

**Temporary Liver Support Using Artificial Lung Technologies:
Ammonia Removal via Peritoneal Dialysis for Acute Liver
Failure with PFC and PDMS at Vacuum Pressure
DATA ANALYSIS**

This Document (DATA ANALYSIS), MATLAB CODE, and Data Repository:

<https://doi.org/10.17605/OSF.IO/GJFPD>

Lukas W. DiBeneditto

A.A., Communications, Jefferson Community & Technical College
B.S., Mechanical Engineering Technology, Purdue University
M.S., Biomedical Engineering Research, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA

October, 2023

© Lukas W. DiBeneditto, 2023

All Rights Reserved

ABSTRACT

This document details the preparation and statistical analysis of the multi-device recorded data for the master's thesis research project titled "Temporary Liver Support Using Artificial Lung Technologies: Ammonia Removal via Peritoneal Dialysis for Acute Liver Failure with PFC and PDMS at Vacuum Pressure". (DiBeneditto, 2023) Central to the analysis is a MATLAB program, "main.m", comprising 3313 lines of code. This program produced the figures, tables, and results presented in this document. The resulting files are stored in the program working directory, "_trials" and "device-calibration" folders, and represent 355 files across 114 folders, totaling 144 MB. The following sections provide in-depth insights into the methodologies employed, findings obtained, and their implications in the broader context of Acute Liver Failure management. The core objective of our study was to evaluate the efficiency and feasibility of a system incorporating a PDMS filter (representative of artificial liver technology) in tandem with perfluorocarbon (PFC, representative of artificial lung technology) to remove ammonia from a liquid solution in a simulated peritoneal environment. To investigate the potential application for acute liver failure (ALF) patients and attempt to answer two central questions: Can the PDMS filter effectively facilitate the diffusion of ammonia from the liquid to the gas phase, thereby removing it? Does the presence of PFC enhance or influence this ammonia removal process? Our data provides substantial evidence regarding the efficacy of the PDMS filter in facilitating ammonia removal. However, the role of PFC in this process remains unclear and requires further investigation. The complexity of the experiment and the variations observed in trials with similar parameters suggest we need additional studies to understand the variables we are exploring fully.

KEYWORDS: acute liver failure (ALF), peritoneal dialysis, polydimethylsiloxane (PDMS) silicone, perfluorocarbons (PFCs), hyperammonemia, ammonia removal

ORCID:

Lukas W. DiBeneditto MS, Carnegie Mellon University, Principal Author: 0000-0001-9262-9110

Keith E. Cook PhD, Carnegie Mellon University, Faculty Advisor: 0000-0002-5604-3718

Rosalyn D. Abbott PhD, Carnegie Mellon University, Faculty Reader: 0000-0003-3287-3175

Liu Yang MBBS, Mayo Clinic, Independent Advisor: 0000-0003-0677-6152

DATA AVAILABILITY: <https://doi.org/10.17605/OSF.IO/GJFPD>

CONFLICTS OF INTEREST: The author declares no conflict of interest.

ACKNOWLEDGEMENTS: The author expresses profound gratitude to the numerous individuals who offered support and guidance. My heartfelt thanks go to my family, friends, teachers, staff, and colleagues at Indiana University, Purdue University, Carnegie Mellon University, the Mayo Clinic, and MathWorks.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION.....	8
§ 1.1. Simplified System Circuit Diagram.....	13
§ 1.2. Normal Flow Path During Trials for the Gas-Pathway.....	14
§ 1.3. Normal Flow Path During Trials for the Liquid-Loop.....	14
§ 1.4. Introduction Summary.....	15
CHAPTER 2: MATERIAL & METHODS.....	16
§ 2.1. Data Recording Devices with Associated Data MATLAB Variable(s) and Details.....	16
§ 2.1.1. Arduino Device.....	17
§ 2.1.2. Omega Device.....	18
§ 2.1.3. ToxiRae Device.....	19
§ 2.1.4. Vernier Device.....	20
§ 2.1.5. Additional Related Hardware and Software.....	21
§ 2.2. Arduino Pressure Sensors Calibration.....	21
§ 2.2.1. Initial Arduino Pressure Sensor Calibration.....	22
§ 2.2.2. Arduino Device Iterative Calibration Approach.....	22
§ 2.2.3. Final Arduino Device Calibration for Pressure Sensor Bias.....	23
§ 2.3. Device Accuracy and Timing Synchronization Considerations.....	24
§ 2.3.1. Pretrial Synchronization Procedure and Device Data Log File Formats.....	25
§ 2.3.2. Device Accuracy Time Drift Compensation.....	27
§ 2.3.3. Synchronization Summary.....	29
§ 2.4. Device Sample Rates.....	30
§ 2.5. Trial Start and Stop Times.....	30
§ 2.6. Gather and Organize Raw Files Recorded by Four Devices.....	33
§ 2.7. Data Preparation for Trial Synchronization.....	33
§ 2.8. Trial and Phase Overlap.....	37
§ 2.9. The Vacuum Pump & Gas Pressure Determine Phase Start and Stop Times.....	38
§ 2.10. Removal of Unused Data.....	38
§ 2.11. Synchronization of Device Logs by Matching Date, Time, and Sample Rates.....	39
§ 2.12. Combine and Synchronize Data For All Devices For All Trials.....	39
§ 2.13. Final Cleanup to Prepare for Statistical Analysis.....	40
§ 2.14. Methods Summary.....	41

CHAPTER 3: RESULTS & DISCUSSION.....	42
§ 3.1. Statistical Analysis Workflow and MATLAB Implementation.....	42
§ 3.1.1. Methods and Tools Employed.....	42
§ 3.1.2. MATLAB Workflow Overview.....	42
§ 3.1.3. The Need for Statistical Analysis.....	44
§ 3.2. Data Normality Checks.....	45
§ 3.2.1. Normality Tests and Preliminary Findings.....	46
§ 3.3. Friedman Test Analysis.....	47
§ 3.3.1. Results from the Friedman Tests Comparing Different Phases.....	48
§ 3.3.2. Discussion of Friedman Test Analysis.....	49
§ 3.4. Exploratory Data Analysis (EDA).....	50
§ 3.4.1. Size Comparisons of Different Phases Across Various Trials.....	51
§ 3.5. Correlation Analysis.....	52
§ 3.5.1. Interpreting Correlation Analysis.....	53
§ 3.5.2. Results of Correlation Analysis.....	54
§ 3.6. Linear Regression Modeling.....	54
§ 3.7. Analysis of Ammonia Readings Based on Presence of PFC.....	57
§ 3.8. Phase 2 Focus.....	59
§ 3.9. Duration of each phase in hours, minutes, and seconds.....	61
§ 3.10. Phase 2 Mean Pressure \pm SD for Liquid-Loop and Gas-Pathway.....	62
§ 3.11. Phase 2 Mean Temperature \pm SD for Liquid-Loop and Gas-Pathway.....	63
§ 3.12. Phase 2 Mean Volumetric Flow Rate \pm SD for Gas-Pathway.....	64
§ 3.13. Phase 2 Mean Ammonium and Electrical Potential \pm SD.....	65
§ 3.14. Phase 2 Mean pH and Ammonia \pm SD.....	66
§ 3.15. Ammonia Readings During Each Phase for Every Trial.....	67
§ 3.16. Phase 2 Ammonium (NH ₄ ⁺) vs. Adjusted Mean Ammonia (NH ₃).....	68
§ 3.16.1. Discussion of Phase 2 Ammonium vs. Adjusted Mean Ammonia.....	68
§ 3.17. Volumetric Flow Rate vs. Ammonia Gas Reading (All Trials).....	70
§ 3.18. Phase 2 Mean psiaA0 (SEM Error Bars).....	71
§ 3.19. Phase 2 Mean psiaA1 (SEM Error Bars).....	72
§ 3.20. Phase 2 Mean psiaA2 (SEM Error Bars).....	73
§ 3.21. Phase 2 Mean absolutePressurePSIA (SEM Error Bars).....	74
§ 3.22. Phase 2 Mean gasTemperatureC (SEM Error Bars).....	75
§ 3.23. Phase 2 Mean volumetricFlowRateRawCCM (SEM Error Bars).....	76

§ 3.24. Phase 2 Mean volumetricFlowRateStandardizedSCCM (SEM Error Bars).....	77
§ 3.25. Phase 2 Mean DataSet1AmmoniummgL (SEM Error Bars).....	78
§ 3.26. Phase 2 Mean DataSet1PotentialmV (SEM Error Bars).....	79
§ 3.27. Phase 2 Mean DataSet1TemperatureC (SEM Error Bars).....	80
§ 3.28. Phase 2 Mean DataSet1pH (SEM Error Bars).....	81
§ 3.28.1. Discussion on Elevated pH Values for Trials 14, 15, and 16.....	81
§ 3.29. Phase 2 Mean Sensor1GasReading (SEM Error Bars).....	84
§ 3.29.1. Discussion of Sensor1GasReading Ammonia Gas Values.....	84
§ 3.29.2. In-Depth Analysis of Select Trials for Ammonia Gas Values.....	86
§ 3.30. Gas-Pathway Ammonia Readings Across All Trials and Phases.....	88
§ 3.30.1. Discussion of Gas-Pathway Ammonia Readings Across All Trials and Phases	89
§ 3.31. Pairwise Comparisons for Similar Ammonium Hydroxide Concentrations.....	91
§ 3.32. Summary Table.....	93
§ 3.32.1. Discussion of Summary Table.....	93
§ 3.33. Additional Insights and Potential Areas of Improvement.....	95
§ 3.34. SUMMARY OF DATA ANALYSIS.....	97
§ 3.34.1. Central Questions.....	97
§ 3.34.2. Central Findings.....	97
§ 3.34.3. Conclusion.....	99
§ 3.35. DISCLAIMER.....	99
CHAPTER 4: BIBLIOGRAPHY.....	101
APPENDIX: MAIN.M MATLAB PROGRAM SOURCE CODE.....	115

CHAPTER 1: INTRODUCTION

The liver plays a crucial role in the human body by removing waste products, including ammonia. However, when the liver's function decreases due to damage or other factors, ammonia levels in the body can rise to fatal concentrations unless there is medical intervention. Existing medical procedures, such as hemodialysis, remove most ammonia from the bloodstream. (Robinson et al., 2018; Summar et al., 1996) Alternatively, peritoneal dialysis employs the peritoneum, the membrane lining the abdomen, to filter ammonia and, in some instances, offer better survival for cirrhosis patients. (Chou et al., 2016; Khan et al., 2018; Ponce et al., 2021) By introducing a dialysate solution into the abdomen through a catheter, waste products, including ammonia, transfer from the blood and peritoneal fluid into the dialysate. (Carr et al., 2006) This dialysate is drained and replaced with a fresh solution to repeat the process. (Nissenson & Fine, 2017; Rastogi et al., 2021)

Artificial liver technologies leverage dialysate solutions to facilitate the diffusion of small solutes across semi-permeable membranes, sometimes employing polydimethylsiloxane (PDMS) filters due to high gas permeability. (Cove et al., 2019; Dabaghi et al., 2019) Similarly, artificial lung technologies support gas exchange, oxygenation, and carbon dioxide removal through perfluorocarbons (PFCs), which possess exceptional gas solubility, a principle seen in liquid breathing. (Bauer et al., 2022; Castro & Briceno, 2010; Jeng et al., 2003) These technologies synergistically offer a promising method for ammonia removal and temporary liver support.

Our research focuses on combining artificial liver and lung technologies by conducting an in vitro simulated peritoneal environment. We aim to assess the feasibility and efficiency of using a PDMS filter (artificial liver technology) with a perfluorocarbon (artificial lung technology) to degas a liquid ammonia solution, employing a novel artificial liver medical device prototype. This document describes the trials with precise amounts of reverse osmosis deionized (RODI) water, ammonium hydroxide (NH₄OH), and a non-medical grade perfluorocarbon (PFC 3M FC-770) for initial proof-of-concept testing. (DiBeneditto, 2023)

The artificial liver medical device prototype, hereafter known as “the system”, was designed, developed, tested, and iterated over two years from August 2021 to August 2023 by Lukas W. DiBeneditto, MS, while he served as a Bio-Engineering Organs Initiative Researcher for the renowned Cook Cardiopulmonary Engineering Group and collaborated with the Mayo Clinic in their Transforming Transplant strategic initiative. Lukas was under the expert guidance of faculty from Carnegie Mellon University (Dr. Keith E. Cook, PhD and Dr. Rosalyn D. Abbott, PhD) and the Mayo Clinic (Dr. Liu Yang, MBBS).

At the heart of the system is the PDMS filter, acting as the interface and pressure gradient that facilitates gas exchange from the Liquid-Loop to the Gas-Pathway. Working in concert with the PDMS filter is the PFC 3M FC-770 liquid, which stands in for the dialysate circulated into and out of the peritoneal cavity. The main idea is that by circulating a PFC into the peritoneal space, we can transport excess ammonia in an acute liver failure (ALF) patient model.

In acute liver failure, the accumulation of ammonia plays a critical role in developing complications like hepatic encephalopathy. (Mohiuddin & Khattar, 2023) Our research aims to understand the interactions between ammonia (NH_3) and its ionized form, ammonium ion (NH_4^+), especially in a simulated peritoneal system. Ammonium hydroxide helps us achieve a controlled environment to study these interactions.

Our system mimics the conditions experienced in ALF, namely elevated ammonia levels. Given the prominence of ionized NH_4^+ in physiological environments, with a smaller fraction of NH_3 , ammonium hydroxide offers an ideal medium to simulate these conditions. (Mohiuddin & Khattar, 2023) We employ different volumes of ammonium hydroxide in our experiments, ranging from 0.00 μL to 1515.00 μL . This range allows us to understand the dose-response relationship, which is pivotal in understanding the efficacy of our ammonia removal system in different ammonia concentration scenarios.

Ammonia, with its molecular dipole moment, displays restricted permeability through lipid membranes due to its inherent polarity. Conversely, ammonium ions utilize specialized transport proteins to navigate biological barriers. (Adeva et al., 2012; Mohiuddin & Khattar, 2023) Our system integrates polydimethylsiloxane (PDMS) filter, which possesses unique properties that mimic certain aspects of lipid membranes while offering enhanced control and replicability in an experimental setup. By pairing this PDMS filter with ammonium hydroxide, we create a mock environment that closely mirrors the dynamics of ammonia and ammonium ions within biological systems. Finally, safety remains paramount in our experimental approach. With ammonium hydroxide, we can introduce defined quantities of ammonia, modify concentrations

as required, and exhaust the toxic ammonia gas into the fume hood to ensure precise and safe study conditions.

An adult typically has about 50 mL of peritoneal fluid in the peritoneal cavity. (Rudralingam et al., 2017; Tarn & Lapworth, 2010) For our experiment, we will use a 50 mL or 100 mL mixture of reverse osmosis deionized (RODI) water with different amounts of ammonium hydroxide to replicate the presence of ammonia in ALF. The patient peritoneal cavity is represented with a vortex mixing tank (VMT), which is a custom-designed and fabricated PVC container consisting of an inlet, outlet, magnetic stir bar, and three sensors measuring ammonium (NH_4^+), pH, and temperature, which are connected to the **Vernier device** which combines the data. In addition to the RODI water and ammonium hydroxide liquid mixture, we will add the PFC 3M FC-770 to the test. However, the sample size was limited due to time constraints. This liquid mixture was circulated through a Liquid-Loop by a roller pump at a constant flow rate, measured in advance. Data for determining the Liquid-Loop flow rate can be found in the “device-calibration” folder.

The Gas-Pathway starts with dry lab air, passes through a calibrated electronic flow meter **Omega device**, PDMS filter, vacuum pump, a custom-designed and fabricated PFC adaptor sealing over a ToxiRae Pro Ammonia Gas meter (**ToxiRae device**), which then vents to a fume hood. The Omega device calibration is National Institute of Standards and Technology (NIST) traceable to be accurate and measures temperature, absolute pressure, raw uncorrected volumetric flow rate, and corrected volumetric flow rate based on the selected gas (dry air) and the other three parameters. The ToxiRae device was calibrated with Ammonia (NH_3) Calibration Gas 50 (ppm, mg/L).

In addition, a custom-designed and fabricated **Arduino device** captures Liquid-Loop and Gas-Pathway absolute pressure. The Arduino device pressure sensors were calibrated with custom Arduino code, barometric pressure, and iterative pressure sensor bias compensation. (DiBeneditto, 2023) Gas-Pathway pressure sensor data is captured by a pre-PDMS filter by the Omega device and a post-PDMS filter by the Arduino device. The Arduino device captures pressure sensor data for the liquid-loop pre- and post-PDMS filter.

§ 1.1. Simplified System Circuit Diagram

Data recording devices are highlighted in yellow in the figure below.

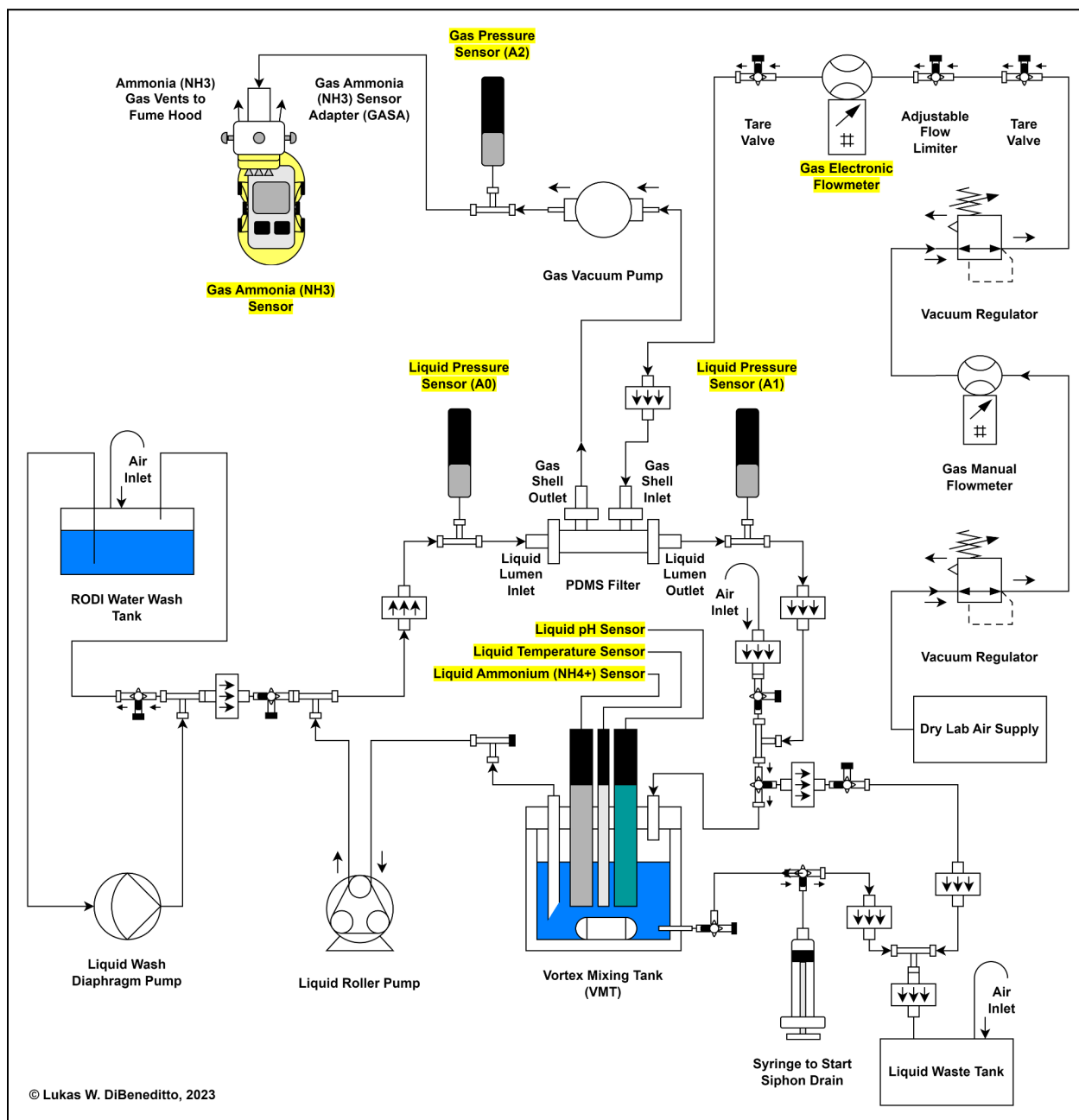


Figure 1. Illustration of Simplified System Circuit (Data Recording Devices Highlighted in Yellow). This diagram represents what the system looks like during trials. Adapted from (DiBeneditto, 2023). The Gas-Pathway starts at the “Dry Lab Air Supply” and continues to “Ammonia (NH₃) Gas Vents to Fume Hood”. The Liquid-Loop starts at the “Vortex Mixing Tank (VMT)” and continues counterclockwise. Gas exchange occurs in the “PDMS Filter”.

§ 1.2. Normal Flow Path During Trials for the Gas-Pathway

1. Beginning at the Dry Lab Air Supply
2. Vacuum Regulator
3. Gas Manual Flowmeter
4. Vacuum Regulator
5. Tare Valve
6. Adjustable Flow Limiter
7. Gas Electronic Flowmeter
8. Tare Valve
9. Nylon Check Valve (one-way valve)
10. Gas Shell Inlet, PDMS Filter, and then Gas Shell Outlet
11. Gas Vacuum Pump
12. Stopcock (3-Way) and Gas Pressure Sensor A2
13. Gas Ammonia (NH₃) Sensor Adapter (GASA)
14. Gas Ammonia (NH₃) Sensor
15. Ending where Ammonia (NH₃) Gas Vents to the Fume Hood, along with the Dry Lab Air

§ 1.3. Normal Flow Path During Trials for the Liquid-Loop

1. Beginning at the Vortex Mixing Tank (VMT) (moving clockwise in the figure), including the Liquid pH Sensor, Liquid Temperature Sensor, Liquid Ammonium (NH₄⁺) Sensor, and Magnetic Stir Bar (unlabeled)
2. Stopcock (3-Way)
3. Liquid Roller Pump

4. Stopcock (3-Way)
5. Nylon Check Valve (one-way valve)
6. Stopcock (3-Way) and Liquid Pressure Sensor A0
7. Liquid Lumen Inlet, PDMS Filter, and then Liquid Lumen Outlet
8. Stopcock (3-Way) and Liquid Pressure Sensor A1
9. Nylon Check Valve (one-way valve)
10. Stopcock (3-Way)
11. Ending at the Vortex Mixing Tank (VMT)

§ 1.4. Introduction Summary

The data consists of 16 trials. RODI water was the only type of water used in the Liquid-Loop and Vortex Mixing Tank. The Liquid-Loop was flushed, washed, and dried (with dry lab air) between Trials 1 to 2, 7 to 8, and 13 to 14. Trial 8 was the control. Iteration testing trials before Trial 1 are not included in the data. With a very high degree of certainty, there were no gas or liquid leaks for the recorded trial data. The data was recorded by the multiple sensors in four devices: Arduino, Omega, ToxiRae, and Vernier, which will be discussed in the next chapter.

CHAPTER 2: MATERIAL & METHODS

This chapter describes the material and methods used for this experiment. We have already described that the data consists of 16 trials recorded by multiple sensors in four devices: Arduino, Omega, ToxiRae, and Vernier. We used Mathworks MATLAB software to prepare, synchronize, and analyze the data. The MATLAB variables associated with and recorded by specific devices will now be described in more detail.

§ 2.1. Data Recording Devices with Associated Data MATLAB Variable(s) and Details

The following information related to this data has been condensed for brevity. Specific information about the PDMS filter, PFC 3M FC-770, tubing, and additional components used in the system can be found in the thesis (DiBeneditto, 2023). The data recording devices are highlighted in yellow, and the MATLAB data variables associated with those devices are highlighted in orange.

The specific variable names are somewhat self-explanatory. However, the specific meaning of each variable is detailed later in this document.

§ 2.1.1. Arduino Device

Table 1. Arduino Device Details.

Arduino Device Item	Matlab Variable(s) & Details
(Arduino Device) Arduino	Variable(s): timeMS, psiaA0, psiaA1, psiaA2 REXQualis UNO R3 Board ATmega328P ATMEGA16U2 Development Board Compatible With Arduino. From REXQualis Super Starter Kit based on Arduino UNO R3 with Tutorial and Controller Board Compatible with Arduino IDE. (UNO R3 Board, 2023) Custom Arduino code is available. See APPENDIX 1 - APPENDIX ARDUINO TIMER CALCULATIONS in DiBeneditto, 2023. NOTE: Since the custom software is basing the pressure transducer voltage calculations on the ATmega328P Analog to Digital Voltage Converter (ADC) and a 9V 1a power supply adapter is required in order to aid in damping the voltage fluctuations from power mains instead of drawing voltage from the laptop USB. (DiBeneditto, 2023)
(software) Arduino IDE	Arduino IDE , Arduino, v2.1.0. (Software Arduino, 2023)
(software) PuTTY Software	Putty , Simon Tatham, v0.78. (Tatham, 2022)
Arduino Power Cable	9V 1A Power Adapter . (REXQualis Super Starter Kit, 2017)
Arduino USB Cable	Assmann WSW Components AK672MB USB 1.1 (USB 1.0) Cable A Male to B Male 6.56' (2.00m) Shielded . (AK672MB Assmann, 2023)
Gas Pressure Sensor (A2)	Variable(s): psiaA2 AUTEX 5823950559 GSND-0601293724 Pressure Transducer Sensor 100 Psi Pressure Sender 316 Stainless Steel Oil Pressure Transmitter 1/8" -27 NPT For Oil Fuel Air Water Pressure. (Amazon.com: AUTEX, 2023)
Liquid Pressure Sensor (A0)	Variable(s): psiaA0 Same as the Gas Pressure Sensor (A2)
Liquid Pressure Sensor (A1)	Variable(s): psiaA1 Same as the Gas Pressure Sensor (A2)

§ 2.1.2. Omega Device

Table 2. Omega Device Details.

Omega Device Item	Matlab Variable(s) & Details
(Omega Device) Gas Electronic Flow Meter	Variable(s): absolutePressurePSIA, gasTemperatureC, volumetricFlowRateRawCCM, volumetricFlowRateStandardizedSCCM Omega Engineering, Inc. FMA-1618A Mass and Volumetric Flow Meter. NOTE: Change Serial Reporting to 500 ms for this study: Per User's Guide "Streaming Mode" instructions. Stop Streaming Mode with "@@=a↵", set interval using "aw91=500↵", and start Mode using "a@=@↵". (DiBeneditto, 2023; FMA-1618A FMA1600, 2023)
(software) PuTTY Software	Putty , Simon Tatham, v0.78. (Tatham, 2022) NOTE: Follow Configuration of PuTTY Software: In PuTTY Configuration, navigate to "Terminal" and check "Implicit LF in every CR", set "Local echo" to "Force on". Go to "Connection" > "Serial", set "Speed" to 19200, "Data bits" to 8, "Stop bits" to 1, with "Parity" and "Flow control" set to "None". In "Session", choose "Serial" under "Connection type", verify COM port and baud rate. (DiBeneditto, 2023)
Omega Adapter Cable 8 Pin Din to DB-9 Adapter Cable	OMEGA FMA1600-C3 1.8M (6') 8 pin male Mini-DIN to DB9 female adapter. (1.8M (6') 8 pin male Mini-DIN, 2023)
Omega Adapter Cable DB-9 to RS-232 to USB	Pluggable PL2303-DB9 USB TO RS-232 DB9 SERIAL ADAPTER (PROLIFIC PL2303HX CHIPSET). (Pluggable USB to RS-232 DB9, 2023)

§ 2.1.3. ToxiRae Device

Table 3. ToxiRae Device Details.

ToxiRae Device Item	Matlab Variable(s) & Details
(ToxiRae Device) Gas Ammonia (NH ₃) Sensor	Variable(s): Sensor1GasReading HONEYWELL G02-B810-100 Single Gas Detector: Ammonia, 0 to 100 ppm, Black, Audible / Vibrating / Visual, Lithium, LCD. (HONEYWELL, Ammonia, 2023) It is also called a ToxiRAE Pro Monitor Single Gas Detector. The data, while collected in real-time, must be downloaded by the Honeywell Safety Suite Device Configurator Software after the trials are completed for that day. As soon as the Honeywell Safety Suite Device Configurator connects to the ToxiRae Device, the ToxiRae Device stops recording data.
(software) Honeywell Safety Suite Device Configurator Software	Honeywell Safety Suite Device Configurator , Desktop Application, Honeywell, v3.7.0. (Safety Suite Solution, 2023)
Gas Electrochemical Sensor	Variable(s): Sensor1GasReading Honeywell RAE Systems, Hughes Co Inc C03-0950-000 4R+ EC NH3-100 sensor module. (RAE Systems Replacement Sensor C03-0950-000, 2023) The electrochemical sensor is installed in the ToxiRae device.
ToxiRAE Pro Charging and PC Communication Cradle	RAE Systems Charging Cradle G02-3008-000, power adapter cable 500-0036-100, and USB communications cable 410-0203-000. (RAE Systems Cradle, 2023; RAE Systems PC Communications Cable, 2023)

§ 2.1.4. Vernier Device

Table 4. Vernier Device Details.

Vernier Device Item	Matlab Variable(s) & Details
(Vernier Device) LabQuest	Variable(s): DataSet1AmmoniummgL, DataSet1PotentialmV, DataSet1pH, DataSet1TemperatureC Vernier LABQ3 LabQuest 3 . NOTE: Use with a manufacturer-supplied USB Cord and Power Adapter. (LabQuest 3 - Vernier, 2023)
(software) Vernier Graphical Analysis Software	Vernier Graphical Analysis , Vernier Science Education, v5.17.2-3067. (Download Vernier Graphical Analysis, 2023)
Liquid Ammonium (NH₄⁺) Sensor	Variable(s): DataSet1AmmoniummgL, DataSet1PotentialmV Vernier GDX-NH4 Go Direct Ammonium Ion-Selective Electrode . (Go Direct Ammonium Ion-Selective Electrode, 2023) NOTE: The USB connection should be utilized solely for charging purposes. A Bluetooth connection is necessary for calibration and experimental measurement because the Liquid pH Sensor induces voltage interference when the pH sensor and the USB-connected Ammonium Electrode are immersed in the same liquid. (DiBeneditto, 2023)
Liquid pH Sensor	Variable(s): DataSet1pH Vernier PH-BTA pH Sensor . (pH Sensor - Vernier, 2023) Wired Connection.
Liquid Temperature Sensor	Variable(s): DataSet1TemperatureC Vernier TMP-BTA Stainless Steel Temperature Probe . (Stainless Steel Temperature Probe, 2023) Wired Connection.

§ 2.1.5. Additional Related Hardware and Software

Table 5. Additional Related Hardware and Software.

Item	Details
Atomic Wristwatch	Casio PRO TREK MULTIFIELD LINE PRW3500T-7. Manual Casio Men's Pro Trek PRW-3500T-7CR Tough Solar Triple Sensor Digital Sport Watch. (PRW3500T-7 Titanium Watch, 2023) Atomic time adjustment radio-controlled watch (DiBeneditto, 2023)
Google Docs	Google Docs , Google. Version: 2023. (Google Docs: Online Document Editor, 2023)
Google Sheets	Google Sheets , Google. Version: 2023. (Google Sheets: Online Spreadsheet Editor, 2023)
MATLAB Software	MATLAB Version: 9.13.0.2049777 (R2022b) , and MATLAB Version: 23.2.0.2380103 (R2023b) Update 1 . (MATLAB, 2023)
Richcopy	Richcopy , Microsoft, v.4.0.217.0, Link from MSDN archives containing old version of RichCopy 4. (Free Utility: RichCopy, 2016; Looking for RoboCopy GUI and RichCopy, 2023; RichCopy HoffmanUtilitySpotlight2009_04.exe Archive Copy, 2019)
Robocopy	Robocopy , Microsoft, v.10.0.19041.2546 (Free Utility: RichCopy, 2016; robocopy Microsoft Learn 2023)
Stopwatch	Capelli Sport CSEF-1014 Digital Chronograph 1/100 Second Stopwatch. (Amazon.com: Capelli Sport Digital Stopwatch Timer, 2023)
USB Hub	Vienon B09MLRPPT2 USB 3.0 Hub. VIENON 4-Port USB Hub USB Splitter USB Expander. (Amazon.com: USB 3.0 Hub, 2023)
Windows Laptop with Power Adapter	Windows 10 or Windows 11. The power adapter has features to minimize voltage spikes, aiding with connected voltage-sensitive data collection devices.

§ 2.2. Arduino Pressure Sensors Calibration

The Arduino Pressure Sensors play a vital role in the experimental setup, so ensuring their accuracy through calibration is paramount. In this context, calibration is an iterative process to reduce the systematic error in the device's readings. The following is the calibration procedure and the rationale behind it.

§ 2.2.1. Initial Arduino Pressure Sensor Calibration

Our Arduino pressure sensor calibration started in Monroeville, PA, on June 26, 2023, at 01:50:21 AM. To establish a reference point, we referred to the reported barometric pressure from Weather.gov at 29.9295 inHg. Converting this to absolute pressure (PSIA) by dividing the pressure value by 2.036 gives us a reference pressure of approximately 14.700 PSIA. We derived calibration statistics for each sensor using the reference barometric pressure and an iterative calibration approach.

§ 2.2.2. Arduino Device Iterative Calibration Approach

The following process was used to calibrate the Arduino device pressure sensors. We hoped to find and compensate for the innate pressure sensor bias by taking a systematic and iterative approach. The calibration process for determining the Arduino pressure sensor bias values was as follows:

1. As the baseline, set the custom Arduino code bias values to zero for psiaA0, psiaA1, and psiaA2.
2. Upload this code to the Arduino Device, which will reset the Arduino to start collecting data via the serial monitor.
3. After at least 15 minutes, disconnect the USB cord, which will pause the scrolling of data on the serial monitor and allow us to copy and paste the data into Google Sheets. Note that this was before we used PuTTY to record the data as text files on the Laptop. We have saved the Google Sheets Arduino Device logs as CSV files in the “device-calibration” folder for audit purposes.

4. In Google Sheets, calculate the mean pressure for psiaA0, psiaA1, and psiaA2 based on the copied and pasted data just recorded.
5. For each pressure sensor, subtract the mean pressure from the reference barometric pressure to get the new values to update the custom Arduino code bias values for psiaA0, psiaA1, and psiaA2.
6. The process repeats once we upload the code to the Arduino Device.

We repeated this process four times with increasing recording length to gather the final Arduino pressure sensor bias cumulative values.

§ 2.2.3. Final Arduino Device Calibration for Pressure Sensor Bias

For the final Arduino code, we implemented the cumulative correction values. Every sensor is flawed, and manufacturing variances can introduce systematic errors. We adjusted for these errors by applying these corrections, making our device as accurate as possible.

Statistics for the three Arduino Device pressure sensors, with standard deviation (SD) and sample size ($N = 57088$) representing the number of rows collected in the data:

- **psiaA0:** 14.7150 ± 0.0149 PSIA
- **psiaA1:** 14.7101 ± 0.0099 PSIA
- **psiaA2:** 14.7088 ± 0.0087 PSIA

Table 6. Arduino Pressure Sensor Bias. The final values are highlighted in yellow.

<u>trial</u>	<u>samples (N)</u>	<u>timeMS</u>	<u>psiaA0</u>	<u>psiaA1</u>	<u>psiaA2</u>
0			0	0	0
1	5121	1058620	0.1699803226	0.1683831151	0.2408455638
2	5158	1067732	0.0014820435	0.0003188213	-0.0078802675
3	5356	1108689	-0.0022986920	-0.0015509325	0.0133464723
4	41453	8577972	-0.0003176177	-0.0069798472	-0.0185387476
Sum Total	57088	11813013	0.1688460564	0.1601711567	0.2277730210

The final Arduino pressure sensor bias values (highlighted in yellow from the table above and below) were then saved in the final Arduino code stored on the Arduino device and used in all the trials. (DiBeneditto, 2023):

```
// these are the averaged correction values
float correctionA0 = -0.1688460564;
float correctionA1 = -0.1601711567;
float correctionA2 = -0.2277730210;
```

These above values are negative because of how they are used in calculations in the custom Arduino code.

§ 2.3. Device Accuracy and Timing Synchronization Considerations

When dealing with data from multiple devices that start and stop recording at different times and have different sample rates, synchronizing the data is a significant challenge. The plan was to collect the data on the different devices and then synchronize the data with MATLAB. The following sections will discuss how we resampled all device logs to 0.5 second intervals. While a 0.5 second interval provides resolution, the synchronization accuracy is not implicitly derived

from this resolution. Therefore, it is essential to discuss device sampling intervals and synchronization accuracy.

§ 2.3.1. Pretrial Synchronization Procedure and Device Data Log File Formats

In anticipation of having to synchronize the data logs from the four different devices for each trial with MATLAB, we initiated a procedure to aid in increasing the accuracy of the data collection, which consisted of a **daily pretrial atomic clock synchronization procedure** (DiBeneditto, 2023), (bold and numbers added for clarity):

1. Confirm and adjust the Date and Time on the LabQuest (**Vernier Device**) to match the Atomic Wristwatch within ± 1 second, accounting for the time drift on the LabQuest.
2. This time should match the time on the Laptop, synchronized with the time.microsoft.com server, within ± 1 second.
3. Synchronize the Stopwatch to the Atomic Wristwatch within ± 1 second.

Each data recording device would create a file which contained the data:

- The Arduino and Omega device was monitored via USB serial connection in real-time and saved the tab-separated value (TSV) or text file (TXT) to the Laptop when data collection stopped with PuTTY software. PuTTY software was configured to log the data collection start time in the device log file.
- The ToxiRae device recorded data on the device and had to be accessed after the fact (usually the next day) with the Honeywell Safety Suite Device Configurator Software to

be exported from the ToxiRae device to a comma-separated value (CSV) file and saved on the Laptop.

- The Vernier device was monitored in real time via USB and Bluetooth. Data was saved on the Laptop as a “gaml” file and then exported to a CSV file with Vernier Graphical Analysis Software.

The Laptop, via the Windows Operating System, is synchronized within ± 1 second of atomic clock-derived Coordinated Universal Time (UTC) via the time.microsoft.com time server. We also confirmed that within ± 1 second, this time matched the wristwatch and Stopwatch used for data collection. When saved to the Laptop, the device data log files by extension of the Windows Operating System are timestamped to within ± 1 second of atomic clock-derived Coordinated Universal Time (UTC).

Using the “Created” and/or “Modified” date and time from the Windows file properties allows us to get the start or stop time for the Arduino, Omega, and Vernier devices, accurately recording data within ± 1 second of UTC. The filenames also contained general date and time start recording times (discussed later). The ToxiRae device synchronized with the Honeywell Safety Suite Device Configurator Software and the Laptop time. Then, we double-checked the ToxiRae device with the Atomic Wristwatch before starting trials, resulting in log accuracy to within ± 1 second of UTC.

§ 2.3.2. Device Accuracy Time Drift Compensation

In the context of device accuracy, “drift” refers to the unintended and often gradual deviation or change in a measurement over time. This deviation can be due to various factors, such as environmental conditions (temperature), aging components, manufacturing defects, or other internal or external influences (crystal oscillator tolerance and stability, power supply instabilities, analog to digital converter limitations). Drift implies that the device’s output or reading changes over time, even if the input or actual measured value remains constant; hence, for accurate results, devices often require recalibration or compensation over time to account for this drift.

Atomic Wristwatch:

- Utilized for LabQuest calibration and trial start and stop times.
- **Without nightly signal calibration:** Drift = ± 0.5 seconds/day. Drift over 2 hours: $(0.5 \text{ seconds}/24 \text{ hours}) * 2 \text{ hours} = 0.0417 \text{ seconds}$. (DiBeneditto, 2023)

Arduino Device:

- **Serial Connection Reset:** When the Arduino Device connects via serial communication tools like PuTTY, the Arduino Device will restart. This inherent behavior in many Arduino boards triggers the `setup()` function, resetting its internal timers and variables. If the Arduino uses the built-in `millis()` function for time tracking, this counter resets to zero with each connection. We restarted the PuTTY log for most of the trials, which restarted the Arduino Device and internal clock, minimizing concerns for the two-hour worst-case drift scenario related to the `timeMS` variable calculation. (DiBeneditto, 2023)

Omega Device:

- **Configuration:**

- **Model:** Omega FMA-1618A.
- **Communication:** 19200 baud.
- **Connection:** Omega FMA-1618A → RS-232 adaptor → USB.
- **OS:** Windows 11.
- **Software:** PuTTY.
- **COM Port:** COM7.
- **Speed:** 19200 (baud).

- **Omega Calibration Data Log:**

- For audit purposes.
- **Filename:** “omega-20230628155736.txt”.
- **Start Time:** Wednesday, June 28, 2023, at 3:57:36 PM.
- **End Time:** Wednesday, June 28, 2023, at 6:33:13 PM.
- **Duration:** 9337 seconds or 2.594 hours.
- **Total Samples (n):** 18760 lines
- Therefore, 18670 lines/9337 seconds

$$\approx 1.99957 \text{ lines per second} \approx 2 \text{ lines per second}$$

- See folder “device-calibration” and files labeled “arduino pressure calibration” for data audit.

Vernier Device:

- **Daily Observed Drift:** ± 8 seconds.
- **For a 2-hour duration:** $(8 \text{ seconds}/24 \text{ hours}) * 2 \text{ hours} = \pm 0.6667 \text{ seconds}$
- The Vernier Device has the feature to connect to GPS satellites to synchronize the internal clock automatically. (LabQuest 3, 2023) However, we could not get this feature to work under the Fume Hood in the Biomedical Engineering Laboratory.

§ 2.3.3. Synchronization Summary

Considering the potential drifts and accuracies of all the devices:

- The Arduino is the primary source of potential drift over a 2-hour duration, with its worst-case scenario presenting a drift of up to 36 seconds. However, given that after each trial, the PuTTY log is restarted, resetting the Arduino's internal clock, this 2-hour worst-case scenario is mitigated, alleviating major concerns regarding data accuracy over longer durations.
- The Vernier Device has a drift of about 0.6667 seconds over 2 hours.
- The Atomic Wristwatch, which serves as a calibration reference, shows a nearly negligible drift.

In a comprehensive assessment, we estimate the data to be accurate within approximately ± 1 second over 2 hours per trial. The Arduino presents the most significant potential drift in its worst-case scenario. However, as long as users correctly calibrate the devices and follow operational guidelines, the measurements will likely maintain this level of accuracy.

§ 2.4. Device Sample Rates

The preparation of data for the analysis process consisted of several phases, which included the creation of an experimental variable spreadsheet “meta.csv” (located in the leading working directory), organization of raw files recorded by the four devices into specific folders, file conversion into text or comma-separated values (CSV) formats when necessary, minimal data cleanup for MATLAB import, MATLAB data import, and synchronization of data for each trial.

We matched the data collection sample rate for all the devices as closely as possible. Additional details on each device’s configuration appear in the tables provided earlier. We manually configured the data collection sample rate for each device as follows:

- **Arduino Device:** Configured via custom code to 0.5 Hz (2 samples per second)
- **Omega Device:** Set via a serial command to 0.5 Hz (2 samples per second)
- **ToxiRae Device:** Configured via Honeywell Safety Suite Device Configurator Software to 1 Hz (1 sample per second)
- **Vernier Device:** Configured via Vernier Graphical Analysis Software in conjunction with the LabQuest to 0.5 Hz (1 sample per second)

§ 2.5. Trial Start and Stop Times

Before writing the MATLAB code, we created a spreadsheet using Google Sheets to keep everything organized and to develop the “import procedure” for all the many different device files.

For transparency and audit purposes, the finalized version of this spreadsheet is in the working directory under the filename “everything.csv”. Initially, the “everything.csv” spreadsheet did not contain all the data the final version now has. An additional goal of this spreadsheet was to narrow in on the exact start and stop times for each trial. To achieve this, we began by incorporating all the immediately available data:

- Written experimental variables and journal observations, with some findings already presented in the thesis.
- Data from individual device log files, especially the “Created” and “Modified” timestamps from Windows file properties. Notably, file copying can sometimes change the “Modified” date. While tools like Microsoft’s “Robocopy /mir” or Richcopy can address this, they were not used in all circumstances to copy the data into the trial folders.
- Individual device log files have the start date and time in the device log filenames, following the format “YYYYMMDDHHmmss”. As an illustration, the filename “omega-20230630155836.txt” indicates an Omega Device log file that began recording on 2023 Jun 30 at 3:58:36 pm (HH:mm:ss).
- The stop time for the Arduino and Omega devices was calculated from the start time, the count of recorded lines, and an acceptable sample rate of 0.5 second interval per line. Here is the basic formula: $\text{start time} + (\text{number of lines} \times 0.5 \text{ seconds}) = \text{stop time}$
- Furthermore, the Arduino Device log file provides an internal clock variable labeled “timeMS”. By comparing the value of this variable from the first and last data lines, we gained insight into the sample rate. Using MATLAB code, the sample rate for the Arduino Device was confirmed to be 504.81 ± 0.47 ms (occurrences per second) across

all trials, which consisted of 100,062 lines or samples, which provides an alternative method to verify the Arduino Device sample rate.

Some of this information from the “everything.csv” spreadsheet made its way to the “meta.csv” spreadsheet, used in MATLAB code to find the exact start and stop times for each trial.

Table 7. Initial trial conditions from the “meta.csv” file. NOTE: Trial 8 is the control.

trial	<u>volumeRODIWaterML</u> Volume of RODI Water (mL)	<u>volumePFCML</u> Volume of PFC (mL)	<u>volumeNH4OHUL</u> Volume of Ammonium Hydroxide (NH4OH) (uL or µL)	<u>volumeNH4OHUMOLL</u> Volume of Ammonium Hydroxide (NH4OH) (umol/L or µmol/L)
1	100.00	0.00	1515.00	222387.00
2	100.00	0.00	15.14	2222.40
3	100.00	0.00	22.70	3333.60
4	100.00	0.00	30.28	4444.80
5	100.00	0.00	37.85	5556.00
6	100.00	0.00	45.42	6667.20
7	100.00	0.00	52.99	7778.40
8	100.00	0.00	0.00	0.00
9	100.00	0.00	15.14	2222.40
10	100.00	0.00	1514.00	222240.10
11	100.00	0.00	151.40	22224.00
12	100.00	0.00	151.40	22224.00
13	100.00	0.00	151.40	22224.00
14	50.00	50.00	757.00	222240.06
15	50.00	50.00	757.00	222240.06
16	50.00	50.00	757.00	222240.06

§ 2.6. Gather and Organize Raw Files Recorded by Four Devices

During data collection, each device log file was stored in a specific folder named after that device, e.g., Arduino Device logs were in the “arduino” folder, and Omega Device logs were in the “omega” folder. Before running the MATLAB program, “main.m”, 16 trials were conducted. The ToxiRae Device created daily log files, so trials on the same day had identical ToxiRae files. Additionally, some trials utilized a single Arduino Device log file since the system generated only one log file across multiple trials. Trials 2 through 7 all employed the same Arduino and Omega PuTTY logs, so there are duplicate files in different trials. We included four raw device log files for auditing purposes in each trial’s “0-raw” folder.

§ 2.7. Data Preparation for Trial Synchronization

The procedure detailed in this section was applied uniformly to all trial folders, creating four preprocessed files in the “1-preprocessed” folder for each trial. The following highlighted text helps delineate variations in file content. We cleaned any residual PuTTY serial communication logs from prior recording attempts stored in the serial cache for the Arduino and Omega Device log files. Also, we eliminated any lines cut off due to an early disconnection from serial communication before registering a new line character.

For illustrative purposes, we will use Trial 1, i.e., “**trial1**” folder, as a representative example below:

Arduino Device log file, clean and change the filename to “arduino.tsv” (tab-separated values).

```
From: <current_working_folder>\_trials\trial1\0-raw\arduino-pressure-20230630155832.log
===== PuTTY log 2023.06.30 15:58:32 =====
4543    14.7490 14.6813 14.6516
5048    14.7192 14.6700 14.6561
:
14639   14.7640 14.7014 14.6600    <-- note from prior recording attempt
151                                           <-- note the truncation

Arduino Uno Pressure Transducer Code, (c) 2023, Lukas W. DiBeneditto, v1.21
timeMS  psiaA0 psiaA1 psiaA2    <-- note header
1010    14.7539 14.6768 14.6676    <-- note data start
1515    14.6860 14.6620 14.6846
2020    14.7608 14.6764 14.6865
:

To: <current_working_folder>\_trials\trial1\1-preprocessed\arduino.tsv
Arduino Uno Pressure Transducer Code, (c) 2023, Lukas W. DiBeneditto, v1.21
timeMS  psiaA0 psiaA1 psiaA2    <-- note header
1010    14.7539 14.6768 14.6676    <-- note data start
1515    14.6860 14.6620 14.6846
2020    14.7608 14.6764 14.6865
:
```

Omega Device log file, clean and change the filename to “omega.txt” (text file).

```
From: <current_working_folder>\_trials\trial1\0-raw\omega-20230630155836.txt
===== PuTTY log 2023.06.30 15:58:36 =====
+014.24 +025.04 +000.00 +000.00    Air    <-- note data start
:
+005.46 +024.92 +255.99 +179.15    Air VOV    <-- note different status
:

To: <current_working_folder>\_trials\trial1\1-processed\omega.txt
<no changes to contents, unless serial communication artifacts>
```

ToxiRae Device log file, clean and change the filename to “toxirae.csv” (comma-separated file).

From: <current_working_folder>_trials\trial1\0-raw\G024017649_DataLog_1_202306301949526956.csv
Device Serial No,Log Time,Log Type,Log Interval,Sensor 1 Type,Sensor 1 Display
Unit,Sensor 1 Serial Number,Sensor 1 Status,Sensor 1 Gas Reading,Sensor 1 STEL
Reading,Sensor 1 TWA Reading,Sensor 1 Last Cal,Sensor 1 Span Setpoint,Sensor 1 High
Alarm,Sensor 1 Low Alarm,Sensor 1 STEL Alarm,Sensor 1 TWA Alarm,Sensor 1 Overrange
Threshold,Unit Status,Running Mode,Log Start Type,Diagnostic Mode,Stop Reason,User
Id,Site Id,Record Number,Session Start Time,Session Stop Time,Firmware Version
G024017649,6/30/2023 7:07:07
PM,Readings,,NH3,,SC03500053D4,Normal,0.0,0.0,0.0,,,,,,,,,,,,,
:
G024017649,6/30/2023 5:14:21
PM,Readings,,NH3,,SC03500053D4,Normal,0.0,1.0,1.0,,,,,,,,,,,,,
G024017649,6/30/2023 5:14:21
PM,CONFIG,1,NH3,ppm,SC03500053D4,,,,,6/13/2023,51.0,50.0,25.0,35.0,25.0,100.0,,,Normal
Mode,Event Full,USER0000,SITE0000,3600,6/30/2023 5:14:21 PM,6/30/2023 6:14:21 PM,V1.86
:

To: <current_working_folder>_trials\trial1\1-processed\toxirae.csv
Device Serial No,Log Time,Log Type,Log Interval,Sensor 1 Type,Sensor 1 Display
Unit,Sensor 1 Serial Number,Sensor 1 Status,Sensor 1 Gas Reading,Sensor 1 STEL
Reading,Sensor 1 TWA Reading,Sensor 1 Last Cal,Sensor 1 Span Setpoint,Sensor 1 High
Alarm,Sensor 1 Low Alarm,Sensor 1 STEL Alarm,Sensor 1 TWA Alarm,Sensor 1 Overrange
Threshold,Unit Status,Running Mode,Log Start Type,Diagnostic Mode,Stop Reason,User
Id,Site Id,Record Number,Session Start Time,Session Stop Time,Firmware Version
G024017649,6/30/2023 7:07:07
PM,Readings,,NH3,,SC03500053D4,Normal,0.0,0.0,0.0,,,,,,,,,,,,,
:
G024017649,6/30/2023 5:14:21
PM,Readings,,NH3,,SC03500053D4,Normal,0.0,1.0,1.0,,,,,,,,,,,,,
G024017649,6/30/2023 5:14:21
:
:

Vernier Device log file, open the “gambl” file using the Vernier Graphical Analysis Software and export it to a comma-separated value (CSV) “verner.csv” file, confirm and reorder in Google Sheets if needed, then clean.

```
From: <current_working_folder>\_trials\trial1\0-raw\vernier-202306301537.gambl
vstudm\data.udm
664                      3606341                      5612
vstudm
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <UDMFileFormat>2</UDMFileFormat>
  <Version>5.16.0-2915</Version>
  <charset>utf-8</charset>
  <ApplicationBuildDateTime>07-03-2023</ApplicationBuildDateTime>
  <Copyright>Copyright (c) 2007-2023 Vernier Software & Technology and its
  licensors</Copyright>
  <FileName>vernier-202306301537.gambl</FileName>
  <CreatorOS>2</CreatorOS>
  <CreatorApp>4</CreatorApp>
  <AbsoluteVersion>5.16.0-2915</AbsoluteVersion>
  <DataWorld>
    <DataWorldRunPrefix>Run</DataWorldRunPrefix>
    <MatrixDefaultPref>0</MatrixDefaultPref>
    <UDMSessionSubtype>0</UDMSessionSubtype>
    <UDMReplayDataSetId>0</UDMReplayDataSetId>
    <UDMReplayActive>0</UDMReplayActive>
    <UDMReplayRate>1</UDMReplayRate>
    <UDMExperimentName>vernier-202306301537.gambl</UDMExperimentName>
  </DataWorld>
  ;

To: <current_working_folder>\_trials\trial1\1-processed\vernier.csv
Data Set 1:Time(s),Data Set 1:Temperature(°C),Data Set 1:Potential(mV),Data Set
1:Ammonium(mg/L),Data Set 1:pH
0,21.923534666276623,-468.17596435546875,50000,11.180719519528797
0.5,21.923534666276623,-468.17596435546875,50000,11.180719519528797
1,21.923534666276623,-468.17596435546875,50000,11.174942128331171
1.5,21.952719810554754,-468.17596435546875,50000,11.180719519528797
;
```

After synchronizing the device logs using MATLAB, we identified segments where all four devices recorded data simultaneously. Based on this analysis, the start and stop times in the “everything.csv” and “meta.csv” files were adjusted to ensure accuracy. This refinement guarantees that subsequent runs of the MATLAB code will utilize the most precise start and stop times, ensuring a comprehensive and gap-free data set.

§ 2.8. Trial and Phase Overlap

Each trial had three different phases:

- **Phase 1:** *(The start of Phase 1 is the start of the Trial.)* The roller pump circulated fluid through the fluid loop while the vacuum pump remained off. This setup supplied only the regular dry lab air to the Gas-Pathway. Before starting Phase 2, the Gas Ammonia (NH₃) Sensor's digital display must read zero for two to three minutes.
- **Phase 2:** With the roller pump on, we manually activated the vacuum pump, creating a vacuum pressure in the Gas-Pathway up to the outlet, as this is the main phase we are interested in analyzing.
- **Phase 3:** While the roller pump stayed on, the operator turned off the vacuum pump. This phase resembles Phase 1 but follows the completion of Phase 2. At this stage, residual ammonia gas might be present in the Gas-Pathway. *(The stop of Phase 3 is the stop of the Trial.)*

The table below will use Trial 8 to demonstrate the overlap:

Table 8. Trial and Phase Overlap Example for Trial 8.

Trial	Phase	MATLAB Variable	Date and Time Value
Trial Start Date and Time	Phase 1 Start Date and Time	phase1VacOffBeg	2023-07-15 20:42:07.000
	Phase 1 Stop Date and Time	phase1VacOffEnd	2023-07-15 20:44:59.500
	Phase 2 Start Date and Time	phase2VacOnBeg	2023-07-15 20:45:00.000
	Phase 2 Stop Date and Time	phase2VacOnEnd	2023-07-15 20:55:02.000
	Phase 3 Start Date and Time	phase3VacOffBeg	2023-07-15 20:55:02.500
Trial Stop Date and Time	Phase 3 Stop Date and Time	phase3VacOffEnd	2023-07-15 20:58:09.000

§ 2.9. The Vacuum Pump & Gas Pressure Determine Phase Start and Stop Times

In determining the start and stop times for each phase within every trial, a manual analysis was conducted on the “absolutePressurePSIA” data values, representing the Gas-Pathway pressure data measured by the Omega Device and, by extension, the result of turning on and off the Vacuum pump. Given the established trial start (beginning of Phase 1) and stop times (end of Phase 3), an analysis of this pressure data revealed a sudden pressure decrease, indicating the activation of the vacuum pump and an abrupt pressure increase, signaling the deactivation of the vacuum pump. These shifts mark the beginning and termination of Phase 2, respectively. By shifting 0.5 seconds before the start of Phase 2, we pinpointed the stop of Phase 1. Similarly, advancing 0.5 seconds after the stop of Phase 2 provided the start of Phase 3. Based on the rapid pressure changes, this method facilitated the accurate identification of start and stop times for each phase across all trials down to 500th of a second.

§ 2.10. Removal of Unused Data

The two pieces of data that we wanted from the ToxiRae Device were the date and time (“LogTime” which eventually becomes “dateTime”) when the device recorded the ammonia level and actual ammonia level (“Sensor1GasReading”) as measured by the Gas Electrochemical Sensor in the Gas Ammonia (NH₃) Sensor. We discarded the remaining data from the ToxiRae Device log.

§ 2.11. Synchronization of Device Logs by Matching Date, Time, and Sample Rates

With the established recording start times for each device, periods of simultaneous recording across all devices, and the known sample rates, we resampled each device’s data to 0.5-second intervals. Specifically, we upsampled the ToxiRae log from its original 1-second interval to 0.5 seconds, effectively aligning each device’s data into 500-millisecond interval buckets using linear interpolation. For accuracy and auditing purposes, we exported the adjusted data as comma-separated value (CSV) files and stored them in the “2-processed” folder within the respective trial folder for each device.

§ 2.12. Combine and Synchronize Data For All Devices For All Trials

The primary goal of data preparation is to synchronize all device recordings for each trial. First, each trial’s device log was sorted by its “dateTime” value, ensuring chronological order. Then, we confirmed that there were no gaps in this chronological order.

The MATLAB synchronize function was crucial in ensuring that the synchronized data set encompassed all unique time points from every device. We used the “union” method, resulting in a time vector that includes all distinct time points from all input time series or timetables. The “union” method guarantees that every data point is considered during synchronization, and each resulting time point consistently holds data (or NaN if data is unavailable) from every device.

(Synchronize, 2023)

Using this approach, we combined the data from four distinct files (one for each device log) into a single synchronized file. For accuracy and auditing purposes, we exported the adjusted data as

comma-separated value (CSV) files and stored them in the “3-synchronized” folder within the respective trial folder for each device.

§ 2.13. Final Cleanup to Prepare for Statistical Analysis

Several cleanup steps ensured the data’s integrity and reliability for statistical analysis:

- **Trial Duration Alignment:** We trimmed the data to match the duration of each experiment, specifically from the trial’s start to its end, which corresponds to the span from the start of Phase 1 to the end of Phase 3.
- **Voltage Correction for Trial 1:** A malfunctioning Liquid Ammonium (NH_4^+) Sensor, subsequently replaced for the remainder of the trials, produced inaccurate voltage readings “DataSet1PotentialmV” that affected the temperature sensor. We identified and corrected these readings by setting them to “NaN”.
- **NaN Value Replacement:** We replaced any “NaN” values in the “Sensor1GasReading” for Trial 1 with zero.
- **Range Validations:**
 - We adjusted values that fell outside the acceptable range for the Liquid Ammonium (NH_4^+) Sensor. Specifically, adjustments pertained to values in “DataSet1AmmoniummgL” outside the range of 1 to 18,000 mg/L (or ppm) and values in “DataSet1PotentialmV” outside the range of 0 to 116 mV.
 - We set out-of-range pH and temperature values to “NaN”.

In practice, the out-of-range values arose during the addition of ammonium hydroxide. Typically, these deviations occurred when we unscrewed the vortex mixing tank's lid (VMT) lid, exposing the Liquid Ammonium (NH_4^+), pH, and Temperature Sensors to air. For accuracy and auditing purposes, we exported the adjusted data as comma-separated value (CSV) files and stored them in the "4-experiment" folder within the respective trial folder for each device.

§ 2.14. Methods Summary

The data preparation phase was undertaken with meticulous attention to detail:

- We organized the raw data logs, making them systematically arranged.
- We prepared and synchronized the four device logs by matching date, time, and sample rates to 0.5 second intervals.
- We implemented necessary adjustments and corrections, from eliminating unused data to refining specific data values for clarity and reliability.
- We addressed anomalies and potential inconsistencies in the data, particularly those related to the Liquid Ammonium (NH_4^+) Sensor and overlaps or glitches in device logs.

Such a comprehensive approach to data preparation ensures that we have robust and reliable data for statistical analysis, which we will now detail in the next chapter.

CHAPTER 3: RESULTS & DISCUSSION

This chapter presents the results, intertwined with a discussion and the relevant statistical methods.

§ 3.1. Statistical Analysis Workflow and MATLAB Implementation

In any scientific study, especially those involving the collection of vast amounts of data from different devices and systems, statistical analysis must be considered. This analysis acts as the bridge between raw, unprocessed data and meaningful, actionable insights.

§ 3.1.1. Methods and Tools Employed

The statistical analysis undertaken for this study incorporates many methods, each chosen to fit best the nature of our data and the questions we seek to answer. The primary tool employed was MATLAB for data processing, statistical modeling, exploratory data analysis (EDA), and visualization. MATLAB has been integral in sifting through the vast data sets, ensuring they are consistent, clean, and prepared for more advanced statistical techniques.

§ 3.1.2. MATLAB Workflow Overview

The following steps were taken to prepare for comparing the three phases to determine if there was a significant difference in ammonia gas removal:

1. **Calculate Basic Statistics:** For each phase in each trial, a custom function, “descStat”, was used to get the basic statistics for each column variable consisting of the following:

- a. n: number of rows or samples
 - b. min: the minimum value
 - c. max: the maximum value
 - d. mean: the average
 - e. mode: the most frequent value
 - f. std: standard deviation (SD)
 - g. var: variance
 - h. rms: root mean square
 - i. sem: standard error of the mean (SEM)
2. **Export Each Phase Data and Statistics:** For accuracy and auditing purposes, we exported the data as comma-separated value (CSV) files and stored them in the “5-statistics” folder within the respective trial folder for each device. MATLAB saves NaN values next to numeric values and commas when saving the data in CSV format. We found it easier to let MATLAB save the CSV file, then reopen the file and have MATLAB remove “NaN” values as if it were processing a text file, which was determined to be the most efficient method.
3. **Data Integrity Check:** We did a simple check to confirm that all the trials, phases, and main variables, e.g., temperature, pH, pressure, and ammonia gas levels, were actually in the data for statistical analysis.
4. **Check and Count for NaNs for Sensor1GasReading Variable:** Not a Number (NaN) values will cause problems with some MATLAB statistics functions. We confirmed that there were none for the Ammonia gas levels represented in the Sensor1GasReading.

5. **Calculated Duration for Each Phase:** The duration for each phase was calculated and then combined into one table. We exported the data as comma-separated value (CSV) files for accuracy and auditing purposes and stored them in the main working directory folder with the filename “duration.csv”.
6. **Create a Statistics Summary Table for Phase 2:** Since phase 2 was the primary phase where the roller pump and the vacuum pump were on, it was the phase that was were most interested in the values since this phase represented the primary goal of the experiment of trying to determine if we could extract ammonia gas through a PDMS filter that the PFC picked up in a simulated peritoneal dialysis Liquid-Loop. We exported the data as comma-separated value (CSV) files for accuracy and auditing purposes and stored them in the main working directory folder with the filename “summary.csv”.

§ 3.1.3. The Need for Statistical Analysis

Our research collected data from four devices with unique calibration, sample rate, and precision. There needs to be more than plotting this data or providing averages to draw accurate conclusions or detect underlying patterns. Among other reasons, statistical analysis helps in several crucial aspects:

- **Identifying Patterns:** Statistics help us understand the distribution, variance, and possible correlations among measurements. This understanding is pivotal when exploring phenomena that might take time to be apparent.
- **Validity and Reliability:** We want to ensure that our findings are valid and are measuring what they are supposed to measure. We also want to ensure that our findings

are reliable and consistent across measurements. Through statistical tests and models, we can quantify the degree of confidence in our results, ensuring that they are not just coincidental but have a grounding in the processes we are studying.

- **Informed Decision Making:** Raw data, on its own, can often be misleading. However, we transform this data through the lens of statistics into information that can guide future trials, inform discussions, and drive the decision-making process.

While our devices provide the raw data, it is through statistical analysis that we can better understand this data, transforming it from mere numbers on a spreadsheet to insights.

§ 3.2. Data Normality Checks

Standardization and normalization of the data are pivotal to achieving consistency in our analysis across data derived from four distinct devices. Such standardization and normalization ensure that measurements, irrespective of which of the four devices produced the data, are interpreted on a comparable scale.

To assess the normality of the ammonia gas meter readings across various trials for each phase, we employed the Lilliefors test. The Lilliefors test is a statistical method specifically designed to test whether a sample comes from a normally distributed population, especially when the sample size is small, and the population's mean and variance are unknown. This test is a modification of the Kolmogorov-Smirnov (K-S) test, a widely recognized method for testing the goodness-of-fit of sample data to a specified distribution. The standard K-S test assumes that the population parameters are known. These parameters are often unknown in real-world scenarios, making the

K-S test less suitable for small sample sizes. The Lilliefors test addresses this limitation by enhancing the K-S test, making it more suitable for small sample sizes, and the population parameters are unknown. Because of these advantages, the Lilliefors test is frequently employed to evaluate the normality of samples in such situations. (Lilliefors, 2023; One-sample, 2023)

§ 3.2.1. Normality Tests and Preliminary Findings

Findings from the Normality Tests for Various Trials and Phases, and our preliminary investigations into the data indicated the following:

- Across trials, for each of the three phases, the ammonia gas readings from the Sensor1GasReading variable (Ammonia gas levels) were subjected to the Lilliefors test. The results consistently indicated that the readings for each phase across all trials were not normally distributed. For all trials:
 - Phase 1: The Sensor1GasReading is not normally distributed.
 - Phase 2: The Sensor1GasReading is not normally distributed.
 - Phase 3: The Sensor1GasReading is not normally distributed.
- Given the consistently non-normal data distribution across all trials and phases, non-parametric statistical tests such as the Friedman test might be more suitable for further analyses, as they do not operate under the normality assumption.

Following the normality checks, indicating that the Sensor1GasReading was not normally distributed, we transitioned to the Friedman test analysis, detailed in the next section.

§ 3.3. Friedman Test Analysis

Analysis of Variance (ANOVA) is a statistical technique designed to analyze the differences among group means in a sample. It determines if statistically significant differences exist between the means of three or more independent groups.

However, not all data sets adhere to the assumptions necessary for ANOVA, especially those concerning normal distribution and homogeneity of variances. In such cases, non-parametric statistical methods, which do not assume a specific distribution shape or form for the underlying population, are more suitable.

The Friedman test is a non-parametric procedure for comparing multiple paired groups. It is particularly valuable when the data does not fit the conditions for repeated measures ANOVA, such as when data deviates from a normal distribution or when the focus is on analyzing ranks instead of raw data values. The procedure works by ranking the data and then comparing these ranks across groups to discern if significant differences exist (Friedman's, 2023; Multiple Comparisons, 2023).

Our study used the Friedman test to determine whether significant differences occurred in ammonia gas readings across the three phases over numerous trials.

§ 3.3.1. Results from the Friedman Tests Comparing Different Phases

After organizing the data in a matrix format where columns represent phases and rows represent trials, the Friedman test was applied to each phase's Sensor1GasReading (ammonia gas levels in ppm). The following table details the variation (or sum of squares) due to the different phases and the Error or within-phase variation:

Table 9. Friedman Test to compare phase to Sensor1GasReading.

Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	7.125	2	3.5625	14.25	0.0008
Error	8.875	30	0.2958		
Total	16	47			

- **Source:** This refers to the source of variation. “Columns” is the variation due to differences among the phases. “Error” refers to the variation within the phases or the inherent variability of the data. “Total” is the sum of the column and error variations.
- **SS (Sum of Squares):** This represents the total variability in the data.
 - **Columns (Between groups):** 7.1250
 - **Error (Within groups):** 8.8750
 - **Total:** 16.0000
- **df (Degrees of Freedom):** This indicates the number of values in the final calculation of a statistic that are free to vary.
 - **Columns (Between groups):** 2
 - **Error (Within groups):** 30
 - **Total:** 47

- **MS (Mean Square):** The average variability is computed by dividing the SS by the df.
 - **Columns:** 3.5625 (which is 7.1250/2)
 - **Error:** 0.2958 (which is 8.8750/30)
- **Chi-sq (Chi-Square):** This is the test statistic for the Friedman test. A larger chi-square value can indicate stronger evidence against the null hypothesis. In this case, the chi-square value is 14.2500.
- **Prob>Chi-sq:** This represents the p-value. Given that the null hypothesis is true, it gauges the likelihood of encountering a chi-square value similar to our observed value. A lower p-value points to more compelling evidence against the null hypothesis. A p-value of 0.0008 is generally deemed highly significant, indicating a marked difference in ammonia gas readings (Sensor1GasReading) across the phases.

§ 3.3.2. Discussion of Friedman Test Analysis

The following are discussion points for the Friedman Test analysis:

- There is a marked difference in the Sensor1GasReading between Phases 1 and 2, which aligns with expectations as Phase 1 has the vacuum pump off while it is on in Phase 2.
- A significant variance was observed between Sensor1GasReading for Phases 1 and 3, which was surprising. Both Phase 1 and Phase 3 have the vacuum pump off. However, the “sticky” nature of ammonia gas means that when the vacuum pump gets turned off in Phase 2, residual ammonia remains, taking time to clear in the dry lab air before the subsequent trial.

- There were no reported significant differences between Phases 2 and 3 based on the Sensor1GasReading. This result was unanticipated, especially since the vacuum pump is active in Phase 2 but inactive in Phase 3. The residual ammonia gas might be a contributing factor.

The findings were further confirmed by a posthoc analysis using the Wilcoxon Signed-Rank Test. (Wilcoxon, 2023). After applying the Bonferroni correction to adjust the alpha level for multiple comparisons, pairwise evaluations pointed out significant differences between:

- Phase 1 and Phase 2
- Phase 1 and Phase 3
- However, no notable difference was observed between Phase 2 and Phase 3 in terms of their means.

The following section discusses Exploratory Data Analysis and will delve deeper into the dataset, analyzing patterns, anomalies, and relationships among variables to draw more insights from the data.

§ 3.4. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a fundamental step in data processing and analysis. It involves an initial investigation of data to uncover patterns, spot anomalies, test hypotheses, and check assumptions using summary statistics and graphical representations. (Exploratory Analysis, 2023) This approach provides a more in-depth insight into the dataset, revealing its

main characteristics and inherent structures. Such understanding serves as a foundational layer for subsequent hypothesis tests or modeling. In the context of this study, EDA proves invaluable by offering both visual and statistical perspectives on how ammonia readings vary across different phases and trials.

§ 3.4.1. Size Comparisons of Different Phases Across Various Trials

As part of the EDA process, the sizes of data available from each phase were analyzed for every trial. Here is a concise breakdown of the findings:

Table 10. Size Comparisons of Different Phases Across Various Trials.

Trial	Phase 1 Size	Phase 2 Size	Phase 3 Size
1	1846	5602	4574
2	1692	2613	859
3	859	652	920
4	920	736	734
5	734	973	863
6	863	566	586
7	586	152	443
8	346	1205	374
9	277	1203	319
10	326	1195	291
11	403	1203	199
12	206	723	112
13	284	1202	219
14	338	1200	262
15	532	1200	153
16	419	1195	185

The table delineates the dataset size for each phase across various trials, offering insights into the distribution of ammonia readings. In this context, “size” signifies the number of data points (or

readings) gathered during each phase of every trial. A higher number suggests more data points were collected, offering a richer, more detailed dataset, which can lead to more reliable results. Conversely, a lower number denotes fewer data points, representing a shorter observation duration, reduced sampling rate, or potential issues during data collection. These data sizes shed light on the volume of data available for analysis from each trial phase. Recognizing these variations assist in discerning if the data size remains consistent across trials or if discrepancies warrant attention.

The following section discusses correlation analysis and will identify how different variables in the dataset might be interrelated, implying a potential underlying relationship or pattern that could be explored further.

§ 3.5. Correlation Analysis

In any data-driven study, understanding the relationships between different parameters is paramount. Correlation analysis aids in determining the linear association between two variables. (Linear or rank correlation, 2023) Within the scope of our investigation, it is crucial to discern the correlation between Gas-Pathway volumetric flow rates and ammonia levels, as this could highlight the efficacy of the gas removal system or potential underlying mechanisms.

§ 3.5.1. Interpreting Correlation Analysis

Correlation coefficients are metrics used to determine the strength and nature of a linear relationship between two variables. These coefficients range from -1 to 1.

Positive Correlation (Close to 1): When the coefficient is near 1, it indicates a strong positive relationship. For instance, if the amount of ammonium hydroxide introduced correlates with higher ammonia gas readings during a particular phase of the experiment, there is a positive correlation. As the amount of ammonium hydroxide increases, the ammonia gas readings also tend to rise.

Negative Correlation (Close to -1): A coefficient near -1 indicates a strong negative relationship. Suppose a particular treatment or condition in a phase leads to decreased ammonia gas readings as the concentration of another agent rises. This relationship would demonstrate a negative correlation.

No or Weak Correlation (Around 0): A coefficient value near 0 means the two variables have little to no linear relationship. In the context of the experiment, a particular condition or variable does not have a predictable effect on the ammonia gas readings.

Understanding that correlation measures association but does not prove causation is crucial. A correlation between two variables doesn't necessarily mean that changes in one lead to changes in the other.

§ 3.5.2. Results of Correlation Analysis

The following are the correlation results:

- For phase 1, the correlation between the volumetric flow rate and ammonia level is NaN, meaning not a number, and suggests no valid correlation could be computed, likely because no ammonia was detected in phase 1.
- For phase 2, the correlation value is 0.28978, indicating a weak positive linear relationship between the volumetric flow rate and ammonia level, which suggests that as the flow rate of phase 2 increases, the ammonia level tends to rise, albeit weakly. This weak relationship emphasizes that while flow rate does play a role in ammonia levels, it alone might not be the sole influential factor.
- For phase 3, the correlation is again NaN, underscoring that no ammonia was detected, in line with experimental expectations.

While there is a discernible positive correlation in phase 2 between flow rates and ammonia levels, its strength is weak. Further studies might be required to understand other contributing factors or refine the experimental setup to accentuate this relationship.

§ 3.6. Linear Regression Modeling

Linear regression modeling is a statistical method used to investigate and model the relationship between two variables. With linear regression modeling, we are trying to fit a straight line (a linear equation) to a set of data points that minimizes the differences (residuals) between the observed values (actual data points) and the values predicted by the model. The dependent or

response variable is what we are trying to predict. The independent or predictor variable is the input. The primary output of linear regression is the regression equation, given typically in the form:

$$y = \beta_0 + \beta_1 x$$

Where:

- y is the dependent variable predicted, i.e., the estimated value based on the regression model.
- x is the known value of the independent, i.e., the predictor variable.
- β_0 is the y-intercept and represents the value of y when x is 0.
- β_1 is the slope of the regression line, which is the average change in y for a one-unit increase in x .

In the context of our study, we use linear regression modeling to interpret how the Gas-Pathway volumetric flow rate potentially impacts ammonia levels. By understanding the regression equation and examining the coefficients, we gain insight into the magnitude and direction of influence. For example, a positive slope (β_1) would suggest that as the Gas-Pathway volumetric flow rate increases, so do ammonia levels. The proportion of variance depicts how much of the total variation in the dependent variable is explained by one or more independent variables. Metrics like R-squared shed light on the model's explanatory power by characterizing the percentage of variance in the dependent variable as predicted by the independent variable.

Phase 1: First Linear Regression Model (lm1)

- **Equation:** $y \cong 1 + x_1$
- **Coefficients:** Both the intercept and the slope (x_1) are 0.
- **Interpretation:** In Phase 1, the model suggests no discernible relationship between the Gas-Pathway volumetric flow rate and ammonia levels. We can determine this by the coefficients equating to zero and the R-squared being NaN (Not a Number), which may indicate no variation in the dependent or independent variable during this phase. This result is expected since the vacuum pump is off during Phase 3.

Phase 2: Second Linear Regression Model (lm2)

- **Equation:** $y \cong 1 + x_2$
- **Coefficients:** The intercept is approximately 0.20101, and the slope (x_2) is about 0.027905.
- **Interpretation:** In Phase 2, for each unit increase in the Gas-Pathway volumetric flow rate, there is a rise of 0.027905 in ammonia levels. However, the relationship is not statistically significant due to a p-value of 0.27631, exceeding the conventional 0.05 threshold. The R-squared value, being 0.084, indicates that the model accounts for about 8.4% of the variance in ammonia levels. While a positive correlation exists between the flow rate and ammonia levels, other factors not encompassed in this model also play a role. This result is expected since the vacuum pump is on during Phase 2.

Phase 3: Third Linear Regression Model (lm3):

- **Equation:** $y \cong 1 + x_3$
- **Coefficients:** The intercept is approximately 1.3862, and the slope (x_3) is about 0.27662.
- **Interpretation:** In Phase 3, the intercept is 1.3862, the slope (x_3) is 0.27662, and the model represents a positive correlation between the Gas-Pathway volumetric flow rate and ammonia levels. However, the predictor p-value of 0.5557 does not meet statistical significance. Furthermore, with an R-squared of 0.0274, the model captures a mere 2.74% of the variance in ammonia levels, hinting at a relatively weak correlation. This result is expected since the vacuum pump is off during Phase 3, likely due to residual sticky ammonia gas in the Gas-Pathway.

In the context of the experiment, these regression models provide insights into the relationship between flow rates and ammonia levels across different phases. Specifically, the relationship is weak for Phase 2 despite a positive association between flow rate and ammonia levels. For Phases 1 and 3, the models could not capture a significant relationship, possibly due to the data's inherent characteristics or lack of sufficient variation due to close to zero values of ammonia gas in the Gas-Pathway.

§ 3.7. Analysis of Ammonia Readings Based on Presence of PFC

This analysis aims to evaluate whether the presence of PFC affects ammonia readings. We use Trial 8 to control and compute the average ammonia reading from its phase 2 sensor gas readings.

The Trials were divided into two groups:

- Trials without PFC: Trials 1 to 13
- Trials with PFC: Trials 14 to 16

We compute the mean ammonia reading from phase 2 sensor gas readings for each trial and adjust it using the control reading (Trial 8), which helps neutralize any inherent biases or systematic errors in the readings. We used two-sample t-tests to determine whether the two groups' differences were statistically significant. (Two-sample t-test, 2023)

Adjusted Average Ammonia Readings:

- Without PFC: 1.771
- With PFC: 2.2976

Statistical Significance:

- p-value: 0.84248

The p-value is much greater than the commonly used significance threshold of 0.05, which indicates that the observed differences between the two groups could have easily occurred due to random chance. **Therefore, based on this data, introducing PFC does not significantly impact ammonia readings, at least as measured in these trials.**

Given the findings, we can infer that while there is a numerical difference in the adjusted ammonia readings between trials with and without PFC, this difference is not statistically significant. It is important to consider other factors or experimental conditions that might have influenced the results. Furthermore, further studies with a larger sample size or different experimental conditions might be necessary to determine the impact of PFC on ammonia readings conclusively.

§ 3.8. Phase 2 Focus

In the following sections, we will concentrate on Phase 2 to present the findings from our experiment. We chose Phase 2 as the focus of our experiment since this is the period when both the roller pump and the vacuum pump are active. In contrast, during both Phase 1 and Phase 3, only the roller pump operates while the vacuum pump remains off.

Phase 2 most accurately represents the data we have the greatest interest in. Although it is feasible to generate a pressure gradient in this setup without the vacuum pump, the most significant pressure difference occurs when both the roller pump and the vacuum pump are functioning, which is during Phase 2.

We use tables and graphs to clarify various parameters and their inter-relationships, which refer to the connections, associations, or dependencies between different parameters or variables.

- **For tables**, we use the **Standard Deviation (SD)** to indicate the spread of individual data points around the mean, helping to clarify the inherent **variability** in the raw data.
- **For graphs**, we use the **Standard Error of the Mean (SEM)** to convey the precision of our sample mean as an estimate of the possible population mean, underlining the **reliability** of our findings.

§ 3.9. Duration of each phase in hours, minutes, and seconds.

Table 11. Duration of each phase in hours, minutes, and seconds.

trial	<u>phase1duration</u> Phase 1 Duration (HH:mm:ss) ± 1 s	<u>phase2duration</u> Phase 2 Duration (HH:mm:ss) ± 1 s	<u>phase3duration</u> Phase 3 Duration (HH:mm:ss) ± 1 s
1	00:15:22	00:46:40	00:38:06
2	00:14:05	00:21:46	00:07:09
3	00:07:09	00:05:25	00:07:39
4	00:07:39	00:06:07	00:06:06
5	00:06:06	00:08:06	00:07:11
6	00:07:11	00:04:42	00:04:52
7	00:04:52	00:01:15	00:03:41
8	00:02:52	00:10:02	00:03:06
9	00:02:18	00:10:01	00:02:39
10	00:02:42	00:09:57	00:02:25
11	00:03:21	00:10:01	00:01:39
12	00:01:42	00:06:01	00:00:55
13	00:02:21	00:10:00	00:01:49
14	00:02:48	00:09:59	00:02:10
15	00:04:25	00:09:59	00:01:16
16	00:03:29	00:09:57	00:01:32

§ 3.10. Phase 2 Mean Pressure \pm SD for Liquid-Loop and Gas-Pathway

Table 12. Phase 2 Mean Pressure \pm SD for Liquid-Loop and Gas-Pathway.

<u>trial</u>	<u>phase2psiaA0</u> Phase 2 Liquid-Loop Before PDMS Filter (PSIA)	<u>phase2psiaA1</u> Phase 2 Liquid-Loop After PDMS Filter (PSIA)	<u>phase2absolutePressurePSIA</u> Phase 2 Gas-Pathway Before PDMS Filter (PSIA)	<u>phase2psiaA2</u> Phase 2 Gas-Pathway After PDMS Filter (PSIA)
1	16.87 \pm 0.87	14.66 \pm 0.01	1.51 \pm 0.06	14.68 \pm 0.02
2	16.07 \pm 0.71	14.67 \pm 0.01	1.60 \pm 0.52	14.68 \pm 0.01
3	16.09 \pm 0.75	14.67 \pm 0.01	1.47 \pm 0.34	14.68 \pm 0.01
4	16.02 \pm 0.69	14.67 \pm 0.01	1.47 \pm 0.38	14.67 \pm 0.02
5	16.09 \pm 0.67	14.67 \pm 0.01	1.45 \pm 0.10	14.67 \pm 0.02
6	16.08 \pm 0.70	14.67 \pm 0.01	1.45 \pm 0.18	14.66 \pm 0.01
7	16.11 \pm 0.75	14.67 \pm 0.01	1.47 \pm 0.23	14.67 \pm 0.01
8	16.94 \pm 0.72	14.65 \pm 0.01	2.07 \pm 0.15	14.65 \pm 0.02
9	17.17 \pm 0.70	14.65 \pm 0.00	2.07 \pm 0.05	14.67 \pm 0.01
10	17.14 \pm 0.94	14.65 \pm 0.01	2.09 \pm 0.19	14.67 \pm 0.02
11	19.23 \pm 0.80	14.65 \pm 0.00	2.10 \pm 0.25	14.67 \pm 0.01
12	18.16 \pm 0.80	14.66 \pm 0.00	2.08 \pm 0.06	14.67 \pm 0.01
13	16.47 \pm 0.72	14.65 \pm 0.00	2.07 \pm 0.10	14.67 \pm 0.01
14	20.46 \pm 1.22	14.67 \pm 0.01	2.09 \pm 0.10	14.67 \pm 0.02
15	18.64 \pm 1.57	14.67 \pm 0.01	2.08 \pm 0.05	14.68 \pm 0.02
16	16.81 \pm 1.48	14.66 \pm 0.00	2.07 \pm 0.07	14.66 \pm 0.01

§ 3.11. Phase 2 Mean Temperature \pm SD for Liquid-Loop and Gas-Pathway

Table 13. Phase 2 Mean Temperature \pm SD for Liquid-Loop and Gas-Pathway.

trial	<u>phase2DataSet1TemperatureC</u> Phase 2 Liquid-Loop Temperature (°C)	<u>phase2gasTemperatureC</u> Phase 2 Gas-Pathway Temperature (°C)
1	0.00 \pm 0.00	25.45 \pm 0.50
2	22.50 \pm 0.05	24.85 \pm 0.11
3	22.68 \pm 0.03	25.33 \pm 0.04
4	22.70 \pm 0.02	25.49 \pm 0.05
5	22.96 \pm 0.05	25.76 \pm 0.04
6	23.18 \pm 0.02	25.89 \pm 0.01
7	23.29 \pm 0.01	25.99 \pm 0.00
8	21.78 \pm 0.11	24.68 \pm 0.08
9	21.93 \pm 0.11	25.01 \pm 0.06
10	21.75 \pm 0.05	24.94 \pm 0.03
11	22.07 \pm 0.03	25.11 \pm 0.04
12	21.99 \pm 0.02	25.13 \pm 0.02
13	21.99 \pm 0.02	25.34 \pm 0.03
14	21.49 \pm 0.01	24.52 \pm 0.02
15	21.58 \pm 0.03	24.80 \pm 0.05
16	21.64 \pm 0.01	25.03 \pm 0.04

For Trial 1 phase2DataSet1TemperatureC values due to an error caused by broken Liquid Ammonium (NH₄⁺) Sensor and data cleanup.

§ 3.12. Phase 2 Mean Volumetric Flow Rate \pm SD for Gas-Pathway

Table 14. Phase 2 Mean Volumetric Flow Rate \pm SD for Gas-Pathway.

<u>trial</u>	<u>phase2volumetricFlowRateRawCCM</u> Phase 2 Gas-Pathway Volumetric Flow Rate Raw Uncorrected (CCM)	<u>phase2volumetricFlowRateStandardizedSCCM</u> Phase 2 Gas-Pathway Volumetric Flow Rate Standardized by Temperature, Pressure, and Gas Type (Air) (SCCM)
1	255.99 \pm 0.00	36.99 \pm 2.04
2	138.13 \pm 112.73	22.49 \pm 34.52
3	98.15 \pm 9.46	10.13 \pm 9.87
4	96.94 \pm 9.22	9.93 \pm 9.38
5	95.68 \pm 7.17	9.50 \pm 4.17
6	96.35 \pm 9.87	9.76 \pm 7.28
7	98.74 \pm 17.36	10.45 \pm 9.68
8	255.99 \pm 0.00	94.23 \pm 4.74
9	255.99 \pm 0.00	94.04 \pm 2.03
10	255.99 \pm 0.00	94.43 \pm 4.78
11	255.99 \pm 0.00	94.44 \pm 4.81
12	255.99 \pm 0.00	94.16 \pm 2.53
13	255.99 \pm 0.00	93.67 \pm 3.87
14	255.99 \pm 0.00	94.69 \pm 4.17
15	255.99 \pm 0.00	94.09 \pm 2.27
16	255.99 \pm 0.00	93.84 \pm 2.84

Values for phase2volumetricFlowRateRawCCM of 255.99 indicate the raw uncorrected flow rate is out of range. However, the phase2volumetricFlowRateStandardizedSCCM values are still valid.

§ 3.13. Phase 2 Mean Ammonium and Electrical Potential \pm SD

Table 15. Phase 2 Mean Ammonium and Electrical Potential \pm SD.

<u>trial</u>	<u>phase2DataSet1AmmoniummgL</u> Phase 2 Liquid-Loop Ammonium (NH ₄ ⁺) (PPM = mg/L)	<u>phase2DataSet1PotentialmV</u> Phase 2 Liquid-Loop Electrical Potential (mV = millivolt)
1	5.50 \pm 5.20	NaN \pm NaN
2	2.23 \pm 0.20	20.07 \pm 2.23
3	4.60 \pm 0.45	38.32 \pm 2.66
4	5.51 \pm 0.28	42.94 \pm 1.29
5	5.58 \pm 0.27	43.28 \pm 1.18
6	7.22 \pm 0.72	49.65 \pm 2.76
7	6.47 \pm 0.11	47.04 \pm 0.42
8	NaN \pm NaN	NaN \pm NaN
9	2.23 \pm 0.05	20.14 \pm 0.52
10	18.66 \pm 0.33	73.70 \pm 0.45
11	5.29 \pm 0.06	41.96 \pm 0.31
12	5.53 \pm 0.09	43.09 \pm 0.42
13	5.78 \pm 0.10	44.20 \pm 0.42
14	5.56 \pm 4.02	36.03 \pm 20.00
15	2.93 \pm 1.39	24.32 \pm 12.03
16	2.92 \pm 1.35	24.16 \pm 12.31

The phase2DataSet1AmmoniummgL and phase2DataSet1PotentialmV are measured by the Liquid Ammonium (NH₄⁺) Sensor. For Trial 1, phase2DataSet1AmmoniummgL values should not be trusted due to a broken Liquid Ammonium (NH₄⁺) Sensor.

§ 3.14. Phase 2 Mean pH and Ammonia \pm SD

Table 16. Phase 2 Mean pH and Ammonia \pm SD.

<u>trial</u>	<u>phase2DataSet1pH</u> Phase 2 Liquid-Loop Potential of Hydrogen (pH)	<u>phase2Sensor1GasReading</u> Phase 2 Gas-Pathway Ammonia (NH ₃) (PPM = mg/L)
1	11.26 \pm 0.04	5.86 \pm 1.20
2	9.54 \pm 0.02	0.00 \pm 0.00
3	9.76 \pm 0.03	0.00 \pm 0.00
4	9.85 \pm 0.01	0.00 \pm 0.00
5	9.93 \pm 0.01	0.00 \pm 0.00
6	9.99 \pm 0.01	0.00 \pm 0.00
7	10.05 \pm 0.01	0.00 \pm 0.00
8	5.39 \pm 0.03	0.00 \pm 0.00
9	10.03 \pm 0.08	0.00 \pm 0.00
10	11.17 \pm 0.00	15.37 \pm 9.58
11	10.59 \pm 0.01	0.60 \pm 0.49
12	10.47 \pm 0.00	0.51 \pm 0.50
13	10.48 \pm 0.00	0.69 \pm 0.46
14	11.36 \pm 0.01	2.15 \pm 1.34
15	11.23 \pm 0.00	2.45 \pm 1.54
16	11.22 \pm 0.00	2.29 \pm 1.54

§ 3.15. Ammonia Readings During Each Phase for Every Trial

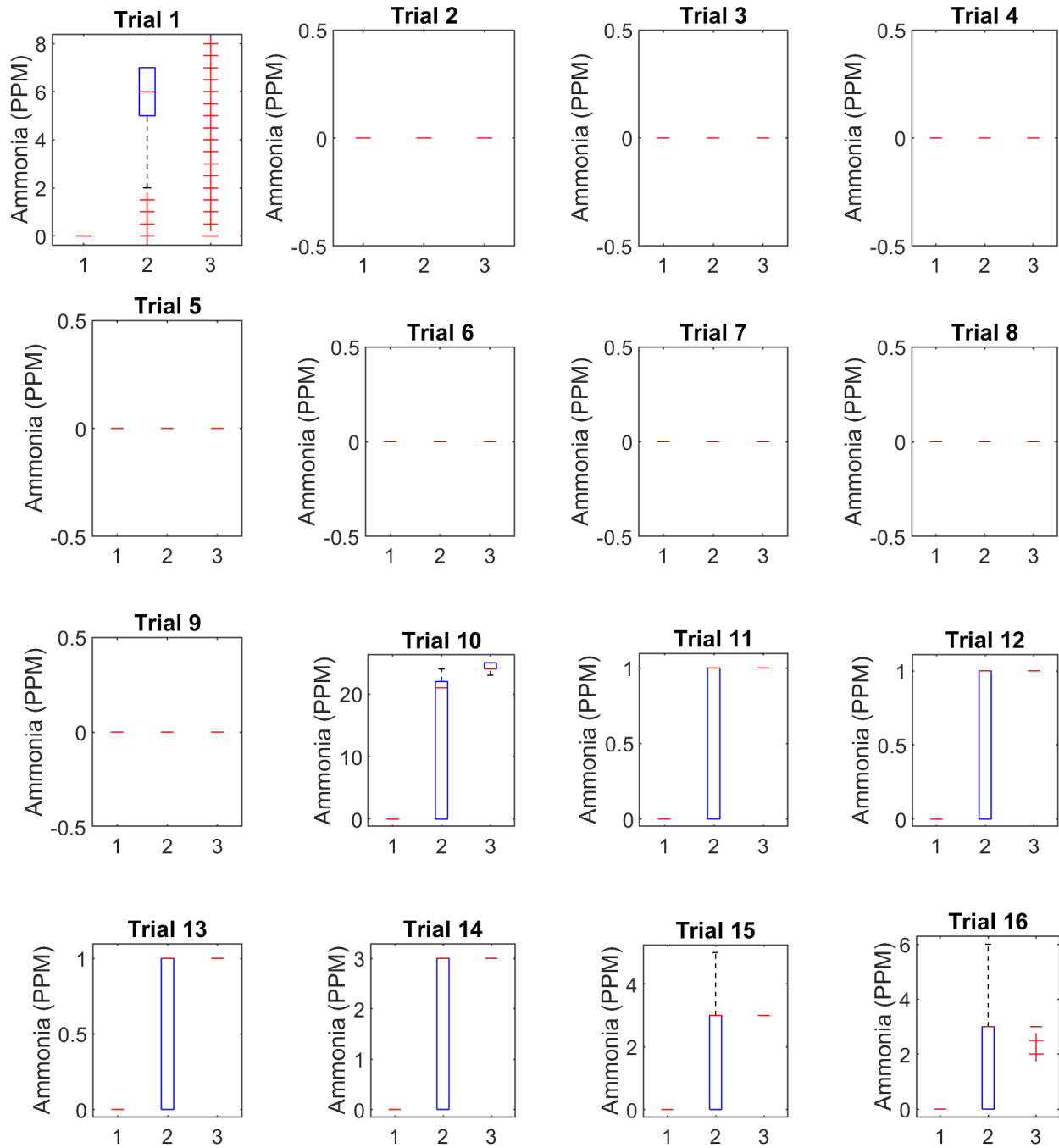


Figure 2. Ammonia Readings During Each Phase for Every Trial. Each subplot illustrates the distribution of ammonia data for a specific trial. The x-axis is categorized by phase, while the y-axis displays ammonia readings in parts per million (PPM), with scales varying per trial. The blue box captures the interquartile range (IQR), representing the middle 50% of the data. This box's lower and upper edges correspond to the 25th and 75th percentiles. A red line within the box marks the median. Whiskers stretch from the box to the most extreme data points, excluding outliers, which are indicated with “+” markers. (Visualize summary statistics with box plot, 2023) As an illustration, in Trial 1 of Phase 2, the median ammonia reading is around 6 PPM, ranging from 2 to 7 PPM.

§ 3.16. Phase 2 Ammonium (NH₄⁺) vs. Adjusted Mean Ammonia (NH₃)

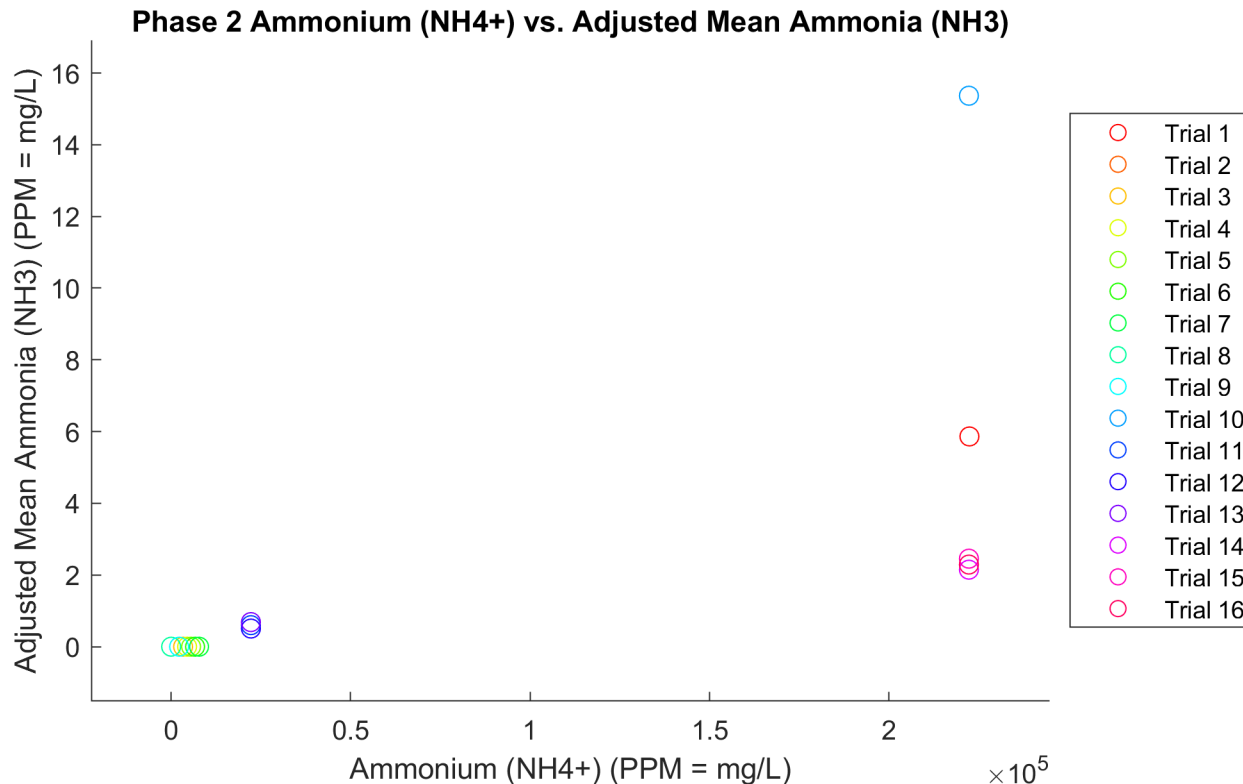


Figure 3. Phase 2 Ammonium (NH₄⁺) vs. Adjusted Mean Ammonia (NH₃). Each circle represents a trial, with the ammonia readings adjusted based on the control. The adjustment involved subtracting the mean ammonia reading from Trial 8 (control) from the mean readings of all other trials to account for baseline ammonia presence.

§ 3.16.1. Discussion of Phase 2 Ammonium vs. Adjusted Mean Ammonia

In our analysis, we utilized the Pearson correlation coefficient to gauge the strength and direction of the linear relationship between the volume of ammonium hydroxide and the mean ammonia gas reading during Phase 2. (Correlation coefficients, 2023) The strength of this relationship is indicated by how closely our data points fit a linear pattern. Essentially, the closer the coefficient value is to 1 (or -1), the stronger the linear relationship between the two datasets. A value close to 0 suggests a weak or no linear relationship. The direction, on the other hand, is indicated by the sign of the coefficient. A positive value indicates that as the volume of ammonium hydroxide

increases, the mean ammonia gas reading typically rises at the same time. Conversely, a negative value would suggest that as one variable rises, the other tends to decrease. This approach enables a clearer understanding of how the volume of ammonium hydroxide might impact ammonia gas readings during Phase 2, offering invaluable insights for our study.

The results indicated that the correlation coefficient of 0.66776 suggests a moderate positive linear relationship between the volume of ammonium hydroxide and the mean ammonia gas reading during Phase 2, which indicates that as the volume of ammonium hydroxide increases, the mean ammonia gas reading during Phase 2 also tends to increase. These results are expected, as more ammonium hydroxide in the Liquid-Loop provides more opportunities for ammonia to diffuse through the PDMS filter and into the Gas-Pathway stream to be measured by the ammonia gas sensor.

The p-value of 0.0047022 indicates that this correlation is statistically significant. A p-value below 0.05 is typically considered statistically significant, meaning there is a low probability that the observed correlation occurred by random chance. We can infer from this that the positive relationship between the volume of ammonium hydroxide and the ammonia gas reading is genuine and not due to random variations in the data.

In summary, the analysis suggests a statistically significant moderate positive relationship between the volume of ammonium hydroxide and the mean ammonia gas readings in phase 2. As more ammonium hydroxide is added, there is a notable increase in the mean ammonia gas reading during phase 2.

§ 3.17. Volumetric Flow Rate vs. Ammonia Gas Reading (All Trials)

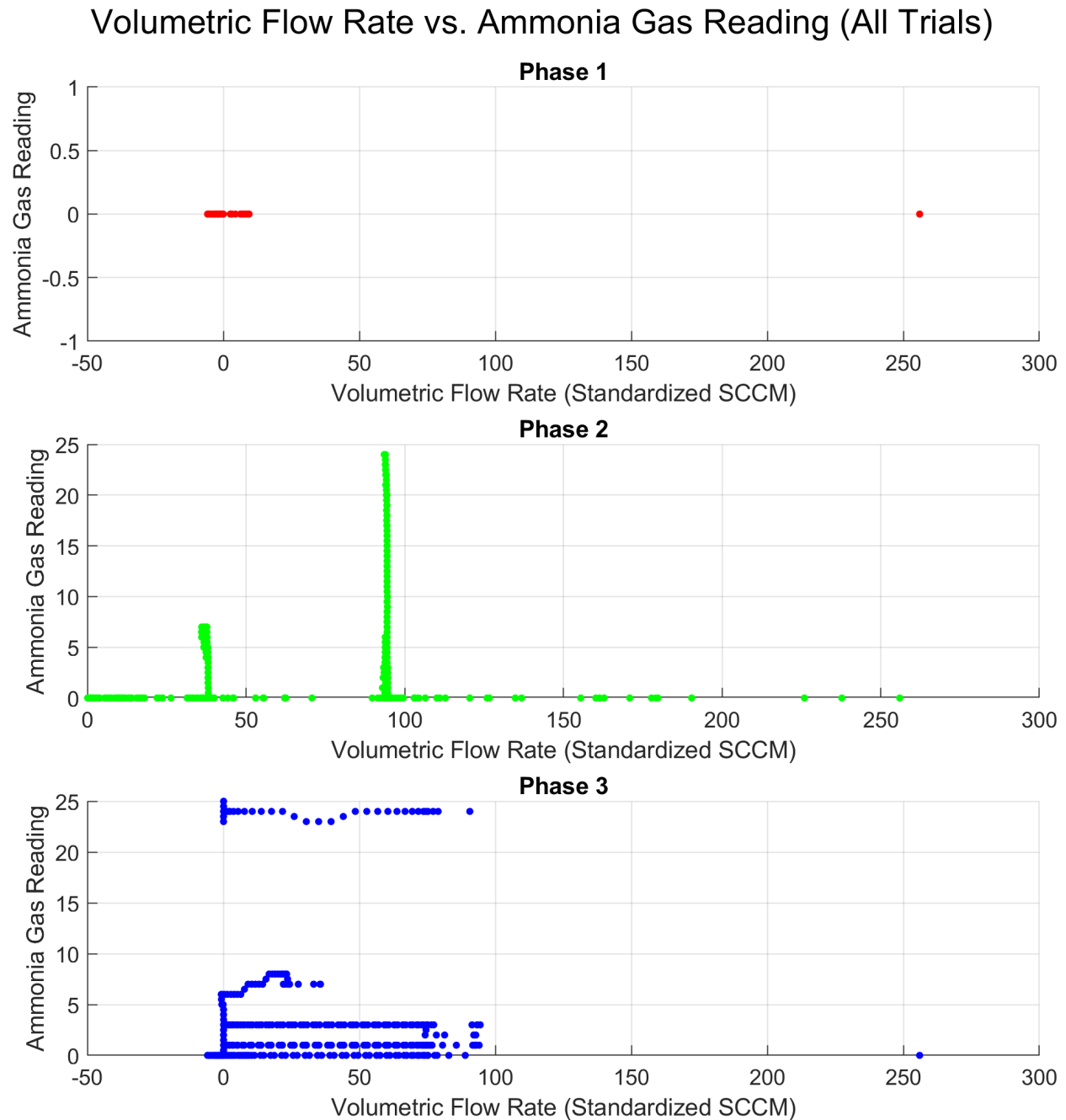


Figure 4. Volumetric Flow Rate vs. Ammonia Gas Reading (All Trials). This figure shows the correlation between the standardized volumetric flow rate (SCCM) and ammonia gas readings throughout three phases. Data scatter plot points are color-coded: red for Phase 1, green for Phase 2, and blue for Phase 3, representing individual observations within and across all trials. Phase 1 (roller pump on, vacuum pump off), data clusters around zero for both parameters, as anticipated. Phase 2 (roller pump on, vacuum pump on), data mostly align vertically around a flow rate of approximately 40 SCCM, with ammonia readings ranging from zero to seven PPM, and another concentration around 95 SCCM with ammonia readings from zero to 25 PPM. Phase 3 (roller pump on, vacuum pump off) displays residual ammonia in the Gas-Pathway.

§ 3.18. Phase 2 Mean psiaA0 (SEM Error Bars)

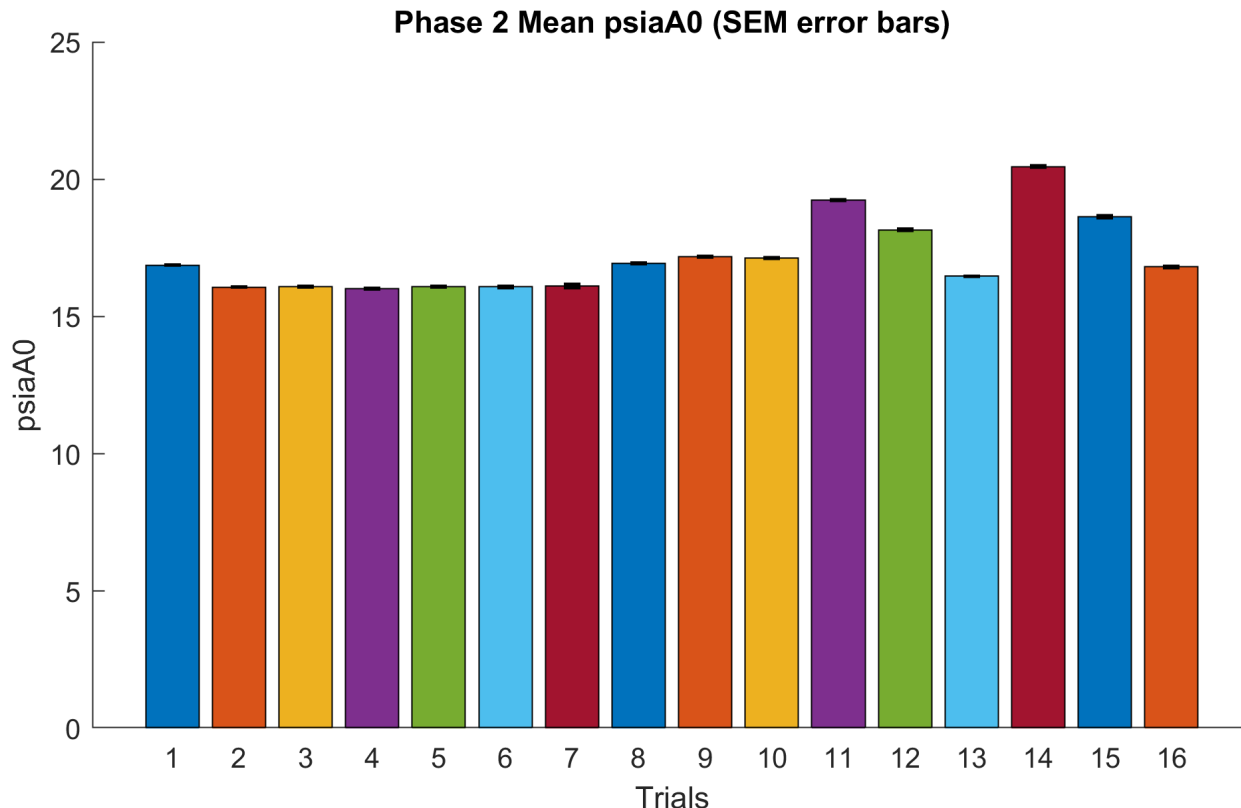


Figure 5. Phase 2 Mean psiaA0 (SEM Error Bars). psiaA0: (PSIA = PSI Absolute = pounds per square inch absolute) units, mean (average) pressure over the entire phase 2 duration, in line with the Liquid-Loop, immediately before the PDMS filter, recorded by Arduino Device. The vertical bars show the mean values from all trials, and the black lines represent each trial's Standard Error of the Mean (SEM). Trial 4 shows the lowest at 16.02 ± 0.69 PSIA, and Trial 14 shows the highest pressure of 20.46 ± 1.22 PSIA. Trials with PFC 3M FC-770 include 14, 15, and 16. Liquid-Loop flow rates for context (mL/min): Trial 1 (5.65 ± 0.16), Trials 2 to 11, and 14: (6.34 ± 0.10), Trials 12, 13, 15: (4.34 ± 0.12), Trial 16: (2.42 ± 0.19). For Trials 14, 15, and 16, the overall higher pressure can be attributed to the PFC's higher viscosity trying to push its way through the PDMS filter. The decrease in pressure from 14 to 16 can be attributed to the decrease in Liquid-Loop flow rates from the roller pump.

§ 3.19. Phase 2 Mean psiaA1 (SEM Error Bars)

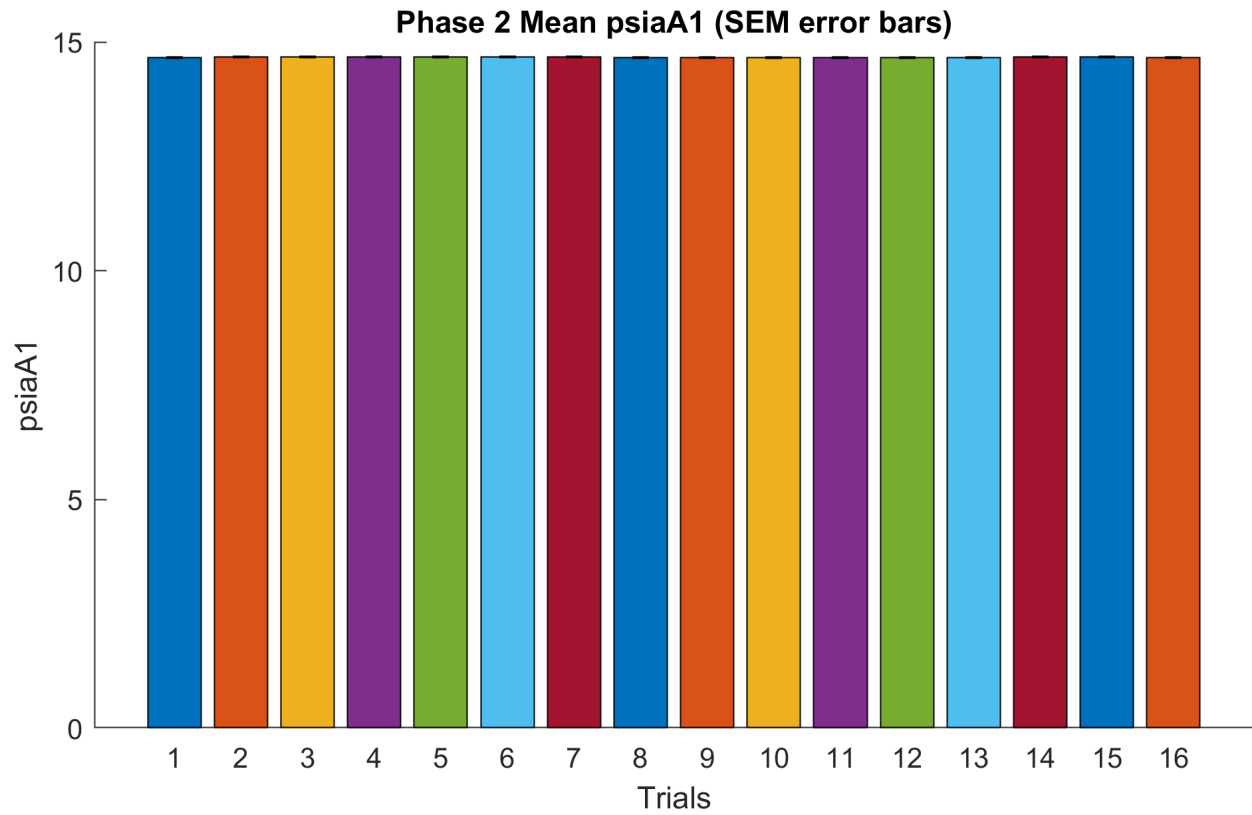


Figure 6. Phase 2 Mean psiaA1 (SEM Error Bars). psiaA1: (PSIA = PSI Absolute = pounds per square inch absolute) units, mean (average) pressure over the entire phase 2 duration, in line with the Liquid-Loop, immediately after the PDMS filter, recorded by Arduino Device. The vertical bars show the mean values from all trials, and the black lines represent each trial's Standard Error of the Mean (SEM). Pressures remain relatively consistent across the trials, with a narrow range between 14.65 ± 0.00 PSIA and 14.67 ± 0.01 PSIA. The tight consistency in these readings underscores a stable after-PDMS filter pressure condition during Phase 2 of the experiment.

§ 3.20. Phase 2 Mean psiaA2 (SEM Error Bars)

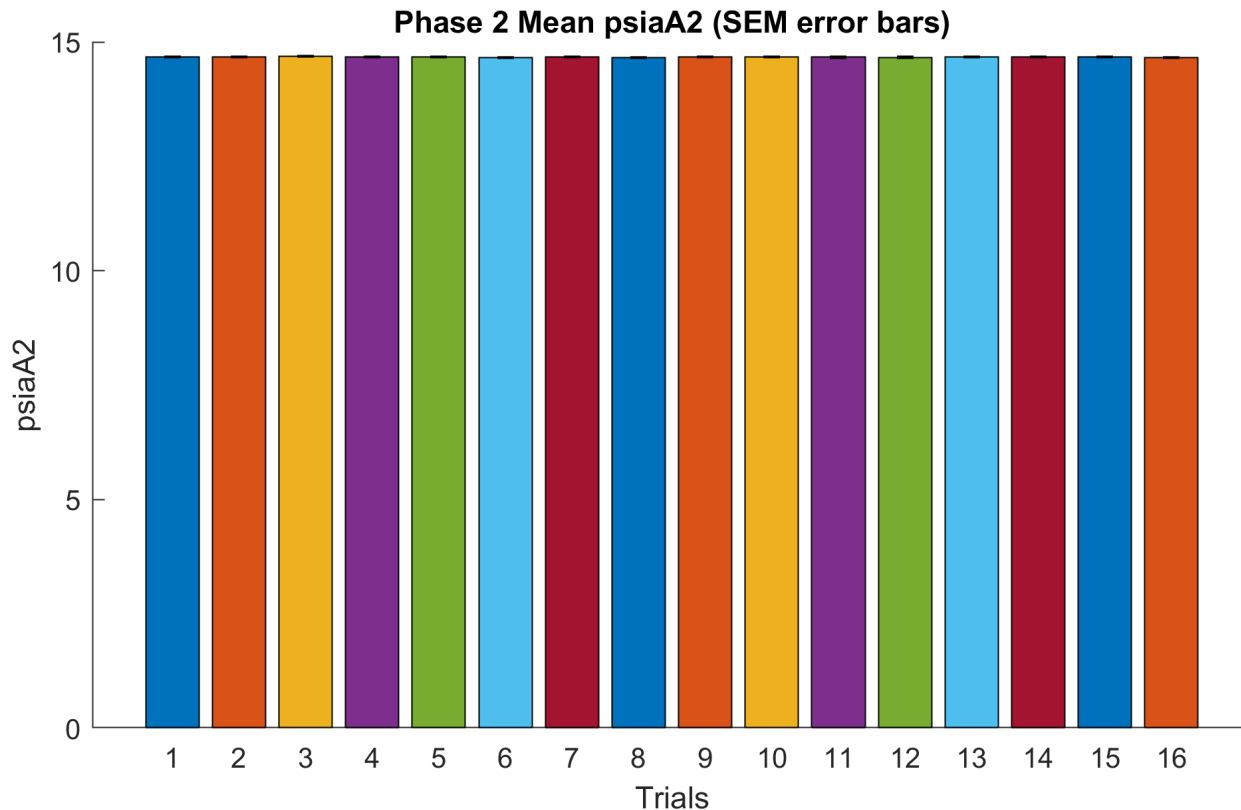


Figure 7. Phase 2 Mean psiaA2 (SEM Error Bars). psiaA2: (PSIA = PSI Absolute = pounds per square inch absolute) units, mean (average) pressure over the entire phase 2 duration, in line with the gas path, immediately before the Omega Device measuring ammonia (NH₃) gas, recorded by Arduino Device, exposed to dry lab air and ammonia gas. The vertical bars show the mean values from all trials, and the black lines represent each trial's Standard Error of the Mean (SEM). The pressures across trials exhibit a remarkable consistency, ranging narrowly from 14.65 ± 0.02 PSIA to 14.68 ± 0.02 PSIA. This consistency underscores a stable pressure condition in the Gas-Pathway after the PDMS filter and right before the ammonia gas measurement with the Gas Ammonia (NH₃) Sensor during Phase 2 of the experiment.

§ 3.21. Phase 2 Mean absolutePressurePSIA (SEM Error Bars)

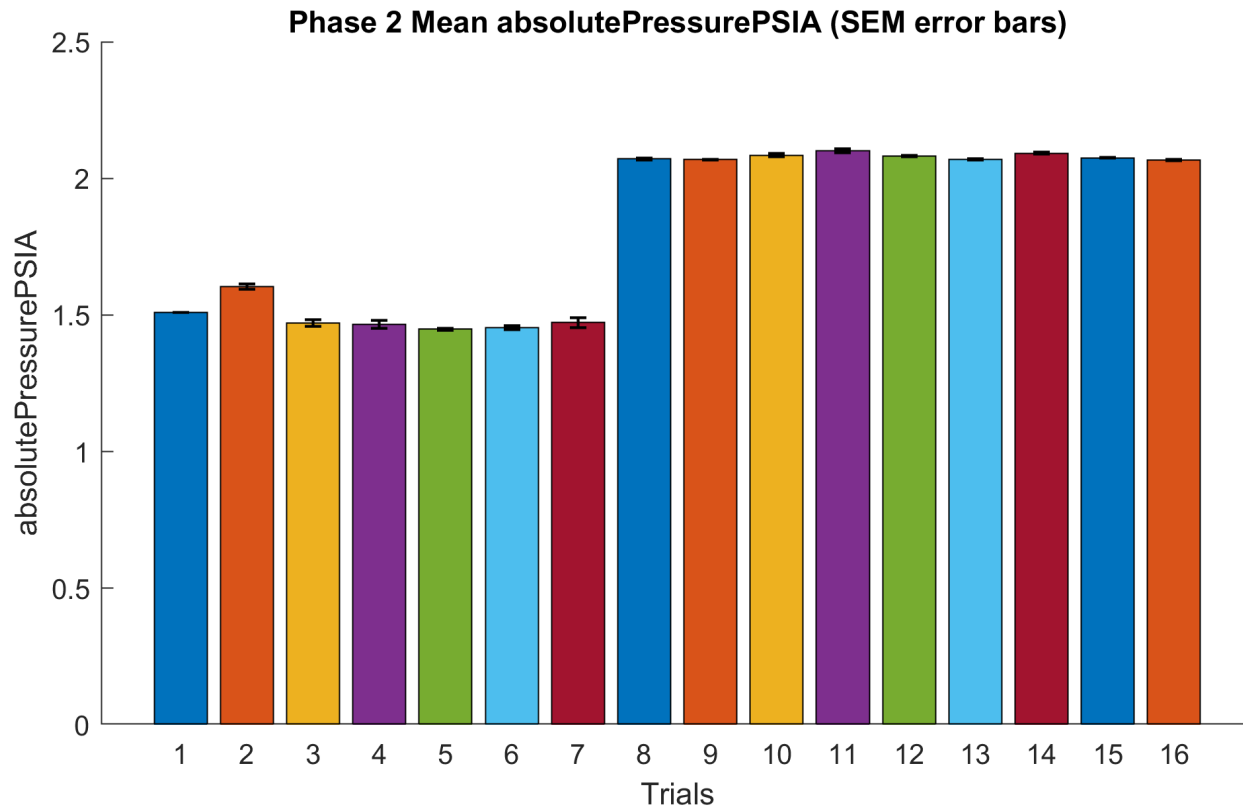


Figure 8. Phase 2 Mean absolutePressurePSIA (SEM Error Bars). absolutePressurePSIA: (PSIA = PSI Absolute = pounds per square inch absolute) units, mean (average) pressure over the entire phase 2 duration, in line with the gas path, before the Polydimethylsiloxane (PDMS) filter, recorded by the Omega Device, exposed to dry lab air only. The vertical bars show the mean values from all trials, and the black lines represent each trial's Standard Error of the Mean (SEM). A distinct difference is evident between Trials 1 to 7 and 8 to 16. The former shows pressures in the 1.45 to 1.60 PSIA range, while the latter consistently falls in the 2.07 to 2.10 PSIA range. The change in absolutePressurePSIA values most likely results from a noticeable shift in the Gas-Pathway Volumetric Flow Rate, which jumps from around 10 SCCM to 94 SCCM starting from Trial 8. This shift highlights the impact of flow rate changes on pressure readings during Phase 2. As the gas flow rate in the pathway increases, the associated pressure also rises. This increase in pressure with a higher flow rate results from the compressibility of gases, viscous effects, and potential constrictions or resistances in the Gas-Pathway. External factors may also influence these readings.

§ 3.22. Phase 2 Mean gasTemperatureC (SEM Error Bars)

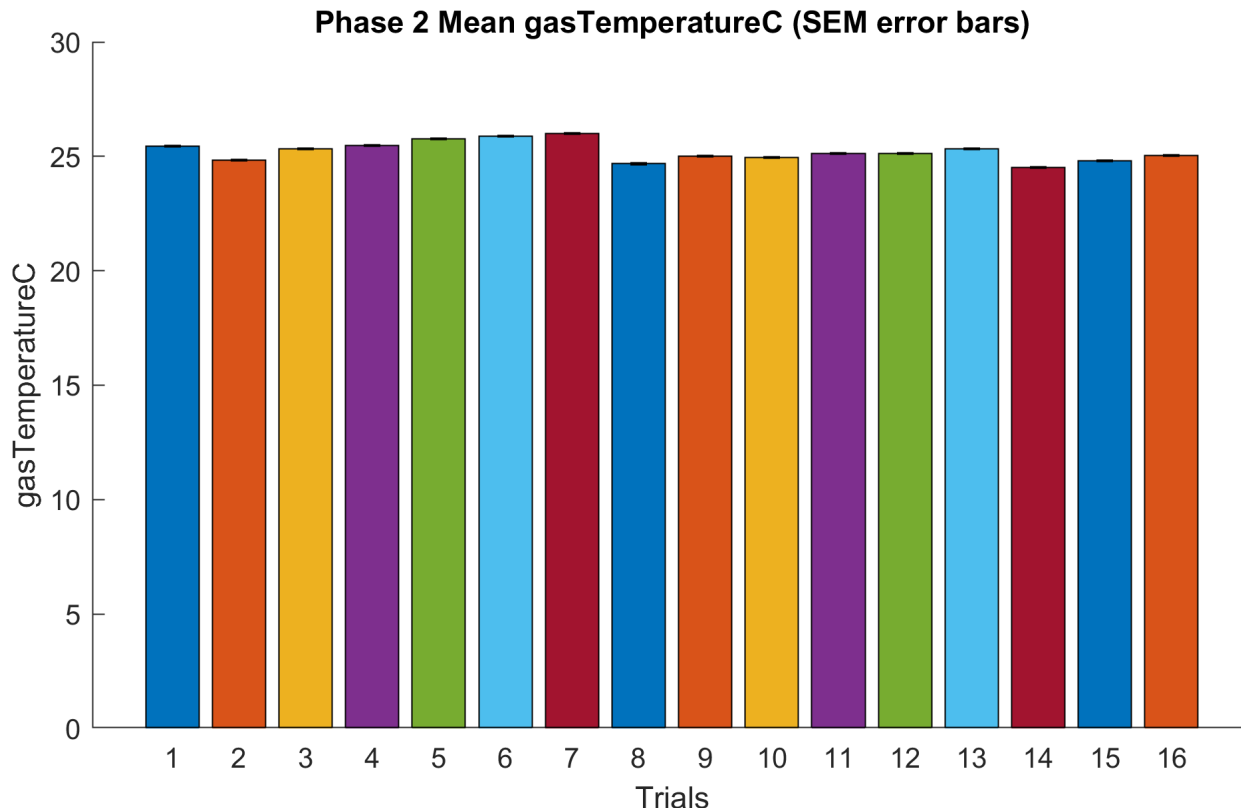


Figure 9. Phase 2 Mean gasTemperatureC (SEM Error Bars). gasTemperatureC: (C = degrees Celsius) units, mean (average) temperature over the entire phase 2 duration, in line with the gas path, before the Polydimethylsiloxane (PDMS) filter, recorded by the Omega Device, exposed to dry lab air only. The vertical bars show the mean values from all trials, and the black lines represent each trial's Standard Error of the Mean (SEM). Temperature values remain relatively consistent throughout the trials, with minimal variance. Trials predominantly range from 24.52°C to 25.99°C. However, it is noteworthy to mention a slight decrease in temperature readings for some trials like Trial 2 and Trial 14. Such variations can be due to multiple factors, including ambient lab conditions, gas flow rates, or potential changes in the experiment setup. The overall stability in temperature readings suggests that the Gas-Pathway remained under controlled conditions during Phase 2, ensuring minimal external interference on the gas temperature. Trials 14 to 16 used the PFC 3M FC-770, which is used for heat transfer fluid applications and may be contributing to the increase in temperature from environmental factors, e.g., friction effects from the magnetic stir bar in fluid (as the PFC is particularly viscous), or from forcing the PFC through the PDMS filter (due to resistance to flow with higher pressures).

§ 3.23. Phase 2 Mean volumetricFlowRateRawCCM (SEM Error Bars)

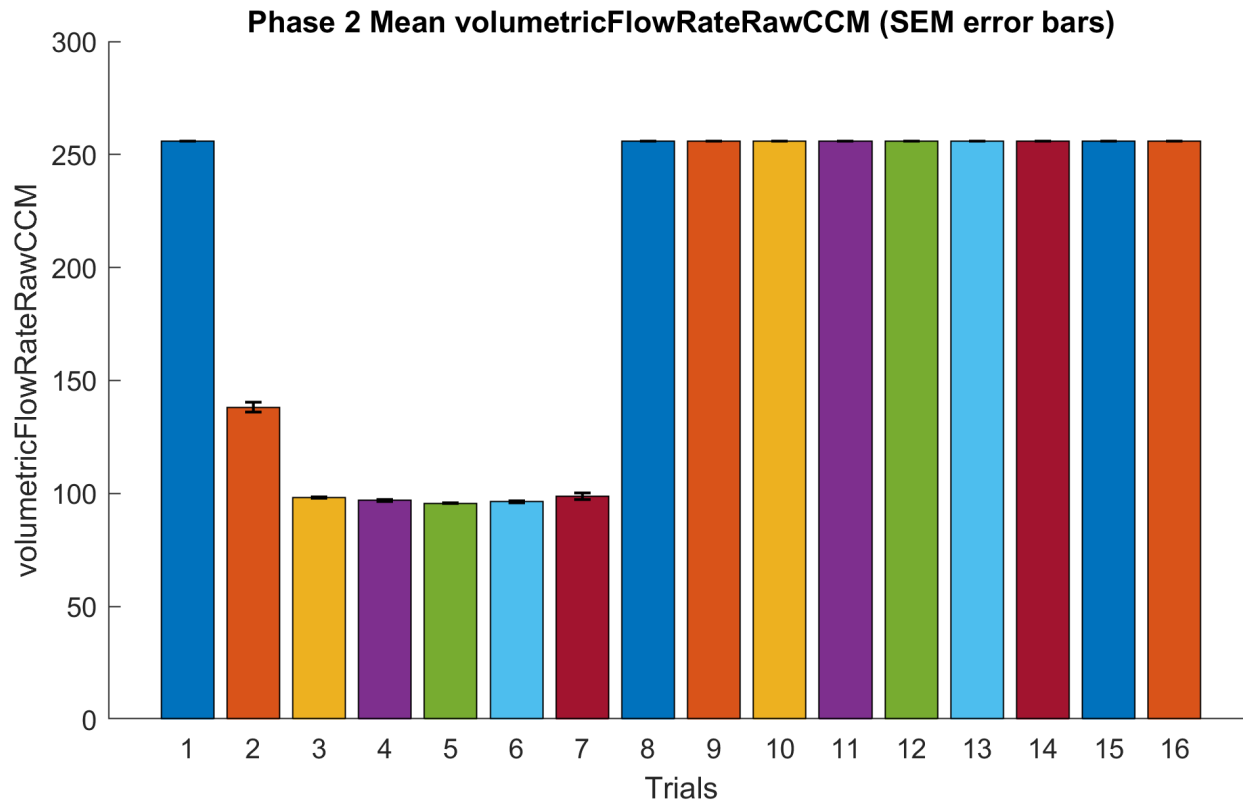


Figure 10. Phase 2 Mean volumetricFlowRateRawCCM (SEM Error Bars). volumetricFlowRateRawCCM, represented in cubic centimeters (CCM), indicates the mean (average) flow rate over the complete duration of Phase 2. Measurements are taken before the Polydimethylsiloxane (PDMS) filter and recorded by the Omega Device when exposed only to dry lab air. Importantly, these readings are raw and uncorrected for temperature, pressure, and the specific gas type used. The vertical bars show the mean values from all trials, and the black lines represent each trial's Standard Error of the Mean (SEM). While most trials register a consistent raw flow rate of 255.99 CCM, this value represents the upper limit of the Omega Device, indicating flow rates that exceed this threshold. The standardized flow rate values, which account for temperature, pressure, and gas type, offer a clearer insight into the actual flow rates, ranging from 9.50 SCCM to 94.69 SCCM. This distinction emphasizes the importance of standardizing flow rate measurements, especially when comparing them with other parameters or datasets within the experiment.

§ 3.24. Phase 2 Mean volumetricFlowRateStandardizedSCCM (SEM Error Bars)

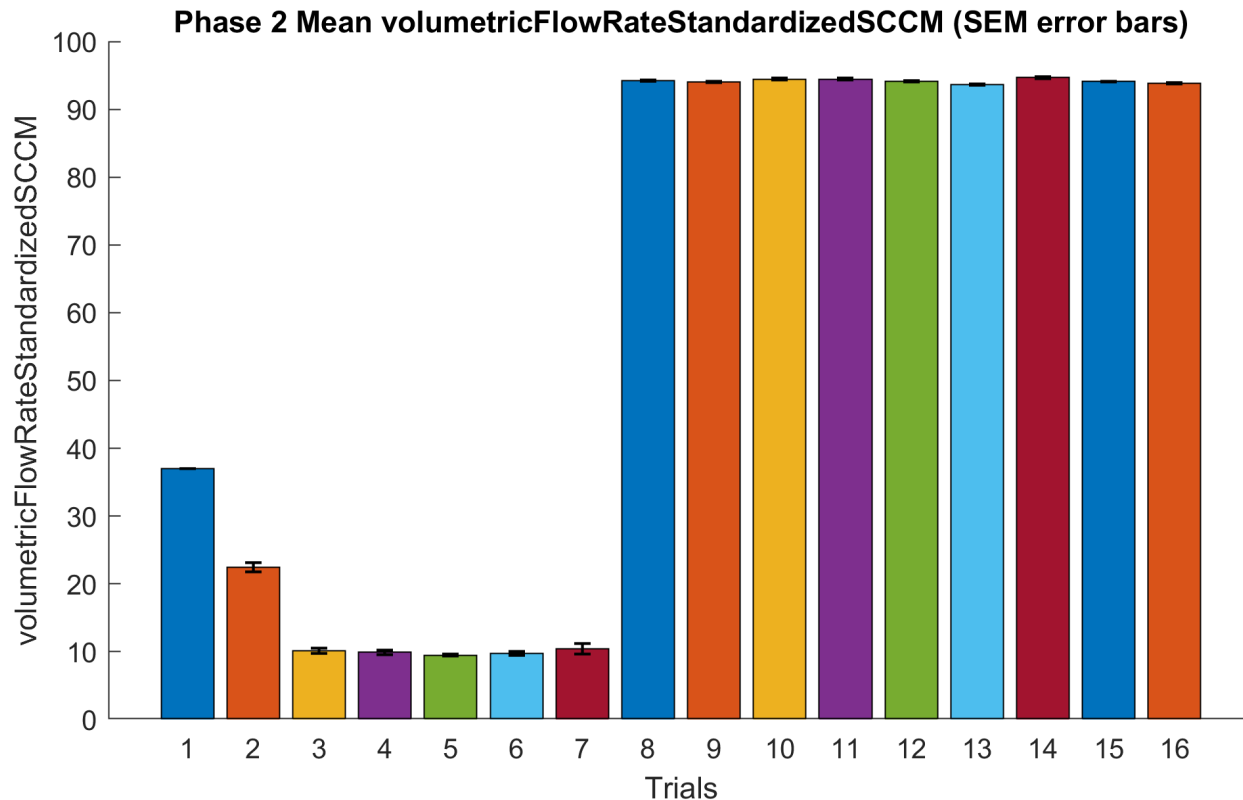


Figure 11. Phase 2 Mean volumetricFlowRateStandardizedSCCM (SEM Error Bars).

volumetricFlowRateStandardizedSCCM: (SCCM = standard cubic centimeter) units, mean (average) flow rate over the entire phase 2 duration, in line with the gas path, before the Polydimethylsiloxane (PDMS) filter, recorded by the Omega Device, exposed to dry lab air only, corrected by Omega Device for temperature, pressure, and gas type (air), National Institute of Standards and Technology (NIST) traceable calibration. The vertical bars show the mean values from all trials, and the black lines represent each trial's Standard Error of the Mean (SEM). Data indicates a significant variability in flow rates: the initial trials (1 to 7) hover between 9.50 SCCM to 36.99 SCCM, while the subsequent Trials (8 to 16) maintain a steadier and markedly higher range around 93.67 SCCM to 94.69 SCCM. The observed contrast underscores potential adjustments in experimental conditions during Phase 2, further highlighting the relevance of standardized flow rate measurements in obtaining precise experimental outcomes.

§ 3.25. Phase 2 Mean DataSet1AmmoniummgL (SEM Error Bars)

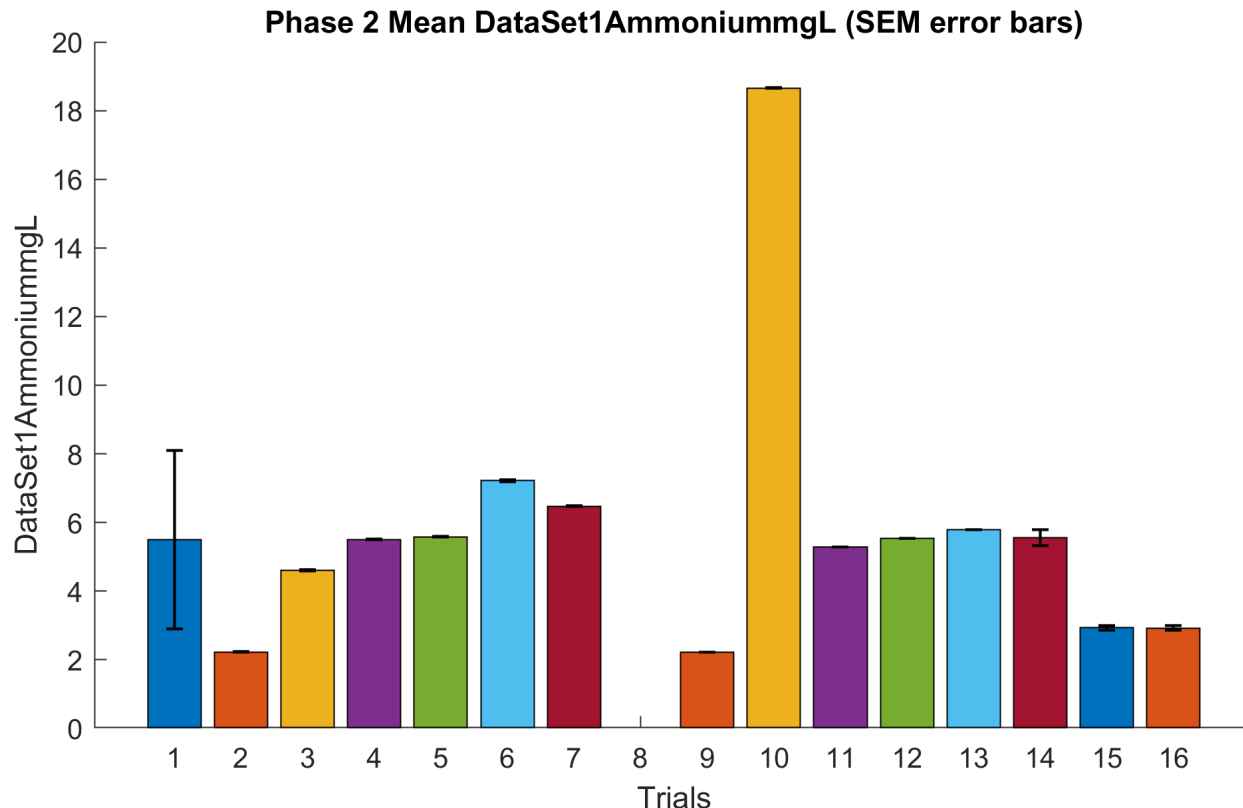


Figure 12. Phase 2 Mean DataSet1AmmoniummgL (SEM Error Bars). DataSet1AmmoniummgL: (mg/L = ppm) units, mean (average) ammonium (NH_4^+) measurement from Vernier Device, in line with the fluid loop, specifically in the vortex mixing tank, measured via Bluetooth (to prevent voltage interference from pH sensor). The vertical bars show the mean values from all trials, and the black lines represent each trial's Standard Error of the Mean (SEM).

Together with the ammonium concentration, the electrical potential, denoted in millivolts (mV) and captured by the Liquid Ammonium (NH_4^+) Sensor, offers valuable insights into the experiment. It is essential to note some anomalies: the values from Trial 1 may not be reliable due to a malfunctioning Liquid Ammonium (NH_4^+) Sensor, hence the significant SEM. In Trial 8, the control recorded values exceeded the expected range because the fluid was purely RODI water, resulting in NaNs. For Trial 10, the ammonium concentration measured 18.66 ± 0.33 ppm, correlating with the introduction of a significant 1514.00 ± 5.0 μL of ammonium hydroxide (NH_4OH). This volume likely boosted the ammonium levels. Concurrently, the electrical potential was recorded at 73.70 ± 0.45 mV, indicating the interplay between the added ammonium hydroxide and the measured potential. Trials 14 to 16 had PFC 3M FC-770 added to the mixture. While the PFC is non-conductive, it may have influenced the measurements, giving rise to the higher SEM when the PFC was first added in Trial 14.

Despite these variances, the broader dataset remains reasonably consistent, underscoring the challenges of monitoring liquid ammonium levels. Ensuring the integrity of this data and effectively addressing any anomalies is crucial for drawing precise and meaningful conclusions from the experiment.

§ 3.26. Phase 2 Mean DataSet1PotentialmV (SEM Error Bars)

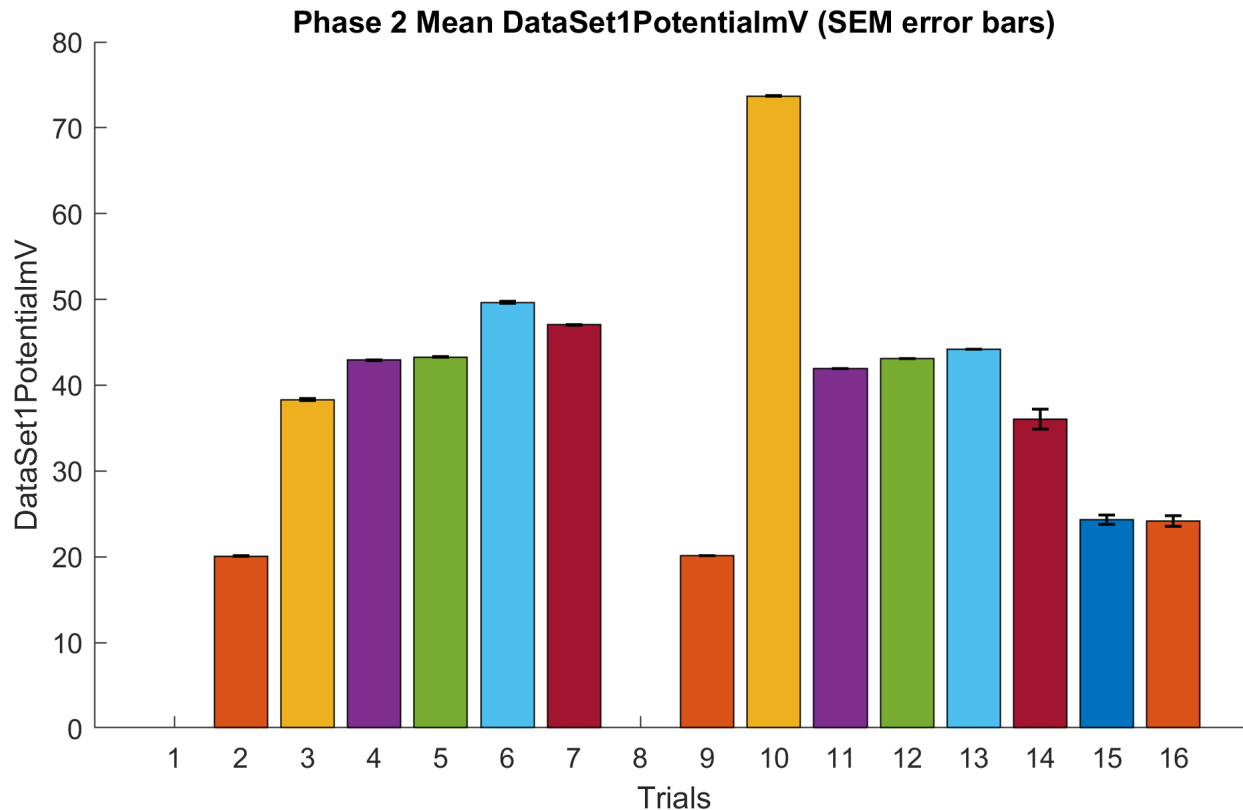


Figure 13. Phase 2 Mean DataSet1PotentialmV (SEM Error Bars). DataSet1PotentialmV: (mV = millivolt) units, mean (average) electric potential measurement from Vernier Device, in line with the fluid loop, specifically in the vortex mixing tank, measured via Bluetooth (to prevent voltage interference from pH sensor). The vertical bars show the mean values from all trials, and the black lines represent each trial's Standard Error of the Mean (SEM). For Trial 1, the values were out of range due to a broken sensor and converted into NaNs. For Trial 8, the only fluid present was RODI water. The Liquid Ammonium (NH_4^+) Sensor, designed to detect specific ions, had difficulty due to the water's purity, leading to out-of-range measurements, subsequently converted into NaNs. Trials 14 to 16 had PFC 3M FC-770 added to the mixture. While the PFC is non-conductive, it may have influenced the measurements, giving rise to the higher SEM when the PFC was first added in Trial 14.

§ 3.27. Phase 2 Mean DataSet1TemperatureC (SEM Error Bars)

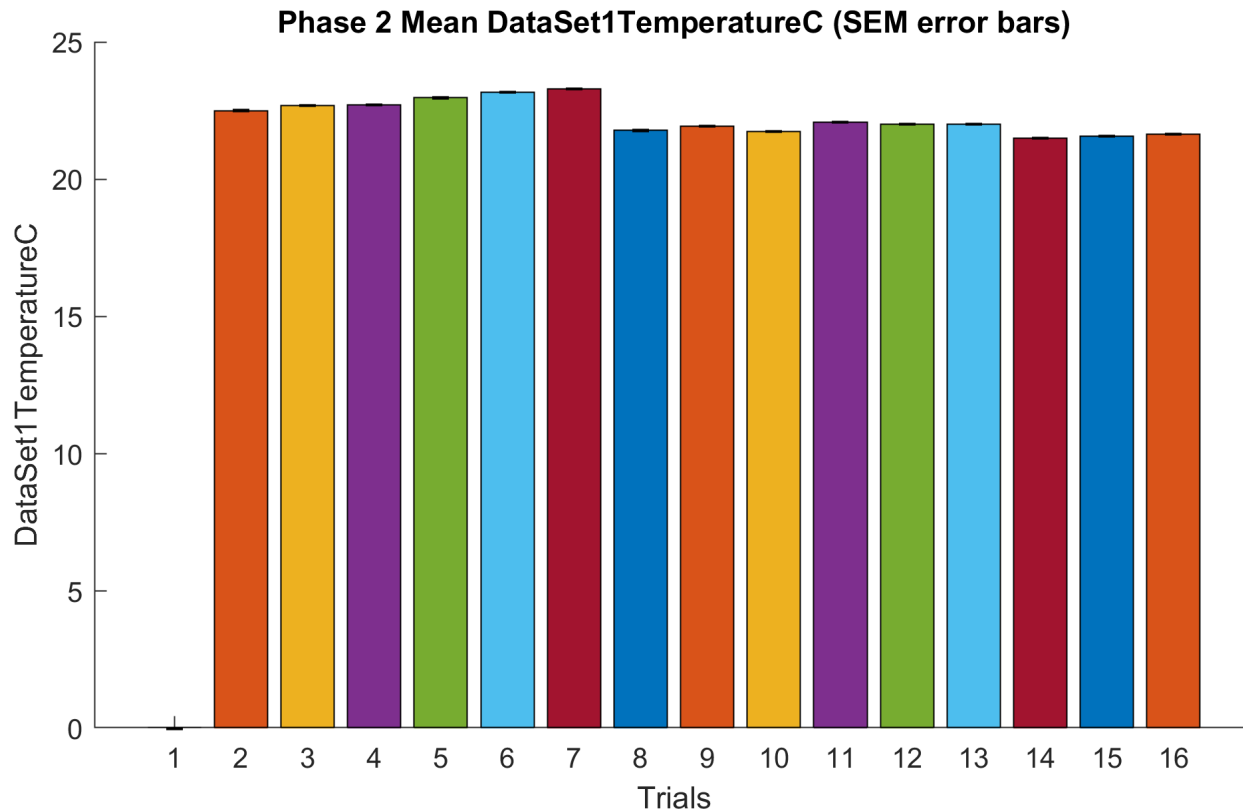


Figure 14. Phase 2 Mean DataSet1TemperatureC (SEM Error Bars). DataSet1TemperatureC: (C = degrees Celsius) units, mean (average) electric potential measurement from Vernier Device, in line with the fluid loop, specifically in the vortex mixing tank, measured via a wire connection. The vertical bars show the mean values from all trials, and the black lines represent each trial's Standard Error of the Mean (SEM). For Trial 1, DataSet1TemperatureC values are due to an error caused by a broken Liquid Ammonium (NH_4^+) Sensor, which we removed during data cleanup.

§ 3.28. Phase 2 Mean DataSet1pH (SEM Error Bars)

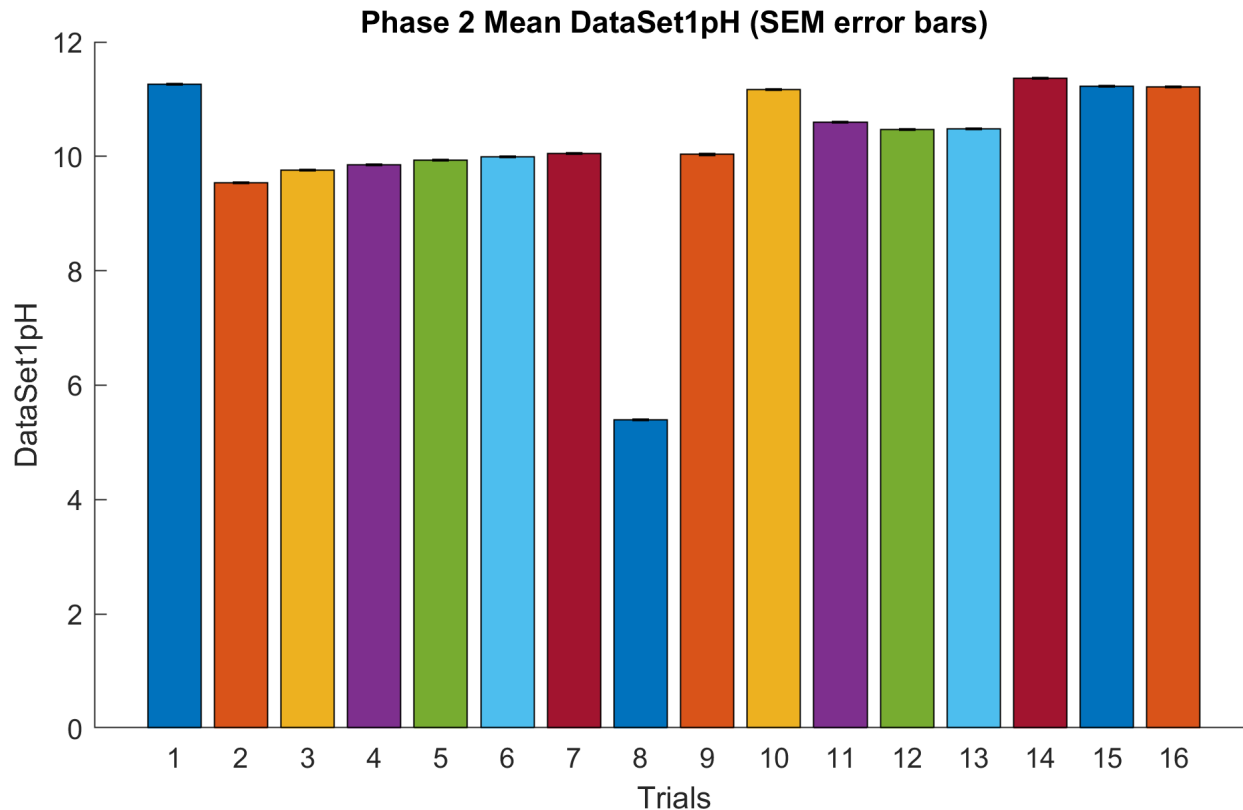


Figure 15. Phase 2 Mean DataSet1pH (SEM Error Bars). DataSet1pH: (pH = potential of hydrogen) units, mean (average) pH measurement from Vernier Device, in line with the fluid loop, specifically in the vortex mixing tank, measured via a wire connection. The vertical bars show the mean values from all trials, and the black lines represent each trial's Standard Error of the Mean (SEM). For Trial 8, the pH notably dropped to 5.39 ± 0.03 . Trial 8 used only RODI water as a control, typically having a neutral to slightly acidic pH. The pH can lean towards the acidic side, especially if atmospheric carbon dioxide exposes the RODI water. The recorded pH matched the expected value for pure RODI water, and we confirmed it with pH paper. In contrast, the remaining trials showed pH values in a more alkaline range, reflecting the addition of ammonium hydroxide.

§ 3.28.1. Discussion on Elevated pH Values for Trials 14, 15, and 16

Additional information and discussion of the PFC 3M FC-770 can be found in (DiBeneditto, 2023). The elevated pH values observed in Trials 14, 15, and 16 might be influenced by the presence of PFC 3M FC-770. However, further investigation is necessary to confirm this. Several properties of 3M FC-770 **may** be at play:

- **Chemical Stability, Non-Conductive Nature, and Inertness:** The inherent inertness and chemical stability of 3M FC-770 ensure non-reactivity with the solution's components, including ammonium hydroxide. (3M Fluorinert Electronic Liquid FC-770, 2023) Its non-conductive nature could interfere with ion mobility, limiting interactions that typically alter pH and potentially stabilizing it at a higher value. This property also ensures that unintended chemical reactions do not affect the solution, maintaining a consistent pH less sensitive to unexpected factors.
- **Gas Absorption and CO₂ Uptake:** PFCs, especially 3M FC-770, are well-known for their gas absorption capabilities. While specific research on its solubility concerning ammonia is lacking, it might absorb significant quantities of other atmospheric gases, including nitrogen and water vapor. Absorbed water can react with CO₂ to form carbonic acid, which could decrease the pH. The elevated pH levels might result from the reduced proton concentration influenced by 3M FC-770's absorption of atmospheric gases and potential reactions within it.
- **Immiscibility, Solute Solubilization, and Species Partitioning:** Due to 3M FC-770's immiscibility, solutes like ammonium hydroxide might be more concentrated in the water phase, potentially leading to higher pH levels. However, using the magnetic stir bar in the Vortex Mixing Tank for mechanical mixing should have mitigated this. Furthermore, while partitioning generally describes how a substance distributes itself between two immiscible phases, it might be more complex in this context. The system's conditions (magnetic stirring, vortex formation, and probe placement) mean that probes could measure a mixture of both phases, which are in constant flux due to agitation, rather than

one dominant phase. Such interactions may render the water phase more alkaline and skew pH measurements.

- **Buffering Potential:** 3M FC-770 does not directly react with the solution, but it might bolster its buffering capacity. PH remains relatively stable in a buffered solution despite adding acids or bases. Given the consistent pH values observed even under conditions typically prompting pH shifts, it is plausible that 3M FC-770 enhances the solution's natural buffering capabilities, ensuring a stable environment for subsequent experiments or reactions.

Given these factors, the elevated pH values in these trials likely stem from a combination of 3M FC-770's unique physicochemical properties and interactions (or lack thereof) with the solution's components. The potential partitioning effect, non-reactive nature, and buffering potential of 3M FC-770 seem pivotal in these observations. The pH changes in the latter trials can be attributed to the complicated interactions between 3M FC-770 and the solution's components.

§ 3.29. Phase 2 Mean Sensor1GasReading (SEM Error Bars)

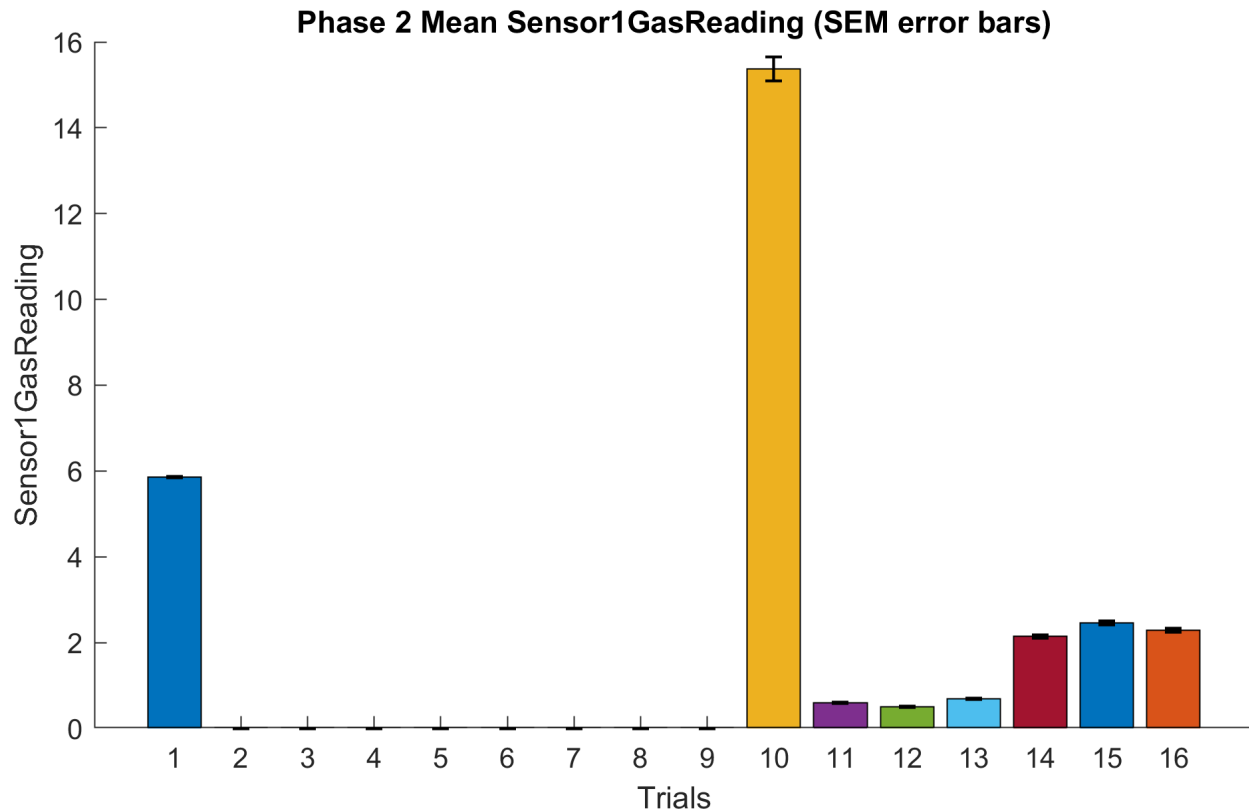


Figure 16. Phase 2 Mean Sensor1GasReading (SEM Error Bars). Sensor1GasReading: (mg/L = ppm) units, mean (average) ammonia (NH₃) gas reading pulled through the Polydimethylsiloxane (PDMS) filter at vacuum pressure through the dry lab air supplied. The vertical bars show the mean values from all trials, and the black lines represent each trial's Standard Error of the Mean (SEM).

§ 3.29.1. Discussion of Sensor1GasReading Ammonia Gas Values

After analyzing the ammonia gas readings (Sensor1GasReading) across various trials, specific patterns become clear:

- **Detection Limit:** In Trials 2 to 9 and 11 to 13, ammonia readings were negligible, possibly because the ToxiRae device has a resolution of 1 ppm and may not detect ammonia concentrations below this threshold.

- **Influence of PFC 3M FC-770:** Trials 14 to 16 showed a deviation, with ammonia readings fluctuating between 2.15 to 2.45 mg/L. The addition of PFC 3M FC-770, known for its gas absorption capabilities, might affect the gas equilibrium in the solution, which could influence the quantity of ammonia available for transfer when the solution is subjected to the PDMS filter with vacuum pressure. However, PFC 3M FC-770 is unlikely to pass through the PDMS filter.
- **Trial 10 Anomaly:** Trial 10 recorded a significant ammonia reading of 15.37 ± 9.58 mg/L. Without PFC 3M FC-770, the high concentration of ammonium hydroxide in this trial is likely responsible for the elevated ammonia levels. It is further highlighted by the subsequent reduction in ammonia readings in Trials 11 to 13, which had similar conditions.
- **Physiological Implications:** While the ammonia concentrations analyzed here far exceed physiological levels, it is crucial to approach these results with caution when making real-world physiological interpretations. Nevertheless, the findings provide valuable insights into ammonia's behavior under different experimental conditions.
- **Other Influences:** Although factors like temperature, pressure, and volumetric flow rate were kept relatively consistent throughout the trials, they may still subtly influence the results. Even slight changes in these parameters could affect gas solubility, movement, and detection.

In summary, the ammonia gas readings appear to be primarily influenced by the solution's physicochemical characteristics, the presence or absence of PFC 3M FC-770, and the concentrations of ammonium hydroxide. Further exploration of these relationships, particularly

concerning the potential effects or interactions of PFC 3M FC-770, is crucial for a thorough understanding.

§ 3.29.2. In-Depth Analysis of Select Trials for Ammonia Gas Values

This section discusses a detailed analysis of selected trials.

Trial 1: Setting the Stage

The first trial registered an ammonia gas reading of 5.86 ± 1.20 mg/L. Given the substantial volume of ammonium hydroxide introduced (1515.00 ± 5.0 μ L) and the absence of PFC 3M FC-770, this result reaffirms the direct correlation between the concentration of ammonium hydroxide and the resultant ammonia gas readings.

Trial 10: An Evident Spike

Trial 10 has a peak ammonia reading of 15.37 ± 9.58 mg/L. Although the other conditions and parameters remained consistent with other trials, the distinct volume of ammonium hydroxide used (1514.00 ± 5.0 μ L) is likely the key contributor to this notable peak. Without PFC 3M FC-770 in this trial, the influence of ammonium hydroxide concentration on ammonia readings is again highlighted.

Trials 11 to 13: Demonstrating Consistency

Trials 11 to 13 show consistent conditions, with unchanging flow rates, the absence of PFC, and equivalent volumes of ammonium hydroxide; even with lower readings, ammonia was consistently detected, with values fluctuating between 0.51 and 0.69 mg/L. Despite minor

variations, this consistent detection attests to the system's dependability and reproducibility under these conditions.

Trials 14 to 16: The Impact of PFC 3M FC-770

In Trials 14 to 16, PFC 3M FC-770 was introduced to the system. Each trial utilized consistent volumes of ammonium hydroxide, measuring $757.00 \pm 5.0 \mu\text{L}$. The ammonia readings in these trials consistently registered values above zero, which significantly differed from the negligible readings observed in Trials 11 through 13, conducted without PFC. The ammonia readings for Trials 14 through 16 ranged from 2.15 to 2.45 mg/L, highlighting a potential interaction or effect facilitated by PFC 3M FC-770. The data implies that adding PFC could alter ammonia's behavior within this system, possibly affecting its solubility, detectability, or diffusion efficiency across the PDMS filter.

In summary, the trials conducted underscore the complex relationship between ammonium hydroxide volume and the presence of PFC 3M FC-770 in influencing ammonia readings. The noticeable variations in ammonia measurements across different trials emphasize the nuanced equilibrium of contributing elements. This comprehensive analysis provides valuable insights, laying a foundation for subsequent investigations and potential improvements, enhancing our understanding of ammonia behavior under diverse experimental conditions.

§ 3.30. Gas-Pathway Ammonia Readings Across All Trials and Phases

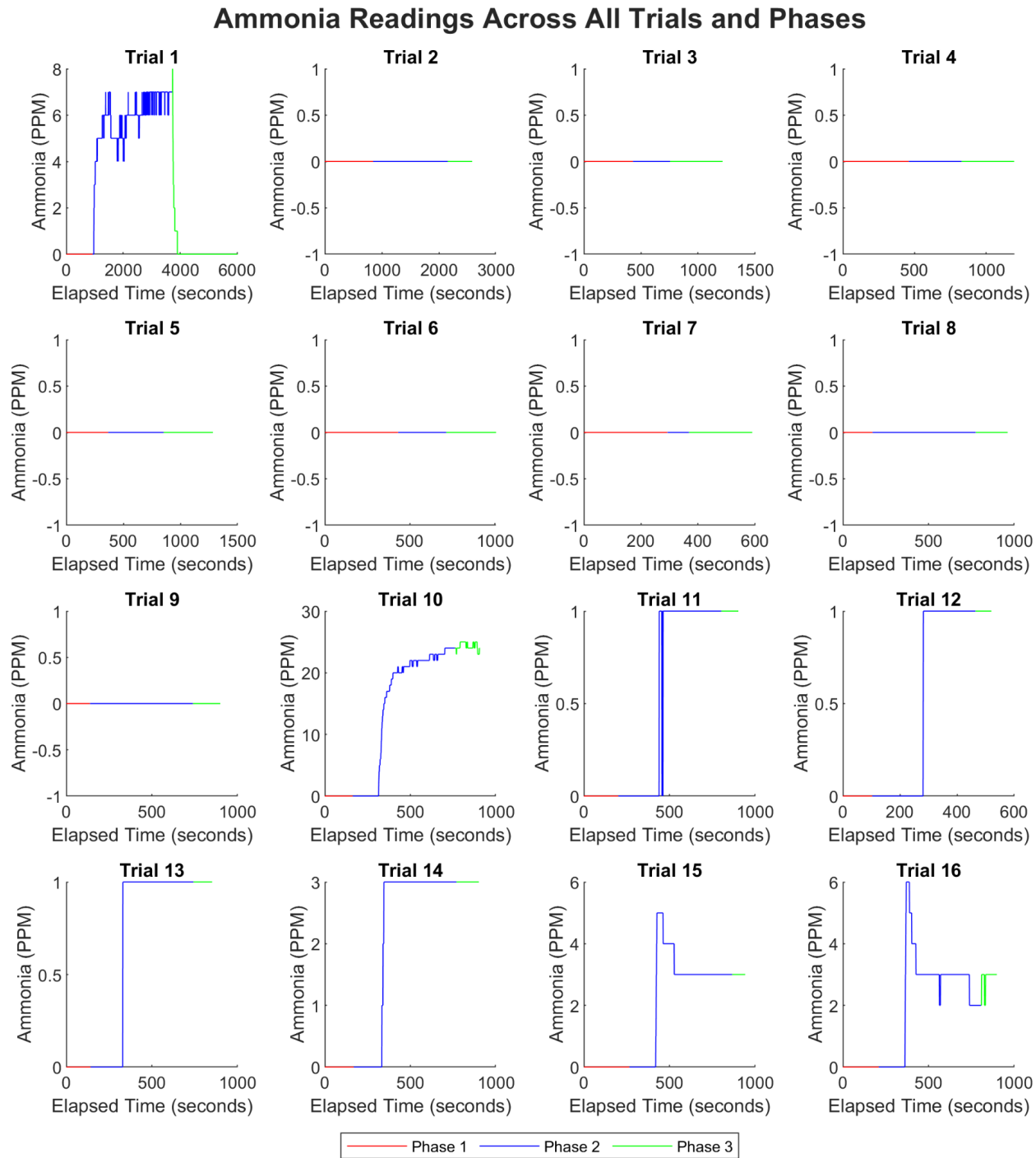


Figure 17. Gas-Pathway Ammonia Readings Across All Trials and Phases. This figure presents the ammonia readings across all trials and phases. Each subplot corresponds to a specific trial, as indicated by its title. The x-axis represents the elapsed time in seconds, and the y-axis depicts the ammonia concentration in PPM (parts per million). Three distinct colors, red, blue, and green, represent Phases 1, 2, and 3 readings, respectively.

§ 3.30.1. Discussion of Gas-Pathway Ammonia Readings Across All Trials and Phases

The graphs of Sensor1GasReading provide visual insight into the ammonia levels across different trials and phases. Each line plot in the graph shows the ammonia gas reading for a given trial across its phases. It allows for easy comparison between the trials, revealing trends, anomalies, and the possible influence of factors like PFC 3M FC-770. The following are some discussion points to consider:

- **Low to Negligible Readings:** Trials 2 through 9 and 11 to 13 predominantly showcase line plots hugging the baseline, indicating minimal ammonia readings. This observation aligns with the tabulated data, confirming these trials' near-zero or extremely low ammonia gas readings.
- **Significance of PFC 3M FC-770:** A notable increase can be seen in the line plots for Trials 14 to 16, where ammonia readings fluctuate between 2.15 mg/L to 2.45 mg/L. The introduction of PFC 3M FC-770 in these trials could be the driving factor behind these elevated levels.
- **Notable Deviation in Trial 10:** The line plot for Trial 10 shows a significant spike, with ammonia readings reaching 15.37 mg/L. This deviation is more pronounced when compared to the plots of adjacent trials.
- **Initial Observations and Ammonium Hydroxide:** The line plot of Trial 1 distinctly highlights an ammonia reading of 5.86 mg/L. As detailed in the data, this rise can be tied back to the volume of ammonium hydroxide introduced.
- **Consistent Conditions and Their Effects:** The line plots of Trials 11 to 13 show similar patterns, reflective of their consistent conditions, specifically the absence of PFC and

comparable volumes of ammonium hydroxide. The data tables validate these visual cues, with ammonia levels between 0.51 mg/L and 0.69 mg/L.

- **Trials with PFC 3M FC-770:** Examining the line plots of Trials 14 to 16 with the data table highlights the potential interactions or effects caused by PFC 3M FC-770. The plots and the data suggest readings spanning 2.15 mg/L to 2.45 mg/L, standing out from the other trials without PFC.
- **Physiological Implications:** While both the line plots and the data table indicate specific trials where ammonia levels exceed typical physiological ranges, it is crucial to contextualize these readings within the constraints of the experimental setup and conditions.
- **Consideration of Other Parameters:** Even as the focus is on ammonia readings, other parameters, such as pressure, temperature, and flow rate, must be considered. Variables like `phase2gasTemperatureC` and `phase2volumetricFlowRateStandardizedSCCM` across various trials may play pivotal roles in shaping the results. Even though these parameters are relatively consistent, slight variances could influence the readings.

In summary, the line plots and the tabulated data offer an in-depth glimpse into the factors impacting ammonia readings across trials. The conjunction of visual and numerical data paves the way for a comprehensive understanding, emphasizing the need for further exploration and potential subsequent experiments, which is discussed in the next section.

§ 3.31. Pairwise Comparisons for Similar Ammonium Hydroxide Concentrations

This section aims to discern the effects of different factors, such as PFC, on the ammonia readings when the concentration of ammonium hydroxide is kept constant or similar. Trials with comparable volumes of ammonium hydroxide were identified and compared.

Identification of Comparable Trials

The dataset was examined to identify trials with similar volumes of ammonium hydroxide. The following pairs and groups of trials were identified based on their ammonium hydroxide concentrations:

- **Trials [1, 10]:** Approximately 1515.00 μL of ammonium hydroxide.
- **Trials [11, 12, 13]:** Approximately 151.40 μL of ammonium hydroxide.
- **Trials [14, 15, 16]:** Approximately 757.00 μL of ammonium hydroxide.

Statistical Comparisons

For each of the identified pairs or groups, pairwise t-tests were conducted to determine if there were statistically significant differences in their ammonia readings despite having similar ammonium hydroxide concentrations. (Two-sample t-test, 2023) The following are the findings:

Trials [1, 10]:

Comparison between Trial 1 and Trial 10 resulted in a p-value of 0, indicating a statistically significant difference between the outcomes of these trials despite having

nearly identical ammonium hydroxide volumes. This p-value of 0 strongly suggests that the variations observed are not due to random chance.

Trials [11, 12, 13]:

For Trial 11 vs. Trial 12, the p-value was 4.3721e-05.

For Trial 11 vs. Trial 13, the p-value was 1.1782e-05.

For Trial 12 vs. Trial 13, the p-value was 1.8034e-15.

These low p-values point to significant differences between the outcomes of these trials, even with their almost identical ammonium hydroxide concentrations.

Trials [14, 15, 16]:

For Trial 14 vs. Trial 15, the p-value was 2.1386e-07.

For Trial 14 vs. Trial 16, the p-value was 0.015725.

For Trial 15 vs. Trial 16, the p-value was 0.009133.

All these comparisons indicate statistically significant differences between these trials' outcomes despite being grouped based on similar ammonium hydroxide concentrations.

In summary, the pairwise comparisons highlight significant differences in ammonia readings between trials, even when the ammonium hydroxide concentrations are closely matched. This finding suggests that factors other than ammonium hydroxide concentration, such as the presence or absence of PFC or other conditions, could influence the results. Further analysis would be necessary to isolate and identify these factors.

§ 3.32. Summary Table

Table 17. Phase 2, Roller Pump On, Vacuum Pump On, Summary Table

<u>trial</u>	Duration (HH:mm:ss)	Liquid Flow Rate (mL/min)	Liquid-Loop Before PDMS Filter (PSIA)	Gas-Pathway Volumetric Flow Rate (SCCM)	Gas-Pathway Before PDMS Filter (PSIA)	Vol. RODI Water (mL) ± 0.1	Vol. PFC 3M FC-770 (mL) ± 0.1	Vol. Ammonium Hydroxide (NH ₄ OH) (uL or µL)	Gas-Pathway Ammonia (NH ₃) (PPM = mg/L)
1	00:46:40	5.65 ± 0.16	16.87 ± 0.87	36.99 ± 2.04	1.51 ± 0.06	100.00	0.00	1515.00 ± 5.0	5.86 ± 1.20
2	00:21:46	6.34 ± 0.10	16.07 ± 0.71	22.49 ± 34.52	1.60 ± 0.52	100.00	0.00	15.14 ± 0.075	0.00 ± 0.00
3	00:05:25	6.34 ± 0.10	16.09 ± 0.75	10.13 ± 9.87	1.47 ± 0.34	100.00	0.00	22.70 ± 0.075	0.00 ± 0.00
4	00:06:07	6.34 ± 0.10	16.02 ± 0.69	9.93 ± 9.38	1.47 ± 0.38	100.00	0.00	30.28 ± 0.075	0.00 ± 0.00
5	00:08:06	6.34 ± 0.10	16.09 ± 0.67	9.50 ± 4.17	1.45 ± 0.10	100.00	0.00	37.85 ± 0.075	0.00 ± 0.00
6	00:04:42	6.34 ± 0.10	16.08 ± 0.70	9.76 ± 7.28	1.45 ± 0.18	100.00	0.00	45.42 ± 0.075	0.00 ± 0.00
7	00:01:15	6.34 ± 0.10	16.11 ± 0.75	10.45 ± 9.68	1.47 ± 0.23	100.00	0.00	52.99 ± 0.075	0.00 ± 0.00
8	00:10:02	6.34 ± 0.10	16.94 ± 0.72	94.23 ± 4.74	2.07 ± 0.15	100.00	0.00	0.00 ± 0.00	0.00 ± 0.00
9	00:10:01	6.34 ± 0.10	17.17 ± 0.70	94.04 ± 2.03	2.07 ± 0.05	100.00	0.00	15.14 ± 0.075	0.00 ± 0.00
10	00:09:57	6.34 ± 0.10	17.14 ± 0.94	94.43 ± 4.78	2.09 ± 0.19	100.00	0.00	1514.00 ± 5.0	15.37 ± 9.58
11	00:10:01	6.34 ± 0.10	19.23 ± 0.80	94.44 ± 4.81	2.10 ± 0.25	100.00	0.00	151.40 ± 5.0	0.60 ± 0.49
12	00:06:01	4.34 ± 0.12	18.16 ± 0.80	94.16 ± 2.53	2.08 ± 0.06	100.00	0.00	151.40 ± 5.0	0.51 ± 0.50
13	00:10:00	4.34 ± 0.12	16.47 ± 0.72	93.67 ± 3.87	2.07 ± 0.10	100.00	0.00	151.40 ± 5.0	0.69 ± 0.46
14	00:09:59	6.34 ± 0.10	20.46 ± 1.22	94.69 ± 4.17	2.09 ± 0.10	50.00	50.00	757.00 ± 5.0	2.15 ± 1.34
15	00:09:59	4.34 ± 0.12	18.64 ± 1.57	94.09 ± 2.27	2.08 ± 0.05	50.00	50.00	757.00 ± 5.0	2.45 ± 1.54
16	00:09:57	2.42 ± 0.19	16.81 ± 1.48	93.84 ± 2.84	2.07 ± 0.07	50.00	50.00	757.00 ± 5.0	2.29 ± 1.54

Some data adapted from (DiBeneditto, 2023)

§ 3.32.1. Discussion of Summary Table

The following are some observations based on the above table:

- **Consistent Liquid Flow Rate:** Most trials, specifically Trials 2 through 11, exhibit a consistent liquid flow rate of 6.34 ± 0.10 mL/min. However, Trials 12 to 16 display variations with flow rates deviating from this consistent measurement.

- **Diverse Duration of Trials:** The durations for each trial significantly differ. For instance, Trial 1 has a notably longer duration (0:46:40) than some other trials, like Trial 7 (0:01:15).
- **Gas-Pathway Volumetric Flow Rate Surge:** The Gas-Pathway volumetric flow rates for Trials 1 to 7 are significantly lower than those of Trials 8 and onwards, which consistently hover around 94 SCCM.
- **Gas-Pathway Pressure Variation:** There are evident fluctuations in the Gas-Pathway Before PDMS Filter (PSIA) across trials. Notably, Trial 14 exhibits a higher pressure of 20.46 ± 1.22 PSIA.
- **Consistent RODI Water Volume and Two-Phase Testing:** The volume of RODI water remains consistent at 100.00 mL for Trials 1 through 13. However, in Trials 14 to 16, this volume is reduced to 50 mL, with an equal addition of PFC 3M FC-770, hinting at a two-phase testing approach.
- **Ammonium Hydroxide Variation:** The volume of ammonium hydroxide added varies across trials. For instance, while Trial 1 used a significant volume of 1515.00 ± 5.0 μ L, Trial 8 did not use any.
- **Ammonia Capture in Gas Phase:** The Gas-Pathway Ammonia (NH₃) measurements offer insights into system performance. For clarity, specific ammonia levels were recorded across trials, with some trials, such as Trial 1, showing 5.86 ± 1.20 PPM and Trial 10 showing 15.37 ± 9.58 PPM.

In summary, the table offers insights into the operational parameters of an ammonia removal system. It showcases how specific parameters, from flow rates to added volumes of reagents, can

influence the system's performance, emphasizing the potential of this method for Acute Liver Failure management treatment.

§ 3.33. Additional Insights and Potential Areas of Improvement

The following are some additional insights and potential areas of Improvement:

- **Optimization for Efficiency:** Although the liquid flow rate remains consistent across many trials, the efficiency of ammonia removal varies across trials. It might be beneficial to analyze the contributions of the PDMS filter, PFC, and vacuum pressure to determine optimal conditions that maximize ammonia removal.
- **Correlation with Duration:** The varied trial durations could be a deliberate attempt to understand the time-dependency of ammonia removal. Investigating how trial duration affects ammonia removal efficiency could provide insights into the influence of exposure time on system performance.
- **Potential Limiting Factors:** Observations of elevated ammonia (mg / L) levels in specific trials (Trial 1: 5.86 ± 1.20 , Trial 10: 15.37 ± 9.58 , Trial 15: 2.45 ± 1.54) suggest possible constraints within the system. These constraints may arise from the limited capacity of the PDMS filter to effectively remove ammonia gas or challenges related to the diffusion rate of ammonia from the ammonium hydroxide and its subsequent capture by the PFC. However, it is essential to consider external factors and the overall system design to determine if these are genuine constraints or results influenced by variable conditions.

- **Significance of Vacuum Pressure:** Investigating the relationship between variations in the Gas-Pathway Before the PDMS Filter (PSIA) and ammonia removal efficiency could elucidate the role and significance of vacuum pressure in the process.
- **Role of PFC:** The introduction of PFC in specific trials aims to assist in the diffusion and capture of ammonia. PFC and ammonia diffusion interplay is crucial for understanding and optimizing the system's overall efficiency.
- **Real-world Applicability:** The ultimate goal is application in Acute Liver Failure treatment. Evaluating the system's practicality, safety, and scalability in real-world conditions is essential.
- **External Variables:** Factors like ambient temperature or the presence of other solutes might influence the system's efficiency. Recognizing and accounting for these external influences is critical for a thorough analysis.
- **Future Iterations:** To boost system performance, there is room for refining certain elements, such as expanding the PDMS filter's surface area or employing medical-grade PFC.

In summary, data offers a comprehensive overview of the system's performance. However, to fully appreciate the findings and their broader implications, it is necessary to account for all variables and consider this system's real-world challenges and applications.

§ 3.34. SUMMARY OF DATA ANALYSIS

The core objective of our study was to evaluate the efficiency and feasibility of a system incorporating a PDMS filter (representative of artificial liver technology) in tandem with perfluorocarbon (PFC, representative of artificial lung technology) to remove ammonia from a liquid solution in a simulated peritoneal environment.

§ 3.34.1. Central Questions

To investigate the potential application for acute liver failure (ALF) patients and attempt to answer two central questions:

1. Can the PDMS filter effectively facilitate the diffusion of ammonia from the liquid to the gas phase, thereby removing it?
2. Does the presence of PFC enhance or influence this ammonia removal process?

§ 3.34.2. Central Findings

Our data analysis has clarified several aspects of these questions:

- We observed expected notable differences in ammonia gas levels (Sensor1GasReading) between Phases 1 and 2 due to the vacuum pump states across these phases.

(Question 1)

- The residual ammonia gas from Phase 2 might account for the significant variance and statistical tests in readings observed between Phases 1 and 3, even though the vacuum pump was off in both phases. **(Question 1)**

- Correlation studies for Phase 1 showed no correlation due to an absence of detected ammonia, consistent with our understanding of the system's initial state. Phase 2 had a weak positive correlation between flow rate and ammonia levels. Phase 3 again yielded a NaN correlation, signifying no detected ammonia. **(Question 1)**
- A statistically significant correlation in Phase 2 pointed towards an increase in ammonia gas readings with an increase in the volume of ammonium hydroxide, suggesting a genuine relationship between the amount of ammonium hydroxide liquid introduced, ammonia gas diffusing through the PDMS filter, entering the Gas-Pathway at vacuum pressure, and then the amount of ammonia gas detected by the ammonia gas levels.
(Question 1)
- Adjusted average ammonia gas readings indicated a higher reading with PFC than without, indicating more dissolved ammonia in the PFC solution. However, the lack of statistical significance, possibly due to the small sample size, implies that our data does not conclusively support the hypothesis that PFC enhances ammonia removal.
(Question 2)
- Trials with equivalent ammonium hydroxide volumes showed significant differences in results. These differences underscore the multi-faceted nature of the experiment and hint at the possible influence of factors beyond just the amount of ammonium hydroxide introduced. The liquid flow rate may have contributed to this.

§ 3.34.3. Conclusion

In conclusion, our data provides substantial evidence regarding the efficacy of the PDMS filter in facilitating ammonia removal. However, the role of PFC in this process remains unclear and requires further investigation. The complexity of the experiment and the variations observed in trials with similar parameters suggest we need additional studies to understand the variables we are exploring fully. Future research would benefit from delving deeper into these variations and optimizing conditions for improved ammonia removal. The author would like to express profound gratitude to everyone who contributed to the completion of this study; thank you again.

§ 3.35. DISCLAIMER

This document has been developed for academic and research purposes only. The contents herein represent the author's views and findings, not those of the institution, the advisor, or the thesis committee.

The content and findings within this document do not constitute legal, professional, or medical advice, nor are they intended to replace such advice. The author is not a licensed medical professional, and the information contained herein should not be interpreted as medical advice or instructions.

All information contained in this document has been researched, analyzed, and presented with the utmost care and accuracy. However, it is the reader's responsibility to verify all information before making decisions based on it.

This document, content, and findings should not be used as a basis for medical diagnosis, treatment, or other health-related decisions. Readers are strongly advised to consult a qualified healthcare professional for health-related concerns.

The author and the institution do not accept any responsibility for any death, damage, or loss caused by reliance on the information contained within this document. (DiBeneditto, 2023)

CHAPTER 4: BIBLIOGRAPHY

1.8M (6') 8 pin male Mini-DIN to DB9 female adapter [Internet]. Omega Engineering, Inc.;

2023 [Cited 2023 Jun 14]. Available from:

<https://www.omega.com/en-us/flow-instruments/flow-meters/mass-flow-meters/p/FMA1600-C3>.

3M Fluorinert Electronic Liquid FC-770 | 3M United States [Internet]. Electronics Materials

Solutions Division, 3M; 2023 [Cited 2022 Jul 3]. Available from:

https://www.3m.com/3M/en_US/p/d/b40006507/.

3M Fluorinert Electronic Liquid FC-770 3M Safety Data Sheet Document group: 22-7641-8

[Internet]. 3M Canada Company; 2021 Nov 17 [Cited 2022 Aug 17]. Available from:

https://multimedia.3m.com/mws/mediawebserver?mwsId=SSSSSuUn_zu8lZNUMx21O8tU5v70k17zHvu9lxtD7SSSSSS--.

3M Fluorinert Electronic Liquid FC-770 Regulatory Data Sheet [Internet]. Electronics Materials

Solutions Division, 3M; 2021 Jul 15 [Updated 2022 Jan 18; Cited 2022 Jul 3]. Available from:

https://multimedia.3m.com/mws/mediawebserver?mwsId=SSSSSu9n_zu8l00xMx21O8tU5v70k17zHvu9lxtD7xt1evSSSSSS-.

3M Fluorinert Electronic Liquid FC-770 Safety Data Sheet Document Group: 22-7641-8 Version

Number: 10.00 Issue Date: 01/03/23 Supersedes Date: 10/19/21 [Internet]. 3M; 2023 Jan 3 [Updated 2023 Apr 14; Cited 2023 Jul 31]. Available from:

https://multimedia.3m.com/mws/mediawebserver?mwsId=SSSSSuUn_zu8l00xMx21O8tU5v70k17zHvu9lxtD7SSSSSS--.

3M Fluorinert Electronic Liquid FC-770 Technical Data | September 2019 [Internet]. Electronics Materials Solutions Division, 3M; 2019 Oct 8 [Cited 2022 Jul 3]. Available from: <https://multimedia.3m.com/mws/media/471785O/3m-fluorinert-electronic-liquid-fc770-product-info-sheet-en.pdf>.

3M Fluorinert Electronic Liquid FC-770, 6.35 kg (14 lb), 1 Bottle/Case | 3M United States [Internet]. 3M; 2023 [Cited 2023 Jul 31]. Available from: https://www.3m.com/3M/en_US/p/dc/v000528116/.

Adeva MM, Souto G, Blanco N, Donapetry C. Ammonium metabolism in humans. *Metabolism*. 2012 Nov;61(11):1495-511. doi: 10.1016/j.metabol.2012.07.007. Epub 2012 Aug 24. PMID: 22921946. Available from: <https://pubmed.ncbi.nlm.nih.gov/22921946/>.

AK672MB Assmann WSW Components | Cable Assemblies | DigiKey [Internet]. DigiKey; 2023 [Cited 2023 Jan 18]. Available from: <https://digikey.com/en/products/detail/assmann-wsw-components/AK672MB/857941>.

Amazon.com: AUTEX Pressure Transducer Sensor 100 Psi Pressure Sender 316 Stainless Steel Oil Pressure Transmitter 1/8" -27 NPT For Oil Fuel Air Water Pressure : Automotive [Internet]. Amazon.com, Inc.; 2023 [Cited 2023 Jan 26]. Available from: <https://www.amazon.com/gp/product/B00NIK9E10/>.

Amazon.com: Capelli Sport Digital Stopwatch Timer, Sports Stopwatch Timer with Large Display Screen, Blue/Black : Sports & Outdoors [Internet]. Amazon.com, Inc.; 2023 [Cited 2023 Mar 24]. Available from: <https://www.amazon.com/-/he/CSEF-1014/dp/B09XPD2HXX>.

Amazon.com: USB 3.0 Hub, VIENON 4-Port USB Hub USB Splitter USB Expander for Laptop, Xbox, Flash Drive, HDD, Console, Printer, Camera, Keyboard, Mouse : Electronics

[Internet]. Amazon.com, Inc.; 2023 [Cited 2023 Mar 24]. Available from:

<https://www.amazon.com/Extender-Splitter-MacBook-Chromebook-Pixelbook/dp/B09MLRPTT2>.

Analysis of variance for linear regression model - MATLAB anova [Internet]. The MathWorks, Inc.; 2023 [Cited 2023 Sep 24]. Available from:

<https://www.mathworks.com/help/stats/linearmodel.anova.html>.

Bauer K, Janke T, Schwarze R. Oxygen transport during liquid ventilation: an in vitro study. *Sci*

Rep. 2022 Jan 24;12(1):1244. doi: 10.1038/s41598-022-05105-1. PMID: 35075158;

PMCID: PMC8786849. Available from: <https://doi.org/10.1038/s41598-022-05105-1>.

Available from: <https://pubmed.ncbi.nlm.nih.gov/35075158/>. Available from:

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8786849/>. Available from:

<https://www.nature.com/articles/s41598-022-05105-1>.

Carr SR, Cantor JP, Rao AS, Lakshman TV, Collins JE, Friedberg JS. Peritoneal perfusion with oxygenated perfluorocarbon augments systemic oxygenation. *Chest*. 2006

Aug;130(2):402-11. doi: 10.1378/chest.130.2.402. PMID: 16899838; PMCID:

PMC7126298. Available from: <https://doi.org/10.1378/chest.130.2.402>. Available from:

<https://pubmed.ncbi.nlm.nih.gov/16899838/>. Available from:

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7126298/>. Available from:

[https://journal.chestnet.org/article/S0012-3692\(15\)51854-0/fulltext](https://journal.chestnet.org/article/S0012-3692(15)51854-0/fulltext).

Castro CI, Briceno JC. Perfluorocarbon-based oxygen carriers: review of products and trials.

Artif Organs. 2010 Aug;34(8):622-34. doi: 10.1111/j.1525-1594.2009.00944.x. PMID:

20698841. Available from: <https://doi.org/10.1111/j.1525-1594.2009.00944.x>. Available

from: <https://pubmed.ncbi.nlm.nih.gov/20698841/>. Available from:

<https://onlinelibrary.wiley.com/doi/10.1111/j.1525-1594.2009.00944.x>.

Chou CY, Wang SM, Liang CC, Chang CT, Liu JH, Wang IK, Hsiao LC, Muo CH, Chung CJ, Huang CC. Peritoneal Dialysis is Associated With A Better Survival in Cirrhotic Patients With Chronic Kidney Disease. *Medicine (Baltimore)*. 2016 Jan;95(4):e2465. doi: 10.1097/MD.0000000000002465. PMID: 26825885; PMCID: PMC5291555. Available from: <https://doi.org/10.1097/MD.0000000000002465>. Available from: <https://pubmed.ncbi.nlm.nih.gov/26825885/>. Available from: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5291555/>. Available from: https://journals.lww.com/md-journal/fulltext/2016/01250/peritoneal_dialysis_is_associated_with_a_better.11.aspx.

Choudhary M, Clavica F, van Mastrigt R, van Asselt E. A novel single compartment in vitro model for electrophysiological research using the perfluorocarbon FC-770. *Physiol Res*. 2016 Jun 20;65(2):341-8. doi: 10.33549/physiolres.933099. PMID: 27322010. Available from: <https://doi.org/10.33549/physiolres.933099>. Available from: http://www.biomed.cas.cz/physiolres/pdf/65/65_341.pdf

Correlation coefficients - MATLAB corrcoef [Internet]. The MathWorks, Inc.; 2023 [Cited 2023 Oct 2]. Available from: <https://www.mathworks.com/help/matlab/ref/corrcoef.html>.

Cove ME, Vu LH, Ring T, May AG, Federspiel WJ, Kellum JA. A Proof of Concept Study, Demonstrating Extracorporeal Carbon Dioxide Removal Using Hemodialysis with a Low Bicarbonate Dialysate. *ASAIO J*. 2019 Aug;65(6):605-613. doi: 10.1097/MAT.0000000000000879. PMID: 30281542. Available from: <https://doi.org/10.1097/mat.0000000000000879>. Available from:

<https://pubmed.ncbi.nlm.nih.gov/30281542/>. Available from:

https://journals.lww.com/asaiojournal/fulltext/2019/08000/a_proof_of_concept_study_demonstrating.14.aspx.

Dabaghi M, Saraei N, Fusch G, Rochow N, Brash JL, Fusch C, Ravi Selvaganapathy P. An ultra-thin, all PDMS-based microfluidic lung assist device with high oxygenation capacity. *Biomicrofluidics*. 2019 Jun 27;13(3):034116. doi: 10.1063/1.5091492. PMID: 31263515; PMCID: PMC6597343. Available from: <https://doi.org/10.1063/1.5091492>. Available from: <https://pubmed.ncbi.nlm.nih.gov/31263515/>. Available from: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6597343/>. Available from: <https://pubs.aip.org/aip/bmf/article-abstract/13/3/034116/134101/An-ultra-thin-all-PDMS-based-microfluidic-lung>.

DiBeneditto LW. Temporary Liver Support Using Artificial Lung Technologies: Ammonia Removal via Peritoneal Dialysis for Acute Liver Failure with PFC and PDMS at Vacuum Pressure [master's thesis]. [Pittsburg (PA)]: Carnegie Mellon University; 2023 Aug. 333 p. ISBN: 9798380335881. Carnegie Mellon University ProQuest Dissertations Publishing. ProQuest document ID: 2864402381. doi: 10.1184/R1/24130056.v1. Available from: <https://doi.org/10.1184/R1/24130056.v1>. Available from: <https://www.proquest.com/docview/2864402381>. Available from: <https://www.andrew.cmu.edu/user/lwd/files/20230808-thesis-temporary-liver-support-using-artificial-lung-technologies-final.pdf>.

Download Vernier Graphical Analysis® - Vernier [Internet]. Vernier Science Education; 2023 [Cited 2023 Feb 18]. Available from: <https://www.vernier.com/downloads/graphical-analysis/>.

Exploratory Analysis of Data - MATLAB & Simulink [Internet]. The MathWorks, Inc.; 2023

[Cited 2023 Sep 25]. Available from:

<https://www.mathworks.com/help/stats/exploratory-analysis-of-data.html>.

Fit linear regression model - MATLAB fitlm [Internet]. The MathWorks, Inc.; 2023 [Cited 2023

Sep 23]. Available from: <https://www.mathworks.com/help/stats/fitlm.html>.

Fluorinert FC-770 Product Specification Sigma-Aldrich [Internet]. Sigma-Aldrich; 2015 Aug 31

[Cited 2023 Apr 12]. Available from:

https://www.sigmaaldrich.com/specification-sheets/398/535/F3556-BULK_____ALDRICH_.pdf.

Fluorinert FC-770, 3M - ChemPoint [Internet]. ChemPoint; 2022 [Cited 2022 Aug 29]. Available from:

<https://www.chempoint.com/products/3m/3m-novec-engineered-fluids/3m-fluorinert-electronic-liquids/fluorinert-fc-770>.

FMA-1618A | FMA1600 | Mass and Volumetric Flow Meters | Omega Engineering [Internet].

Omega Engineering, Inc.; 2023 [Cited 2023 Feb 28]. Available from:

<https://www.omega.com/en-us/flow-instruments/flow-meters/mass-flow-meters/fma1600-series/p/FMA-1618A>.

Free Utility: RichCopy, an Advanced Alternative to RoboCopy | Microsoft Learn [Internet].

Microsoft; 2016 Sep 8 [Cited 2023 Aug 26]. Available from:

[https://learn.microsoft.com/en-us/previous-versions/technet-magazine/dd547088\(v=msdn.10\)](https://learn.microsoft.com/en-us/previous-versions/technet-magazine/dd547088(v=msdn.10)).

Friedman's test - MATLAB friedman [Internet]. The MathWorks, Inc.; 2023 [Cited 2023 Sep

12]. Available from: <https://www.mathworks.com/help/stats/friedman.html>.

Go Direct Ammonium Ion-Selective Electrode - Vernier [Internet]. Vernier Science Education; 2023 [Cited 2023 Mar 24]. Available from:

<https://www.vernier.com/product/go-direct-ammonium-ion-selective-electrode/>.

Google Docs: Online Document Editor | Google Workspace [Internet]. Google; 2023 [Updated 2023 Apr 26; Cited 2023 Mar 11]. Available from: <https://www.google.com/docs/about/>.

Google Sheets: Online Spreadsheet Editor | Google Workspace [Internet]. Google; 2023 [Updated 2023 Apr 26; Cited 2023 Jan 10]. Available from:

<https://www.google.com/sheets/about/>.

HONEYWELL, Ammonia (NH₃), HONEYWELL RAE Systems ToxiRAE Pro, Single-Gas Detector - 498Z32[G02-B810-100 - Grainger [Internet]. W.W. Grainger, Inc.; 2023 [Cited 2023 Feb 14]. Available from:

<https://www.grainger.com/product/HONEYWELL-Single-Gas-Detector-Ammonia-498Z32>.

Jeng MJ, Yang SS, Wolfson MR, Shaffer TH. Perfluorochemical (PFC) combinations for acute lung injury: an in vitro and in vivo study in juvenile rabbits. *Pediatr Res*. 2003

Jan;53(1):81-8. doi: 10.1203/00006450-200301000-00015. PMID: 12508085. Available from: <https://doi.org/10.1203/00006450-200301000-00015>. Available from:

<https://pubmed.ncbi.nlm.nih.gov/12508085/>. Available from:

<https://www.nature.com/articles/pr200315>.

Khan S, Rosner MH. Peritoneal Dialysis for Patients with End-Stage Renal Disease and Liver Cirrhosis. *Perit Dial Int*. 2018 Nov-Dec;38(6):397-401. doi: 10.3747/pdi.2018.00008.

PMID: 30413635. Available from: <https://doi.org/10.3747/pdi.2018.00008>. Available

from: <https://pubmed.ncbi.nlm.nih.gov/30413635/>. Available from:
<https://journals.sagepub.com/doi/10.3747/pdi.2018.00008>.

LabQuest 3 - Vernier [Internet]. Vernier Science Education; 2023 [Cited 2023 Apr 11]. Available from: <https://www.vernier.com/product/labquest-3/>.

LabQuest 3 - Vernier [Internet]. Vernier Science Education; 2023 [Cited 2023 Apr 11]. Available from: <https://www.vernier.com/product/labquest-3/>.

Lanaro R, De Capitani EM, Costa JL, Bucarechi F, Togni L, Linden R, Barbosa F, Tessaro EP, Bataglion GA, Eberlin MN and Zappa JEB. Sudden deaths due to accidental intravenous injection of perfluorocarbon during MRI cranial examinations. Forensic toxicology. 2014;32:pp. 323–330. doi: 10.1007/s11419-014-0231-z. Available from:
<https://doi.org/10.1007/s11419-014-0231-z>. Available from:
<https://link.springer.com/article/10.1007/s11419-014-0231-z>.

Lilliefors test - MATLAB lillietest [Internet]. The MathWorks, Inc.; 2023 [Cited 2023 Sep 16]. Available from: <https://www.mathworks.com/help/stats/lillietest.html>.

Linear or rank correlation - MATLAB corr [Internet]. The MathWorks, Inc.; 2023 [Cited 2023 Sep 20]. Available from: <https://www.mathworks.com/help/stats/corr.html>.

Looking for RoboCopy GUI and RichCopy - Microsoft Q&A [Internet]. Microsoft; 2022 Dec 17 [Cited 2023 Aug 26]. Available from:
<https://learn.microsoft.com/en-us/answers/questions/1132681/looking-for-robocopy-gui-and-richcopy>.

MATLAB [Internet]. The MathWorks, Inc.; 2023 [Cited 2023 Jul 31]. Available from:
<https://www.mathworks.com/products/matlab.html>.

Mohiuddin SS, Khattar D. Biochemistry, Ammonia. [Updated 2022 Feb 25; Cited 2022 Aug 15].

In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2022 Jan-. Available from: <https://www.ncbi.nlm.nih.gov/books/NBK541039/>.

Multiple Comparisons - MATLAB & Simulink [Internet]. The MathWorks, Inc.; 2023 [Cited 2023 Sep 8]. Available from:

<https://www.mathworks.com/help/stats/multiple-comparisons.html>.

Nissenson AR, Fine RN. Handbook of Dialysis Therapy. 5th ed. Philadelphia, PA, USA:

Elsevier; v2017. 1266 p. ISBN: 9780323391542. Available from:

<https://www.sciencedirect.com/book/9780323391542/handbook-of-dialysis-therapy>.

One-sample Kolmogorov-Smirnov test - MATLAB kstest [Internet]. The MathWorks, Inc.; 2023 [Cited 2023 Sep 10]. Available from: <https://www.mathworks.com/help/stats/kstest.html>.

Perfluoro-compound FC-770(TM), Thermo Scientific | Fisher Scientific [Internet]. Thermo

Fisher Scientific Inc; 2022 [Cited 2022 Aug 17]. Available from:

<https://www.fishersci.pt/shop/products/perfluoro-compound-fc-770-tm-thermo-scientific/10412785>.

pH Sensor - Vernier [Internet]. Vernier Science Education; 2023 [Cited 2023 Feb 08]. Available from: <https://www.vernier.com/product/ph-sensor/>.

Plugable USB to RS-232 DB9 Serial Adapter (Prolific PL2303HX Chipset) – Plugable

Technologies [Internet]. Plugable Technologies; 2023 [Updated 2023 Jul 24; Cited 2023 May 25]. Available from: <https://plugable.com/products/pl2303-db9>.

Ponce D, Zamonier W, Dias DB, Pires da Rocha E, Kojima C, Balbi AL. The Role of Peritoneal Dialysis in the Treatment of Acute Kidney Injury in Patients With Acute-on-Chronic Liver Failure: A Prospective Brazilian Study. *Front Med (Lausanne)*. 2021 Sep

23;8:713160. doi: 10.3389/fmed.2021.713160. PMID: 34631735; PMCID: PMC8496932. Available from: <https://doi.org/10.3389/fmed.2021.713160>. Available from: <https://pubmed.ncbi.nlm.nih.gov/34631735/>. Available from: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8496932/>. Available from: <https://www.frontiersin.org/articles/10.3389/fmed.2021.713160/full>.

PRW3500T-7 | Titanium Watch - PRO TREK | CASIO [Internet]. Casio America, Inc.; 2023 [Cited 2023 Jan 26]. Available from: <https://www.casio.com/us/watches/protrek/product.PRW-3500T-7/>.

RAE Systems Cradle for Charging and PC Communications (AC adapter or PC cable NOT included) - G02-3008-000 [Internet]. JJS Technical Services; 2023 [Cited 2023 Mar 01]. Available from: <https://www.rae-gasmonitors.com/g02-3008-000.html>.

RAE Systems PC Communications Cable - 410-0203-000 [Internet]. JJS Technical Services; 2023 [Cited 2023 Apr 09]. Available from: <https://www.rae-gasmonitors.com/410-0203-000.html>.

RAE Systems Replacement Sensor C03-0950-000, Ammonia (NH₃) 0-100 ppm | RSHughes.com [Internet]. R.S. Hughes Co., Inc.; 2023 [Cited 2023 Jan 18]. Available from: https://www.rshughes.com/p/RAE-Systems-Replacement-Sensor-C03-0950-000-Ammonia-NH3-0-100-Ppm-For-Use-With-ToxiRAE-Pro-Electrochemical-Toxic-MultiRAE-Lite-MultiRAE-MultiRAE-Pro/c03_0950_000/.

Rastogi A, Lerma EV, Bargman JM (Editors). Applied Peritoneal Dialysis Improving Patient Outcomes. 1st ed. Cham, Switzerland: Springer Nature Switzerland; c2021. 471 p. doi:

10.1007/978-3-030-70897-9. ISBN: 9783030708962. ISBN: 9783030708979. Available from: <https://doi.org/10.1007/978-3-030-70897-9>.

REXQualis Super Starter Kit based on Arduino UNO R3 with Tutorial and Controller Board Compatible with Arduino IDE - Rexqualis Industries, Ingenious & fun DIY electronics and kits [Internet]. REXQualis, Inc.; 2017 [Cited 2023 Jun 05]. Available from: <https://www.rexqualis.com/product/uno-project-super-starter-kit-for-arduino-w-uno-r3-development-board-detailed-tutorial/>.

RichCopy HoffmanUtilitySpotlight2009_04.exe Archive Copy. Wayback Machine [Internet]. Microsoft; 2019 Sep 21 [Cited 2023 Aug 26]. Available from: https://web.archive.org/web/20190921232753/http://download.microsoft.com/download/f/d/0/fd05def7-68a1-4f71-8546-25c359cc0842/HoffmanUtilitySpotlight2009_04.exe.

Robinson JR, Conroy PC, Hardison D, Hamid R, Grubb PH, Pietsch JB, Lovvorn HN 3rd. Rapid resolution of hyperammonemia in neonates using extracorporeal membrane oxygenation as a platform to drive hemodialysis. *J Perinatol*. 2018 Jun;38(6):665-671. doi: 10.1038/s41372-018-0084-0. Epub 2018 Feb 21. PMID: 29467521; PMCID: PMC6030490. Available from: <https://doi.org/10.1038/s41372-018-0084-0>. Available from: <https://pubmed.ncbi.nlm.nih.gov/29467521/>. Available from: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6030490/>. Available from: <https://www.nature.com/articles/s41372-018-0084-0>.

robocopy | Microsoft Learn [Internet]. Microsoft; 2023 Sep 1 [Cited 2023 Aug 26]. Available from: <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/robocopy>.

Rudralingam V, Footitt C, Layton B. Ascites matters. *Ultrasound*. 2017 May;25(2):69-79. doi: 10.1177/1742271X16680653. Epub 2016 Dec 1. PMID: 28567101; PMCID: PMC5438051. Available from: <https://doi.org/10.1177/1742271X16680653>. Available from: <https://pubmed.ncbi.nlm.nih.gov/28567101/>. Available from: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5438051/>. Available from: <https://journals.sagepub.com/doi/10.1177/1742271X16680653>.

Safety Suite Solution | Honeywell [Internet]. Honeywell International Inc; 2023 Jun [Cited 2023 Jan 07]. Available from: <https://sps.honeywell.com/us/en/software/safety/connected-safety/safety-suite/safety-suite-device-configurator#download>.

Software | Arduino [Internet]. Arduino; 2023 [Updated 2023 Jul 18; Cited 2023 May 29]. Available from: <https://www.arduino.cc/en/software>.

Spieß BD. Perfluorocarbon Gas Transport: an Overview of Medical History With Yet Unrealized Potentials. *Shock*. 2019 Oct;52(1S Suppl 1):7-12. doi: 10.1097/SHK.0000000000001150. PMID: 29677087. Available from: <https://doi.org/10.1097/SHK.0000000000001150>. Available from: <https://pubmed.ncbi.nlm.nih.gov/29677087/>. Available from: https://journals.lww.com/shockjournal/abstract/2019/10001/perfluorocarbon_gas_transport__an_overview_of.3.aspx.

Stainless Steel Temperature Probe - Vernier [Internet]. Vernier Science Education; 2023 [Cited 2023 Jan 31]. Available from: <https://www.vernier.com/product/stainless-steel-temperature-probe/>.

Summar M, Pietsch J, Deshpande J, Schulman G. Effective hemodialysis and hemofiltration driven by an extracorporeal membrane oxygenation pump in infants with hyperammonemia. *J Pediatr*. 1996 Mar;128(3):379-82. doi: 10.1016/s0022-3476(96)70287-1. PMID: 8774510. Available from: [https://doi.org/10.1016/s0022-3476\(96\)70287-1](https://doi.org/10.1016/s0022-3476(96)70287-1). Available from: <https://pubmed.ncbi.nlm.nih.gov/8774510/>. Available from: [https://www.jpeds.com/article/S0022-3476\(96\)70287-1/fulltext](https://www.jpeds.com/article/S0022-3476(96)70287-1/fulltext).

Synchronize timetables to common time vector, and resample or aggregate data from input timetables - MATLAB synchronize [Internet]. The MathWorks, Inc.; 2023 [Cited 2023 Sep 9]. Available from: <https://www.mathworks.com/help/matlab/ref/timetable.synchronize.html>.

Tarn AC, Lapworth R. Biochemical analysis of ascitic (peritoneal) fluid: what should we measure? *Ann Clin Biochem*. 2010 Sep;47(Pt 5):397-407. doi: 10.1258/acb.2010.010048. Epub 2010 Jul 1. PMID: 20595402. Available from: <https://doi.org/10.1258/acb.2010.010048>. Available from: <https://pubmed.ncbi.nlm.nih.gov/20595402/>. Available from: <https://journals.sagepub.com/doi/full/10.1258/acb.2010.010048>.

Tatham S. Download PuTTY - a free SSH and telnet client for Windows [Internet]. Putty.org; 2022 [Updated 2022 Jun 5; Cited 2023 Jun 09]. Available from: <https://www.putty.org/>.

ThermoFisher SCIENTIFIC SAFETY DATA SHEET Perfluoro-compound FC-770. [Internet]. ThermoFisher Scientific; 2011 Mar 29 [Updated 2020 Dec 17; Cited 2022 Aug 17]. Available from:

<https://www.fishersci.co.uk/store/msds?partNumber=10704184&productDescription=50ML+Perfluoro-compound+FC-770&countryCode=GB&language=en>.

Two-sample t-test - MATLAB ttest2 [Internet]. The MathWorks, Inc.; 2023 [Cited 2023 Sep 26].

Available from: <https://www.mathworks.com/help/stats/ttest2.html>.

UNO R3 Board ATmega328P ATMEGA16U2 Development Board Compatible With Arduino -

Rexqualis Industries, Ingenious & fun DIY electronics and kits [Internet]. REXQualis,

Inc.; 2023 [Cited 2023 Mar 19]. Available from:

<https://www.rexqualis.com/product/uno-r3-board-atmega328p-atmega16u2-development-board-with-usb-cable-compatible-with-arduino-uno-r3-robot/>.

Visualize summary statistics with box plot - MATLAB boxplot [Internet]. The MathWorks, Inc.;

2023 [Cited 2023 Sep 17]. Available from:

<https://www.mathworks.com/help/stats/boxplot.html>.

Wilcoxon signed rank test - MATLAB signrank [Internet]. The MathWorks, Inc.; 2023 [Cited

2023 Sep 19]. Available from: <https://www.mathworks.com/help/stats/signrank.html>.

APPENDIX: MAIN.M MATLAB PROGRAM SOURCE CODE

The following is the MATLAB program source code that produced the figures, tables, and results presented in this document.

```
%% reset MATLAB workspace
close all;
clear;
clc;

% ---
% TITLE:
%   Preparation and Statistics MATLAB Program for Thesis Project Data
% AUTHOR:
%   Lukas W. DiBeneditto, https://dibeneditto.com/
% CREATED:
%   2023 Aug 08
% DESCRIPTION:
%   This MATLAB program prepares the recorded data and then runs various
%   statistics tests on that data for the Master of Science in Biomedical
%   Engineering Thesis titled "Temporary Liver Support Using Artificial
%   Lung Technologies: Ammonia Removal via Peritoneal Dialysis for Acute
%   Liver Failure with PFC and PDMS at Vacuum Pressure", found here:
%   https://www.proquest.com/docview/2864402381 and an up to date version
%   of this code and data can be found at:
%   https://doi.org/10.17605/OSF.IO/GJFPD
% KEYWORDS:
%   acute liver failure (ALF), peritoneal dialysis, polydimethylsiloxane
%   (PDMS) silicone, perfluorocarbons (PFCs), hyperammonemia, ammonia
%   removal
% PUBLISHER:
%   Lukas W. DiBeneditto, Biomedical Engineering, Carnegie Mellon
%   University, Pittsburgh PA USA
% VERSION:
%   2023.10.18.102
% ---

programVersion = 'v.2023.10.18.102';

% Data Recorded: 2023 Jun 30 to 2023 Jul 16

%% NOTE
% ## This program assumes and requires:
%   - same number trials in meta.csv (16) as trial folders (16)
%   - there is a "_trials" folder that contains all trial folders
%   - the trial folders are labeled "trial1", "trial2", ... "trial16"
```

```

% - in each trial folder has a sub required folder is labeled
%   "1-preprocessed", where the minimal processing has been done to
%   prepare them for MATLAB
% - in each "1-preprocessed" folder there are (4) required files
%   labeled "arduino.tsv", "omega.txt", "toxirae.csv", and "vernier.csv"
%
% ## Contents of meta.csv (Note 16 trials):
% trial,phase1VacOffBeg,...
% 1,2023-06-30 15:58:36.000,...
% 2,2023-07-12 21:22:02.000,...
% :
% 16,2023-07-16 06:18:34.000,...
%
% ## Required Folder Structure (Note 16 trials):
% Working Directory
% |
% | meta.csv          (REQUIRED)
% | main.m            (REQUIRED, this current file)
% | data-analysis.pdf (Report for this MATLAB Program)
% | README.md
% +---_trials          (REQUIRED folder)
%   +---trial1         (REQUIRED folder)
%     | +---0-raw
%     | | (original device recorded files)
%     | \---1-preprocessed (REQUIRED folder)
%     |   arduino.tsv      (REQUIRED file)
%     |   omega.txt        (REQUIRED file)
%     |   toxirae.csv      (REQUIRED file)
%     |   vernier.csv      (REQUIRED file)
%     :
%   \---trial16         (REQUIRED folder)
%     +---0-raw
%     | (original device recorded files)
%     \---1-preprocessed (REQUIRED folder)
%       arduino.tsv      (REQUIRED file)
%       omega.txt        (REQUIRED file)
%       toxirae.csv      (REQUIRED file)
%       vernier.csv      (REQUIRED file)
%
% ## Information about the contents of the meta.csv file:
% - 17 rows, 1 header row, 16 trial rows - 29 columns
%
% Data Created Before this MATLAB Program Running (manually):
% trial:
%   trial number for each row
% phase1VacOffBeg:
%   datetime phase 1 started, roller pump on vacuum pump off
%   (this is also the start of the experiment trial)
% phase1VacOffEnd:
%   datetime phase 1 stopped, roller pump on vacuum pump off
% phase2VacOnBeg:
%   datetime phase 2 started, roller pump on vacuum pump on
%   (the moment the vacuum pump was turned on)

```

```

%
phase2VacOnEnd:
%
    datetime phase 2 stopped, roller pump on vacuum pump on
%
phase3VacOffBeg:
%
    datetime phase 3 started, roller pump on vacuum pump off
%
    (the moment the vacuum pump was turned off)
%
phase3VacOffEnd:
%
    datetime phase 3 stopped, roller pump on vacuum pump off
%
    (this is also the stop of the experiment trial)
%
arduinoBeg:
%
    date and time the arduino device started recording
%
omegaBeg:
%
    date and time the omega device started recording
%
vernierBeg:
%
    date and time the vernier device started recording
%
volumeRODIWaterML:
%
    volume of reverse osmosis deionized (RODI or RO/DI) water
%
    that circulated in the Liquid-Loop
%
volumePFCML:
%
    volume of 3M FC-770 perfluorinated compound (PFC) or
%
    perfluoro compound that circulated in the Liquid-Loop
%
volumeNH4OHUL:
%
    (μL) units, volume of ammonium hydroxide (NH4OH) Stock
%
    Solution, ~25% NH3 basis (w/w), 14.679 M liquid NH4OH,
%
    circulated in the Liquid-Loop, some eventually diffuses
%
    through the Polydimethylsiloxane (PDMS) filter and turns
%
    into ammonia (NH3) gas
%
volumeNH4OHUMOLL:
%
    (μmol / L) units, volume of ammonium hydroxide (NH4OH)
%
    Stock Solution, ~25% NH3 basis (w/w), 14.679 M liquid
%
    NH4OH, circulated in the Liquid-Loop, some eventually
%
    diffuses through the Polydimethylsiloxane (PDMS) filter
%
    and turns into ammonia (NH3) gas
%
%
Data Created After this MATLAB Program Running (manually):
%
phase1duration:
%
    (hours minutes and seconds) units, time duration phase 1
%
phase2duration:
%
    (hours minutes and seconds) units, time duration phase 2
%
phase3duration:
%
    (hours minutes and seconds) units, time duration phase 3
%
phase2psiaA0:
%
    (PSIA = PSI Absolute = pounds per square inch absolute)
%
    units, mean (average) pressure over entire phase 2
%
    duration, in line of the Liquid-Loop, immediately before
%
    the PDMS filter, recorded by arduino device
%
phase2psiaA1:
%
    (PSIA = PSI Absolute = pounds per square inch absolute)
%
    units, mean (average) pressure over entire phase 2
%
    duration, in line of the Liquid-Loop, immediately after
%
    the PDMS filter, recorded by arduino device
%
phase2psiaA2:
%
    (PSIA = PSI Absolute = pounds per square inch absolute)
%
    units, mean (average) pressure over entire phase 2
%

```

```

%           duration, in line of the Gas-Pathway, immediately before
%           the omega device measuring ammonia (NH3) gas, recorded by
%           arduino device, exposed to dry lab air and ammonia gas
% phase2absolutePressurePSIA:
%           (PSIA = PSI Absolute = pounds per square inch absolute)
%           units, mean (average) pressure over entire phase 2
%           duration, in line of the Gas-Pathway, before the
%           Polydimethylsiloxane (PDMS) filter, recorded by the omega
%           device, exposed to dry lab air only
% phase2gasTemperatureC:
%           (C = degrees Celsius) units, mean (average) temperature
%           over entire phase 2 duration, in line of the Gas-Pathway,
%           before the Polydimethylsiloxane (PDMS) filter, recorded
%           by the omega device, exposed to dry lab air only
% phase2volumetricFlowRateRawCCM:
%           (CCM = cubic centimeter) units, mean (average) flow rate
%           over entire phase 2 duration, in line of the Gas-Pathway,
%           before the Polydimethylsiloxane (PDMS) filter, recorded
%           by the omega device, exposed to dry lab air only, not
%           corrected for temperature, pressure, and gas type
% phase2volumetricFlowRateStandardizedSCCM:
%           (SCCM = standard cubic centimeter) units, mean (average)
%           flow rate over entire phase 2 duration, in line of the
%           Gas-Pathway, before the Polydimethylsiloxane (PDMS)
%           filter, recorded by the omega device, exposed to dry lab
%           air only, corrected by omega device for temperature,
%           pressure, and gas type (air), National Institute of
%           Standards and Technology (NIST) traceable calibration
% phase2DataSet1AmmoniummgL:
%           (mg/L = ppm) units, mean (average) ammonium (NH4+)
%           measurement from vernier device, in line of the fluid
%           loop, specifically in the vortex mixing tank, measured
%           via bluetooth (to prevent voltage interference from pH
%           sensor)
% phase2DataSet1PotentialmV:
%           (mV = millivolt) units, mean (average) electric potential
%           measurement from vernier device, in line of the fluid
%           loop, specifically in the vortex mixing tank, measured
%           via bluetooth (to prevent voltage interference from pH
%           sensor)
% phase2DataSet1TemperatureC:
%           (C = degrees Celsius) units, mean (average) electric
%           potential measurement from vernier device, in line of the
%           fluid loop, specifically in the vortex mixing tank,
%           measured via wire connection
% phase2DataSet1pH:
%           (pH = potential of hydrogen) units, mean (average) pH
%           measurement from vernier device, in line of the fluid
%           loop, specifically in the vortex mixing tank, measured
%           via wire connection
% phase2Sensor1GasReading:
%           (mg/L = ppm) units, mean (average) ammonia (NH3) gas
%           reading pulled through the Polydimethylsiloxane (PDMS)

```

```

%           filter at vacuum pressure through the dry lab air
%           supplied

%% logging start
% start logging MATLAB command window to text file
current = string(datetime("now", "Format", "yyyyMMddHHmmss"));
diaryFilename = strcat('log-', current, '.txt');
diary(diaryFilename);

%% program started
disp('program started')
disp('M.S. Thesis Project Data Preparation and Statistics')
disp(['(c) 2023, Lukas W. DiBeneditto, ', programVersion])
disp(['MATLAB ', version])

%% PREPARATION
% -----

%% preparation started
disp('====preparation started====')

%% import datetime data from meta.csv
disp('import datetime data from meta.csv')

% check if 'meta.csv' exists
if ~isfile('meta.csv')

    % throw warning if meta.csv does not exist
    warning(['File meta.csv does not exist. ',...
            'Please ensure the file is in the current directory.']);

    % Exit the script if file doesn't exist
    return;

end % <-- end if

try
    % set up import options and import data
    opts = delimitedTextImportOptions("NumVariables", 29);

    % specify range and delimiter
    opts.DataLines = [2, Inf];
    opts.Delimiter = ",";

```

```

% specify column names
opts.VariableNames = ["trial", ...
    "phase1VacOffBeg", ...
    "phase1VacOffEnd", ...
    "phase2VacOnBeg", ...
    "phase2VacOnEnd", ...
    "phase3VacOffBeg", ...
    "phase3VacOffEnd", ...
    "arduinoBeg", ...
    "omegaBeg", ...
    "vernierBeg", ...
    "volumeRODIWaterML", ...
    "volumePFCML", ...
    "volumeNH4OHUL", ...
    "volumeNH4OHUMOLL"];

% specify column types
opts.VariableTypes = ["double", ...
    "datetime", ...
    "datetime", ...
    "datetime", ...
    "datetime", ...
    "datetime", ...
    "datetime", ...
    "datetime", ...
    "datetime", ...
    "datetime", ...
    "double", ...
    "double", ...
    "double", ...
    "double"];

% specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% specify variable properties
opts = setvaropts(opts, "phase1VacOffBeg", ...
    "InputFormat", "yyyy-MM-dd HH:mm:ss.SSS");
opts = setvaropts(opts, "phase1VacOffEnd", ...
    "InputFormat", "yyyy-MM-dd HH:mm:ss.SSS");
opts = setvaropts(opts, "phase2VacOnBeg", ...
    "InputFormat", "yyyy-MM-dd HH:mm:ss.SSS");
opts = setvaropts(opts, "phase2VacOnEnd", ...
    "InputFormat", "yyyy-MM-dd HH:mm:ss.SSS");
opts = setvaropts(opts, "phase3VacOffBeg", ...
    "InputFormat", "yyyy-MM-dd HH:mm:ss.SSS");
opts = setvaropts(opts, "phase3VacOffEnd", ...
    "InputFormat", "yyyy-MM-dd HH:mm:ss.SSS");
opts = setvaropts(opts, "arduinoBeg", ...
    "InputFormat", "yyyy-MM-dd HH:mm:ss.SSS");
opts = setvaropts(opts, "omegaBeg", ...

```



```

                                "InputFormat", "yyyy-MM-dd HH:mm:ss.SSS");
opts = setvaropts(opts, "vernierBeg", ...
                                "InputFormat", "yyyy-MM-dd HH:mm:ss.SSS");

% import data
M = readtable("meta.csv", opts);

% clear temporary variables
clear opts

% display successful import message
disp('  successfully imported meta.csv');

catch errorInfo
    % throw warning if there was an error during import
    warning(['  An error occurred while importing meta.csv. ',...
            '  Error details: ', errorInfo.message]);

    % exit script if there was an error
    return;
end

%% import data from trials
disp('import data from trials')

% get path to current directory where this matlab script is being
% executed
scriptPathChar = pwd;

% build full path to "_trials" folder, assuming it's a subdirectory
% of current directory
dataRootPathChar = fullfile(scriptPathChar, '_trials');

% get a list of all directories starting with "trial" under data root
% path. NOTE: they are not sorted correctly at this point
trialDirsStruct = dir(fullfile(dataRootPathChar, 'trial*'));

% get trial number strings from directory names
%   assumes directory name starts with "trial"
trialNumberCell = {trialDirsStruct.name};

% get numeric part from each trial string and convert to numbers, still
% unsorted, i.e. 1, 10, 11, 12 ...
trialNumbersDouble = cellfun(@(x) str2double(x(6:end)), trialNumberCell);

% sort numbers
[~, sortedIndicesDouble] = sort(trialNumbersDouble);

% use sorted indices to reorder trial directories in numeric order, while
% not strictly required, this ensures consistent handling of trials in

```

```

% subsequent operations
sortedTrialDirsStruct = trialDirsStruct(sortedIndicesDouble);

% initialize a structure to store data tables for each trial
DS = struct();

% loop through each sorted trial directory and populate DS data structure
for i = 1:length(sortedTrialDirsStruct)

    % display current trial number
    disp([' trial', num2str(i)])

    % placeholder structures for each file in a trial
    currentTrialDataStruct = struct('omegaTable', [], ...
                                    'arduinoTable', [], ...
                                    'toxiraeTable', [], ...
                                    'vernierTable', []);

    % build full file name from parts
    trialPathChar = fullfile(dataRootPathChar, ...
                             sortedTrialDirsStruct(i).name);

    % check if '1-preprocessed' folder exists in current trial
    % directory, where 7 indicates it's a directory, 'dir' is
    % recommended
    if exist(fullfile(trialPathChar, '1-preprocessed'), 'dir') == 7

        % get all files and folders in '1-preprocessed' directory
        filesStruct = dir(fullfile(trialPathChar, ...
                                   '1-preprocessed', '*.*'));

        % loop through each file found
        for j = 1:length(filesStruct)

            % only process if it's not a directory
            %     since dir command returns "." as current directory,
            %     and ".." as parent directory
            if ~filesStruct(j).isdir

                % create full file path
                filePathChar = fullfile(trialPathChar, ...
                                        '1-preprocessed', ...
                                        filesStruct(j).name);

                try

                    % choose what to do based on name of file
                    switch filesStruct(j).name

                        case 'arduino.tsv'

                            % set up import options and import data
                            arduinoImportOptions = ...

```

```

        delimitedTextImportOptions("NumVariables", 4);

% specify range and delimiter
arduinoImportOptions.DataLines = [4, Inf];
arduinoImportOptions.Delimiter = "\t";

% specify column names and types
arduinoImportOptions.VariableNames = ...
    ["timeMS", "psiaA0", "psiaA1", "psiaA2"];
arduinoImportOptions.VariableTypes = ...
    ["double", "double", "double", "double"];

% specify file level properties
arduinoImportOptions.ExtraColumnsRule = "ignore";
arduinoImportOptions.EmptyLineRule = "read";

% import data
currentTrialDataStruct.arduinoTable = ...
    readtable(filePathChar, arduinoImportOptions);

% report import success
disp('    arduino import success')

case 'omega.txt'

% initialize variables
omegaStartRowDouble = 2;

% format for each line of text
omegaFormatSpecificationChar = ...
    '%f %f %f %f %s^[^\\n\\r]';

% open text file
omegaFileIDDouble = fopen(filePathChar, 'r');

% read columns of data based on format
% specified
omegaDataCell = ...
    textscan(omegaFileIDDouble, ...
        omegaFormatSpecificationChar, ...
        'Delimiter', ' ', ...
        'MultipleDelimsAsOne', true, ...
        'TextType', 'string', ...
        'HeaderLines', omegaStartRowDouble-1, ...
        'ReturnOnError', false, ...
        'EndOfLine', '\r\n');

% close text file
fclose(omegaFileIDDouble);

% create output variable
currentTrialDataStruct.omegaTable = ...
    table(omegaDataCell{1}, ...

```

```

        omegaDataCell{2}, ...
        omegaDataCell{3}, ...
        omegaDataCell{4}, ...
        omegaDataCell{5}, ...
        'VariableNames', {'absolutePressurePSIA', ...
                           'gasTemperatureC', ...
                           'volumetricFlowRateRawCCM', ...
                           'volumetricFlowRateStandardizedSCCM',
...
                           'gasTypeAndStatus'}});

% remove leading and trailing white spaces from
% gasTypeAndStatus column
currentTrialDataStruct.omegaTable.gasTypeAndStatus = ...
    strtrim(currentTrialDataStruct.omegaTable.gasTypeAndStatus);

% report import success
disp('    omega import success')

case 'toxirae.csv'

% set up import options and import data
toxiraeImportOptions = ...
    delimitedTextImportOptions("NumVariables", 29, ...
                               "Encoding", "UTF-8");

% specify to read all lines after header
% with "Inf"
toxiraeImportOptions.DataLines = [2, Inf];

% specify delimiter
toxiraeImportOptions.Delimiter = ",";

% specify column names
toxiraeImportOptions.VariableNames = ...
    ["DeviceSerialNo", ...
     "LogTime", ...
     "LogType", ...
     "LogInterval", ...
     "Sensor1Type", ...
     "Sensor1DisplayUnit", ...
     "Sensor1SerialNumber", ...
     "Sensor1Status", ...
     "Sensor1GasReading", ...
     "Sensor1STELReading", ...
     "Sensor1TWARReading", ...
     "Sensor1LastCal", ...
     "Sensor1SpanSetpoint", ...
     "Sensor1HighAlarm", ...
     "Sensor1LowAlarm", ...
     "Sensor1STELAlarm", ...
     "Sensor1TWAAlarm", ...
     "Sensor1OvertimeThreshold", ...

```

```
"UnitStatus", ...
"RunningMode", ...
"LogStartType", ...
"DiagnosticsMode", ...
"StopReason", ...
"UserId", ...
"SiteId", ...
"RecordNumber", ...
"SessionStartTime", ...
"SessionStopTime", ...
"FirmwareVersion"];

% specify column types
toxiraeImportOptions.VariableTypes = ...
["string", ...
 "datetime", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "double", ...
 "string", ...
 "string", ...
 "string", ...
 "double", ...
 "double", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...
 "string", ...];

% specify date-time format for LogTime
% column
toxiraeImportOptions = ...
    setvaropts(toxiraeImportOptions, ...
                'LogTime', ...
                'InputFormat', 'M/dd/yyyy h:mm:ss a');

% specify file level properties
toxiraeImportOptions.ExtraColumnsRule = "ignore";
```

```

toxiraeImportOptions.EmptyLineRule = "read";

% import data
currentTrialDataStruct.toxiraeTable = ...
    readtable(filePathChar, toxiraeImportOptions);

% report import success
disp('    toxirae import success')

case 'vernier.csv'

% since the order of the column variables
% varies, i.e., trial8 vs trial9, we need to do
% the following

% read the table
dataTable = ...
    readtable(filePathChar, ...
        'VariableNamingRule', 'preserve');

% get column names to determine the order
colNames = dataTable.Properties.VariableNames;

% create a temporary table to store columns in
% the desired order
tempTable = table();

% for each column variable name, find the
% corresponding column and store data
% accordingly

if any(contains(colNames, 'Time(s)'))
    tempTable.DataSet1Times = ...
        dataTable{:, contains(colNames, 'Time(s)')};
end % <-- end if

if any(contains(colNames, 'Temperature'))
    tempTable.DataSet1TemperatureC = ...
        dataTable{:, contains(colNames, 'Temperature')};
end % <-- end if

if any(contains(colNames, 'Potential'))
    tempTable.DataSet1PotentialmV = ...
        dataTable{:, contains(colNames, 'Potential')};
end % <-- end if

if any(contains(colNames, 'Ammonium'))
    tempTable.DataSet1Ammoniummgl = ...
        dataTable{:, contains(colNames, 'Ammonium')};
end % <-- end if

if any(contains(colNames, 'pH'))
    tempTable.DataSet1pH = ...

```

```

        dataTable{:, contains(colNames, 'pH')});
    end % <-- end if

    % put the temporary table in the correct
    % variable order into what will be used in the
    % data store DS
    currentTrialDataStruct.vernierTable = tempTable;

    % report import success
    disp('    vernier import success');

otherwise

    % throw warning because there should only be
    % four files in "0-raw" folder
    warning('Unrecognized file: %s', ...
        filesStruct(j).name);

    % report import failure
    disp(['    ' filesStruct(j).name ' unrecognized'])

    end % <-- end switch filesStruct(j).name

catch exception

    % throw warning if there is any problems with
    % processing file
    warning('Error processing file: %s\n%s', ...
        filePathChar, exception.message);

    % report import failure
    disp(['    ' filesStruct(j).name ' failure'])

    end % <-- end try

    end % <-- end if

    end % <-- end for

else

    % throw warning that folder was not found
    warning(['Folder "1-preprocessed" not found in directory: %s. ', ...
        'Adding an empty entry for this trial.'], ...
        trialPathChar);

    end % <-- end if

    % store data for this trial number %d = i
    DS.(sprintf('trial%d', i)) = currentTrialDataStruct;

end % <-- end for

```

```

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
                DS ...
                M ...
                scriptPathChar ...
                sortedTrialDirsStruct ;

%% remove unused data from
disp('remove unused data')

% get field names of all trials in structure
trialNamesCell = fieldnames(DS);

% loop over each trial
for i = 1:length(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % check if table 'omegaTable' exists within current trial
    if isfield(DS.(trialNameChar), 'omegaTable')

        % get omegaTable from current trial into T
        T = DS.(trialNameChar).omegaTable;

        % keep only specified columns
        T = T(:, {'absolutePressurePSIA', ...
                  'gasTemperatureC', ...
                  'volumetricFlowRateRawCCM', ...
                  'volumetricFlowRateStandardizedSCCM'});

        % update table in structure
        DS.(trialNameChar).omegaTable = T;

    end % <-- end if

    % check if table 'toxiraeTable' exists within current trial
    if isfield(DS.(trialNameChar), 'toxiraeTable')

        % get current trial toxirae table
        T = DS.(trialNameChar).toxiraeTable;

        % keep only specified columns
        T = T(:, {'LogTime', ...
                  'Sensor1GasReading'});
    end
end

```



```

        % update table in structure
        DS.(trialNameChar).toxiraeTable = T;

    end % <-- end if

end % <-- end for

%% cleanup workspace
disp('  cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
            DS ...
            M ...
            scriptPathChar ...
            sortedTrialDirsStruct ;

%% prepare arduino data with dateTime using table M
disp('prepare arduino data with dateTime using table M');

% instead of trial names, we will now loop over row indices in M
numTrialsDouble = size(M, 1);

% loop over each trial
for i = 1:numTrialsDouble

    % get arduino beginning dateTime
    arduinoBegDateTime = M.arduinoBeg(i);

    % each row in M corresponds to a trial in DS
    trialNameChar = ['trial', num2str(i)];

    % check if table 'arduinoTable' exists within current trial
    if isfield(DS.(trialNameChar), 'arduinoTable')

        % get arduino table for current trial
        T = DS.(trialNameChar).arduinoTable;

        % convert timeMS column to duration
        timeDuration = milliseconds(T.timeMS);

        % generate datetime array using converted duration
        arduinoDateTime = arduinoBegDateTime + timeDuration;

        % add datetime array as a new column to table
        T.dateTime = arduinoDateTime;

        % reorder columns to make dateTime first column
        T = [T(:, 'dateTime'), ...

```

```

        T(:, setdiff(T.Properties.VariableNames, {'dateTime', 'timeMS'}));

        % update table in structure
        DS.(trialNameChar).arduinoTable = T;

    end % <-- end if

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
            DS ...
            M ...
            scriptPathChar ...
            sortedTrialDirsStruct ;

%% prepare omega data with dateTime using table M
disp('prepare omega data with dateTime using table M');

% loop over trials
numTrialsDouble = size(M, 1);

% loop over each trial
for i = 1:numTrialsDouble

    % get starting time omega device began recording
    omegaBegDateTime = M.omegaBeg(i);

    % each row in M corresponds to a trial in DS
    trialNameChar = ['trial', num2str(i)];

    % check if table 'omegaTable' exists within current trial
    if isfield(DS.(trialNameChar), 'omegaTable')

        % get omega table for current trial
        T = DS.(trialNameChar).omegaTable;

        % get number of rows in table
        numRows = height(T);

        % 500 ms increment
        incrementDuration = milliseconds(500);

        % generate datetime array
        omegaDateTime = ...
            omegaBegDateTime + ...

```

```

        (0 : incrementDuration : (numRows-1)*incrementDuration)';

    % add datetime array as a new column to table
    T.dateTime = omegaDateTime;

    % reorder columns by moving 'dateTime' as first column because the
    % timetable function requires datetime to be first column in
    % subsequent operations
    T = [T(:, 'dateTime'), ...
        T(:, setdiff(T.Properties.VariableNames, 'dateTime'))];

    % update table in structure
    DS.(trialNameChar).omegaTable = T;

end % <-- end if

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
        DS ...
        M ...
        scriptPathChar ...
        sortedTrialDirsStruct ;

%% prepare toxirae data with dateTime using table M
disp('prepare toxirae data with dateTime using table M')

% get field names of all trials in structure
trialNamesCell = fieldnames(DS);

% input dateTime format
inputFormatChar = 'MM/dd/yyyy h:mm:ss a';

% output dateTime format
outputFormatChar = 'yyyy-MM-dd HH:mm:ss.SSS';

% original toxirae LogTime format is reverse chronological order, however,
% for synchronizing all timetables we need to convert LogTime format
% and reverse order so it becomes correct chronological order
for i = 1:length(trialNamesCell)

    % get current trial
    trialNameChar = trialNamesCell{i};

    % check if table 'toxiraeTable' exists within current trial

```

```

if isfield(DS.(trialNameChar), 'toxiraeTable')

    % get toxirae table for current trial
    T = DS.(trialNameChar).toxiraeTable;

    % convert LogTime to datetime format
    T.dateTime = datetime(T.LogTime, ...
        'InputFormat', inputFormatChar, ...
        'Format', outputFormatChar);

    % rearrange columns to make dateTime first column
    T = T(:, {'dateTime', 'Sensor1GasReading'});

    % reverse order of rows for T
    T = flipud(T);

    % update table in DS
    DS.(trialNameChar).toxiraeTable = T;

end % <-- end if

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
    DS ...
    M ...
    scriptPathChar ...
    sortedTrialDirsStruct ;

%% prepare vernier data with dateTime using table M
disp('prepare vernier data with dateTime using table M');

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % get starting time vernier device began recording
    vernierBegDateTime = M.vernierBeg(i);

    % check if table 'vernierTable' exists within current trial

```

```

    if isfield(DS.(trialNameChar), 'vernierTable')

        % get vernier table for current trial
        T = DS.(trialNameChar).vernierTable;

        % generate dateTime based on DataSet1Times and starting time
        T.dateTime = vernierBegDateTime + seconds(T.DataSet1Times);

        % rearrange columns to make dateTime first column
        T = T(:, [{'dateTime'}, ...
                  setdiff(T.Properties.VariableNames, 'dateTime')]);

        % remove DataSet1Times column
        T = removevars(T, 'DataSet1Times');

        % update table in structure DS
        DS.(trialNameChar).vernierTable = T;

    end % <-- end if

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
           DS ...
           M ...
           scriptPathChar ...
           sortedTrialDirsStruct ;

%% convert each table in each trial to timetable
disp('convert each table in each trial to timetable')

% we are doing this so that we can sync all tables together later for
% each trial

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % convert tables to timetables and store as new fields in DS
    DS.(trialNameChar).arduinoTT = ...

```

```

        table2timetable(DS.(trialNameChar).arduinoTable, 'RowTimes', 'dateTime');
DS.(trialNameChar).omegaTT = ...
        table2timetable(DS.(trialNameChar).omegaTable, 'RowTimes', 'dateTime');
DS.(trialNameChar).toxiraeTT = ...
        table2timetable(DS.(trialNameChar).toxiraeTable, 'RowTimes', 'dateTime');
DS.(trialNameChar).vernierTT = ...
        table2timetable(DS.(trialNameChar).vernierTable, 'RowTimes', 'dateTime');

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
        DS ...
        M ...
        scriptPathChar ...
        sortedTrialDirsStruct ;

%% calculate average deviation of time for arduinoTT
disp('calculate average deviation of time for arduinoTT');

% calculate average deviation between consecutive timestamps for arduinoTT
% for each trial and then get overall average and standard deviation
% across trials. to determine if deviation lies within range of 500
% ms (+/-) 25 ms

% initialize a counter for total number of arduinoTT lines across all trials
totalArduinoTTLines = 0;

% get list of trials
trialNamesCell = fieldnames(DS);

% initialize a cell array to temporarily store time differences for each
% trial
timeDifferencesCell = cell(numel(trialNamesCell), 1);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % get dateTime from arduino timetable for this trial
    timeStamps = DS.(trialNameChar).arduinoTT.dateTime;

    % update the total number of arduinoTT lines
    totalArduinoTTLines = totalArduinoTTLines + length(timeStamps);

```

```

    % compute differences between consecutive timestamps in seconds
    timeDifferences = diff(timeStamps);
    timeDifferencesInSeconds = seconds(timeDifferences);

    % store time differences in current cell
    timeDifferencesCell{i} = timeDifferencesInSeconds;

end % <-- end for

% concatenate all time differences into one array
totalTimeDifferences = vertcat(timeDifferencesCell{:});

% calculate average and standard deviation of time differences
% across all trials
avgSamplingInterval = mean(totalTimeDifferences);
stdSamplingInterval = std(totalTimeDifferences);

% convert to milliseconds for display
avgSamplingInterval_ms = avgSamplingInterval * 1000;
stdSamplingInterval_ms = stdSamplingInterval * 1000;

% display results in desired format
fprintf('  average sampling interval across all trials: %.2f (+/- %.2f) ms\n', ...
        avgSamplingInterval_ms, stdSamplingInterval_ms);
disp(['  arduinoTT lines count, n = ', num2str(totalArduinoTTLines) ]);

%% cleanup workspace
disp('  cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
           DS ...
           M ...
           scriptPathChar ...
           sortedTrialDirsStruct ;

%% resample arduino data for each trial
disp('resample arduino data for each trial')

% we resample arduinoTT in DS to a regular 500 ms interval, for easier
% synchronization with other regularly sampled devices, this is acceptable
% because arduino is already sampled at 504.81 (+/- 0.47) ms

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

```

```

% get current trial name
trialNameChar = trialNamesCell{i};

% get arduino timetable for this trial
irregularArduinoTT = DS.(trialNameChar).arduinoTT;

% resample to a regular interval of 500 ms using linear interpolation
regularArduinoTT = ...
    retime(irregularArduinoTT, 'regular', 'linear', 'TimeStep', seconds(0.5));

% store regularized timetable back into struct
DS.(trialNameChar).arduinoTT = regularArduinoTT;

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
    DS ...
    M ...
    scriptPathChar ...
    sortedTrialDirsStruct ;

%% upsample toxirae data to 500 ms interval using linear interpolation
disp('upsample toxirae data to 500 ms interval using linear interpolation')

% toxirae device samples data every 1 second by default. in order to
% align its sampling rate with other devices, we need to upsample its data
% to 500 ms intervals
%
% toxirae device samples whole numbers (e.g., 1, 2, 3), during
% interpolation, a value between 1 and 2 becomes 1.5, which is a linear
% interpolation artifact

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % ensure unique timestamps in toxiraeTT
    % - duplicate timestamps are highly unlikely, to ensure data
    % consistency, we check and remove any duplicates
    % - 'unique' function identifies unique date/Time and returns their

```



```

%     indices (idx)
%     - 'stable' ensures original order of data is preserved
[~, idx] = unique(DS.(trialNameChar).toxiraeTT.dateTime, 'stable');

% determine number of non-unique (duplicate) timestamps
numDuplicates = ...
    length(DS.(trialNameChar).toxiraeTT.dateTime) - length(idx);

% display if there are any non-unique timestamps for current trial
if (numDuplicates > 0)
    fprintf(' Trial %s has %d duplicate timestamps.\n', ...
        trialNameChar, numDuplicates);
end % <-- end if

% use indices from 'unique' function, to only keep unique rows from
% original data
DS.(trialNameChar).toxiraeTT = DS.(trialNameChar).toxiraeTT(idx,:);

% create new time vector for 500 ms intervals, here we are getting the
% first and last time and then spacing every 500 ms
newTimeVector = ...
    min(DS.(trialNameChar).toxiraeTT.dateTime) : ...
    seconds(0.5) : ...
    max(DS.(trialNameChar).toxiraeTT.dateTime) ;

% ensure newTimeVector is unique, highly unlikely, but just in case
newTimeVector = unique(newTimeVector);

% remember we are upscaling, we do this by adjust for potential missing
% values by syncing to newly created time vector
%     - 'synchronize' function aligns two timetables by their row times
%     - first input is original toxirae timetable we want to
%       upscale at a sample rate of 1 second
%     - second input is new time vector at a sample rate of 500 ms
%     - 'union' method will synchronize uses linear interpolation to
%       resample data from input timetables to output row
%       times
%     - 'linear' will fill in any missing data in first
%       timetable using linear interpolation based on neighboring
%       values

DS.(trialNameChar).toxiraeTT = ...
    synchronize(DS.(trialNameChar).toxiraeTT, ...
        timetable(newTimeVector'), ...
        'union', ...
        'linear');

% retime function to adjust timetable to regular time intervals,
% this is not really needed, but just in case
%     - 'regular' specifies that timetable should be adjusted to
%       have regular time intervals
%     - 'linear' is used to fill in any missing data with linear
%       interpolation

```

```

% - 'TimeStep' with seconds(0.5) specifies desired time interval as
%     500 milliseconds or 0.5 seconds
DS.(trialNameChar).toxiraeTT = ...
    retime(DS.(trialNameChar).toxiraeTT, ...
        'regular', ...
        'linear', ...
        'TimeStep', seconds(0.5));

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
    DS ...
    M ...
    scriptPathChar ...
    sortedTrialDirsStruct ;

%% save each timetable for each trial to individual csv files
disp('save each timetable for each trial to individual csv files');

% list of timetables we want to save
timetablesToSave = {'arduinoTT', 'omegaTT', 'toxiraeTT', 'vernierTT'};

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % display current trial number
    disp([' trial', num2str(i)])

    % check if '2-processed' exists in current trial's directory
    currentProcessedDir = ...
        fullfile(dataRootPathChar, trialNameChar, '2-processed');

    if ~exist(currentProcessedDir, 'dir')

        % if it doesn't exist, create it
        mkdir(currentProcessedDir);

    end % <-- end if

```

```

% loop over each timetable in timetablesToSave list
for j = 1:length(timetablesToSave)

    % get current table
    currentTT = timetablesToSave{j};

    % confirm that current timetable exists in current trial
    if isfield(DS.(trialNameChar), currentTT)

        % remove 'TT' suffix from filename
        baseName = strrep(currentTT, 'TT', '');

        % create full path and filename for CSV
        fileName = fullfile(currentProcessedDir, [baseName, '.csv']);

        % attempt to save timetable to CSV
        try

            % write timetable to a CSV
            writetimetable(DS.(trialNameChar).(currentTT), fileName);

            % post-process CSV to remove "NaN"s
            removeNaNsFromCSV(fileName)

            % report export success
            disp([' ', baseName, ' export success'])

        catch ME

            % Report error if there's an issue exporting
            disp([' Error exporting ', baseName, ': ', ME.message]);

        end % <-- end try

    end % <-- end if

end % <-- end for

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
    DS ...
    M ...
    scriptPathChar ...
    sortedTrialDirsStruct ;

```

```

%% combine and synchronize data for all devices for that trial
disp('combine and synchronize data for all devices for that trial')

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % display current trial number
    disp([' trial', num2str(i)])

    % ensure each timetable is sorted and check for missing dateTime values
    timetableNames = {'arduinoTT', 'omegaTT', 'vernierTT', 'toxiraeTT'};

    % loop through each timetable to sort and check dateTime
    for ttName = timetableNames

        % get current timetable
        currentTT = DS.(trialNameChar).(ttName{1});

        % sort timetable based on dateTime
        currentTT = sortrows(currentTT, 'dateTime');

        % check for missing dateTime values and throw error if found
        if any(ismissing(currentTT.dateTime))

            error([' Missing dateTime values in ' ...
                ttName{1} ' of ' trialNameChar]);

        end % <-- end if

        % update timetable in struct
        DS.(trialNameChar).(ttName{1}) = currentTT;

    end % <-- end for

try

    % synchronize all time tables for this trial
    % use 'union' to ensure all times are considered
    synchronizedTimeTable = ...
        synchronize(DS.(trialNameChar).arduinoTT, ...
            DS.(trialNameChar).omegaTT, ...
            DS.(trialNameChar).vernierTT, ...
            DS.(trialNameChar).toxiraeTT, ...
            'union');

    % store synchronized data back into main structure

```

```

DS.(trialNameChar).synchronizedTT = synchronizedTimeTable;

% report synchronized success
disp('    synchronized success')

catch ME

% handle specific errors related to timetable synchronization
if strcmp(ME.identifier, 'MATLAB:timetable:synchronize:InvalidTimes')

    % report where error found
    disp(['Found an issue with dateTime values in trial: ' trialNameChar]);

    % to provide more detailed error information
    rethrow(ME);

else

    % for all other errors, rethrow them
    rethrow(ME);

end % <-- end if

end % <-- end try

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
            DS ...
            M ...
            scriptPathChar ...
            sortedTrialDirsStruct ;

%% save combined and synchronized data to csv files
disp('save combined and synchronized data to csv files')

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

```

```

% display current trial number
disp([' trial', num2str(i)])

% determine path for '3-synchronized' in current trial's directory
currentSynchronizedDir = ...
    fullfile(dataRootPathChar, trialNameChar, '3-synchronized');

% check for existence and create directory if not found
if ~exist(currentSynchronizedDir, 'dir')

    % if it doesn't exist, create it
    mkdir(currentSynchronizedDir);
end

% retrieve synchronized timetable from dataset
synchronizedTimeTable = DS.(trialNameChar).synchronizedTT;

% construct path and filename for saving CSV
fileName = fullfile(currentSynchronizedDir, 'synchronized.csv');

% try to save timetable and handle potential errors
try

    % save timetable directly into a CSV file
    writetimetable(synchronizedTimeTable, fileName);

    % post-process CSV to remove "NaN"s
    removeNaNsFromCSV(fileName)

    % report export success
    disp('    synchronized export success');

catch ME

    % report error if there's an issue exporting
    disp(['    Error exporting synchronized: ', ME.message]);

end % <-- end try

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
    DS ...
    M ...
    scriptPathChar ...
    sortedTrialDirsStruct ;

```

```

%% for each trial trim to only include experiment rows using table M
disp('for each trial trim to only include experiment rows using table M');

% we are trimming to when experiment starts and stops

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % get start and stop times directly from M
    startTime = M.phase1VacOffBeg(i);
    stopTime = M.phase3VacOffEnd(i);

    % get synchronizedTT for current trial
    synchronizedTT = DS.(trialNameChar).synchronizedTT;

    % subset synchronizedTT to only include rows between start and stop
    % times
    subsetTT = ...
        synchronizedTT(startTime <= synchronizedTT.dateTime ...
            & synchronizedTT.dateTime <= stopTime, :);

    % save subsetting timetable to structure
    DS.(trialNameChar).experimentTT = subsetTT;

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
    DS ...
    M ...
    scriptPathChar ...
    sortedTrialDirsStruct ;

%% data cleanup for broken or out of range values
disp('data cleanup for broken or out of range values')

% due to broken vernier liquid ammonium probe for trial 1 change all
% vernier temperature readings to zero

```

```

DS.trial11.experimentTT.DataSet1TemperatureC(:) = 0;

% unknown cause of NaNs for ammonia sensor
% locate NaN values in specific field
nanIndices = isnan(DS.trial11.experimentTT.Sensor1GasReading);
% replace NaN values with zeros
DS.trial11.experimentTT.Sensor1GasReading(nanIndices) = 0;

%% fix liquid ammonium sensor out of range values
disp('fix liquid ammonium sensor out of range values')
% liquid ammonium sensor
% https://www.vernier.com/files/manuals/gdx-nh4/gdx-nh4.pdf
%   Range:
%       1 to 18,000 mg/L (or ppm)
%   Accuracy after calibration (precision):
%       ±10% of full scale (calibrated 1 to 100 mg/L)
%   pH range:
%       2-7 (no pH compensation)
%   Temperature range:
%       0-40°C (no temperature compensation)
%   Electrode slope:
%       +56 ±4 mV/decade at 25°C
%   Standard voltages, typical:
%       High (100 mg/L) 116 mV, Low 0 mV (1 mg/L)
%   Electrode resistance:
%       0.1 to 5 MΩ
% DS.(trialNameChar).experimentTT.DataSet1AmmoniummgL
% DS.(trialNameChar).experimentTT.DataSet1PotentialmV

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % display current trial number
    disp([' trial', num2str(i)])

    % access the ammonium (mg/L) values for the current trial
    ammoniumValues = DS.(trialNameChar).experimentTT.DataSet1AmmoniummgL;

    % find number of values outside the range (1, 18000)
    ammoniumOutOfBound = ammoniumValues < 1 | ammoniumValues > 18000;

    % report found values out of range
    disp([' found ', num2str(sum(ammoniumOutOfBound)), ...
        ' values for DataSet1AmmoniummgL']);

    % set values outside the range (1, 18000) to NaN

```



```

ammoniumValues(ammoniumOutOfBound) = NaN;

% assign the modified values back to the structure
DS.(trialNameChar).experimentTT.DataSet1AmmoniummGL = ammoniumValues;

% access the Potential (mV) values for the current trial
potentialValues = DS.(trialNameChar).experimentTT.DataSet1PotentialmV;

% find number of values outside the range (0, 116)
potentialOutOfBound = potentialValues < 0 | potentialValues > 116;

% report found values out of range
disp(['    found ', num2str(sum(potentialOutOfBound)), ...
      ' values for DataSet1PotentialmV out of range']);

% set values outside the range (0, 116) to NaN
potentialValues(potentialOutOfBound) = NaN;

% assign the modified values back to the structure
DS.(trialNameChar).experimentTT.DataSet1PotentialmV = potentialValues;

end

%% fix liquid pH sensor out of range values
disp('fix liquid pH sensor out of range values')
% liquid pH sensor
% https://www.vernier.com/files/manuals/ph-bta/ph-bta.pdf
%   Range:
%       pH 0-14
%   Response time:
%       90% of final reading in 1 second in a buffer
%   Temperature range:
%       5 to 80°C (readings not compensated)
%   Accuracy:
%       ± 0.2 pH units
%   Isopotential pH:
%       pH 7 (point at which temperature has no effect)
%   Default calibration values:
%       slope: -3.838; intercept: 13.720;
% DS.(trialNameChar).experimentTT.DataSet1pH

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

```

```

% display current trial number
disp([' trial', num2str(i)])

% access the pH values for the current trial
pHValues = DS.(trialNameChar).experimentTT.DataSet1pH;

% find the number of values outside the range (0, 14)
pHOutOfBounds = pHValues < 0 | pHValues > 14;

% report found values out of range
disp([' found ', num2str(sum(pHOutOfBounds)), ...
    ' values for pH out of range']);

% set the out-of-range values to NaN
pHValues(pHOutOfBounds) = NaN;

% assign the modified values back to the structure
DS.(trialNameChar).experimentTT.DataSet1pH = pHValues;

end

%% fix liquid temperature sensor out of range values
disp('fix liquid temperature sensor out of range values')

% liquid temperature sensor
% https://www.vernier.com/files/manuals/tmp-bta/tmp-bta.pdf
% Temperature range:
% -40 to 135°C (-40 to 275°F)
% Temperature sensor:
% 20 kΩ NTC Thermistor
% Accuracy:
% ±0.2°C at 0°C, ±0.5°C at 100°C
% Response time (time for 90% change in reading):
% 10 seconds (in water, with stirring)
% DS.(trialNameChar).experimentTT.DataSet1TemperatureC

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % display current trial number
    disp([' trial', num2str(i)])

    % access the temperature values for the current trial
    tempValues = DS.(trialNameChar).experimentTT.DataSet1TemperatureC;

```

```

% find the number of values outside the range (-40, 135)
tempOutOfBound = tempValues < -40 | tempValues > 135;

% report found values out of range
disp(['    found ', num2str(sum(tempOutOfBound)), ...
      ' values for Temperature out of range']);

% set the out-of-range values to NaN
tempValues(tempOutOfBound) = NaN;

% assign the modified values back to the structure
DS.(trialNameChar).experimentTT.DataSet1TemperatureC = tempValues;

end % <-- end for

%% cleanup workspace
disp('  cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
           DS ...
           M ...
           scriptPathChar ...
           sortedTrialDirsStruct ;

%% save experiment rows to csv files
disp('save experiment rows to csv files')

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % display current trial number
    disp(['  trial', num2str(i)])

    % check if '4-experiment' exists in current trial's directory
    currentExperimentDir = ...
        fullfile(dataRootPathChar, trialNameChar, '4-experiment');

    if ~exist(currentExperimentDir, 'dir')

        % if it doesn't exist, create it
        mkdir(currentExperimentDir);

```

```

end

% only save experimentTT
experimentTimeTable = DS.(trialNameChar).experimentTT;

% form filename for CSV
fileName = fullfile(currentExperimentDir, 'experiment.csv');

% try to save timetable and handle potential errors
try

    % write timetable to a CSV file
    writetimetable(experimentTimeTable, fileName);

    % post-process CSV to remove "NaN"s
    removeNaNsFromCSV(fileName)

    % report export success
    disp('    experiment export success');

catch ME

    % report error if there's an issue exporting
    disp(['    Error exporting experiment: ', ME.message]);

end % <-- end try

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
            DS ...
            M ...
            scriptPathChar ...
            sortedTrialDirsStruct ;

%% get subsets for all three phases for each trial
disp('get subsets for all three phases for each trial');

% we breaking apart experiment into three phases based on the
% dateTime data from meta.csv --> M data structure
%     - phase 1: roller pump on, vacuum off
%     - phase 2: roller pump on, vacuum on
%     - phase 3: roller pump on, vacuum off

% get list of trials

```

```

trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % get experimentTT from current trial
    experimentData = DS.(trialNameChar).experimentTT;

    % phase 1 extraction based on datetime interval from M
    phase1Begin = M.phase1VacOffBeg(i); % phase 1 start (start of trial)
    phase1End = M.phase1VacOffEnd(i);   % phase 1 stop
    phase1Rows = experimentData.dateTime >= phase1Begin ...
        & experimentData.dateTime <= phase1End;
    DS.(trialNameChar).phase1TT = experimentData(phase1Rows,:);

    % phase 2 extraction based on datetime interval from M
    phase2Begin = M.phase2VacOnBeg(i); % phase 2 start
    phase2End = M.phase2VacOnEnd(i);   % phase 2 stop
    phase2Rows = experimentData.dateTime >= phase2Begin ...
        & experimentData.dateTime <= phase2End;
    DS.(trialNameChar).phase2TT = experimentData(phase2Rows,:);

    % phase 3 extraction based on datetime interval from M
    phase3Begin = M.phase3VacOffBeg(i); % phase 3 start
    phase3End = M.phase3VacOffEnd(i);   % phase 3 stop (end of trial)
    phase3Rows = experimentData.dateTime >= phase3Begin ...
        & experimentData.dateTime <= phase3End;
    DS.(trialNameChar).phase3TT = experimentData(phase3Rows,:);

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
    DS ...
    M ...
    scriptPathChar ...
    sortedTrialDirsStruct ;

%% preparation finished
disp('====preparation finished====')

```

```

%% STATISTICS
% -----

%% statistics started
disp('====statistics started====')

%% get statistics for all three phases for each trial
disp('get statistics for all three phases for each trial')

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % get statistics for each phase and store them in DS
    DS.(trialNameChar).phase1Stats = descStat(DS.(trialNameChar).phase1TT);
    DS.(trialNameChar).phase2Stats = descStat(DS.(trialNameChar).phase2TT);
    DS.(trialNameChar).phase3Stats = descStat(DS.(trialNameChar).phase3TT);

end

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
           DS ...
           M ...
           scriptPathChar ...
           sortedTrialDirsStruct ;

%% export phase 1 to 3 data and statistics
disp('export phase 1 to 3 data and statistics')

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

```

```

% get current trial name
trialNameChar = trialNamesCell{i};

% display current trial number
disp([' trial', num2str(i)])

% construct path to 5-statistics directory for current trial
currentStatsDir = fullfile(dataRootPathChar, ...
                           trialNameChar, ...
                           '5-statistics');

% create 5-statistics directory if it doesn't exist
if ~exist(currentStatsDir, 'dir')

    mkdir(currentStatsDir);

end % <-- end if

% attempt to save statistics for each phase into CSVs
try

    for phaseNum = 1:3

        % save raw phase data
        phaseDataName = strcat('phase', num2str(phaseNum), 'TT');
        phaseData = DS.(trialNameChar).(phaseDataName);
        csvFileName = ...
            fullfile(currentStatsDir, ...
                    strcat('phase', num2str(phaseNum), '.csv'));

        % write CSV
        writetimetable(phaseData, csvFileName);

        % post-process CSV to remove "NaN"s
        removeNaNsFromCSV(csvFileName);

        % report export success
        disp([' ', strcat('phase', num2str(phaseNum)), ...
            ' export success']);

        % save phase statistics, "S" in "Stats" should be uppercase
        phaseStatsName = strcat('phase', num2str(phaseNum), 'Stats');
        phaseStats = DS.(trialNameChar).(phaseStatsName);

        % prepare to add "stat" (should be lowercase) column name into
        % CSV file, we need to do this because MATLAB does not
        % normally include row variable names, we want this:
        %     CSV file contents:
        %         stat,psiaA0,...
        %         min,...
        %         max,...
        %         :

```

```

phaseStats = addvars(phaseStats, ...
                    phaseStats.Properties.RowNames, ...
                    'Before', 'psiaA0', ...
                    'NewVariableNames', 'stat');

% NOTE lowercase filename preference for
% "phase(phaseNum)stats.csv"
csvStatsFileName = ...
    fullfile(currentStatsDir, ...
              strcat('phase', num2str(phaseNum), 'stats.csv'));

% export CSV file
writetable(phaseStats, csvStatsFileName);

% post-process CSV to remove "NaN"s
removeNaNsFromCSV(csvStatsFileName);

% report export success
disp(['    ', strcat('phase', num2str(phaseNum), 'stats'), ...
      ' export success']);

end % <-- end for

catch ME

% report error if there's an issue exporting
disp(['    Error exporting trial: ', trialNameChar, ...
      '. Details: ', ME.message]);

end % <-- end try-catch

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
          DS ...
          M ...
          scriptPathChar ...
          sortedTrialDirsStruct ;

%% data integrity check
disp('data integrity check')

% to verify presence of essential fields in DS structure for
% statistical analysis

```



```

% expected fields in each trial
expectedFields = {'experimentTT', 'phase1TT', 'phase2TT', 'phase3TT', ...
                  'phase1Stats', 'phase2Stats', 'phase3Stats'};

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % display current trial number
    disp([' trial', num2str(i)])

    % check if trial exists in DS
    if ~isfield(DS, trialNameChar)

        error([' ', trialNameChar, ' is missing from DS!']);

    end % <-- end if

    % get current trial
    currentTrial = DS.(trialNameChar);

    % loop through expected fields
    for j = 1:length(expectedFields)

        % Check if field exists in current trial
        if ~isfield(currentTrial, expectedFields{j})

            error([' field ', expectedFields{j}, ...
                  ' is missing from ', trialNameChar, '!']);

        else

            % report data integrity check success
            disp([' ', expectedFields{j}, ' integrity check success'])

        end % <-- end if

    end % <-- end for

end % <-- end for

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
          DS ...

```

```

M ...
scriptPathChar ...
sortedTrialDirsStruct ;

%% count NaNs for each trial and phase
disp('count NaNs for each trial and phase')

% NaNs will cause problems with some statistics functions

% get list of trials
trialNamesCell = fieldnames(DS);

% initialize a count for NaN values
totalNaNCount = 0;

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % display current trial number
    disp([' trial', num2str(i), ': Counting NaNs for ', trialNameChar]);

    for phaseNum = 1:3

        phaseDataName = strcat('phase', num2str(phaseNum), 'TT');
        phaseData = DS.(trialNameChar).(phaseDataName);

        % count NaN values in current phase's gas reading
        currentNaNCount = sum(isnan(phaseData.Sensor1GasReading));

        % display NaN count for current phase
        disp([' phase ', num2str(phaseNum), ...
            ': ', num2str(currentNaNCount), ' NaNs']);

        % update total NaN count
        totalNaNCount = totalNaNCount + currentNaNCount;

    end % <-- end for

end % <-- end for

% display total NaN count
disp([' Total NaNs across all trials and phases: ', ...
    num2str(totalNaNCount)]);

%% cleanup workspace
disp(' cleanup workspace')

```

```

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
                DS ...
                M ...
                scriptPathChar ...
                sortedTrialDirsStruct ;

%% get duration for each phase
disp('get duration for each phase')

% get list of trials
trialNamesCell = fieldnames(DS);

% Initialize a matrix to store all durations
allDurations = strings(numel(trialNamesCell), 3); % assuming there are 3 phases

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % display current trial number
    disp([' trial', num2str(i)])

    for phaseNum = 1:3
        % create phase data name based on current phaseNum
        phaseDataName = strcat('phase', num2str(phaseNum), 'TT');

        % get specific phase data based on current trial and phase
        phaseData = DS.(trialNameChar).(phaseDataName);

        % get dateTime values from timetable
        dateTimeValues = phaseData.dateTime;

        % get start and end times
        startTime = dateTimeValues(1);
        endTime = dateTimeValues(end);

        % get duration of phase
        duration = endTime - startTime;

        % store duration in the matrix
        allDurations(i, phaseNum) = char(duration);

        % display duration of current phase for current trial
        disp([' phase', num2str(phaseNum), ': ', char(duration), ' (HH:mm:ss)']);
    end
end
end

```

```

% Convert the matrix to a table
durationTable = array2table(allDurations, 'VariableNames', {'phase1duration',
    'phase2duration', 'phase3duration'});

% Add the trial column to the table
trialNumbers = (1:numel(trialNamesCell))'; % Creates a column vector
durationTable = addvars(durationTable, trialNumbers, 'Before', 'phase1duration',
    'NewVariableNames', 'trial');

% Write the table to a CSV file
writetable(durationTable, 'duration.csv');

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
    DS ...
    M ...
    scriptPathChar ...
    sortedTrialDirsStruct ;

%% get summary for phase 2
disp('get summary for phase 2')

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)
    % get current trial name
    trialNameChar = trialNamesCell{i};

    % display current trial number
    disp([' trial', num2str(i)])

    for phaseNum = 2
        % create phase data name based on current phaseNum Stats
        phaseDataName = strcat('phase', num2str(phaseNum), 'Stats');

        % get mean and standard deviation values from phaseDataName
        meanRow = DS.(trialNameChar).(phaseDataName)('mean', :);
        stdRow = DS.(trialNameChar).(phaseDataName)('std', :);

        % get variable names (column names) from the table
        columnNames = meanRow.Properties.VariableNames;

        % loop through each column variable and display its mean and standard deviation
        for j = 1:numel(columnNames)
            columnName = columnNames{j};

```

```

        meanValue = meanRow.(columnName);
        stdValue = stdRow.(columnName);

        % Convert NaN to 'NaN' for display, else round to 2 decimal places
        if isnan(meanValue)
            meanValueStr = 'NaN';
        else
            meanValueStr = sprintf('%.2f', meanValue);
        end

        if isnan(stdValue)
            stdValueStr = 'NaN';
        else
            stdValueStr = sprintf('%.3f', stdValue);
        end

        disp([' ', columnName, ': ', meanValueStr, ' ± ', stdValueStr]);
    end
end

%% cleanup workspace
disp(' cleanup workspace')

% debug clear all variables except listed
clearvars -except dataRootPathChar ...
            DS ...
            M ...
            scriptPathChar ...
            sortedTrialDirsStruct ;

%% get summary table for phase 2
disp('get summary table phase 2')

% get list of trials
trialNamesCell = fieldnames(DS);

% get variable names (column names) from first trial for reference
columnNames = DS.(trialNamesCell{1}).phase2Stats.Properties.VariableNames;

% prepare new variable names
varNames = strcat('phase2', columnNames);

% initialize dataTable to store formatted values with an extra column for
% "trial"
dataTable = cell(numel(trialNamesCell), numel(columnNames) + 1);

% loop through each trial
for i = 1:numel(trialNamesCell)

```

```

% get current trial name
trialNameChar = trialNamesCell{i};

% add trial number to the first column of dataTable
dataTable{i, 1} = num2str(i);

% get mean and standard deviation values for phase2
meanRow = DS.(trialNameChar).phase2Stats('mean', :);
stdRow = DS.(trialNameChar).phase2Stats('std', :);

% loop through each column variable and store its formatted value in
% dataTable
for j = 1:numel(columnNames)

    columnName = columnNames{j};
    meanValue = meanRow.(columnName);
    stdValue = stdRow.(columnName);

    % convert NaN to 'NaN' for display, else round to 2 decimal places
    if isnan(meanValue)

        meanValueStr = 'NaN';

    else

        meanValueStr = num2str(meanValue, '%.2f');

    end % <-- end if

    if isnan(stdValue)

        stdValueStr = 'NaN';

    else

        stdValueStr = num2str(stdValue, '%.2f');

    end % <-- end if

    dataTable{i, j+1} = [meanValueStr, ' ± ', stdValueStr];
end
end

% we are doing all this just to display the table without any {''} or ""
% omg MATLAB

% add "trial" to start of varNames
varNames = ['trial', varNames];

% convert cell array dataTable into a MATLAB table
summaryTable = cell2table(dataTable, 'VariableNames', varNames);

```

```

% convert to string table
summaryTable = convertvars(summaryTable, varNames, 'string');

% capture the output of disp(summaryTable) with strings "value"
outputStr = evalc('disp(summaryTable)');

% replace the quotes with spaces
outputStrWithoutQuotes = strrep(outputStr, '"', ' ');

% display the processed string
disp(outputStrWithoutQuotes);

% overwrite previous summary.csv
writetable(summaryTable, 'summary.csv');

%% comparing phases
disp('comparing phases')

% compare three phases to assess if there's a significant difference in
% ammonia removal

disp('check for normality')

% use Lilliefors test on ammonia gas meter readings for each phase in each
% trial

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % display current trial number
    disp([' trial', num2str(i)])

    for phaseNum = 1:3

        phaseDataName = strcat('phase', num2str(phaseNum), 'TT');
        phaseData = DS.(trialNameChar).(phaseDataName);

        if length(phaseData.Sensor1GasReading) >= 4

            h = lillietest(phaseData.Sensor1GasReading);

            if h == 1

                disp([' phase', num2str(phaseNum), ...
                    ' Sensor1GasReading is not normally distributed'])
            end
        end
    end
end

```

```

        end % <-- end if

    else

        disp(['    phase', num2str(phaseNum), ' Sensor1GasReading ', ...
            'has less than 4 valid observations, ' ...
            'Lilliefors test skipped'])

        end % <-- end if

    end % <-- end for

end % <-- end for

% NOTE: all data across trials and phases are not normally distributed

%% use Friedman tests
disp('use Friedman tests')

disp(' compare phase to Sensor1GasReading (ammonia gas in ppm)')

% since we are comparing three phases across different trials, the Friedman
% test is appropriate, as it is non-parametric alternative to repeated
% measures ANOVA

% organize data such that it's in a matrix format where each column
% represents a phase (i.e., phase 1, phase 2, and phase 3) and each row
% represents a trial

numTrials = length(fieldnames(DS));

% pre-allocate space for 3 phases
dataMatrix = zeros(numTrials, 3);

% trial names fetched for the Lilliefors normality test will also be used in
% constructing the dataMatrix for the Friedman test
for i = 1:numTrials

    trialNameChar = trialNamesCell{i};
    dataMatrix(i, 1) = mean(DS.(trialNameChar).phase1TT.Sensor1GasReading);
    dataMatrix(i, 2) = mean(DS.(trialNameChar).phase2TT.Sensor1GasReading);
    dataMatrix(i, 3) = mean(DS.(trialNameChar).phase3TT.Sensor1GasReading);

end % <-- end for

% Friedman test is a non-parametric alternative to repeated measures ANOVA.
% we use it when data does not meet assumptions for a parametric test.
% - dataMatrix: matrix where each column represents a phase, and each
%   row represents a trial

```



```

% - 1: ranking of data is done by rows, which compares phases across
% trials
% - 'off': suppresses default display of the test statistics and plots
[p, tbl, stats] = friedman(dataMatrix, 1, 'off');

% display friedman table
% loop over rows
fprintf(' %s\n', repmat('-', 1, 72)) % horizontal line
for r = 1:size(tbl, 1)

    fprintf(' ') % indent

    % loop over columns
    for c = 1:size(tbl, 2)

        cell_content = tbl{r, c};

        % check if the content is a string (char array)
        if ischar(cell_content)

            fprintf('%-12s', cell_content)

        % check if the content is a numeric value
        elseif isnumeric(cell_content)

            if isempty(cell_content)

                fprintf('%-12s', ' ')

            else

                fprintf('%-12.4f', cell_content)

            end % <-- end if

        end % <-- end if

    end % <-- end for

    % new line after printing all columns of the current row
    fprintf('\n');

end % <-- end for
fprintf(' %s\n', repmat('-', 1, 72)) % horizontal line

% Post-Hoc Analysis with Wilcoxon Signed-Rank Test

% check to see if p value from Friedman test is significant (typically less
% than 0.05), which would mean that at least one of phases is significantly
% different from others. to determine which phases are different, we can
% conduct post-hoc pairwise comparisons using the Wilcoxon signed-rank test

```

```

% NOTE: since we will be conducting multiple comparisons, we need to
% correct for this to avoid inflated Type I errors, we will use Bonferroni
% correction for this. therefore, we will perform pairwise comparisons,
% where we are trying to determine which pairs of phases have significant
% differences in their means
%
% our primary interest is phase 2, so if h12 and/or h23 are 1, that
% indicates phase 2 is significantly different from either phase 1 or phase
% 3, respectively.

if p < 0.05

    % adjusted alpha level for Bonferroni correction (3 comparisons)
    alpha_adjusted = 0.05 / 3;

    % pairwise comparison between phase 1 and phase 2
    [p12, h12] = signrank(dataMatrix(:,1), dataMatrix(:,2), ...
        'alpha', alpha_adjusted);

    % pairwise comparison between phase 1 and phase 3
    [p13, h13] = signrank(dataMatrix(:,1), dataMatrix(:,3), ...
        'alpha', alpha_adjusted);

    % pairwise comparison between phase 2 and phase 3
    [p23, h23] = signrank(dataMatrix(:,2), dataMatrix(:,3), ...
        'alpha', alpha_adjusted);

    % display significant differences (if any)

    if h12 == 1

        disp(' phase 1 and phase 2 are significantly different.');
```

```

% - phase 1 and phase 3 have significantly different gas readings
% - No significant difference was reported between phase 2 and phase 3

%% exploratory data analysis (EDA)
disp('exploratory data analysis (EDA)')

disp(' compare trials to Sensor1GasReading (ammonia gas in ppm)')

% visualize ammonia readings across phases and trials

% Sensor1GasReading contains ammonia readings

% number of trials
numTrials = length(fieldnames(DS));

% plot ammonia readings
figure;
for i = 1:numTrials

    subplot(4,4,i, "replace");

    % get readings for three phases
    phase1 = DS.(['trial', num2str(i)]).phase1TT.Sensor1GasReading;
    phase2 = DS.(['trial', num2str(i)]).phase2TT.Sensor1GasReading;
    phase3 = DS.(['trial', num2str(i)]).phase3TT.Sensor1GasReading;

    % convert each phase data into a cell array, and then concatenate them
    allPhases = [{phase1}; {phase2}; {phase3}];

    % Check if each phase is numeric and print its size
    for j = 1:length(allPhases)

        if isnumeric(allPhases{j})

            numericStatus = 'true';

        else

            numericStatus = 'false';

        end % <--end if

        disp([' Is phase ', num2str(j), ' numeric? ', numericStatus]);

        disp([' Size of phase ', num2str(j), ': ', ...
            num2str(size(allPhases{j}))]);

    end % <--end for

    % create a grouping variable
    group = [ones(size(phase1)); 2*ones(size(phase2)); 3*ones(size(phase3))];

```

```

    % plot using boxplot
    boxplot(cell2mat(allPhases), group, "Widths", 0.25);
    ylabel('Ammonia (PPM)');
    title(['Trial ' num2str(i)]);

end % <--end for

% save current figure
saveCurrentFigureAsImage('phases_for_each_trial_vs_ammonia_gas', ...
    7.5, 7.5);

%% correlation analysis
disp('correlation analysis')

% number of trials
numTrials = length(fieldnames(DS));

% preallocate arrays to store flow rates for each phase
flowRatesPhase1 = zeros(1, numTrials);
flowRatesPhase2 = zeros(1, numTrials);
flowRatesPhase3 = zeros(1, numTrials);

% preallocate arrays to store mean ammonia readings for each phase
meanAmmoniaPhase1 = zeros(1, numTrials);
meanAmmoniaPhase2 = zeros(1, numTrials);
meanAmmoniaPhase3 = zeros(1, numTrials);

for i = 1:numTrials

    flowRatesPhase1(i) = ...
        mean(DS.(['trial', num2str(i)]).phase1TT.volumetricFlowRateStandardizedSCCM);
    flowRatesPhase2(i) = ...
        mean(DS.(['trial', num2str(i)]).phase2TT.volumetricFlowRateStandardizedSCCM);
    flowRatesPhase3(i) = ...
        mean(DS.(['trial', num2str(i)]).phase3TT.volumetricFlowRateStandardizedSCCM);

    meanAmmoniaPhase1(i) = ...
        mean(DS.(['trial', num2str(i)]).phase1TT.Sensor1GasReading);
    meanAmmoniaPhase2(i) = ...
        mean(DS.(['trial', num2str(i)]).phase2TT.Sensor1GasReading);
    meanAmmoniaPhase3(i) = ...
        mean(DS.(['trial', num2str(i)]).phase3TT.Sensor1GasReading);

end % <-- end for

% linear or rank correlation
R1 = corr(flowRatesPhase1', meanAmmoniaPhase1');
R2 = corr(flowRatesPhase2', meanAmmoniaPhase2');
R3 = corr(flowRatesPhase3', meanAmmoniaPhase3');

```

```

disp(' for all trials')
disp([' correlation between phase 1 flow rate and phase 1 mean ammonia: ', num2str(R1)])
disp([' correlation between phase 2 flow rate and phase 2 mean ammonia: ', num2str(R2)])
disp([' correlation between phase 3 flow rate and phase 3 mean ammonia: ', num2str(R3)])

% The correlation coefficient for phase 2 flow rate and ammonia is
% approximately 0.28978, signifying a weak positive linear association

% a positive value suggests that as phase 2 flow rate increases, the
% ammonia level also has a tendency to increase, this relationship is weak,
% implying that flow rate alone might not be a strong predictor of ammonia
% levels. also no ammonia was detected in phase 1 or phase 3, which is
% consistent with experimental expectations

%% linear regression
disp('linear regression');

% fit a simple linear regression model to see how flow rate predicts the
% mean ammonia reading for each phase

lm1 = fitlm(flowRatesPhase1, meanAmmoniaPhase1, 'Linear');
lm2 = fitlm(flowRatesPhase2, meanAmmoniaPhase2, 'Linear');
lm3 = fitlm(flowRatesPhase3, meanAmmoniaPhase3, 'Linear');

disp(lm1);
disp(lm2);
disp(lm3);

% First model: All coefficients are 0, and R-squared is NaN. This indicates
% an issue with model, possibly no variation in dependent or
% independent variable.

% Second model: predictor x1 has a p-value of 0.27631, meaning it's not
% statistically significant at conventional levels (like 0.05). The
% R-squared of 0.084 suggests that model explains about 8.4% of the
% variability in dependent variable.

% Third model: Has an insignificant predictor with a p-value of 0.5557.
% This model explains only about 2.74% of variability in dependent
% variable.

%% visualization
disp('visualization')

% plotting volumetric flow rate against ammonia gas reading for each trial
% or phase may reveal patterns and trends, if flow rate is a significant
% factor, we might observe a consistent trend across trials or phases

```

```

% set up figure
figure;

% colors for phase 1, 2, and 3
colors = {'r', 'g', 'b'};

% loop through each phase
for phaseNum = 1:3

    % preallocate for storing data across all trials
    allFlowRates = [];
    allAmmoniaReadings = [];

    % loop through each trial
    for i = 1:numTrials

        trialNameChar = ['trial', num2str(i)];
        phaseName = ['phase', num2str(phaseNum), 'TT'];

        % get volumetric flow rate for current trial and phase
        flowRates = DS.(trialNameChar).(phaseName).volumetricFlowRateStandardizedSCCM;

        % get ammonia readings for current trial and phase
        ammoniaReadings = DS.(trialNameChar).(phaseName).Sensor1GasReading;

        % append data to arrays
        allFlowRates = [allFlowRates; flowRates];
        allAmmoniaReadings = [allAmmoniaReadings; ammoniaReadings];

    end % <-- end for

    % plot data for current phase
    subplot(3, 1, phaseNum);
    scatter(allFlowRates, allAmmoniaReadings, 10, colors{phaseNum}, 'filled'); % '10' sets
        marker size
    title(['Phase ', num2str(phaseNum)]);
    xlabel('Volumetric Flow Rate (Standardized SCCM)');
    ylabel('Ammonia Gas Reading');
    grid on;

end % <-- end for

% adjust layout
sgtitle('Volumetric Flow Rate vs. Ammonia Gas Reading (All Trials)');

% save current figure
saveCurrentFigureAsImage('volumetric_flow_rate_vs_ammonia_gas_reading', ...
    7.5, 7.5);

%% plot mean variables for each variable name for phase 2
disp('plot mean variables for phase 2')

```

```

% initialize cell arrays to store results
varNames = {};
meanRows = [];
semRows = [];

% get list of trials
trialNamesCell = fieldnames(DS);

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % get variable names for phase2Stats for current trial we only need to
    % do this once since variable names are the same across trials
    if i == 1

        varNames = DS.(trialNameChar).phase2Stats.Properties.VariableNames;

    end % <-- end if

    % get mean and sem rows for each variable for plotting
    for j = 1:numel(varNames)

        varName = varNames{j};
        meanRows(i, j) = DS.(trialNameChar).phase2Stats{'mean', varName};
        semRows(i, j) = DS.(trialNameChar).phase2Stats{'sem', varName};

    end % <-- end for

end % <-- end for

% number of trials and variables
[numTrials, numVars] = size(meanRows);

% Create a colormap for 16 different colors
colors = lines(numTrials);

% loop through each variable to create individual plots
for v = 1:numVars

    % create a new figure for each variable
    figure;
    hold on;

    % bar graph with different colors for each bar
    b = bar(meanRows(:, v), 'FaceColor', 'flat');
    b.CData = colors; % Assign the colors to the bars

    % error bars
    xCenters = b.XEndPoints; % X centers of the bars

```

```

hErrBar = errorbar(xCenters, meanRows(:, v), semRows(:, v), ...
    'k', 'LineStyle', 'none');
hErrBar.LineWidth = 1; % Set error bar line width

% x-axis labels
xticks(1:numTrials);

% numbers 1 to 16
xticklabels(arrayfun(@num2str, 1:numTrials, 'UniformOutput', false));

% label for the x-axis
xlabel('Trials');

% title, y-axis label
title(['Phase 2 Mean ' varNames{v} ' (SEM error bars)']);
ylabel(varNames{v});

hold off;

% save current figure
saveCurrentFigureAsImage(strcat('mean_', num2str(v), '_', varNames{v}));

end % <-- end for

%% compare volume of ammonium hydroxide to ammonia gas readings
disp('compare volume of ammonium hydroxide to ammonia gas readings');

% get list of trials
trialNamesCell = fieldnames(DS);
volumeAmmonium = zeros(numel(trialNamesCell), 1);
meanAmmoniaPhase2 = zeros(numel(trialNamesCell), 1);

% get mean reading for control trial (Trial 8)
controlMean = mean(DS.trial8.phase2TT.Sensor1GasReading, "omitnan");

% loop through each trial
for i = 1:numel(trialNamesCell)

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % get ammonium volume for current trial
    volumeAmmonium(i) = M.volumeNH4OHUMOLL(i);

    % get mean sensor reading for phase 2 and adjust by subtracting control
    % value
    if i ~= 8 % don't adjust the control trial

        meanAmmoniaPhase2(i) = ...
            mean(DS.(['trial', num2str(i)]).phase2TT.Sensor1GasReading, ...
                "omitnan") - controlMean;
    end
end

```



```

else

    meanAmmoniaPhase2(i) = controlMean;

end % <-- end if

end % <-- end for

% scatter plot for phase 2
figure;

% loop again for scatter plot with colors and legend
hold on;
colors = hsv(numel(trialNamesCell));

for i = 1:numel(trialNamesCell)

    scatter(volumeAmmonium(i), meanAmmoniaPhase2(i), ...
        50, colors(i,:), 'DisplayName', sprintf('Trial %d', i));

end % <-- end for

xlabel('Ammonium (NH4+) (PPM = mg/L)');
xRange = max(volumeAmmonium) - min(volumeAmmonium);
xlim([min(volumeAmmonium)-0.10*xRange, max(volumeAmmonium)+0.10*xRange]);
ylabel('Adjusted Mean Ammonia (NH3) (PPM = mg/L)');
yRange = max(meanAmmoniaPhase2) - min(meanAmmoniaPhase2);
ylim([min(meanAmmoniaPhase2)-0.10*yRange, max(meanAmmoniaPhase2)+0.10*yRange]);
title('Phase 2 Ammonium (NH4+) vs. Adjusted Mean Ammonia (NH3)');
legend('Location', 'eastoutside');
hold off;

% calculate correlation for phase 2
[R, P] = corrcoef(volumeAmmonium, meanAmmoniaPhase2);
disp([' correlation coefficient for phase 2: ', num2str(R(1,2))]);
disp([' p-value for phase 2: ', num2str(P(1,2))]);

saveCurrentFigureAsImage('volume_of_ammonium_and_adjusted_mean_ammonia_phase_2')

% calculate correlation for phase 2
% the Pearson correlation coefficient is trying to quantify the
% strength and direction of the linear relationship between the two
% datasets, and the p-value, represents the statistical significance of
% the observed correlation

% correlation coefficient for phase 2: 0.66776: suggests a moderate positive
% linear relationship between the volume of ammonium hydroxide and the mean
% ammonia gas reading during phase 2. In other words, as the volume of
% ammonium hydroxide increases, the mean ammonia gas reading during phase 2
% tends to increase to a moderate extent.
%
% p-value for phase 2: 0.0047022: a p-value below 0.05 is considered

```

```

% statistically significant, implying that there's a low probability that
% the observed correlation occurred by random chance. This means that the
% positive correlation between the volume of ammonium hydroxide and the
% mean ammonia gas reading during phase 2 is statistically significant, and
% it's unlikely to be due to random chance
%
% conclusion: analysis suggests that there's a statistically significant
% moderate positive relationship between the volume of ammonium hydroxide
% and the mean ammonia gas readings in phase 2, since as we add more
% ammonium hydroxide, we can expect an increase in the mean ammonia gas
% reading during phase 2, at least to a certain extent

%% analysis based on presence of PFC
disp('analysis based on presence of PFC');

% control trial ammonia reading
controlAmmonia = mean(DS.trial8.phase2TT.Sensor1GasReading, "omitnan");

% group data based on PFC
trialsWithoutPFC = 1:13;
trialsWithPFC = 14:16;

% preallocate for mean ammonia readings adjusted for control
adjustedAmmoniaWithoutPFC = zeros(length(trialsWithoutPFC), 1);
adjustedAmmoniaWithPFC = zeros(length(trialsWithPFC), 1);

% mean ammonia readings for trials without PFC adjusted for control
for i = trialsWithoutPFC
    adjustedAmmoniaWithoutPFC(i) = ...
        mean(DS.(['trial', num2str(i)]).phase2TT.Sensor1GasReading, ...
            "omitnan") - controlAmmonia;
end

% mean ammonia readings for trials with PFC adjusted for control
for i = trialsWithPFC
    adjustedAmmoniaWithPFC(i - length(trialsWithoutPFC)) = ...
        mean(DS.(['trial', num2str(i)]).phase2TT.Sensor1GasReading, ...
            "omitnan") - controlAmmonia;
end

% get overall average for both groups
avgAmmoniaWithoutPFC = mean(adjustedAmmoniaWithoutPFC);
avgAmmoniaWithPFC = mean(adjustedAmmoniaWithPFC);

% display results
disp([' Adjusted average ammonia reading without PFC: ', ...
    num2str(avgAmmoniaWithoutPFC)]);
disp([' Adjusted average ammonia reading with PFC: ', ...
    num2str(avgAmmoniaWithPFC)]);

% statistical comparison using t-test

```

```

[h, p] = ttest2(adjustedAmmoniaWithoutPFC, adjustedAmmoniaWithPFC);
disp([' p-value for difference between groups: ', num2str(p)]);

% significance level (e.g., 0.05)
if p < 0.05

    disp([' The difference in adjusted ammonia readings between ' ...
        'the groups is statistically significant.']);

else

    disp([' The difference in adjusted ammonia readings between ' ...
        'the groups is not statistically significant.']);

end % <-- end if

% adjusted average ammonia readings: after accounting for the control, on
% average, the trials without PFC showed an adjusted ammonia reading of
% 1.771, while the trials with PFC showed an adjusted ammonia reading of
% 2.2976
%
% statistical significance: a p-value of 0.84248 is much greater than the
% commonly used significance threshold of 0.05, indicating that the
% observed differences between the two groups could easily have occurred by
% random chance. based on this data, the introduction of PFC doesn't
% have a statistically significant impact on ammonia readings, at least as
% measured in these trials

%% pairwise comparisons for similar ammonium hydroxide concentrations
disp('pairwise comparisons for similar ammonium hydroxide concentrations')

% compare trials that have similar or same volume of ammonium hydroxide
% (volumeNH4OHUL) to discern the effect of other factors like PFC

% trial  volumeRODIWaterML  volumePFCML  volumeNH4OHUL  Sensor1GasReading
% 1      100.00              0.00             1515.00         5.86
% 2      100.00              0.00             15.14          0.00
% 3      100.00              0.00             22.70          0.00
% 4      100.00              0.00             30.28          0.00
% 5      100.00              0.00             37.85          0.00
% 6      100.00              0.00             45.42          0.00
% 7      100.00              0.00             52.99          0.00
% 8      100.00              0.00              0.00          0.00
% 9      100.00              0.00             15.14          0.00
% 10     100.00              0.00            1514.00        15.37
% 11     100.00              0.00             151.40         0.60
% 12     100.00              0.00             151.40         0.51
% 13     100.00              0.00             151.40         0.69
% 14      50.00             50.00             757.00         2.15

```

```

% 15      50.00      50.00      757.00      2.45
% 16      50.00      50.00      757.00      2.29

% identify pairs of trials with similar ammonium hydroxide concentrations
% based on the given data above from meta.csv, the possible pairs could be:
pairs = {[ 1, 10], ... % ~1515.00 uL of ammonium hydroxide
         [11, 12, 13], ... % 151.40 uL of ammonium hydroxide
         [14, 15, 16], ... % 757.00 uL of ammonium hydroxide
        };

% loop through each pair and perform pairwise comparison
for k = 1:length(pairs)

    currentPair = pairs{k};

    % get ammonia readings for each trial in pair
    ammoniaReadingsCell = cell(1, length(currentPair));
    for j = 1:length(currentPair)

        ammoniaReadingsCell{j} = ...
            DS(['trial', num2str(currentPair(j))]).phase2TT.Sensor1GasReading;

    end % <-- end for

    % pairwise t-tests between trials in pair
    for m = 1:(length(currentPair) - 1)

        for n = (m+1):length(currentPair)

            [h, p] = ttest2(ammoniaReadingsCell{m}, ammoniaReadingsCell{n});
            disp([' Pairwise comparison between Trial ', ...
                  num2str(currentPair(m)), ' and Trial ', ...
                  num2str(currentPair(n)), ': p-value = ', num2str(p)]);

        end % <-- end for

    end % <-- end for

end % <-- end for

% Trials [1, 10]: ~1515 uL of ammonium hydroxide
%   - Trial 1 vs. Trial 10: p-value = 0
%   - p-value indicates statistically significant difference between
%     outcomes of Trials 1 and 10, even though both have almost the same
%     volume of ammonium hydroxide
%   - p-value of 0 suggests differences between these two trials are not
%     due to random chance
%
% Trials [11, 12, 13]: ~151.4 uL of ammonium hydroxide
%   - Trial 11 vs. Trial 12: p-value = 4.3721e-05
%   - Trial 11 vs. Trial 13: p-value = 1.1782e-05
%   - Trial 12 vs. Trial 13: p-value = 1.8034e-15

```

```

% - these low p-values indicate significant differences between
% outcomes of these trials, despite them having very similar ammonium
% hydroxide concentrations
%
% Trials [14, 15, 16]: ~757 uL of ammonium hydroxide
% - Trial 14 vs. Trial 15: p-value = 2.1386e-07
% - Trial 14 vs. Trial 16: p-value = 0.015725
% - Trial 15 vs. Trial 16: p-value = 0.009133
% - all comparisons here show statistically significant differences
% between the outcomes of these trials

%% get ammonia plot of all trials
disp('get ammonia plot of all trials')

% get list of trials
trialNamesCell = fieldnames(DS);

% number of trials
numTrials = numel(trialNamesCell);

% prepare a matrix for subplots
numRows = ceil(sqrt(numTrials));
numCols = ceil(numTrials / numRows);

% create a new figure
figure;

% use tiledlayout for plotting
t = tiledlayout(numRows, numCols, 'TileSpacing', 'compact');
title(t, 'Ammonia Readings Across All Trials and Phases', ...
      'FontSize', 14, 'FontWeight', 'bold');

% create empty arrays to store handles for plot lines
h = zeros(3,1);

% colors for each phase
phaseColors = ['r', 'b', 'g'];

% define the gap between phases
gap = 0.5; % in seconds

% loop through each trial
for i = 1:numTrials

    % get current trial name
    trialNameChar = trialNamesCell{i};

    % create a new tile for each trial
    ax = nexttile;
    hold on;

```

```

% initialize elapsed time
elapsed = 0;

% loop through each phase
for phaseNum = 1:3

    % fetch phase data
    phaseDataName = strcat('phase', num2str(phaseNum), 'TT');
    phaseData = DS.(trialNameChar).(phaseDataName);

    % calculate duration of current phase in seconds
    duration = seconds(phaseData.dateTime(end) - phaseData.dateTime(1));

    % calculate elapsed time for current phase
    elapsedTime = elapsed + ...
        (0:duration/(numel(phaseData.Sensor1GasReading)-1):duration)';
    elapsed = elapsed + duration + gap;

    % plot data for current phase
    plot(ax, elapsedTime, phaseData.Sensor1GasReading, ...
        phaseColors(phaseNum), ...
        'DisplayName', ['Phase ' num2str(phaseNum)]);
    % xline(elapsed, '--', strcat('phase', num2str(phaseNum)), ...
    %       'LabelHorizontalAlignment', 'left', ...
    %       'LabelVerticalAlignment', 'bottom');

end % end for

title(ax, ['Trial ' num2str(i)]);
ylabel(ax, 'Ammonia (PPM)');
xlabel(ax, 'Elapsed Time (seconds)');
% grid(ax, 'on');
% ylim([0, 30]);
% xlim([0, 6000]);

hold off;

end % end for

% create a single legend at bottom using handles from first trial
lg = legend(ax, 'Location', 'southoutside', 'Orientation', 'horizontal');
lg.Layout.Tile = 'south';

% save current figure as png
saveCurrentFigureAsImage('gas_ammonia', 8.5, 8.5);

%% statistics finished
disp('====statistics finished====')

```

```
%% program finished
disp('program finished')
```

```
%% logging stop
% stop logging to txt file
diary off;
```

```
%% FUNCTIONS
```

```
% -----
```

```
function removeNaNsFromCSV(fileName)
```

```
% INPUT EXPECTS fileName
%   - full path to CSV file to be processed
% OUTPUT NONE
%   - specified CSV file is modified in-place, with "NaN" entries
%     replaced by empty strings
```

```
fileContent = fileread(fileName);
% replace 'NaN' with empty string
fileContent = strrep(fileContent, 'NaN', '');
fid = fopen(fileName, 'w');
fwrite(fid, fileContent);
fclose(fid);
```

```
end % <-- end function
```

```
function outputTable = descStat(inputTable)
```

```
% this function computes various descriptive statistics (n, min, max,
% mean, median, mode, std, var, rms, sem) for columns of an input
% table, and then returns a table where each row contains those
% statistics. the input can have NaNs in the column values
%
```

```
% INPUT:
```

```
%   - inputTable: a 2D table or timetable where each column will have
%     statistics computed
```

```
%
```

```
% OUTPUT:
```

```
%   - outputTable: a 2D table containing computed statistics for each
%     column of input table, where each row of tableOutput corresponds
%     to a statistic (e.g., n, min, max, ...)
```

```
% get column names from input table or timetable
```

```

inputColNamesCell = inputTable.Properties.VariableNames;

% convert table to array for statistical functions
inputDouble = table2array(inputTable);

% compute basic statistics.
N = sum(~isnan(inputDouble)); % number of rows (samples)
STD = std(inputDouble, 'omitnan'); % standard deviation (STD)
SEM = STD ./ sqrt(N); % standard error of the mean (SEM)

% consolidate statistics for table creation.
stats = ...
    [ N; ...
      min(inputDouble, [], 'omitnan'); ... % minimum elements of an array
      max(inputDouble, [], 'omitnan'); ... % maximum elements of an array
      mean(inputDouble, 'omitnan'); ... % average or mean value
      median(inputDouble, 'omitnan'); ... % median value
      mode(inputDouble); ... % mode, or most frequent value (NaN ignored)
      STD; ...
      var(inputDouble, 'omitnan'); ... % variance
      rms(inputDouble, 'omitnan'); ... % root mean square
      SEM];

% list of computed statistics
statsFunctions = ...
    [ "n", ...
      "min", ...
      "max", ...
      "mean", ...
      "median", ...
      "mode", ...
      "std", ...
      "var", ...
      "rms", ...
      "sem"];

% convert statistics array to output table
outputTable = ...
    array2table(stats, ...
        'VariableNames', inputColNamesCell, ...
        'RowNames', statsFunctions);

end % <-- end function

function saveCurrentFigureAsImage(figureTitle, width, height)

% this function saves the current figure as a PNG image file

% INPUT:
% - figureTitle (string): title of the figure, used as the filename
% - width (numeric, optional): width of the image in inches

```



```

% - height (numeric, optional): height of the image in inches
% OUTPUT:
%   a PNG image file saved to the current working directory

% check if width is provided, if not, set to default
if nargin < 2 || isempty(width)

    width = 7.5; % default width in inches

end % <-- end if

% check if height is provided, if not, set to default
if nargin < 3 || isempty(height)

    height = 4.21875; % default height in inches

end % <-- end if

% replace non-alphanumeric characters in figureTitle with "_"
figureTitle = regexprep(figureTitle, "\W", "_");

% set the size and position of the figure
%   position [left bottom width height] in inches
set(gcf, 'units', 'inches', 'position', [0, 0, width, height]);

% try to save the figure as a PNG file
try

    exportgraphics(gcf, strcat("image_", figureTitle, ".png"), ...
        "Resolution", 300);

catch ME

    % display an error message if saving fails
    disp(['Error: ', ME.message]);

end % <-- end try

end % <-- end function

```
