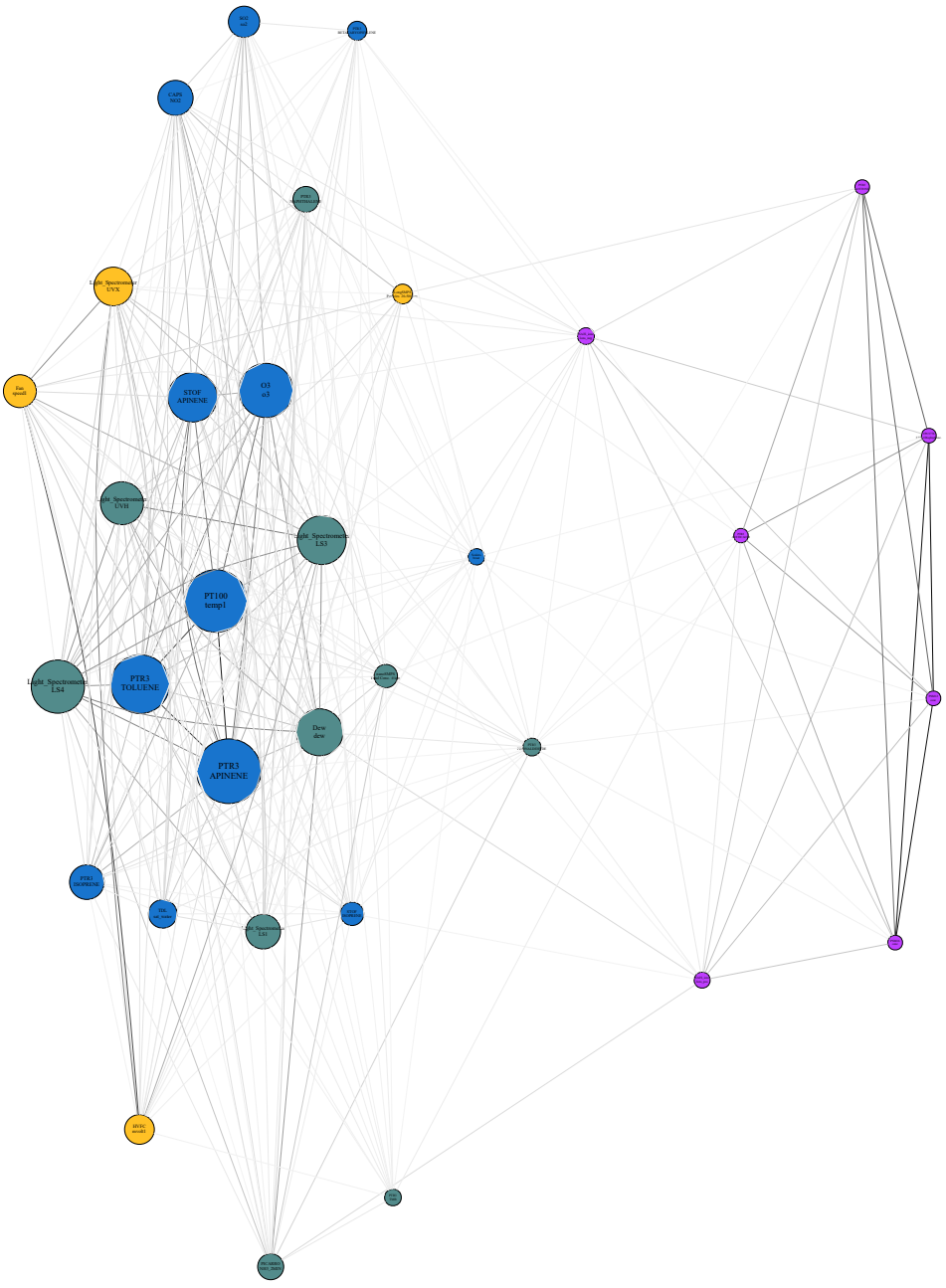



*Figure B.10: Pearson correlation network graph for the CLOUD experimental run 1962.03. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table B.10: Eigenvector ($\epsilon$) centrality node ranking for run 1962.03 using the Pearson correlation as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | PTR3 | APINENE | 0.3761 |
| 2 | PT100 | temp1 | 0.3612 |
| 3 | PTR3 | TOLUENE | 0.3332 |
| 4 | O3 | o3 | 0.2988 |
| 5 | Light_Spectrometer | LS4 | 0.2912 |
| 6 | STOF | APINENE | 0.2638 |
| 7 | Light_Spectrometer | LS3 | 0.2592 |
| 8 | Dew | dew | 0.2451 |
| 9 | Light_Spectrometer | UVH | 0.2152 |
| 10 | Light_Spectrometer | UVX | 0.1836 |
| 11 | CAPS | NO2 | 0.1576 |
| 12 | Light_Spectrometer | LS1 | 0.1543 |
| 13 | PTR3 | ISOPRENE | 0.1534 |
| 14 | Fan | speed1 | 0.1407 |
| 15 | SO2 | so2 | 0.1288 |
| 16 | HVFC | mvolt1 | 0.1173 |
| 17 | TDL | sat_water | 0.1083 |
| 18 | PICARRO | NH3_2MIN | 0.0899 |
| 19 | PTR3 | NAPHTHALENE | 0.0885 |
| 20 | STOF | ISOPRENE | 0.0706 |
| 21 | nanoSMPS | Total Conc. #/cm | 0.0699 |
| 22 | LongSMPS | Particles 20-500 nm | 0.0435 |
| 23 | PTR3 | BETACARYOPHYLENE | 0.0391 |
| 24 | PTR3 | PINONALDEHYDE | 0.0291 |
| 25 | NAIS_ions | Ions_neg | 0.0214 |
| 26 | PTR3 | TMB | 0.0203 |
| 27 | Scalers | brust | 0.019 |
| 28 | NAIS_ions | Ions_pos | 0.0153 |
| 29 | PSM12 | conc | 0.006 |
| 30 | CPC3776 | 3776-1DisplayConc. | 0.006 |
| 31 | PSM16 | conc | 0.0058 |
| 32 | PTR3 | ACETONE | 0.0055 |
| 33 | PTR3 | ACETICACID | 0.0037 |

measurements, meaning that there is a potential delayed relationship between time series.

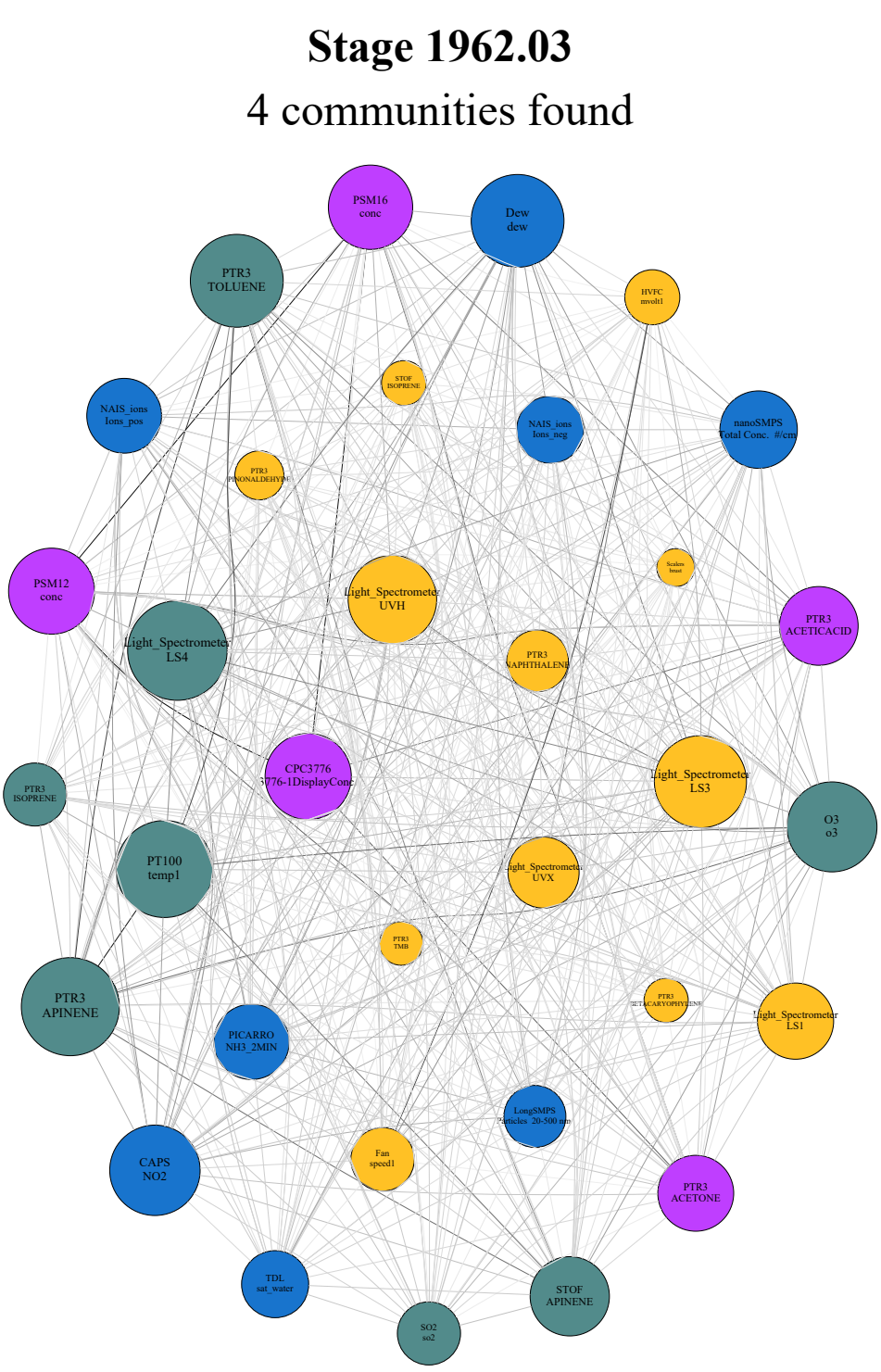



*Figure B.11: Maximum cross-correlation network graph for the CLOUD experimental run 1962.03. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

**B.2.2.2.3 DTW similarity** Figure B.12 shows the similarity network with DTW distance as a similarity measurement, with nodes sized according to their eignevector centrality. Table B.12 shows the ranking of nodes according to their eigenvector centrality, easing the identification of individual nodes in figure B.12.

*Table B.11: Eigenvector ($\epsilon$) centrality node ranking for run 1962.03 using the maximum cross-correlation as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | Light_Spectrometer | LS4 | 0.2428 |
| 2 | PTR3 | APINENE | 0.2386 |
| 3 | PT100 | temp1 | 0.2353 |
| 4 | PTR3 | TOLUENE | 0.2232 |
| 5 | Dew | dew | 0.2231 |
| 6 | Light_Spectrometer | LS3 | 0.2211 |
| 7 | CAPS | NO2 | 0.2161 |
| 8 | O3 | o3 | 0.2148 |
| 9 | Light_Spectrometer | UVH | 0.2105 |
| 10 | CPC3776 | 3776-1DisplayConc. | 0.2037 |
| 11 | PSM12 | conc | 0.2031 |
| 12 | PSM16 | conc | 0.1982 |
| 13 | STOF | APINENE | 0.1833 |
| 14 | PTR3 | ACETICACID | 0.1815 |
| 15 | nanoSMPS | Total Conc. #/cm | 0.1779 |
| 16 | Light_Spectrometer | LS1 | 0.1741 |
| 17 | PTR3 | ACETONE | 0.1733 |
| 18 | PICARRO | NH3_2MIN | 0.1717 |
| 19 | NAIS_ions | Ions_pos | 0.17 |
| 20 | Light_Spectrometer | UVX | 0.1593 |
| 21 | NAIS_ions | Ions_neg | 0.1476 |
| 22 | TDL | sat_water | 0.1474 |
| 23 | Fan | speed1 | 0.1367 |
| 24 | SO2 | so2 | 0.1358 |
| 25 | PTR3 | ISOPRENE | 0.1354 |
| 26 | LongSMPS | Particles 20-500 nm | 0.1323 |
| 27 | PTR3 | NAPHTHALENE | 0.1315 |
| 28 | HVFC | mvolt1 | 0.112 |
| 29 | PTR3 | PINONALDEHYDE | 0.0943 |
| 30 | STOF | ISOPRENE | 0.0819 |
| 31 | PTR3 | BETACARYOPHYLENE | 0.0799 |
| 32 | PTR3 | TMB | 0.0787 |
| 33 | Scalers | brust | 0.0615 |

*Table B.12: Eigenvector ($\epsilon$) centrality node ranking for run 1962.03 using the DTW distance as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | Fan | speed1 | 0.7063 |
| 2 | HVFC | mvolt1 | 0.7061 |
| 3 | PTR3 | APINENE | 0.0214 |
| 4 | PT100 | temp1 | 0.0169 |
| 5 | Light_Spectrometer | UVX | 0.0143 |
| 6 | Light_Spectrometer | LS3 | 0.0138 |
| 7 | LongSMPS | Particles 20-500 nm | 0.0101 |
| 8 | Light_Spectrometer | LS4 | 0.0085 |
| 9 | Dew | dew | 0.0083 |
| 10 | STOF | APINENE | 0.0081 |
| 11 | CAPS | NO2 | 0.008 |
| 12 | Light_Spectrometer | LS1 | 0.008 |
| 13 | Light_Spectrometer | UVH | 0.0079 |
| 14 | O3 | o3 | 0.0079 |
| 15 | PICARRO | NH3_2MIN | 0.0077 |
| 16 | SO2 | so2 | 0.0074 |
| 17 | NAIS_ions | Ions_pos | 0.0074 |
| 18 | PTR3 | TOLUENE | 0.0074 |
| 19 | PTR3 | ISOPRENE | 0.0068 |
| 20 | PTR3 | ACETICACID | 0.0067 |
| 21 | PTR3 | PINONALDEHYDE | 0.0063 |
| 22 | STOF | ISOPRENE | 0.0062 |
| 23 | PTR3 | TMB | 0.0061 |
| 24 | PTR3 | BETACARYOPHYLENE | 0.0059 |
| 25 | NAIS_ions | Ions_neg | 0.0058 |
| 26 | PTR3 | ACETONE | 0.0058 |
| 27 | nanoSMPS | Total Conc. #/cm | 0.0056 |
| 28 | PSM12 | conc | 0.0055 |
| 29 | PTR3 | NAPHTHALENE | 0.0054 |
| 30 | CPC3776 | 3776-1DisplayConc. | 0.0053 |
| 31 | PSM16 | conc | 0.0049 |
| 32 | Scalers | brust | 0.0049 |
| 33 | TDL | sat_water | 0.0043 |

Analysing the figure, it is clear that there is a high relevance of the fan speed and high voltage measurement, while all other measurements appear to be equally as irrelevant. This result is further stressed by analysing the table, which ranks the first two time series extremely above all others, meaning that both these measurements are very important to all other measurements. While this is indeed a true result, there are relationships that are left out and once again it is important to note that the evaluated similarities are extremely small, as evidenced by the reversed grey scale used in figure B.12. Further investigation is required to determine whether this measurement is valid for this particular stage.

**B.2.2.2.4 Coherence similarity** Figure B.12 shows the similarity network graph with coherence as the similarity measurement, with nodes sized according to their eigenvector centrality. Table B.13 ranks nodes according to their eigenvector centrality, thus simplifying the identification of nodes.

Unlike all other network graphs analysed for this stage, the coherence graph is extremely monotone, finding very

# Stage 1962.03

# 3 communities found

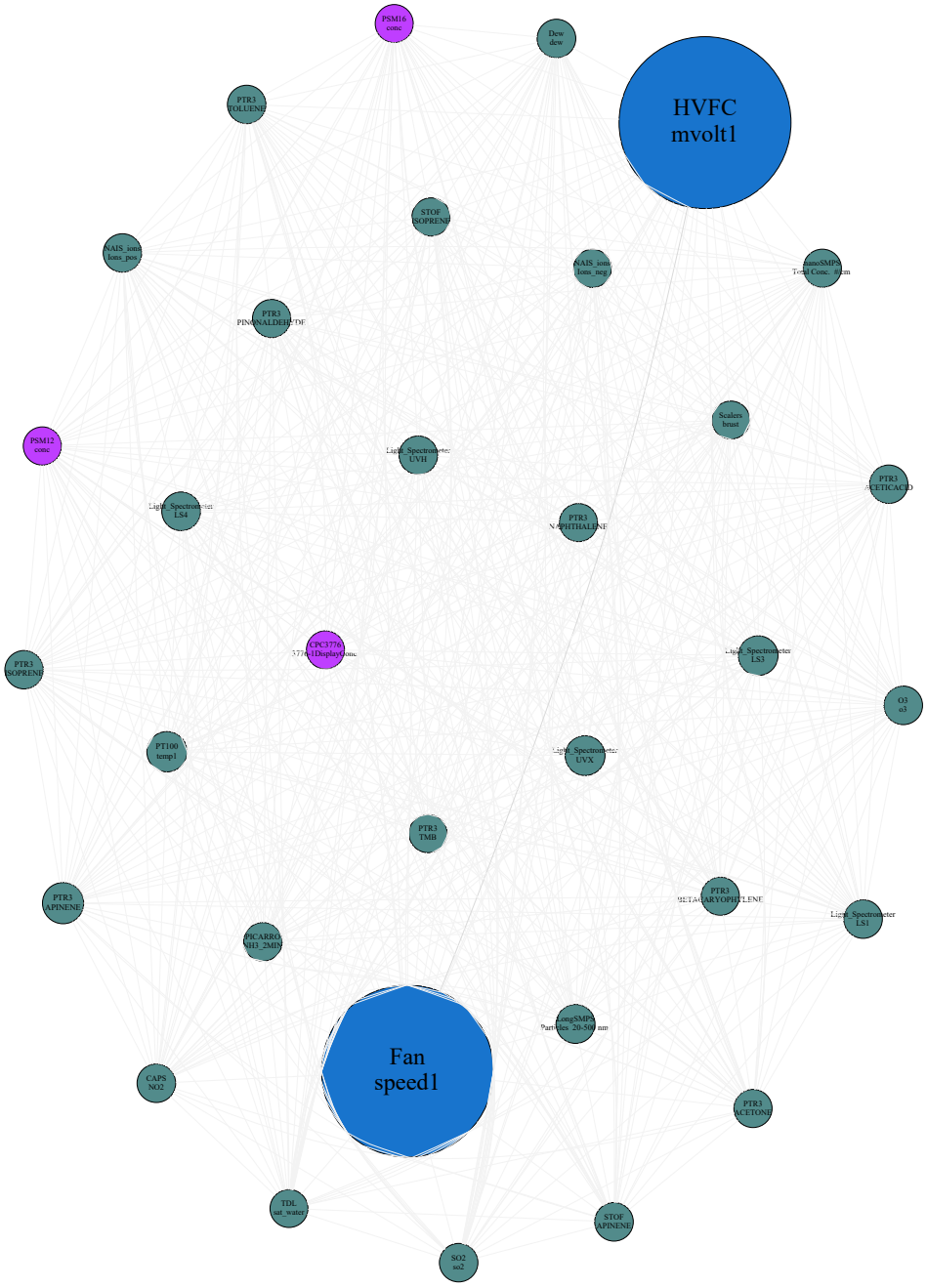



*Figure B.12: DTW similarity network graph for the CLOUD experimental run 1962.03. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

little nodes to highlight in favor of others. Two communities are found, again apparently stratifying the set in terms of the eigenvector centrality, again with little scientific relevance. However, the coherence graph does rank nodes similarly to the rankings proposed by the other similarity measurements, giving slightly more relevance to the light intensity measurements.

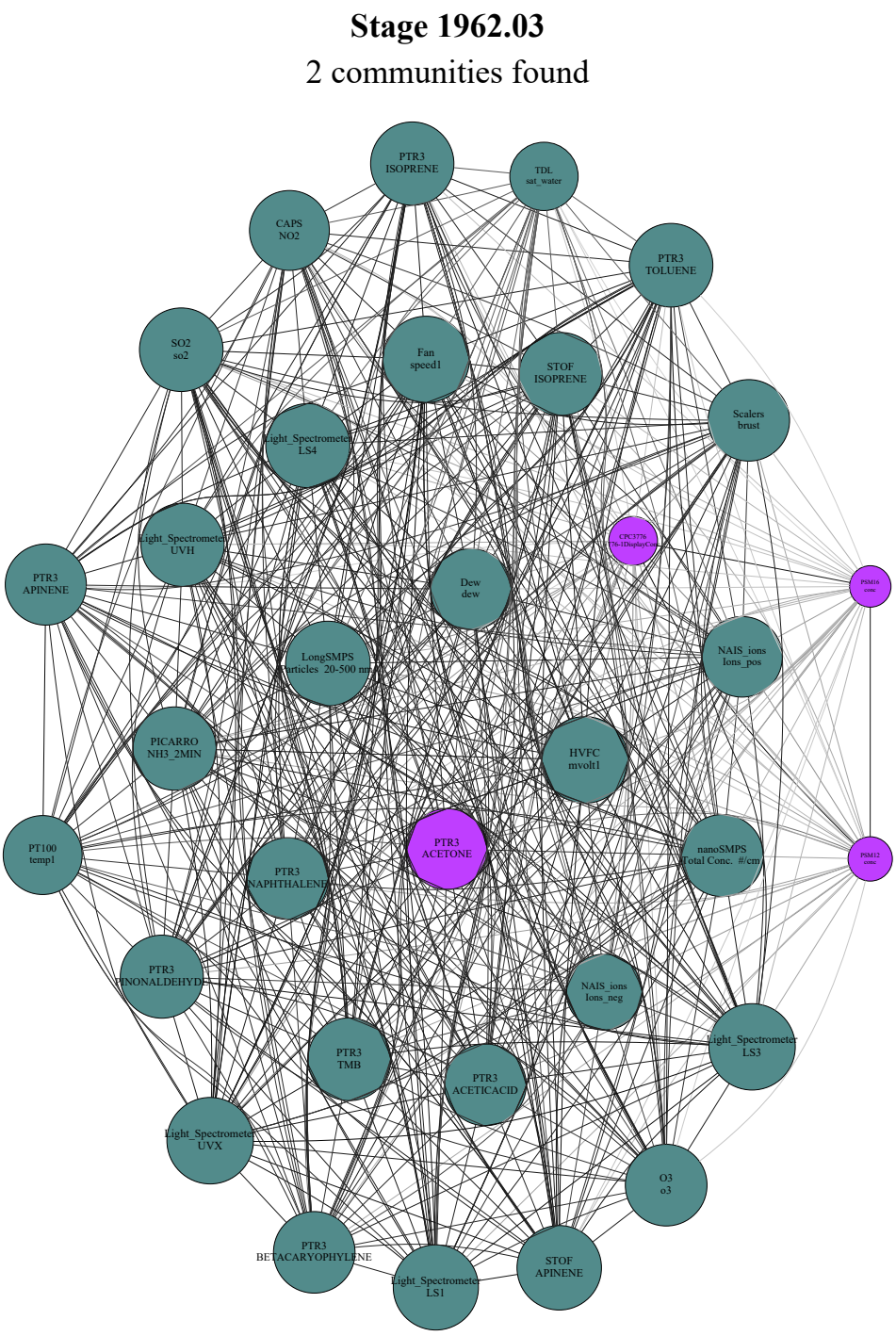



*Figure B.13: Coherence network graph for the CLOUD experimental run 1962.03. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table B.13: Eigenvector ($\epsilon$) centrality node ranking for run 1962.03 using the coherence as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | HVFC | mvolt1 | 0.1917 |
| 2 | Light_Spectrometer | UVX | 0.1914 |
| 3 | Fan | speed1 | 0.1913 |
| 4 | Light_Spectrometer | LS3 | 0.1911 |
| 5 | Light_Spectrometer | LS1 | 0.1879 |
| 6 | LongSMPS | Particles 20-500 nm | 0.1866 |
| 7 | STOF | APINENE | 0.1862 |
| 8 | Light_Spectrometer | LS4 | 0.1851 |
| 9 | PTR3 | TOLUENE | 0.184 |
| 10 | SO2 | so2 | 0.1838 |
| 11 | PICARRO | NH3_2MIN | 0.1831 |
| 12 | PTR3 | ISOPRENE | 0.1831 |
| 13 | Light_Spectrometer | UVH | 0.183 |
| 14 | PTR3 | PINONALDEHYDE | 0.1825 |
| 15 | STOF | ISOPRENE | 0.1824 |
| 16 | PTR3 | TMB | 0.1818 |
| 17 | nanoSMPS | Total Conc. #/cm | 0.1799 |
| 18 | PTR3 | ACETICACID | 0.1791 |
| 19 | O3 | o3 | 0.179 |
| 20 | PTR3 | NAPHTHALENE | 0.1787 |
| 21 | PTR3 | BETACARYOPHYLENE | 0.1786 |
| 22 | Scalers | brust | 0.1784 |
| 23 | PTR3 | APINENE | 0.1776 |
| 24 | Dew | dew | 0.1772 |
| 25 | PTR3 | ACETONE | 0.1765 |
| 26 | NAIS_ions | Ions_pos | 0.1759 |
| 27 | CAPS | NO2 | 0.1742 |
| 28 | PT100 | temp1 | 0.172 |
| 29 | NAIS_ions | Ions_neg | 0.1638 |
| 30 | TDL | sat_water | 0.1443 |
| 31 | CPC3776 | 3776-1DisplayConc. | 0.094 |
| 32 | PSM12 | conc | 0.0823 |
| 33 | PSM16 | conc | 0.0749 |

#### B.2.2.3 Relevant Period Selection

Figure B.14 shows the event detection step of the experimental run summary applied to run 1962.03 using the reduced set of time series since its relevance in this particular stage was stated in B.2.2.1. Table B.14 shows the distribution of found events over each individual channel. It is shown that the most relevant time period is the first and last quarter of the experimental run, with some emphasis being given to the third quarter. As stated previously, this run also had the goal of increasing levels of $\alpha$-pinene concentrations, this particular increase levels out at the third period (which can be seen from the toluene time series). Looking at table B.14, several particle concentration measurements contribute to the total number of events (i.e: *LongSMPS*, *NAIS*). The highlighting of these particular periods is expected for this run since most of the particle changes take place during the first quarter of the run and during the last quarter their value plateaus in preparation for the following run, signaling again a change of state, which should trigger more event detection.

*Table B.14: Events in each period for run 1962.03 using wavelet ME partitioning.*

| Channel | Period 1 | Period 2 | Period 3 | Period 4 | Total |
|---|---|---|---|---|---|
| Light_Spectrometer LS3 | 5 | 2 | 8 | 4 | 19 |
| Scalers brust | 9 | 0 | 0 | 9 | 18 |
| LongSMPS Particles 20-500 nm | 5 | 8 | 3 | 1 | 17 |
| NAIS_ions Ions_pos | 0 | 9 | 0 | 7 | 16 |
| Light_Spectrometer UVX | 8 | 2 | 3 | 3 | 16 |
| HVFC mvolt2 | 9 | 0 | 7 | 0 | 16 |
| Light_Spectrometer LS4 | 4 | 2 | 4 | 4 | 14 |
| NAIS_ions Ions_neg | 0 | 0 | 5 | 8 | 13 |
| nanoSMPS Total Conc. #/cm | 0 | 0 | 4 | 8 | 12 |
| DMAtrain Ch1 C [#/cc] | 3 | 0 | 7 | 2 | 12 |
| Light_Spectrometer UVH | 2 | 3 | 2 | 4 | 11 |
| Light_Spectrometer LS1 | 3 | 1 | 0 | 5 | 9 |
| Fan speed1 | 9 | 0 | 0 | 0 | 9 |
| Fan speed2 | 9 | 0 | 0 | 0 | 9 |
| Dew dew | 0 | 0 | 6 | 0 | 6 |
| DMAtrain Ch0 Ch [#/cc] | 5 | 0 | 0 | 0 | 5 |
| PTR3 APINENE | 1 | 1 | 1 | 1 | 4 |
| PTR3 PINONALDEHYDE | 1 | 1 | 1 | 1 | 4 |
| PTR3 BETACARYOPHYLENE | 1 | 1 | 1 | 1 | 4 |
| PTR3 ACETICACID | 1 | 1 | 1 | 1 | 4 |
| PTR3 ISOPRENE | 1 | 1 | 1 | 1 | 4 |
| PTR3 NAPHTHALENE | 1 | 1 | 1 | 1 | 4 |
| CAPS NO2 | 0 | 0 | 0 | 3 | 3 |
| PTR3 TOLUENE | 1 | 1 | 1 | 0 | 3 |
| PTR3 ACETONE | 1 | 1 | 1 | 0 | 3 |
| PTR3 TMB | 1 | 1 | 1 | 0 | 3 |
| STOF ACETICACID | 0 | 0 | 2 | 0 | 2 |
| STOF NAPHTHALENE | 0 | 0 | 0 | 2 | 2 |
| STOF ISOPRENE | 0 | 0 | 0 | 1 | 1 |
| STOF BETACARYOPHYLENE | 0 | 0 | 0 | 1 | 1 |
| STOF APINENE | 0 | 0 | 0 | 1 | 1 |
| STOF TOLUENE | 0 | 0 | 0 | 1 | 1 |
| TDL sat_water | 0 | 0 | 1 | 0 | 1 |
| Total | 80 | 36 | 61 | 70 | 247 |

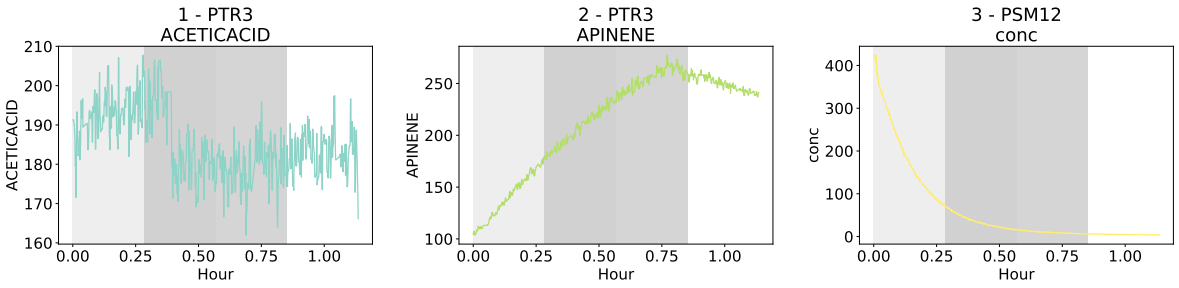



*Figure B.14: Most relevant time series for CLOUD experimental run 1962.03 coupled with ranking of specific time periods to the number of events observed in each event. The time period was divided into 4 distinct sub-periods, which were then tested for the existence of events. The ranking of each sub-period shades each sub-period more the less events are found. In this case, the first and fourth period show most detected events, respectively.*

### B.2.3 Run 1962.04

This run is very similar run to 1962.01 since the goal is to generate neutral particles where the only difference is an increase in the $\alpha$-pinene concentration. It is thus expected that particle generation be highlighted except for charged particle counters.

#### B.2.3.1 Relevant Time Series Highlighting

Figure B.15 shows the most relevant channels according to the PCA highlighting step. As expected, a good part of the highlighted time series are particle generation measurements. Of note the highlighting of the light spectrometer time series, indicating the relevance of light sources towards new particle formation. However, the highlighting of chemical measurements such as the pinonaldehyde and $SO_2$ time series, seem to be measuring at a constant rate and at very small and possibly below detection limit, which is not expected to happen. The reduced set should be considered for this stage.

Figure B.16 shows the result of the highlighting using the reduced set of time series. This result more accurately shows the relevant time series of this run, highlighting the same particle concentration and measurements as in the full set, but highlighting the light spectrometer, $\alpha$-pinene and $O_3$ time series, which are all relevant series for particle generation. Another noticeable fact is that the reduced set produces a reduced amount of time series, meaning that reducing the set removed some of the variance of the full set, thus information was lost with the removal of certain channels. However, it is more likely that the information contained in the removed channels was that of unimportant noise. Since a significant amount of time series was maintained by using the reduced set, this one will be used in further analysis of this stage, though this decision is debatable.

#### B.2.3.2 Time Series Relationships

As discussed in the previous section, the reduced set was (though this decision is debatable), chosen to be the representative set to analyse.

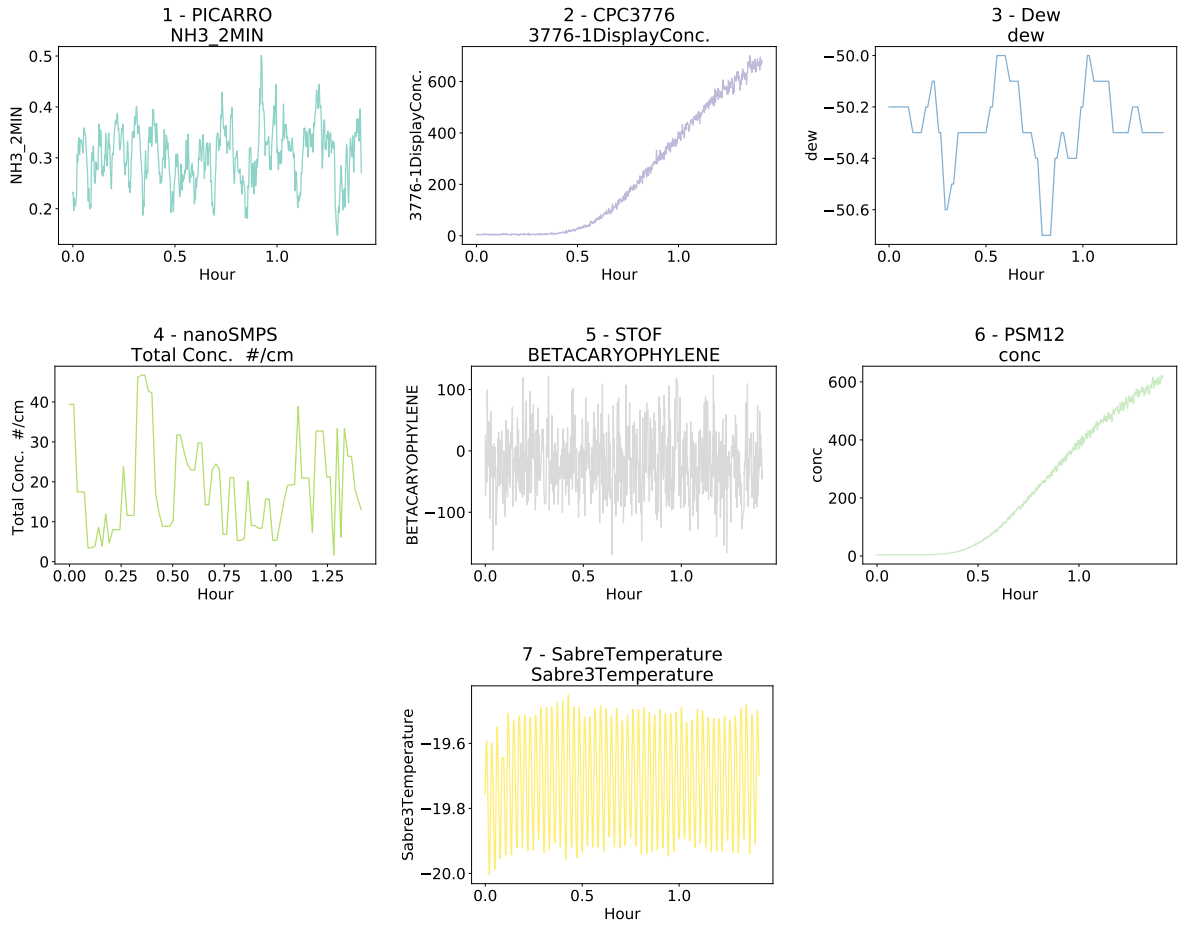



*Figure B.15: Most relevant time series for CLOUD experimental run 1962.04 using the full data set. The compared time series are shown in their original sampled values, before normalization to give the reader an idea of the absolute change of values. This visualization can be shown to users to allow for a more detailed study of the results.*

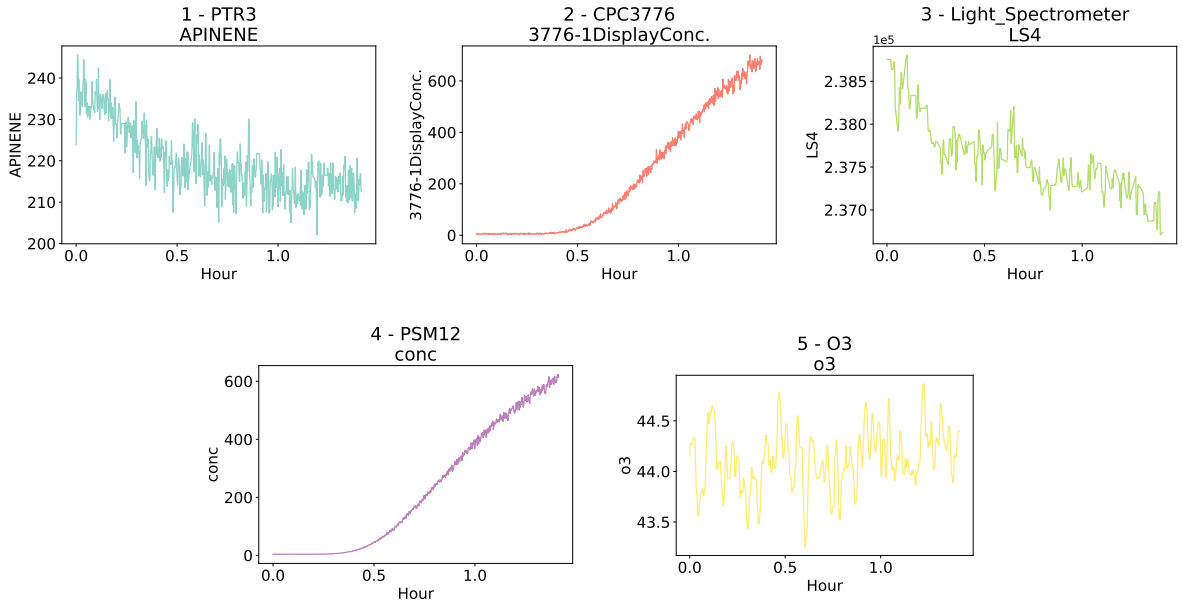



*Figure B.16: Most relevant time series for CLOUD experimental run 1962.04 using the reduced data set. The compared time series are shown in their original sampled values, before normalization to give the reader an idea of the absolute change of values. This visualization can be shown to users to allow for a more detailed study of the results.*

*Table B.15: Eigenvector ($\epsilon$) centrality node ranking for run 1962.04 using the Pearson correlation as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | Light_Spectrometer | LS4 | 0.3743 |
| 2 | PT100 | temp1 | 0.3708 |
| 3 | PTR3 | APINENE | 0.3634 |
| 4 | Light_Spectrometer | UVH | 0.343 |
| 5 | PTR3 | TOLUENE | 0.3118 |
| 6 | NAIS_ions | Ions_neg | 0.2959 |
| 7 | PTR3 | ACETONE | 0.2911 |
| 8 | Light_Spectrometer | LS3 | 0.2705 |
| 9 | Light_Spectrometer | LS1 | 0.2425 |
| 10 | Light_Spectrometer | UVX | 0.1874 |
| 11 | NAIS_ions | Ions_pos | 0.1121 |
| 12 | Fan | speed1 | 0.0763 |
| 13 | PTR3 | ISOPRENE | 0.0747 |
| 14 | STOF | APINENE | 0.0624 |
| 15 | Dew | dew | 0.0456 |
| 16 | SO2 | so2 | 0.045 |
| 17 | CAPS | NO2 | 0.0385 |
| 18 | Scalers | brust | 0.0245 |
| 19 | PTR3 | NAPHTHALENE | 0.0212 |
| 20 | PICARRO | NH3_2MIN | 0.0177 |
| 21 | CPC3776 | 3776-1DisplayConc. | 0.0126 |
| 22 | PSM12 | conc | 0.0124 |
| 23 | PSM16 | conc | 0.0122 |
| 24 | LongSMPS | Particles 20-500 nm | 0.0111 |
| 25 | PTR3 | ACETICACID | 0.0106 |
| 26 | O3 | o3 | 0.0097 |
| 27 | nanoSMPS | Total Conc. #/cm | 0.0069 |
| 28 | PTR3 | PINONALDEHYDE | 0.0067 |
| 29 | PTR3 | TMB | 0.0059 |
| 30 | STOF | ISOPRENE | 0.0056 |
| 31 | TDL | sat_water | 0.0055 |
| 32 | PTR3 | BETACARYOPHYLENE | 0.0051 |
| 33 | HVFC | mvolt1 | 0.005 |

**B.2.3.2.1 Pearson correlation similarity** Figure B.17 shows the similarity network graph using the Pearson correlation coefficient as a similarity measurement. Each node is sized according to their respective eigenvector centrality, thus the larger the centrality, the larger the node. Table B.15 ranks each node according to its eigenvector centrality and thus is useful to identify individual nodes in figure B.17.

Analysing figure B.17 and aided by the coloring provided by the communities found by the Louvain algorithm, it is clear that there are three different sets of time series, loosely related to their rank in table B.15. The first set contains several particle concentration, light intensity measurements, temperature and organic chemical species ($\alpha$-pinene and toluene). While most relationships are in line with the ones found in the sames stage at lower $\alpha$-pinene concentrations (discussed in 5.2.5.2), the highlighting of the time series for organic compounds, temperature and charged particle concentrations is not expected. One explanation for this behavior is due to the fact that the transition between stages is abrupt and could have been set during a time when proper balance was not achieved. This would explain the relevance in temperature change (which fluctuates over 0.1°C during this run, something very unexpected in CLOUD).

**Stage 1962.04**

3 communities found

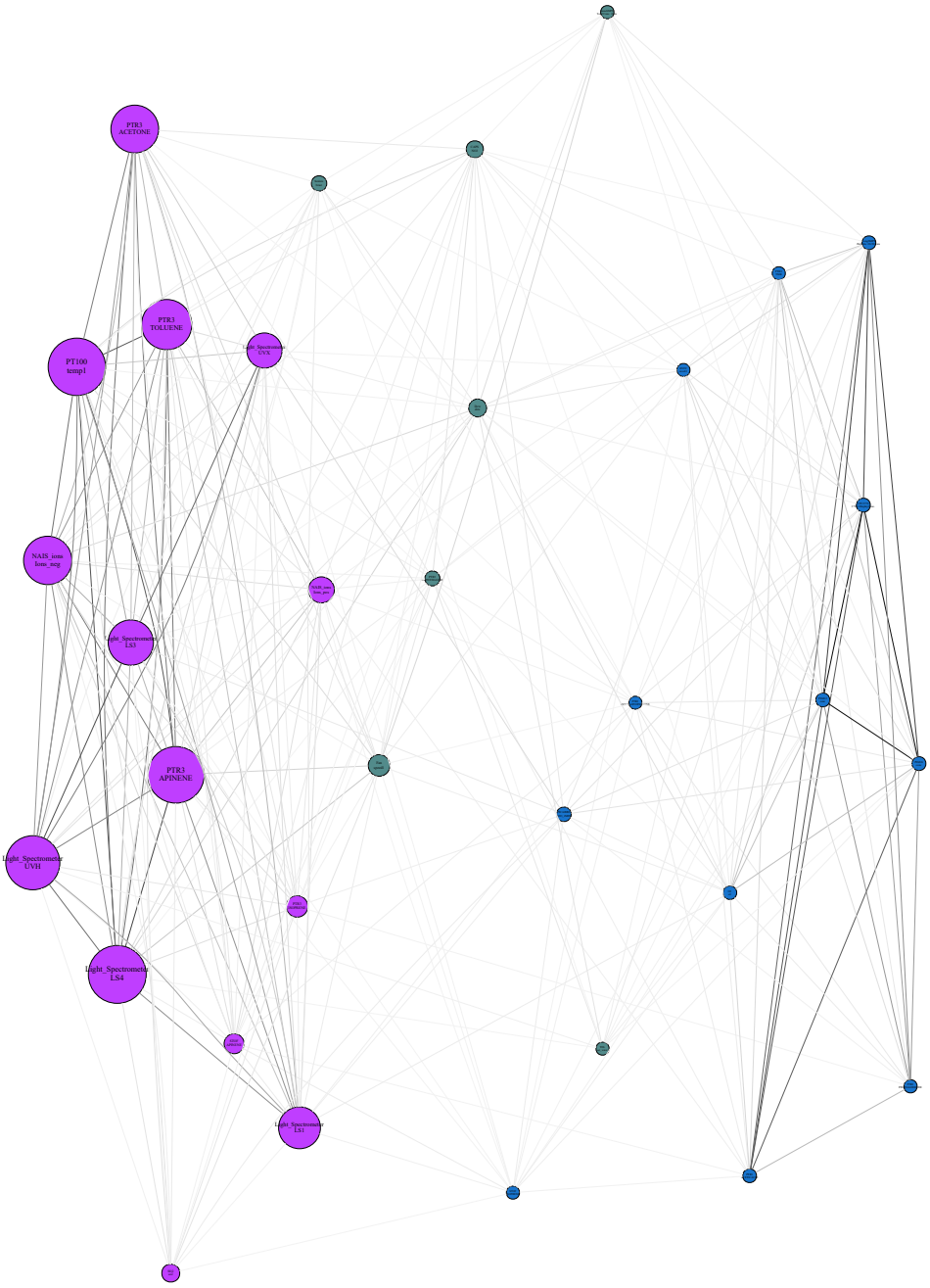



*Figure B.17: Pearson correlation network graph for the CLOUD experimental run 1962.04. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table B.16: Eigenvector ($\epsilon$) centrality node ranking for run 1962.04 using the maximum cross-correlation as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | PT100 | temp1 | 0.2536 |
| 2 | Light_Spectrometer | LS4 | 0.2522 |
| 3 | PTR3 | APINENE | 0.2474 |
| 4 | Light_Spectrometer | UVH | 0.2344 |
| 5 | NAIS_ions | Ions_neg | 0.2309 |
| 6 | PTR3 | TOLUENE | 0.2272 |
| 7 | CPC3776 | 3776-1DisplayConc. | 0.2233 |
| 8 | PSM12 | conc | 0.2229 |
| 9 | PSM16 | conc | 0.2221 |
| 10 | Light_Spectrometer | LS3 | 0.2065 |
| 11 | LongSMPS | Particles 20-500 nm | 0.2041 |
| 12 | PTR3 | ACETICACID | 0.2037 |
| 13 | PTR3 | ACETONE | 0.2036 |
| 14 | Light_Spectrometer | LS1 | 0.1848 |
| 15 | CAPS | NO2 | 0.1813 |
| 16 | NAIS_ions | Ions_pos | 0.1644 |
| 17 | Light_Spectrometer | UVX | 0.1643 |
| 18 | PTR3 | PINONALDEHYDE | 0.1643 |
| 19 | nanoSMPS | Total Conc. #/cm | 0.1606 |
| 20 | O3 | o3 | 0.1536 |
| 21 | Dew | dew | 0.1531 |
| 22 | PICARRO | NH3_2MIN | 0.1265 |
| 23 | Fan | speed1 | 0.1219 |
| 24 | TDL | sat_water | 0.1219 |
| 25 | SO2 | so2 | 0.1089 |
| 26 | PTR3 | TMB | 0.1062 |
| 27 | HVFC | mvolt1 | 0.0966 |
| 28 | STOF | APINENE | 0.0935 |
| 29 | PTR3 | ISOPRENE | 0.0781 |
| 30 | PTR3 | NAPHTHALENE | 0.0769 |
| 31 | PTR3 | BETACARYOPHYLENE | 0.0725 |
| 32 | Scalers | brust | 0.07 |
| 33 | STOF | ISOPRENE | 0.0656 |

**B.2.3.2.2 Maximum cross-correlation similarity** Figure B.18 shows the similarity network graph with the maximum cross-correlation between time series as a similarity measurement. Once again, table B.16 ranks each node according to their eigenvector centrality, which aids in identifying individual nodes in figure B.18, which sizes each node also according to their eigenvector centrality.

Analysing table B.16 shows very similar results to the previous analysis, with higher importance provided to charged particle concentration and temperature measurements and keeping organic compound signals on a prominent rank. On the positive side, several other particle concentration measurements rise in rank. Once again three communities are found, again dividing the set of time series according to their eigenvector centrality. The relative similarities between this method and the previous one show that even when considering delays between signals, there is a statistically significant behavior in the temperature and charged species time series and their behavior should be studied.

**B.2.3.2.3 DTW similarity** Figure B.19 shows the similarity network graph using DTW distance as a similarity measurement. As with all other network graphs, nodes are sized according to their eigenvector centrality and table

# Stage 1962.04

3 communities found

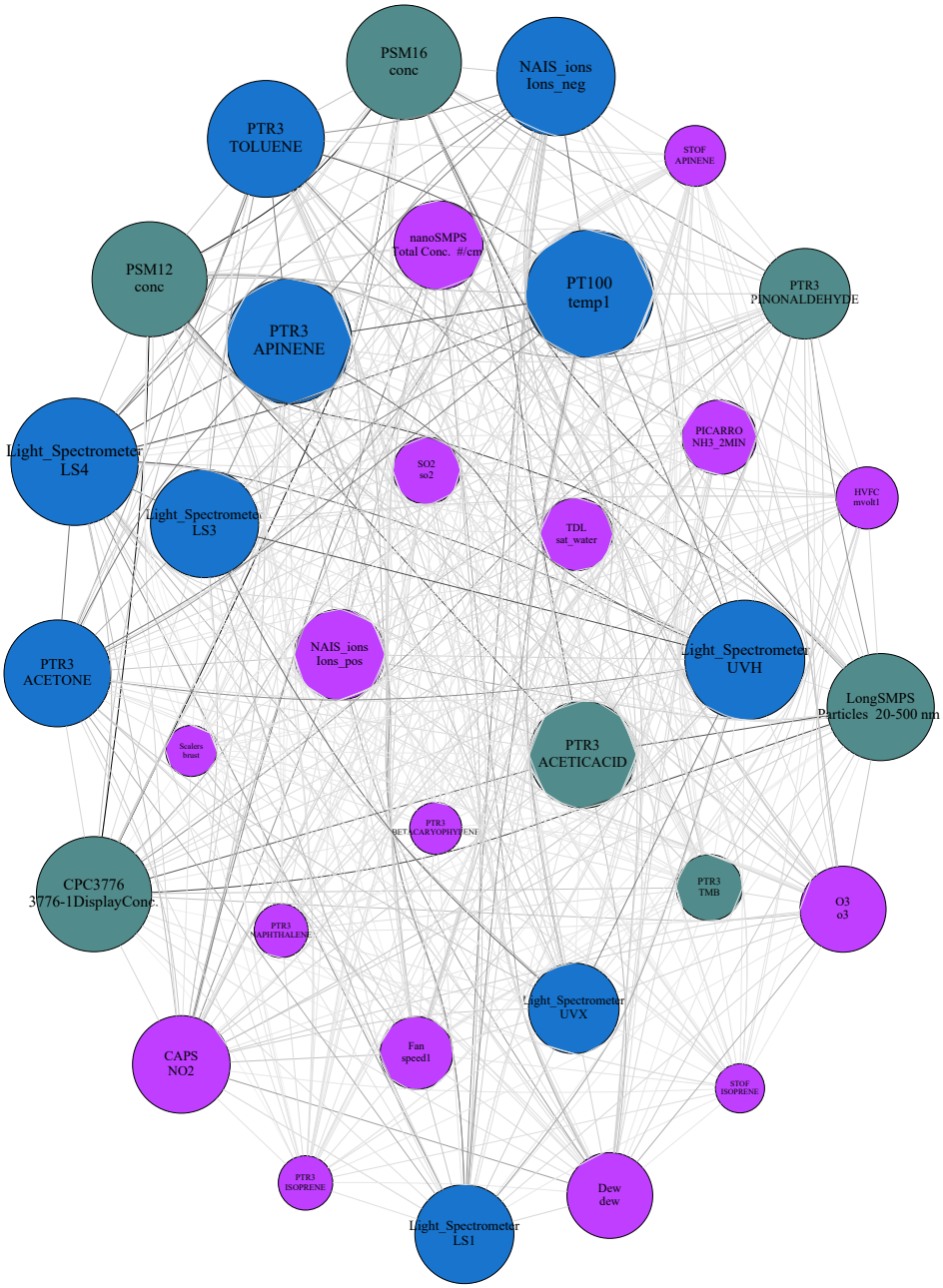



*Figure B.18: Maximum cross-correlation network graph for the CLOUD experimental run 1962.04. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table B.17: Eigenvector ($\epsilon$) centrality node ranking for run 1962.04 using the DTW distance as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | CPC3776 | 3776-1DisplayConc. | 0.4837 |
| 2 | PSM12 | conc | 0.482 |
| 3 | LongSMPS | Particles 20-500 nm | 0.2694 |
| 4 | PSM16 | conc | 0.2299 |
| 5 | PTR3 | PINONALDEHYDE | 0.1486 |
| 6 | PICARRO | NH3_2MIN | 0.1441 |
| 7 | PTR3 | ACETICACID | 0.1427 |
| 8 | Light_Spectrometer | LS1 | 0.1362 |
| 9 | O3 | o3 | 0.1351 |
| 10 | PTR3 | ACETONE | 0.1338 |
| 11 | STOF | ISOPRENE | 0.1336 |
| 12 | PTR3 | BETACARYOPHYLENE | 0.1328 |
| 13 | PTR3 | ISOPRENE | 0.1312 |
| 14 | SO2 | so2 | 0.1274 |
| 15 | PTR3 | TMB | 0.1264 |
| 16 | nanoSMPS | Total Conc. #/cm | 0.1262 |
| 17 | PTR3 | NAPHTHALENE | 0.1259 |
| 18 | STOF | APINENE | 0.1244 |
| 19 | PTR3 | APINENE | 0.1224 |
| 20 | CAPS | NO2 | 0.1204 |
| 21 | Light_Spectrometer | LS4 | 0.1189 |
| 22 | PTR3 | TOLUENE | 0.1188 |
| 23 | NAIS_ions | Ions_neg | 0.1165 |
| 24 | NAIS_ions | Ions_pos | 0.1155 |
| 25 | Dew | dew | 0.1052 |
| 26 | Scalers | brust | 0.1026 |
| 27 | PT100 | temp1 | 0.1025 |
| 28 | TDL | sat_water | 0.1018 |
| 29 | Light_Spectrometer | UVX | 0.0968 |
| 30 | Light_Spectrometer | UVH | 0.0912 |
| 31 | HVFC | mvolt1 | 0.0708 |
| 32 | Light_Spectrometer | LS3 | 0.0703 |
| 33 | Fan | speed1 | 0.0546 |

B.17 ranks each node according to its eigenvector centrality, thus simplifying the identification of each node.

Analysing the figure and table shows a departure from the two previous methods, relegating the temperature and charged particle concentration measurements to the bottom half of table B.17 and highlighting several other particle concentration measurements, such that the Louvain algorithm also highlights the 4 most highly ranked time series as a single community. This result in more in line with the scientific relevance of the run, however, some importance is given to clearly noisy time series such as the pinonaldehyde signal, which casts doubt as to whether this similarity measurement is proper to apply to CLOUD measurements. Another argument against DTW, which is also shared with all other DTW network graphs, is the consistently weak relationships found between nodes when using this similarity.

**B.2.3.2.4 Coherence similarity** Figure B.20 shows the similarity network graph using coherence as the similarity measurement. Once again, table B.23 helps identifying nodes in the network graph by relating nodes sizes to their eigenvector centrality.

# Stage 1962.04

3 communities found

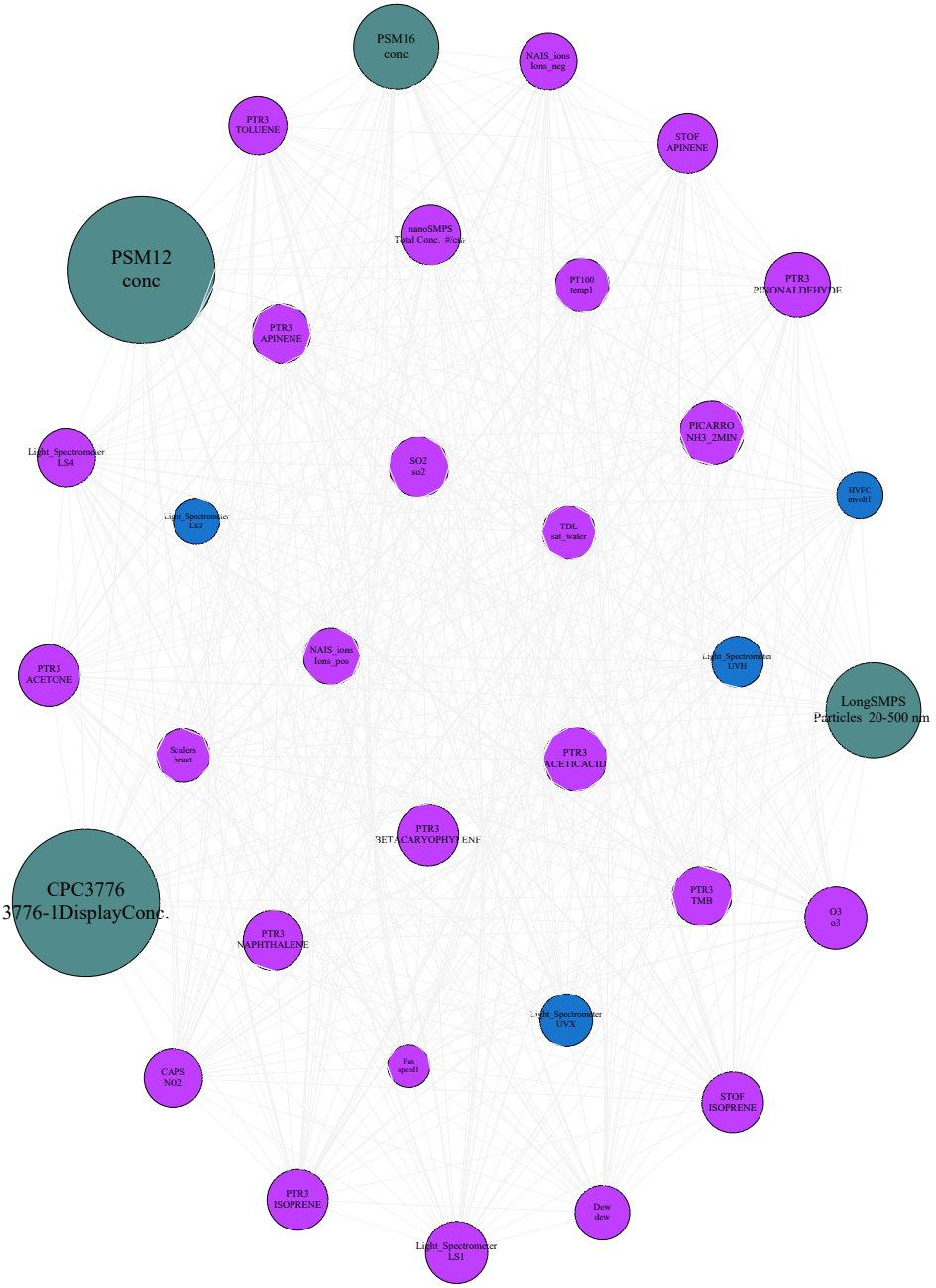



*Figure B.19: DTW similarity network graph for the CLOUD experimental run 1962.04. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

Analysing figure B.20 one can see another monotone graph with very hardly any node visually highlighted, much like the same graph of stage 1962.01 (figure 5.37). Table B.23 shows that the light intensity and organic compound measurements are highlighted over others but by a small margin. This is not in line with the results produced by other similarity measurements and casts doubt once again in the validity of using coherence as a similarity measurement for CLOUD measurements.

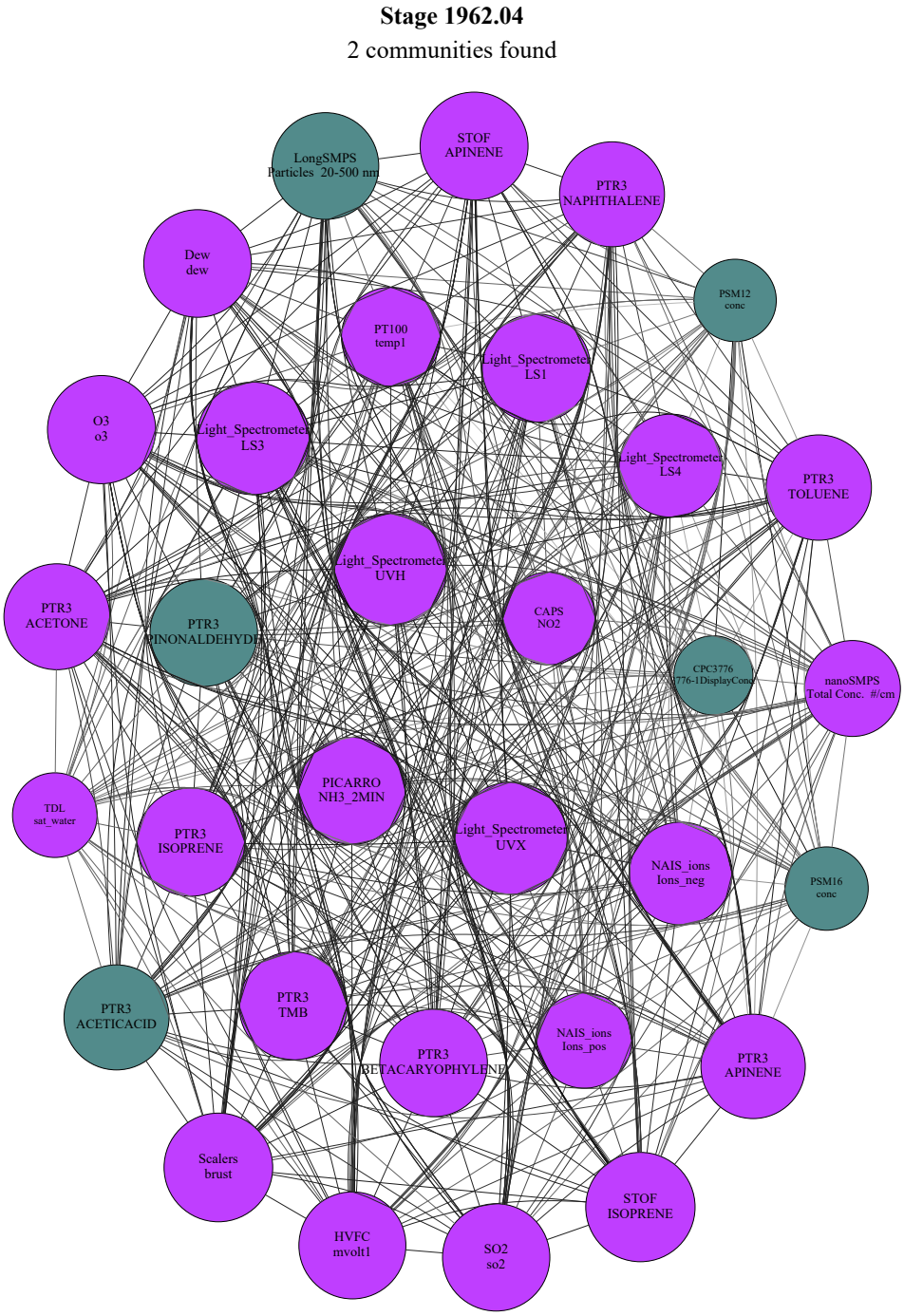



*Figure B.20: Coherence network graph for the CLOUD experimental run 1962.04. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table B.18: Eigenvector ($\epsilon$) centrality node ranking for run 1962.04 using the coherence as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | Light_Spectrometer | LS3 | 0.1942 |
| 2 | Light_Spectrometer | UVX | 0.1937 |
| 3 | Light_Spectrometer | UVH | 0.193 |
| 4 | STOF | ISOPRENE | 0.1881 |
| 5 | Scalers | brust | 0.1869 |
| 6 | PTR3 | ISOPRENE | 0.1863 |
| 7 | Light_Spectrometer | LS1 | 0.1862 |
| 8 | O3 | o3 | 0.1859 |
| 9 | PTR3 | TMB | 0.1859 |
| 10 | STOF | APINENE | 0.1855 |
| 11 | PTR3 | BETACARYOPHYLENE | 0.185 |
| 12 | Dew | dew | 0.1849 |
| 13 | PTR3 | PINONALDEHYDE | 0.1847 |
| 14 | SO2 | so2 | 0.1844 |
| 15 | PICARRO | NH3_2MIN | 0.1839 |
| 16 | HVFC | mvolt1 | 0.1837 |
| 17 | LongSMPS | Particles 20-500 nm | 0.1833 |
| 18 | PTR3 | ACETONE | 0.1817 |
| 19 | PTR3 | TOLUENE | 0.1807 |
| 20 | PTR3 | ACETICACID | 0.1801 |
| 21 | PTR3 | NAPHTHALENE | 0.1799 |
| 22 | PTR3 | APINENE | 0.1789 |
| 23 | Light_Spectrometer | LS4 | 0.1763 |
| 24 | NAIS_ions | Ions_neg | 0.1758 |
| 25 | PT100 | temp1 | 0.1686 |
| 26 | nanoSMPS | Total Conc. #/cm | 0.1629 |
| 27 | NAIS_ions | Ions_pos | 0.1619 |
| 28 | CAPS | NO2 | 0.1571 |
| 29 | TDL | sat_water | 0.1418 |
| 30 | PSM16 | conc | 0.1393 |
| 31 | PSM12 | conc | 0.1375 |
| 32 | CPC3776 | 3776-1DisplayConc. | 0.1315 |

#### B.2.3.3 Relevant Period Selection

Figure B.21 shows the shaded version of figure B.16, since the reduced set was deemed more relevant than the full set, taking into account detected events while table **??** shows the distribution of detected events during run 1962.04. For this particular run, the most relevant time period is the first quarter, while all other three periods are ranked in a very similar fashion. This is not the expected behavior, since the particle generation is mostly detected from the second period onward. One possible explanation is the influence of the cleaning run 1962.03 in the particle generation run 1962.04 that might have created the conditions for several events to be focused on the beginning of the run due to the change in operational mode. This could be resolved by ignoring an initial fraction of each run period before applying the run summary algorithm. Another more likely explanation is the existence non-filtered noise coming from several time series during this run. Looking at table B.19 it can be seen that most particle counter time series correctly detect events from the second period onward, but the number of events detected in said periods are small compared to the events (possibly spurious) detected from all other time series during the first period. Regarding the highlighted time series, 4 time series contribute a total of 57 events (around 50% of total events).

*Table B.19: Events in each period for run 1962.04 using wavelet ME partitioning.*

| Channel | Period 1 | Period 2 | Period 3 | Period 4 | Total |
|---|---|---|---|---|---|
| HVFC mvolt1 | 4 | 9 | 6 | 0 | 19 |
| nanoSMPS Total Conc. #/cm | 3 | 1 | 7 | 3 | 14 |
| LongSMPS Particles 20-500 nm | 8 | 5 | 0 | 0 | 13 |
| TDL sat_water | 0 | 3 | 2 | 6 | 11 |
| Light_Spectrometer LS3 | 4 | 2 | 0 | 3 | 9 |
| Fan speed1 | 6 | 0 | 2 | 0 | 8 |
| Light_Spectrometer UVX | 5 | 0 | 1 | 1 | 7 |
| Light_Spectrometer UVH | 4 | 0 | 1 | 1 | 6 |
| NAIS_ions Ions_pos | 0 | 0 | 0 | 6 | 6 |
| NAIS_ions Ions_neg | 0 | 0 | 4 | 2 | 6 |
| Light_Spectrometer LS4 | 2 | 1 | 0 | 0 | 3 |
| PSM12 conc | 0 | 3 | 0 | 0 | 3 |
| Light_Spectrometer LS1 | 0 | 1 | 0 | 2 | 3 |
| PTR3 APINENE | 1 | 0 | 1 | 0 | 2 |
| PTR3 ACETONE | 1 | 0 | 1 | 0 | 2 |
| PTR3 TMB | 1 | 0 | 1 | 0 | 2 |
| CAPS NO2 | 0 | 0 | 0 | 2 | 2 |
| PTR3 BETACARYOPHYLENE | 1 | 0 | 1 | 0 | 2 |
| PTR3 ACETICACID | 1 | 0 | 0 | 0 | 1 |
| PTR3 TOLUENE | 1 | 0 | 0 | 0 | 1 |
| PTR3 PINONALDEHYDE | 1 | 0 | 0 | 0 | 1 |
| PTR3 ISOPRENE | 1 | 0 | 0 | 0 | 1 |
| PTR3 NAPHTHALENE | 1 | 0 | 0 | 0 | 1 |
| Total | 45 | 25 | 27 | 26 | 123 |

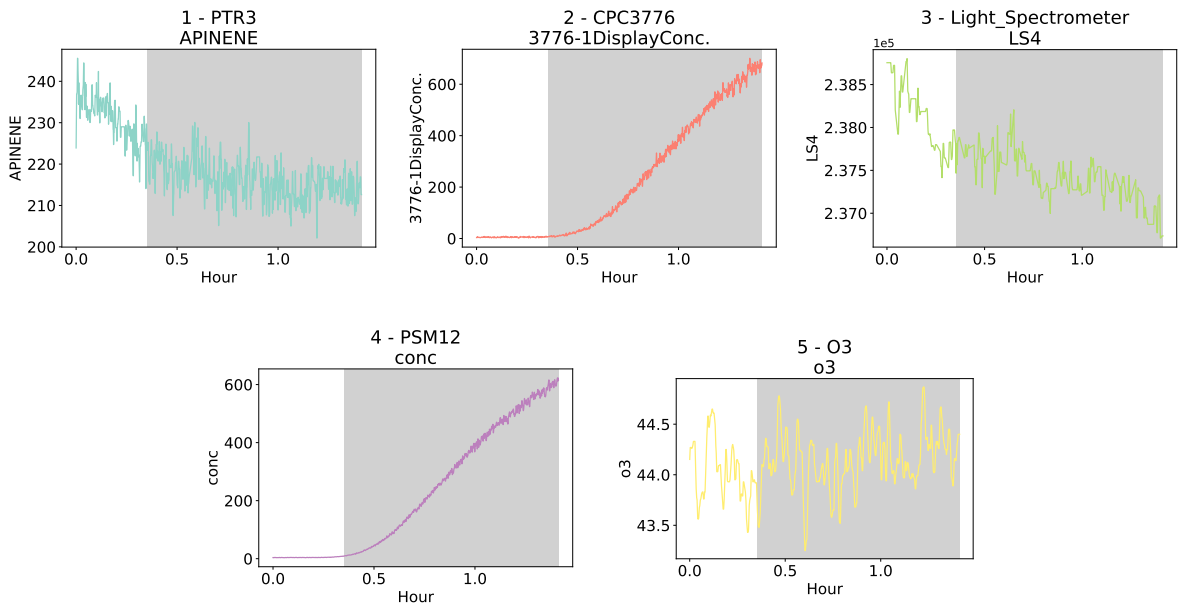



*Figure B.21: Most relevant time series for CLOUD experimental run 1962.04 coupled with ranking of specific time periods to the number of events observed in each event. In this case, most events were found in the first sub-period.*

### B.2.4 Run 1962.05

This last run is most similar in scope to run 1962.02, using increased $\alpha$-pinene concentrations. It is expected that charged particles be a highlight of this run along with other particle counter time series.

#### B.2.4.1 Relevant Time Series Highlighting

Figure B.22 shows the most relevant time series for run 1962.05. Contrary to other experiments this run presents the least amount of highlighted channels and does not show the expected charged particle counters. Again, particle concentration time series are highlighted, bringing emphasis to the particle generation during this run. Also, two light spectrometer time series are also highlighted which emphasises the importance of light during this run. The Dew Point time series is also highlighted, which could still be of spurious as explained in the analysis of run 1962.03. Relevance is also given to the $NO_2$ and $O_3$ while no particular explanation exists for their highlighting.

Figure B.23 show the result of the highlighting using the reduced time series set. Both sets highlight the same number of time series, meaning that little information was lost when switching to a reduced set. Both sets highlight the ion particle concentration measurement as well as other particle concentration measurements, however, using the full set highlights the measurement of acetic acid, while the reduced set highlights the light spectrometer measurements. Since either seem to measure at the same degree of variance it is unclear whether the reduced set improves the scientific analysis of the data set, thus the full set will be maintained for this stage.

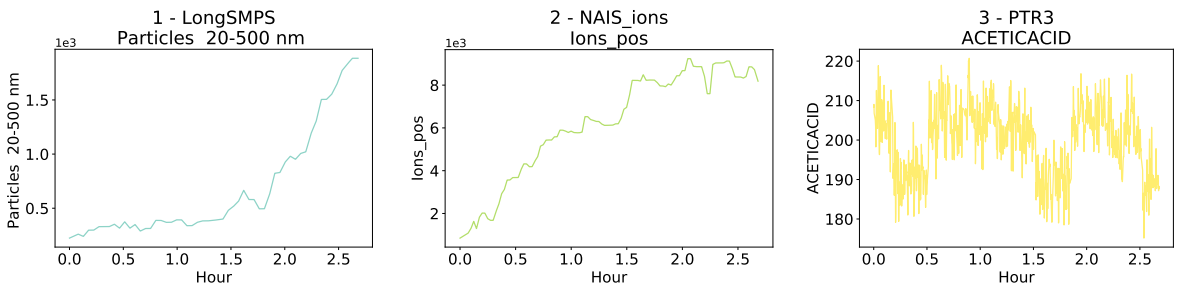



*Figure B.22: Most relevant time series for CLOUD experimental run 1962.05. The compared time series are shown in their original sampled values, before normalization to give the reader an idea of the absolute change of values. This visualization can be shown to users to allow for a more detailed study of the results.*

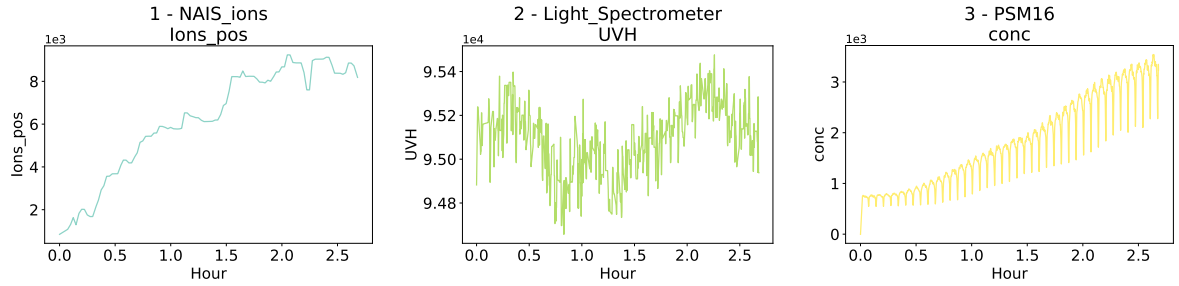



*Figure B.23: Most relevant time series for CLOUD experimental run 1962.05 using the reduced time series set. The compared time series are shown in their original sampled values, before normalization to give the reader an idea of the absolute change of values. This visualization can be shown to users to allow for a more detailed study of the results.*

### B.2.4.2 Time Series Relationships

Since in the previous section it was determined that the full time series set is the one that will more reliably contain the most relevant information to analyse, that will be the data set used to analyse the relationships between time series for this stage

**B.2.4.2.1 Pearson correlation similarity** Figure B.24 shows the similarity network graph using the Pearson correlation coefficient as the similarity measurement. Each node in the figure is sized according to the its eigenvector centrality and table B.20 ranks each node according to its eigenvector centrality and thus can be used to identify individual nodes in figure B.24.

Analysing figure B.24 shows a clear separation between two sets of nodes on the left and right side of the graph. The coloring is a result of the Louvain algorithm separating each node into their community, four of which were found for this graph. While each community stratifies the data set further by grouping loosely based on their eigenvector centrality, very little scientific explanation is available to explain the reason for each community. Analysing table B.20 it is clear the main focus is on highlighting particle concentration measurements, such that the first third of the table is comprises almost entirely of particle concentration measurements. This is a similar result as the one obtained in the analogous charged particle stage 1962.02 (discussed in B.2.1.2) except that for that stage there was a larger relevance given to the light intensity measurements.

**Stage 1962.05**

4 communities found

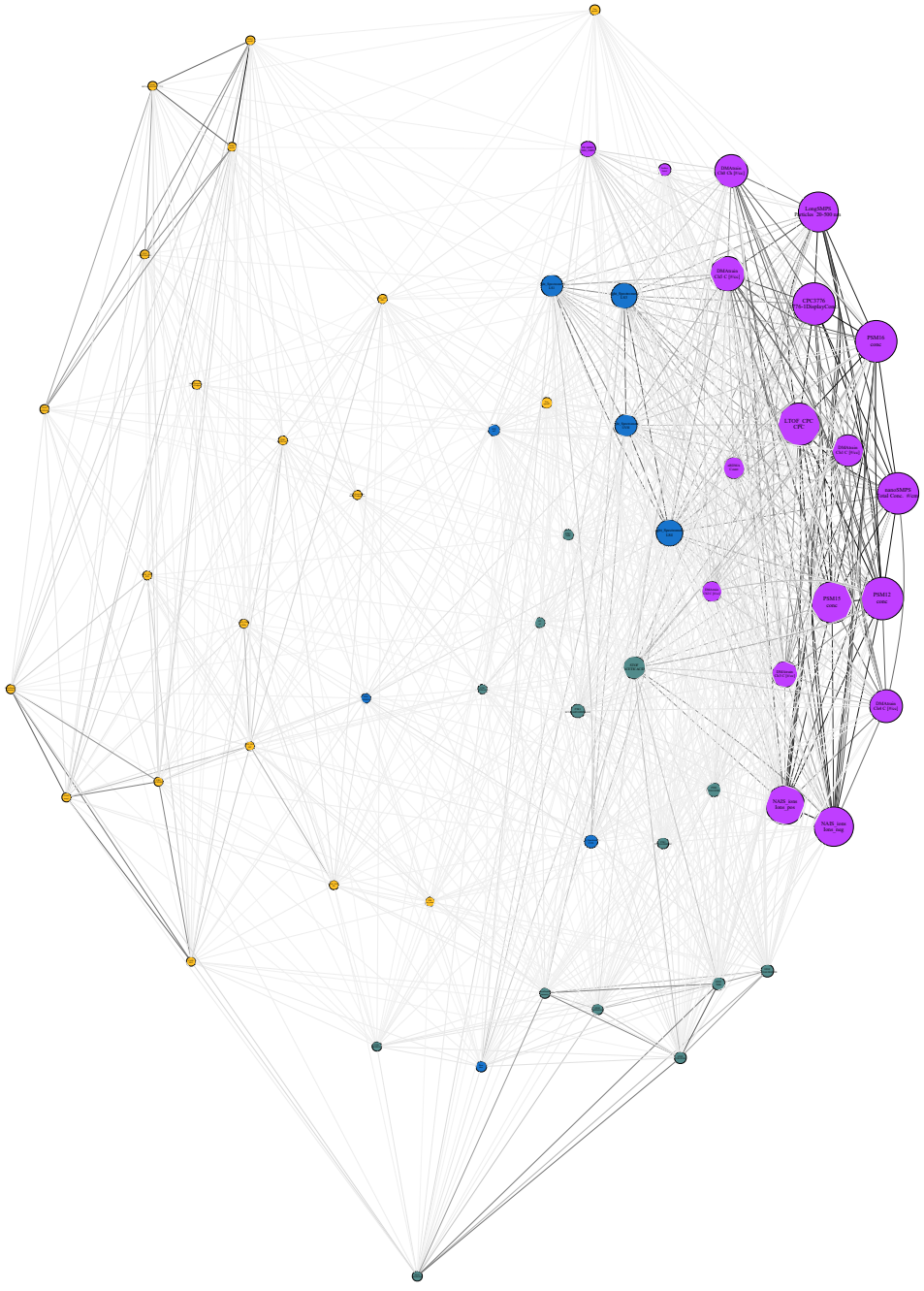


*Figure B.24: Pearson correlation network graph for the CLOUD experimental run 1962.05. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table B.20: Eigenvector ($\epsilon$) centrality node ranking for run 1962.05 using the Pearson correlation as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | LTOF_CPC | CPC | 0.2887 |
| 2 | PSM12 | conc | 0.2887 |
| 3 | CPC3776 | 3776-1DisplayConc. | 0.2885 |
| 4 | PSM16 | conc | 0.2816 |
| 5 | nanoSMPS | Total Conc. #/cm | 0.2789 |
| 6 | PSM15 | conc | 0.2776 |
| 7 | LongSMPS | Particles 20-500 nm | 0.2658 |
| 8 | NAIS_ions | Ions_neg | 0.262 |
| 9 | NAIS_ions | Ions_pos | 0.2573 |
| 10 | DMAtrain | Ch5 C [#/cc] | 0.2211 |
| 11 | DMAtrain | Ch0 Ch [#/cc] | 0.2066 |
| 12 | DMAtrain | Ch4 C [#/cc] | 0.2049 |
| 13 | DMAtrain | Ch1 C [#/cc] | 0.1939 |
| 14 | Light_Spectrometer | LS4 | 0.1505 |
| 15 | DMAtrain | Ch3 C [#/cc] | 0.1466 |
| 16 | Light_Spectrometer | LS3 | 0.1444 |
| 17 | STOF | ACETICACID | 0.1164 |
| 18 | Light_Spectrometer | UVH | 0.1138 |
| 19 | Light_Spectrometer | LS1 | 0.1102 |
| 20 | nRDMA | Count | 0.1094 |
| 21 | DMAtrain | Ch2 C [#/cc] | 0.0975 |
| 22 | PICARRO | NH3_2MIN | 0.0568 |
| 23 | PTR3 | TOLUENE | 0.0427 |
| 24 | PTR3 | BETACARYOPHYLENE | 0.0409 |
| 25 | Light_Spectrometer | UVX | 0.036 |
| 26 | STOF | TMB | 0.0327 |
| 27 | STOF | PINONALDEHYDE | 0.0289 |
| 28 | Scalers | brust | 0.0281 |
| 29 | STOF | TOLUENE | 0.0235 |
| 30 | SO2 | so2 | 0.0216 |
| 31 | PTR3 | PINONALDEHYDE | 0.0188 |
| 32 | PTR3 | TMB | 0.0178 |
| 33 | Fan | speed2 | 0.0166 |
| 34 | STOF | APINENE | 0.0162 |
| 35 | Dew | dew | 0.0136 |
| 36 | PhotoDiode12 | intensity | 0.0128 |
| 37 | O3 | o3 | 0.0127 |
| 38 | Fan | speed1 | 0.01 |
| 39 | STOF | ISOPRENE | 0.0069 |
| 40 | PTR3 | APINENE | 0.0061 |
| 41 | PTR3 | NAPHTHALENE | 0.0046 |
| 42 | TDL | sat_water | 0.0044 |
| 43 | LTOF_UFRA | HIO | 0.0037 |
| 44 | PTR3 | ISOPRENE | 0.0032 |
| 45 | SabreTemperature | Sabre3Temperature | 0.0018 |
| 46 | LTOF_UFRA | SA | 0.0016 |
| 47 | LTOF_UFRA | NIT | 0.0015 |
| 48 | SabreTemperature | Sabre4Temperature | 0.0013 |
| 49 | PTR3 | ACETICACID | 0.0009 |
| 50 | HVFC | mvolt2 | 0.0009 |
| 51 | HVFC | mvolt1 | 0.0008 |
| 52 | STOF | BETACARYOPHYLENE | 0.0006 |
| 53 | LTOF_UFRA | DMA | 0.0006 |
| 54 | CAPS | NO2 | 0.0005 |
| 55 | STOF | NAPHTHALENE | 0.0005 |
| 56 | PTR3 | ACETONE | 0.0005 |
| 57 | STOF | ACETONE | 0.0004 |
| 58 | PT100 | temp1 | 0.0003 |
| 59 | PhotoDiode12 | intensity2 | 0.0003 |
| 60 | HTOF_UFRA | H30 | 0.0002 |

**B.2.4.2.2 Maximum cross-correlation similarity** Figure B.25 shows the similarity network graph using the maximum cross-correlation between time series as the similarity measurement. Once again, table B.21 ranks nodes according to their eigenvector centrality and helps identifying individual nodes in figure B.25.

Analysing the figure, one is drawn to a set of highlighted nodes containing mostly light intensity and particle concentration measurements. This set is also further highlighted by the coloring provided by community detection via the Louvain algorithm. For this graph the communities seem to separate the data set according to their proximity in eigenvector centrality value but no further scientific meaning can be found to explain the reason for each run. Looking at table B.21 one can notice the same trend as in the previous analysis a lot more emphasis is given to particle concentration measurements and less emphasis given to light intensity measurements when compared to the previous charged particle generation run. This could be explained by the fact that at higher $\alpha$-pinene concentrations, it is expected that the rate of particle generations increases and thus a sharper increase in particle concentrations is noticed and thus a larger relevance be given in this run to particle concentration measurements. The similarities between using maximum cross-correlation and the Pearson correlation mean that no significant delays are found between the analysed time series.

**B.2.4.2.3 DTW similarity** Figure B.26 shows the similarity network graph using DTW distance as a similarity measurement. As always, table B.22 aids in identifying individual nodes in figure B.26 by ranking each node by their eigenvector centrality (each node in the figure is also sized according to their eigenvector centrality).

Analysing figure B.26 shows clearly spurious relationships between both electrical field voltage time series, which are turned off at the same time and thus is expected to be highlighted. However, this relationship is so high that it saturates other possibly relevant relationships. The Louvain algorithm highlights the existence of four communities, once again, relating nodes to their eigenvector centrality values. It is once again clear that the DTW distance measurement is very sensitive to same instrument spurious relationships and an analysis with this similarity measurement would be improved using a reduced set of time series.

**B.2.4.2.4 Coherence similarity** Figure B.27 shows the similarity network graph with coherence as the similarity measurement. Once again, each node is sized according to its eigenvector centrality and table B.23 aids in identifying each individual node by ranking each node according to the same centrality.

Analysing figure B.27 the reader is once again presented with a mostly monotone graph with the exception of a completely isolated and obviously low ranked group containing electrical field voltage measurements and one light intensity measurement. There is no immediate reason for the relationship between voltage and light intensity, since they are completely separated systems, thus it is assumed that this is a spurious relationship. Table B.23 shows that there is not particularly stand out time series, which also points that most relationships found are spurious. Once again coherence results further support the argument that this similarity measurement has little use for this set of CLOUD measurements.

## Stage 1962.05

3 communities found

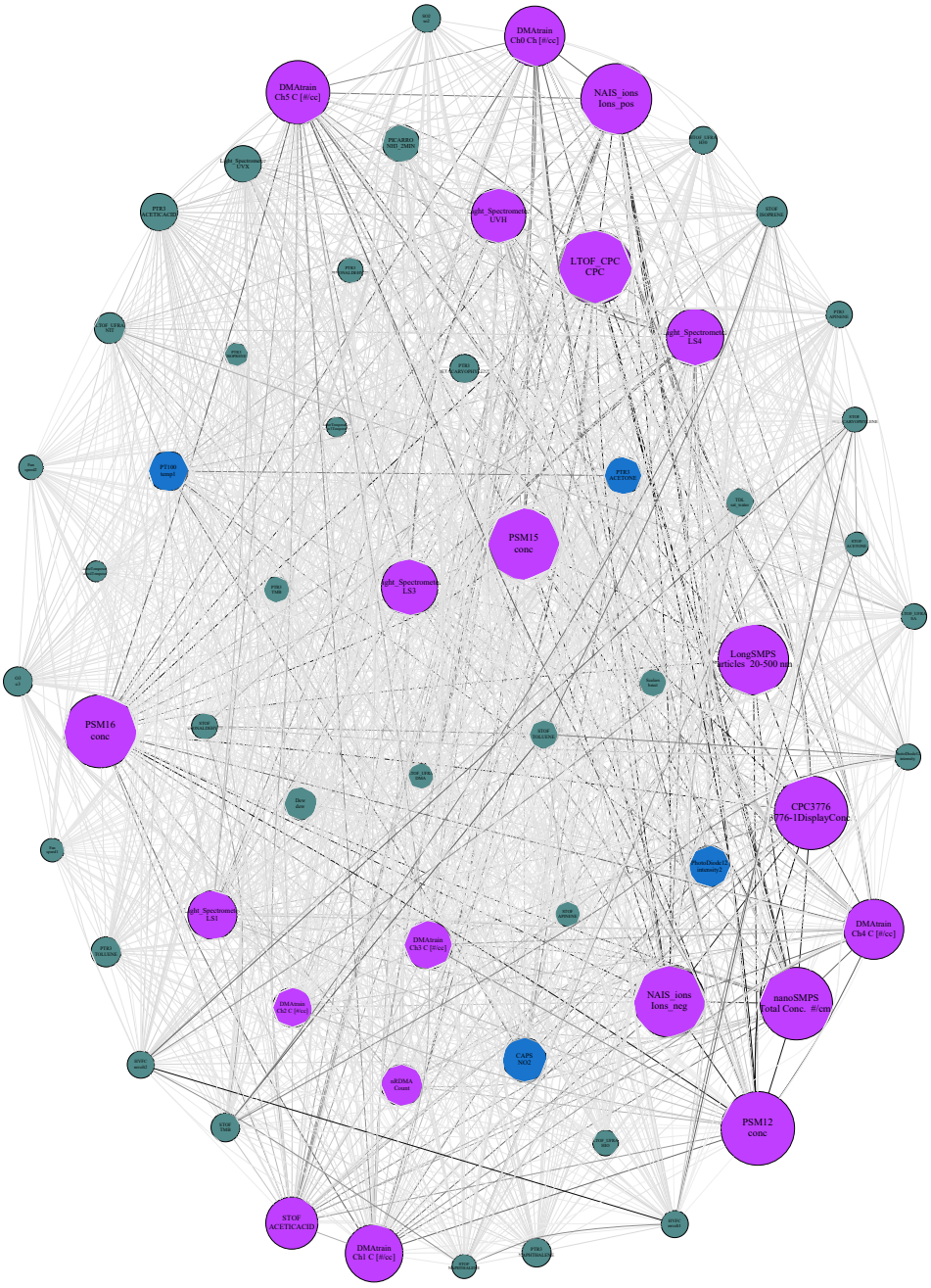



*Figure B.25: Maximum cross-correlation network graph for the CLOUD experimental run 1962.05. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table B.21: Eigenvector ($\epsilon$) centrality node ranking for run 1962.05 using the maximum cross-correlation as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | PSM12 | conc | 0.2448 |
| 2 | LTOF_CPC | CPC | 0.2447 |
| 3 | CPC3776 | 3776-1DisplayConc. | 0.2446 |
| 4 | PSM16 | conc | 0.2425 |
| 5 | PSM15 | conc | 0.2401 |
| 6 | nanoSMPS | Total Conc. #/cm | 0.2401 |
| 7 | NAIS_ions | Ions_neg | 0.2372 |
| 8 | NAIS_ions | Ions_pos | 0.2334 |
| 9 | LongSMPS | Particles 20-500 nm | 0.2321 |
| 10 | DMAtrain | Ch5 C [#/cc] | 0.2031 |
| 11 | DMAtrain | Ch0 Ch [#/cc] | 0.1891 |
| 12 | DMAtrain | Ch4 C [#/cc] | 0.1886 |
| 13 | DMAtrain | Ch1 C [#/cc] | 0.1783 |
| 14 | Light_Spectrometer | LS4 | 0.1775 |
| 15 | Light_Spectrometer | LS3 | 0.1738 |
| 16 | Light_Spectrometer | UVH | 0.1665 |
| 17 | STOF | ACETICACID | 0.1555 |
| 18 | Light_Spectrometer | LS1 | 0.1434 |
| 19 | DMAtrain | Ch3 C [#/cc] | 0.1412 |
| 20 | CAPS | NO2 | 0.1234 |
| 21 | nRDMA | Count | 0.1127 |
| 22 | PhotoDiode12 | intensity2 | 0.1115 |
| 23 | PT100 | temp1 | 0.1058 |
| 24 | DMAtrain | Ch2 C [#/cc] | 0.1042 |
| 25 | PTR3 | ACETONE | 0.0958 |
| 26 | PTR3 | ACETICACID | 0.0944 |
| 27 | PICARRO | NH3_2MIN | 0.0942 |
| 28 | Light_Spectrometer | UVX | 0.0914 |
| 29 | Dew | dew | 0.0768 |
| 30 | LTOF_UFRA | NIT | 0.0678 |
| 31 | STOF | ISOPRENE | 0.0668 |
| 32 | PTR3 | TOLUENE | 0.0641 |
| 33 | O3 | o3 | 0.0605 |
| 34 | PTR3 | NAPHTHALENE | 0.06 |
| 35 | STOF | TOLUENE | 0.0598 |
| 36 | PTR3 | BETACARYOPHYLENE | 0.0598 |
| 37 | STOF | TMB | 0.059 |
| 38 | TDL | sat_water | 0.0558 |
| 39 | SO2 | so2 | 0.0547 |
| 40 | HTOF_UFRA | H30 | 0.0541 |
| 41 | Scalers | brust | 0.053 |
| 42 | HVFC | mvolt2 | 0.0521 |
| 43 | HVFC | mvolt1 | 0.0517 |
| 44 | PTR3 | TMB | 0.049 |
| 45 | PTR3 | APINENE | 0.0488 |
| 46 | STOF | PINONALDEHYDE | 0.0486 |
| 47 | PhotoDiode12 | intensity | 0.0468 |
| 48 | STOF | APINENE | 0.0463 |
| 49 | LTOF_UFRA | HIO | 0.0462 |
| 50 | LTOF_UFRA | DMA | 0.0462 |
| 51 | LTOF_UFRA | SA | 0.0455 |
| 52 | PTR3 | PINONALDEHYDE | 0.045 |
| 53 | STOF | BETACARYOPHYLENE | 0.043 |
| 54 | Fan | speed2 | 0.0413 |
| 55 | STOF | NAPHTHALENE | 0.0402 |
| 56 | STOF | ACETONE | 0.0402 |
| 57 | Fan | speed1 | 0.037 |
| 58 | PTR3 | ISOPRENE | 0.0364 |
| 59 | SabreTemperature | Sabre4Temperature | 0.0266 |
| 60 | SabreTemperature | Sabre3Temperature | 0.0239 |

# Stage 1962.05

4 communities found

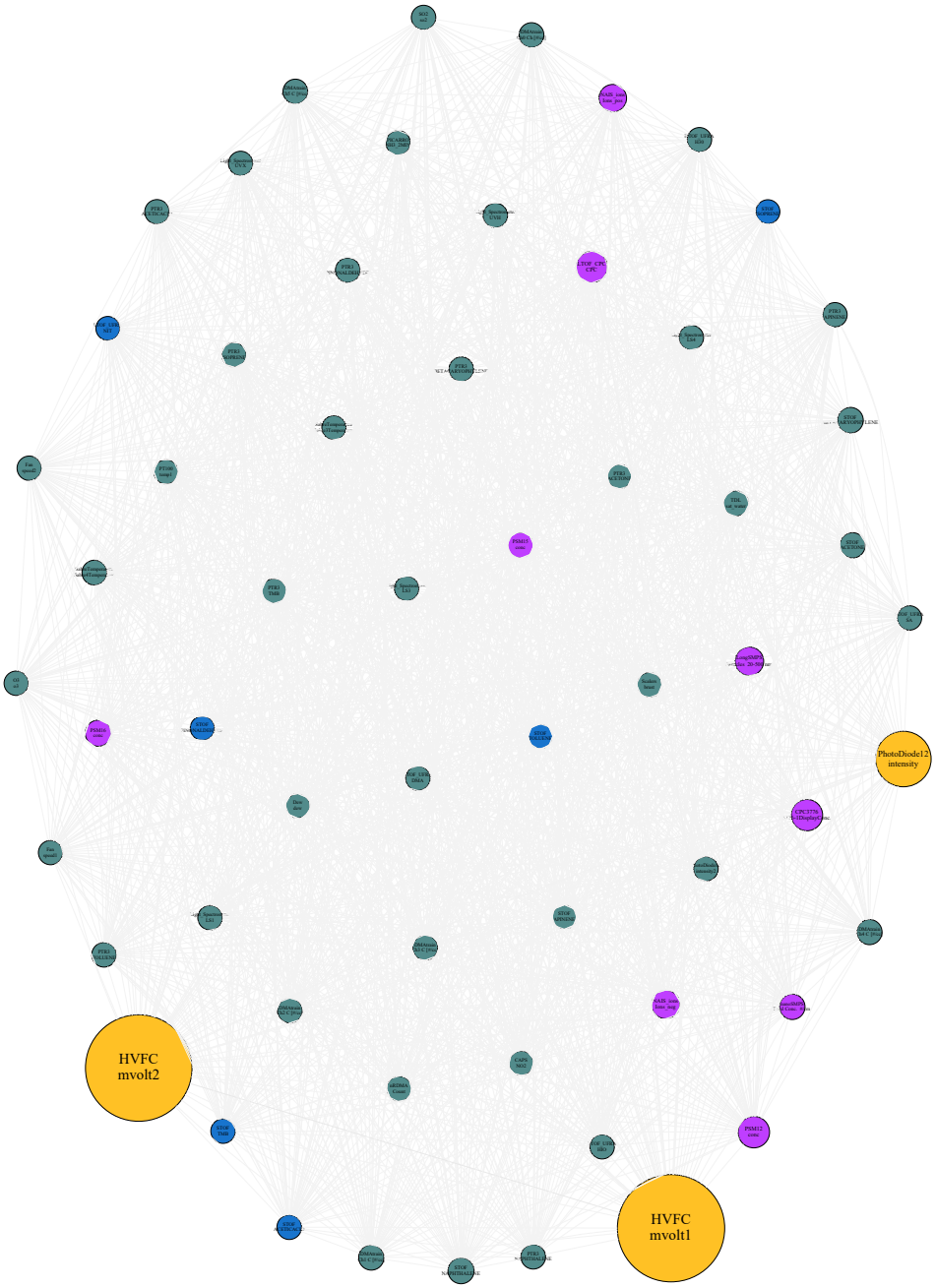



*Figure B.26: DTW similarity network graph for the CLOUD experimental run 1962.05. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table B.22: Eigenvector ($\epsilon$) centrality node ranking for run 1962.05 using the DTW distance as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | HVFC | mvolt1 | 0.6752 |
| 2 | HVFC | mvolt2 | 0.6692 |
| 3 | PhotoDiode12 | intensity | 0.2604 |
| 4 | PSM12 | conc | 0.0674 |
| 5 | LTOF_CPC | CPC | 0.0673 |
| 6 | CPC3776 | 3776-1DisplayConc. | 0.0663 |
| 7 | NAIS_ions | Ions_neg | 0.0444 |
| 8 | LongSMPS | Particles 20-500 nm | 0.044 |
| 9 | NAIS_ions | Ions_pos | 0.0362 |
| 10 | PSM16 | conc | 0.0248 |
| 11 | STOF | NAPHTHALENE | 0.0241 |
| 12 | PSM15 | conc | 0.0231 |
| 13 | STOF | BETACARYOPHYLENE | 0.0212 |
| 14 | TDL | sat_water | 0.0189 |
| 15 | nanoSMPS | Total Conc. #/cm | 0.0186 |
| 16 | STOF | ACETONE | 0.0185 |
| 17 | DMAtrain | Ch1 C [#/cc] | 0.0179 |
| 18 | DMAtrain | Ch4 C [#/cc] | 0.017 |
| 19 | DMAtrain | Ch3 C [#/cc] | 0.0159 |
| 20 | nRDMA | Count | 0.0157 |
| 21 | SabreTemperature | Sabre4Temperature | 0.0144 |
| 22 | Scalers | brust | 0.0144 |
| 23 | DMAtrain | Ch2 C [#/cc] | 0.0139 |
| 24 | DMAtrain | Ch0 Ch [#/cc] | 0.0137 |
| 25 | PTR3 | TMB | 0.0135 |
| 26 | DMAtrain | Ch5 C [#/cc] | 0.0134 |
| 27 | PTR3 | NAPHTHALENE | 0.0125 |
| 28 | PTR3 | PINONALDEHYDE | 0.0125 |
| 29 | STOF | ACETICACID | 0.0125 |
| 30 | PTR3 | TOLUENE | 0.0124 |
| 31 | PTR3 | BETACARYOPHYLENE | 0.0122 |
| 32 | PhotoDiode12 | intensity2 | 0.0121 |
| 33 | Light_Spectrometer | LS1 | 0.0121 |
| 34 | PTR3 | ISOPRENE | 0.0119 |
| 35 | Light_Spectrometer | LS4 | 0.0118 |
| 36 | Light_Spectrometer | LS3 | 0.0117 |
| 37 | STOF | TOLUENE | 0.0117 |
| 38 | STOF | TMB | 0.0116 |
| 39 | Light_Spectrometer | UVH | 0.0115 |
| 40 | PTR3 | ACETONE | 0.0115 |
| 41 | LTOF_UFRA | HIO | 0.0114 |
| 42 | PICARRO | NH3_2MIN | 0.0114 |
| 43 | LTOF_UFRA | DMA | 0.0114 |
| 44 | PTR3 | ACETICACID | 0.0113 |
| 45 | SO2 | so2 | 0.011 |
| 46 | LTOF_UFRA | SA | 0.0108 |
| 47 | Light_Spectrometer | UVX | 0.0108 |
| 48 | O3 | o3 | 0.0108 |
| 49 | PT100 | temp1 | 0.0106 |
| 50 | HTOF_UFRA | H30 | 0.0105 |
| 51 | Fan | speed2 | 0.0105 |
| 52 | CAPS | NO2 | 0.0104 |
| 53 | PTR3 | APINENE | 0.0102 |
| 54 | Dew | dew | 0.0101 |
| 55 | Fan | speed1 | 0.01 |
| 56 | STOF | APINENE | 0.0097 |
| 57 | STOF | PINONALDEHYDE | 0.0086 |
| 58 | SabreTemperature | Sabre3Temperature | 0.0085 |
| 59 | STOF | ISOPRENE | 0.0075 |
| 60 | LTOF_UFRA | NIT | 0.0064 |

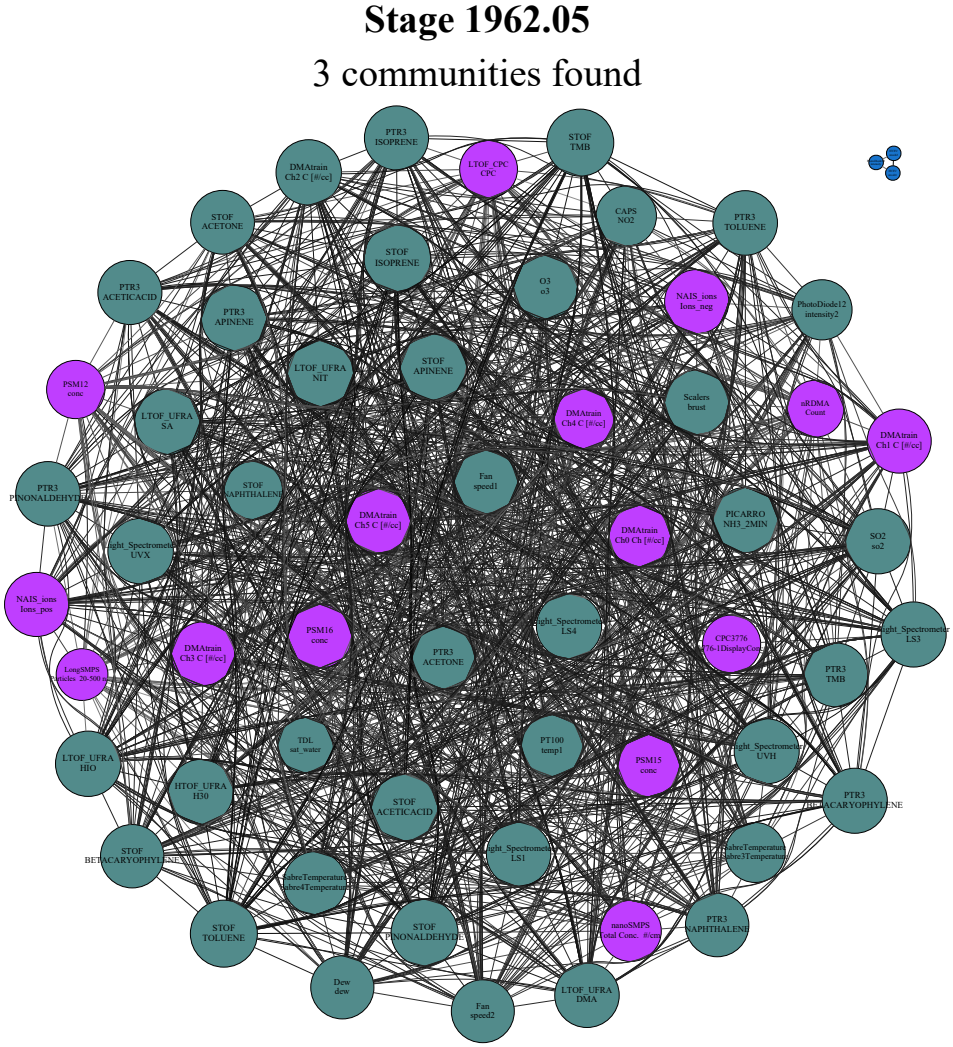



*Figure B.27: Coherence network graph for the CLOUD experimental run 1962.05. A reversed grey scale based on connection strength is applied to the line color connecting each node for visualization purposes.*

*Table B.23: Eigenvector ($\epsilon$) centrality node ranking for run 1962.04 using the coherence as similarity*

| Rank | Instrument | Channel | $\epsilon$ |
|---|---|---|---|
| 1 | Light_Spectrometer | LS3 | 0.1411 |
| 2 | Light_Spectrometer | UVX | 0.1408 |
| 3 | HTOF_UFRA | O2 | 0.1407 |
| 4 | HTOF_UFRA | NH3 | 0.1406 |
| 5 | HVFC | mvolt2 | 0.1403 |
| 6 | Light_Spectrometer | UVH | 0.14 |
| 7 | HTOF_UFRA | H3OH2O | 0.1399 |
| 8 | HTOF_UFRA | H7O3 | 0.1392 |
| 9 | DMAtrain | Ch2 C [#/cc] | 0.1383 |
| 10 | DMAtrain | Ch3 C [#/cc] | 0.1378 |
| 11 | STOF | ISOPRENE | 0.1367 |
| 12 | Scalers | brust | 0.1357 |
| 13 | LTOF_UFRA | NIT | 0.1354 |
| 14 | STOF | BETACARYOPHYLENE | 0.1354 |
| 15 | O3 | o3 | 0.1352 |
| 16 | PTR3 | TMB | 0.1351 |
| 17 | PTR3 | ISOPRENE | 0.135 |
| 18 | LTOF_UFRA | SA | 0.1349 |
| 19 | PTR3 | PINONALDEHYDE | 0.1349 |
| 20 | Light_Spectrometer | LS1 | 0.1348 |
| 21 | LTOF_UFRA | HIO | 0.1346 |
| 22 | STOF | APINENE | 0.1344 |
| 23 | PTR3 | BETACARYOPHYLENE | 0.1343 |
| 24 | STOF | ACETONE | 0.1341 |
| 25 | STOF | TOLUENE | 0.134 |
| 26 | SO2 | so2 | 0.134 |
| 27 | LongSMPS | Particles 20-500 nm | 0.134 |
| 28 | PICARRO | NH3_2MIN | 0.1339 |
| 29 | Dew | dew | 0.1338 |
| 30 | HVFC | mvolt1 | 0.1337 |
| 31 | LTOF_UFRA | DMA | 0.1332 |
| 32 | STOF | PINONALDEHYDE | 0.1329 |
| 33 | STOF | TMB | 0.1327 |
| 34 | HTOF_UFRA | H30 | 0.1321 |
| 35 | STOF | ACETICACID | 0.1321 |
| 36 | PTR3 | ACETICACID | 0.1313 |
| 37 | PTR3 | ACETONE | 0.131 |
| 38 | PTR3 | NAPHTHALENE | 0.1304 |
| 39 | PTR3 | TOLUENE | 0.1304 |
| 40 | DMAtrain | Ch4 C [#/cc] | 0.1303 |
| 41 | DMAtrain | Ch1 C [#/cc] | 0.129 |
| 42 | PTR3 | APINENE | 0.1284 |
| 43 | DMAtrain | Ch0 Ch [#/cc] | 0.1277 |
| 44 | PhotoDiode12 | intensity2 | 0.1267 |
| 45 | Light_Spectrometer | LS4 | 0.1265 |
| 46 | SabreTemperature | Sabre3Temperature | 0.1265 |
| 47 | NAIS_ions | Ions_neg | 0.1264 |
| 48 | SabreTemperature | Sabre4Temperature | 0.1244 |
| 49 | PT100 | temp1 | 0.121 |
| 50 | nanoSMPS | Total Conc. #/cm | 0.1182 |
| 51 | STOF | NAPHTHALENE | 0.1167 |
| 52 | NAIS_ions | Ions_pos | 0.1164 |
| 53 | CAPS | NO2 | 0.1132 |
| 54 | TDL | sat_water | 0.103 |
| 55 | PSM16 | conc | 0.1026 |
| 56 | PSM12 | conc | 0.1012 |
| 57 | nRDMA | Count | 0.1 |
| 58 | PSM15 | conc | 0.099 |
| 59 | LTOF_CPC | CPC | 0.0972 |
| 60 | CPC3776 | 3776-1DisplayConc. | 0.0966 |
| 61 | Fan | speed1 | 0.0 |
| 62 | Fan | speed2 | 0.0 |

### B.2.4.3 Relevant Period Selection

Figure B.28 shows the event highlighting periods and table B.24 shows the distribution of the detected events over each channel. The first period was found to be the most relevant and the third period also showing significant emphasis. The latter was attributed to the correcting of a failure of the charged particle beam used in CLOUD during the first half of the run. A large amount of detected events are found for the beam particle counter (*scalers brust*) as well as other time series (of note are several particle counters such as *NAIS_ions Ions_neg* and *LongSMPS Particles 20-500 nm*) which react to this change in the particle beam behavior during this thrid period.

*Table B.24: Events in each period for run 1962.05 using wavelet ME partitioning.*

| Channel | Period 1 | Period 2 | Period 3 | Period 4 | Total |
|---|---|---|---|---|---|
| Dew dew | 5 | 3 | 2 | 3 | 13 |
| NAIS_ions Ions_pos | 4 | 6 | 2 | 0 | 12 |
| Scalers brust | 0 | 0 | 9 | 3 | 12 |
| nanoSMPS Total Conc. #/cm | 1 | 2 | 1 | 6 | 10 |
| HVFC mvolt2 | 9 | 0 | 0 | 0 | 9 |
| HVFC mvolt1 | 9 | 0 | 0 | 0 | 9 |
| DMAtrain Ch3 C [#/cc] | 7 | 0 | 1 | 0 | 8 |
| LongSMPS Particles 20-500 nm | 0 | 0 | 5 | 2 | 7 |
| Fan speed2 | 0 | 3 | 0 | 4 | 7 |
| NAIS_ions Ions_neg | 1 | 0 | 5 | 0 | 6 |
| Light_Spectrometer UVH | 2 | 0 | 3 | 0 | 5 |
| TDL sat_water | 1 | 1 | 1 | 1 | 4 |
| DMAtrain Ch0 Ch [#/cc] | 3 | 0 | 0 | 0 | 3 |
| Light_Spectrometer UVX | 0 | 2 | 0 | 1 | 3 |
| DMAtrain Ch2 C [#/cc] | 0 | 0 | 3 | 0 | 3 |
| Light_Spectrometer LS1 | 0 | 1 | 0 | 1 | 2 |
| Light_Spectrometer LS4 | 1 | 0 | 0 | 0 | 1 |
| Light_Spectrometer LS3 | 0 | 0 | 0 | 1 | 1 |
| Fan speed1 | 0 | 0 | 0 | 1 | 1 |
| DMAtrain Ch4 C [#/cc] | 1 | 0 | 0 | 0 | 1 |
| Total | 44 | 18 | 32 | 23 | 117 |

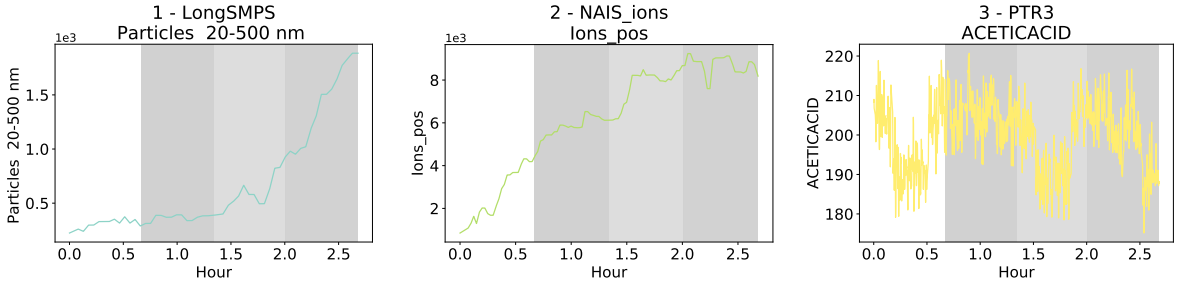



*Figure B.28: Most relevant time series for CLOUD experimental run 1962.05 coupled with ranking of specific time periods to the number of events observed in each event. The time period was divided into 4 distinct sub-periods, which were then tested for the existence of events. The ranking of each sub-period shades each sub-period more the less events are found. The first period was considered the most important, followed by a slight relevance of the thrid period.*