# AN ELECTROCHEMICAL STRATEGY FOR DECENTRALIZED TREATMENT AND AMMONIA RECOVERY FROM HIGH-STRENGTH WASTEWATER

By

Blake Smerigan

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Biosystems Engineering – Master of Science

2025

## ABSTRACT

Water scarcity is a growing global concern, particularly in decentralized and rural areas where novel methods of decentralized high-strength wastewater treatment must be developed. One such technology is a combined electrochemical process of electrocoagulation (EC) and electrodialysis (ED). This study analyzed the viability of this process by performing pilot-scale testing, life cycle assessment (LCA), and techno-economic assessment (TEA) on the treatment of three different high-strength blackwaters: sewage sludge, port-o-john, and latrine. The process was the most viable in the treatment of sewage-sludge blackwater achieving a COD reduction of ~96% (to 75.3 ± 25.8 mg/L) and a $NH_3$-N reduction of ~97% (to 3.9 ± 1.3 mg/L) with a treatment cost of 39.6 $\$/m^3$ treated water. The process was least viable for the port-o-john blackwater as its high pollutant loading and pH causing a COD and $NH_3$-N reduction of ~90% (to 245 ± 136) and ~65% (to 156 ± 9.54) respectively with a cost of 39.8 $\$/m^3$ treated water. The latrine blackwater had a COD and NH3-N reduction of ~88% (to 201 ± 40.7) and ~70% (to 78.6 ± 7.5) respectively with a cost of 40.4 $\$/m^3$ treated water. The decrease in NH3-N removal for the port-o-john and latrine blackwater is likely due to their high pH. This also had negative impacts on their LCA. Sewage sludge, port-o-john, and latrine blackwater had negative global warming potentials (-2.42, -2.11, and -1.59 metric tons $CO_2$-eq/year) due to their low energy demand and compatibility with solar powered operation, however; they had high water eutrophication potentials of 5.33, 71.18, and 54.83 kg N eq/year respectively. Overall, the EC-ED process represents a low-carbon, decentralized alternative to conventional wastewater treatment, but nutrient leakage remains a concern especially for highly concentrated wastewaters. Additional studies are needed to further develop it as a circular sanitation process by improving its nutrient recovery to complement its energy and climate advantages.

## ACKNOWLEDGEMENTS

I would like to take this time to thank everyone who has helped with this thesis. Without their contribution, this work would not have been possible.

I will begin by thanking my major advisor, Dr. Wei Liao. Your knowledge, guidance, and support throughout this research was instrumental. Your passion for research and work ethic are inspirational.

Thank you to Dr. Yan (Susie) Liu for serving on my committee and aiding me through the masters program. Your help both in research and throughout the enrollment process helped reduce the number of late nights and procedural stress.

Thank you to Dr. Christopher Saffron for serving on my committee.

Thank you to Dr. Ben Thomas, Dr. James Dusenbury, and the entire Ground Vehicle Systems Center team for their knowledge, guidance, and expertise in pilot scale design and field testing.

I would also like to thank the entire Liao research group. I would not have been able to do this without your support and comradery throughout this process. Thank you to Carter Monson, Emilia Emerson, John Grivins, Gus Aburto, Meicai Xu, Annalease Marks, Daniel Aburto, and Kelsey Andrews. Special thanks to Dr. Sibel Uludag-Demirer who's mentorship, guidance, and friendship helped both my academic and personal growth.

Thank you to the Department of Defense for funding this research.

Finally, I would like to thank my family and Charlotte Sass for their love and encouragement. You have always been there for me when I need a shoulder to lean on.

**TABLE OF CONTENTS**

# CHAPTER 1: LITERATURE REVIEW

## INTRODUCTION

Across the globe, water scarcity is an ever-increasing problem due to increased demand and climate change (Liu et al. 2017). A major contributor to water scarcity is inadequate wastewater treatment which contaminates water sources (Tzanakakis, Paranychianakis, and Angelakis 2020). According to the World Health Organization, over 1.5 billion people lack access to adequate sanitation services with approximately 419 million people practicing open defecation resulting in severe environmental, human health, and social concerns. This is especially prevalent in rural, decentralized, and developing areas (Anon n.d.-b). These areas disproportionately lack adequate sanitation services due to additional challenges associated with the cost of wastewater transport and infrastructure (Muzioreva et al. 2022). An emphasis has been placed on improving sanitation especially in areas where practices such as open defecation are used as these practices perpetuate a cycle of poverty and result in severe health issues and death due to disease transmission (Geetha Varma et al. 2022).

Wastewater offers a unique opportunity for resource recovery and circular economies as it contains high concentrations of valuable resources such as ammonia (Chen et al. 2021). Traditionally, wastewater treatment processes focus on removing these compounds to prevent environmental pollution, often converting ammonia into nitrogen gas through biological nitrification-denitrification (Zhang and Liu 2021). However, with growing concerns over resource scarcity and sustainability, there is increasing interest in recovering ammonia as a valuable product rather than simply removing it. Ammonia recovery is particularly important because it serves as a key raw material for fertilizers, contributing to global food production. Additionally, recovering ammonia from wastewater can reduce the reliance on energy-intensive

Haber-Bosch synthesis, which is responsible for significant greenhouse gas emissions (Anon n.d.-a). Various technologies, such as struvite precipitation, air stripping, membrane-based separation, and electrochemical processes, have been explored for ammonia recovery, each offering different efficiencies depending on wastewater characteristics (Robles et al. 2020a). By shifting from removal to recovery, wastewater treatment can transition toward a more circular approach, reducing environmental impact while generating valuable byproducts.

## LITERATURE REVIEW

### ***Decentralized wastewater treatment***

Rural and decentralized areas face unique challenges that cannot be effectively addressed by traditional municipal wastewater treatment systems. The lower population densities in these regions mean that wastewater must be transported over much greater distances to reach centralized treatment facilities, significantly increasing infrastructure costs (Massoud, Tarhini, and Nasr 2009; Muzioreva et al. 2022). Municipal systems often rely on economies of scale, using the high wastewater throughput to justify advanced treatment technologies such as anaerobic digestion of activated sludge and large biological treatment units like aeration tanks (Massoud et al. 2009). However, these capital-intensive infrastructures are often not feasible for decentralized communities, forcing them to rely on alternative solutions such as septic systems, constructed wetlands, or on-site treatment technologies (Massoud et al. 2009).

In addition to financial constraints, decentralized areas frequently lack the technical expertise needed to operate and maintain complex biological treatment systems. Improper maintenance or system failures can lead to groundwater contamination, public health risks, and environmental degradation. As a result, there is a growing need for innovative, low-cost, and low-maintenance treatment solutions tailored to these communities (Shahedi et al. 2020; Thomas et al. 2024). Technologies such as electrocoagulation, membrane filtration, and modular bioelectrochemical systems offer promising alternatives that require minimal operator intervention while still achieving high treatment efficiencies (Shahedi et al. 2020). Further advancements in decentralized treatment strategies could significantly improve wastewater management in remote areas, enhancing both environmental protection and resource recovery while reducing infrastructure dependency.

***Electrocoagulation***

Electrocoagulation (EC) is an efficient and versatile method for treating highly concentrated wastewater, making it particularly suitable for small-scale and decentralized communities (Aburto Vazquez 2023). Although the technology has been in use since the nineteenth century, its adoption has increased in recent years due to its effectiveness in treating industrial effluents from sectors such as mining, pulp and paper, and metal processing (Mollah et al. 2001; Shahedi et al. 2020). Unlike conventional coagulation, which relies on the addition of chemical coagulants, electrocoagulation generates coagulants in situ through the electrolytic dissolution of a sacrificial anode, typically made of iron or aluminum (Hakizimana et al. 2017).

The electrocoagulation process consists of three main stages. First, metal ions are released into the wastewater through anodic oxidation, forming coagulants that facilitate pollutant removal. Second, contaminants, including suspended solids, heavy metals, oils, and emulsions, are destabilized due to charge neutralization and electrostatic interactions. Finally, these destabilized particles aggregate to form flocs, which can then be separated from the treated water via sedimentation, flotation, or filtration (Hakizimana et al. 2017; Tahreen, Jami, and Ali 2020).

One of the key advantages of electrocoagulation is its ability to efficiently remove a wide range of pollutants without the need for chemical additives, reducing sludge production and secondary pollution. Additionally, it is relatively simple to operate, making it a promising solution for decentralized wastewater treatment where access to skilled operators and chemical reagents may be limited (Hakizimana et al. 2017). Continued research into optimizing electrode materials, energy consumption, and operational parameters can further enhance the viability of electrocoagulation for wastewater treatment in both industrial and decentralized applications.

## Working Principle

Electrocoagulation (EC) is governed by Faraday's first law of electrolysis, which dictates the release of metal ions from the anode and the reduction of water into hydrogen gas and hydroxide ions at the cathode (Hakizimana et al. 2017). This initiates a series of interrelated treatment mechanisms, with electrocoagulation, electro-flotation, and electrooxidation being the most significant contributors to pollutant removal (Tahreen, Jami, and Ali 2020).

At the anode, metal oxidation occurs, releasing metal cations into the solution, as represented by Equation 1:

$$M \rightarrow M^{Z+} + Ze^{-} \qquad 1$$

where *M* represents the metal released from the anode, and *Z* is the number of electrons transferred in the anodic dissolution process per mole of metal. Additionally, water oxidation may occur, producing oxygen gas and hydronium ions, as described in Equation 2:

$$2\,H_2O \rightarrow O_2 + 4H^{+} + 4e^{-}\ \left(E° = \frac{1.23V}{ENH}\right) \qquad 2$$

In the presence of chloride ions, oxidation can lead to chlorine gas formation, which has a strong oxidation potential, as shown in Equation 3:

$$2Cl^- \rightarrow Cl_2 + 2e^- \left(E° = \frac{1.36V}{ENH}\right) \tag{3}$$

At the cathode, water is reduced, generating hydrogen gas and hydroxide ions, as shown in Equation 4:

$$3H_2O + 3e^- \rightarrow \frac{3}{2}H_2 + 3OH^- \left(E° = \frac{0.00V}{ENH}\right) \tag{4}$$

Iron Electrocoagulation Mechanisms

For an iron anode, the dissolution mechanism is somewhat complex. While $Fe^{3+}$ formation is possible, $Fe^{2+}$ is the predominant species released due to the negligible dissolution rate of $Fe^{3+}$. The oxidation of $Fe^{2+}$ into $Fe^{3+}$ is influenced by pH and dissolved oxygen concentration. Under acidic conditions, $Fe^{2+}$ is slowly oxidized to $Fe^{3+}$, as shown in Equation 5. In alkaline conditions, $Fe^{2+}$ rapidly forms ferrous hydroxide precipitates via Equation 6 (Hakizimana et al. 2017). Further oxidation and hydrolysis reactions result in the formation of ferric hydroxide, $Fe(OH)_3$, as shown in Equation 7.

$$Fe^{2+} + O_2 + H_2O \rightarrow Fe^{3+} + 4OH^- \tag{5}$$

$$Fe^{2+} + 2OH^- \rightarrow Fe(OH)_2 \tag{6}$$

$$Fe^{2+} \xrightarrow{Oxidation\ and\ Hydrolysis} Fe(OH)_3 \tag{7}$$

Pollutant Removal Mechanisms

While these electrochemical reactions occur near the electrodes, they facilitate the destabilization of suspended particles, emulsions, and dissolved contaminants, allowing them to aggregate into flocs. These iron hydroxide complexes interact with pollutants through multiple mechanisms, including (Hakizimana et al. 2017), charge neutralization – reducing electrostatic repulsion between colloidal particles, precipitation and co-precipitation – capturing dissolved species within floc matrices, and adsorption and complexation – binding contaminants to metal hydroxide surfaces.

The generated flocs are removed from the treated water via electro-flotation (buoyancy-driven separation by $H_2$ gas) or sedimentation (gravity-driven settling). These combined mechanisms enable the efficient removal of various pollutants, including heavy metals, suspended solids, oils, dyes, and organic matter (Butler et al. 2011).

## Factors Influencing Electrocoagulation Performance

The efficiency of electrocoagulation is dependent on several operational and environmental factors, including (Butler et al. 2011; Hakizimana et al. 2017): wastewater composition and conductivity – affecting current efficiency and reaction kinetics, pH and alkalinity – influencing metal hydroxide formation and coagulation dynamics, temperature – affecting reaction rates and solubility of metal hydroxides, current density and voltage – determining the rate of metal ion release and gas generation, and electrode properties – including material type (e.g., iron, aluminum), spacing, orientation, and surface area.

Optimizing these parameters is crucial for enhancing electrocoagulation performance, minimizing energy consumption, and achieving sustainable wastewater treatment.

## Inline Ozonation

Numerous studies have demonstrated that incorporating inline ozonation during the electrocoagulation (EC) process can significantly enhance treatment efficiency while reducing the overall energy demand (Ahangarnokolaei, Ayati, and Ganjidoust 2021; Akbari, Ghanbari, and Moradi 2016; Bernal-Martínez et al. 2013). The synergy between ozonation and electrocoagulation arises from the enhanced generation of reactive oxidative species, particularly ozone ($O_3$), hydroxyl radicals (•OH), and other highly reactive intermediates, which promote contaminant degradation and increase removal efficiencies (Ahangarnokolaei et al. 2021). Furthermore, oxygen ($O_2$) formation during ozonation enhances the precipitation of $Fe(OH)_3$, improving flocculation and facilitating the removal of destabilized contaminants (Asaithambi et al. 2012).

Mechanisms of Hydroxyl Radical Formation

The integration of ozonation with iron-based electrocoagulation provides multiple pathways for the generation of hydroxyl radicals, •OH, which are among the most powerful oxidants in advanced oxidation processes. One major pathway is the decomposition of ozone, which follows a chain reaction mechanism involving the formation of superoxide radicals, $O_2^{\bullet-}$, hydroperoxyl radicals, $HO_2^{\bullet}$, and hydrogen peroxide, $H_2O_2$, as illustrated in Equations 8–15 (Asaithambi et al. 2012; Gardoni, Vailati, and Canziani 2012).

$$O_3 + OH^- \rightarrow O_2 + OH_2^- \qquad 8$$

$$O_3 + OH_2^- \rightarrow O_3^{\bullet-} + HO_2^{\bullet} \qquad 9$$

$$HO_2^{\bullet} \leftrightarrow O_2^{\bullet-} + H^+ \qquad 10$$

$$2HO_2^{\bullet} \rightarrow O_2 + H_2O_2 \qquad 11$$

$$O_2^{\bullet-} + O_3 \rightarrow O_3^{\bullet-} + O_2 \qquad 12$$

$$O_3^{\bullet-} + H^+ \leftrightarrow HO_3^{\bullet} \qquad 13$$

$$HO_3^{\bullet} \rightarrow OH^{\bullet} + O_2 \qquad 14$$

$$O_3^{\bullet-} + OH^{\bullet} \rightarrow O_3 + OH^- \qquad 15$$

One of the key byproducts of ozone decomposition is hydrogen peroxide, $H_2O_2$, which, in the presence of iron ions, undergoes the Fenton reaction, generating additional hydroxyl radicals and further enhancing contaminant degradation (Villaseñor-Basulto et al. 2022). Additionally, ozone can directly react with $Fe^{2+}$ to form intermediate species that promote hydroxyl radical formation, as shown in Equations 16–18 (Asaithambi et al. 2012).

$$O_3 + Fe^{2+} \rightarrow FeO^{2+} + O_2 \qquad 16$$

$$FeO^{2+} + H_2O \rightarrow Fe^{3+} + HO^{\bullet} + HO^{-} \qquad 17$$

$$FeO^{2+} + Fe^{2+} + 2H^{+} \rightarrow 2Fe^{3+} + H_2O \qquad 18$$

The in situ formation of $Fe(OH)_3$ flocs further aids in the removal of degraded organic compounds, heavy metals, and suspended solids through coagulation, adsorption, and precipitation mechanisms.

## Combined Treatment Efficiency and Limitations

The combined electrocoagulation–ozonation process has proven effective in removing a wide range of contaminants, including (Butler et al. 2011): chemical oxygen demand (COD) – due to enhanced oxidation of organic matter, total solids (TS) and total suspended solids (TSS) – through improved flocculation, and heavy metals and dyes – via oxidation, adsorption, and precipitation.

However, the efficiency of this hybrid process in treating nutrient contaminants such as total phosphorus (TP) and nitrogen species (TN, $NH_3$-N, $NO_3^{-}$, $NO_2^{-}$) remains variable. Some studies report high phosphorus and nitrogen removal efficiencies, while others report poor removal, especially of ammonia, indicating that treatment performance depends on specific wastewater characteristics and operating conditions (Aburto Vazquez 2023; Hakizimana et al. 2017; Smoczynskia et al. 2017). For high-strength municipal wastewater, electrocoagulation alone is often insufficient to meet total nitrogen discharge limits, primarily due to its limited ability to remove ammonia ($NH_3$) (Aburto Vazquez 2023). As a result, EC is frequently paired with secondary treatment processes such as membrane filtration, biological treatment, and additional advanced oxidation processes (Das, Sharma, and Purkait 2022).

### ***Electrodialysis***

Electrodialysis (ED) is a well-established membrane-based separation technology widely used in water treatment. It has been extensively applied for desalination of brackish water and brine in regions facing water scarcity, offering an efficient method to reduce salinity while maintaining water quality (Gurreri et al. 2020). However, beyond desalination, electrodialysis has gained significant attention in various industries—including biochemistry, food processing, wastewater treatment, and pharmaceuticals—due to its ability to selectively remove, concentrate, and recover valuable ions while simultaneously eliminating toxic compounds (Gurreri et al. 2020).

Over the last two decades, the application of electrodialysis in wastewater treatment has grown exponentially (Gurreri et al. 2020). This is primarily driven by the increasing demand for resource recovery technologies, as wastewater contains valuable nutrients and chemicals that can be reused in agricultural and industrial processes. In municipal wastewater treatment, electrodialysis has been explored for the recovery of key nutrients such as ammonia ($NH_4^+$), phosphate ($PO_4^{3-}$), nitrate ($NO_3^-$), and potassium ($K^+$), which are essential components in fertilizer production (Gurreri et al. 2020) (Ward et al. 2018). The application of electrodialysis for resource recovery differs significantly from traditional desalination, necessitating further research to optimize process parameters, membrane selectivity, and operating conditions (Mohammadi, Tang, and Sillanpää 2021). Unlike desalination, which primarily focuses on removing dissolved salts, resource recovery requires selective extraction of valuable ions such as ammonia, phosphate, and potassium while minimizing unwanted ion transport and energy consumption. The complexity of wastewater composition further complicates the process, as variations in pH, organic matter, and ionic strength can influence membrane performance and fouling potential.

One of the primary challenges associated with electrodialysis in resource recovery is its energy demand. The process is often energy-intensive, with consumption rates dependent on factors such as feedwater conductivity, ion concentration, membrane resistance, and system configuration. While electrodialysis can be optimized to improve efficiency, the overall economic feasibility remains uncertain, particularly for large-scale municipal applications (Gurreri et al. 2020). Further cost-benefit analyses are required to assess whether electrodialysis can compete with other nutrient recovery technologies in terms of capital investment, operational expenses, and long-term sustainability. Additionally, the heterogeneous nature of municipal wastewater poses challenges in standardizing electrodialysis systems for broad implementation. Fluctuations in wastewater composition—including seasonal variations in organic load, nutrient concentrations, and salinity—make it difficult to predict performance, efficiency, and membrane lifespan. A 2021 review study on municipal electrodialysis applications emphasized the need for further pilot-scale studies and techno-economic assessments to determine its long-term viability and scalability for wastewater treatment and resource recovery (Mohammadi et al. 2021).

## Working Principal

Electrodialysis is an electrochemical membrane separation technology that leverages ion-selective membranes and an applied electric field to facilitate the separation and concentration of ionic species in a solution. The system consists of alternating cation exchange membranes and anion exchange membranes, which are arranged between an anode and a cathode. These membranes divide the system into alternating dilute and concentrate channels, creating a structured pathway for ion migration (Gurreri et al. 2020). When an external direct current (DC) voltage is applied across the electrodes, an electric potential gradient is generated, driving cations toward the negatively charged cathode and anions toward the positively charged anode. The ion exchange membranes selectively permit the passage of specific ions while blocking the movement of counter-ions, leading to an increase in ion concentration in the concentrate channels and depletion of ions in the dilute channels (Gurreri et al. 2020). Each unit of this repeating structure—composed of a dilute channel, a concentrate channel, a cation exchange membrane, and an anion exchange membrane—is referred to as a cell pair. An electrodialysis stack consists of multiple cell pairs, ranging from just a few in small bench-scale laboratory systems to hundreds or even thousands in large pilot-scale or industrial-scale stacks (Gurreri et al. 2020). The most common configuration used in electrodialysis systems is the plate-and-frame design, which allows for scalability, modularity, and ease of maintenance (Gurreri et al. 2020).

## Ammonia Purification

Due to the high global demand for ammonia, various methods for its synthesis, concentration, and purification have been developed. Ammonia is a critical component of the world's food supply chain, primarily used in fertilizer production. However, the conventional Haber-Bosch process for ammonia synthesis is both energy-intensive and environmentally costly, accounting for approximately 2% of global energy consumption (Anon n.d.-a). This underscores the importance of circular ammonia management, as large amounts of ammonia are lost in wastewater and subsequently converted back into nitrogen gas during biological treatment—representing a missed opportunity for recovery and reuse (Anon 2020; Zhang and Liu 2021). To improve ammonia circularity, several recovery technologies have been explored, though their economic feasibility depends on ammonia concentration (Zhang and Liu 2021).

Electrodialysis has emerged as a promising technology for ammonia recovery due to its ability to selectively concentrate ammonium ions, making it particularly useful for wastewater streams with high ammonia content. However, for dilute ammonia feeds, electrodialysis often requires high energy inputs, limiting its viability. To further purify and concentrate the recovered ammonia, post-treatment methods such as struvite precipitation, gas stripping with absorption, and thermal separation techniques are commonly employed (Meng et al. 2024; Robles et al. 2020b).

One widely used method for ammonia recovery is struvite precipitation, which facilitates the formation of magnesium ammonium phosphate ($MgNH_4PO_4 \cdot 6H_2O$) from $NH_4^+$-N and $PO_4^{3-}$-P. This method is particularly beneficial for simultaneously recovering phosphorus and nitrogen. However, its efficiency is highly dependent on achieving an equimolar ratio of ammonium, phosphate, and magnesium—conditions that are often not met in wastewater streams (Robles et al. 2020a). Moreover, electrocoagulation, when used as a pre-treatment, has been found to effectively remove phosphate but performs poorly in ammonia removal, making struvite precipitation unsuitable for recovery in electrocoagulation-treated blackwater.

A more viable alternative for ammonia purification is gas stripping followed by absorption, particularly in cases where electrodialysis yields high concentrations of ammonia (Meng et al. 2024). In this process, ammonia is volatilized under alkaline conditions and subsequently absorbed using acidic solutions, such as sulfuric acid, to produce ammonium sulfate $(NH_4)_2SO_4$, a widely used fertilizer (Chen et al. 2021; Kinidi et al. 2018). This method effectively separates ammonia from non-volatile nutrients like potassium and phosphate, making it a preferred technique for high-purity ammonia recovery.

While less commonly applied in wastewater treatment, thermal separation techniques such as distillation and direct condensation can be employed to recover pure ammonia gas. These methods involve heating the ammonia-rich stream to release gaseous ammonia, which is then cooled and condensed into a purified form. However, they are highly energy-intensive and often economically unfeasible compared to stripping and absorption or struvite precipitation. Nevertheless, given the unique composition of blackwater after electrochemical treatment, these methods may be worth investigating as potential solutions for high-purity ammonia recovery.

## SUMMARY OF KNOWLEDGE GAPS

While electrocoagulation and electrodialysis are well-established wastewater treatment technologies, their application to blackwater treatment remains insufficiently studied due to the high variability of blackwater composition and the lack of standardization in treatment methodologies. These technologies offer several advantages for decentralized wastewater treatment, including low capital costs, operational simplicity, and robustness, making them attractive alternatives to traditional biological treatment, which often requires long start-up periods and complex maintenance.

Beyond their potential for efficient blackwater treatment, these technologies enable ammonia recovery, presenting both economic and environmental benefits. However, there is a significant gap in research regarding their long-term viability, efficiency, and scalability for decentralized wastewater applications. Moreover, while ammonia recovery has been demonstrated, further studies are needed to explore methods for upgrading recovered ammonia to higher purity levels suitable for use in energy applications—such as diesel additives or hydrogen fuel cells. The feasibility of these conversion processes and their economic and technical viability remains poorly understood, representing a crucial area for future investigation

## OBJECTIVE AND HYPOTHESIS

The goal of this study is to determine the viability of a combined electrocoagulation and electrodialysis system for the decentralized treatment of high-strength wastewater while recovering ammonia. The hypothesis is that the technologies will synergistically treat the blackwater to meet discharge standards while maintaining and recovering the ammonia which can be directly utilized or purified into fuel grade ammonia. There are 3 objectives to this study:

1. Determine electrodialysis' viability as a post electrocoagulation treatment option.
2. Create a pilot scale electrocoagulation and electrodialysis system for blackwater treatment.
3. Perform life cycle and techno-economic assessment of the system compared to other treatment options.

# CHAPTER 2: ELECTROCOAGULATION AND ELECTRODIALYSIS

## LAB SCALE ELECTRODIALYSIS

### *Materials and Methods*

Electrodialysis (ED) experiments were conducted using the PCCell BED2-2 Compact system, integrated with the ED200 stack and PC Frontend software for data acquisition and control. The synthetic dilute solution was prepared by dissolving Sigma-Aldrich ammonium chloride and Morton table salt in deionized (DI) water produced using a Thermo Fisher Scientific Pacific TII 7 purification unit. Sodium chloride was added to simulate background salinity and more closely match the ionic strength and conductivity of latrine blackwater. The initial concentrate compartment was filled with DI water, while the electrode rinse solution consisted of 2.67 g/L Sigma-Aldrich sodium sulfate dissolved in DI water to maintain ionic conductivity and prevent electrode fouling.

The ED system operated under current regulation, with a maximum voltage of 22 V and a current limit of 0.6 A. Experimental runs were conducted across varying dilute-phase concentrations and flow rates. Specifically, dilute solutions containing 200 mg $NH_3$-N/L and 600 mg $NH_3$-N/L were tested at flow rates of 150 L/h and 75 L/h, respectively.

Ammonia concentrations in the dilute compartment were quantified using Hach TNTplus vial test kits, with absorbance readings taken via a Hach DR3900 spectrophotometer.

A batch-mode configuration was used for the dilute solution, while the concentrate solution was retained between runs, following the process flow diagram shown in Figure 1. Each run was terminated once the dilute stream conductivity dropped below 0.1 mS/cm, indicating near-complete ion removal. After each run, the treated dilute solution was discarded, and a fresh

8-liter batch was prepared. All solutions were stored in 5-gallon high-density polyethylene (HDPE) containers.

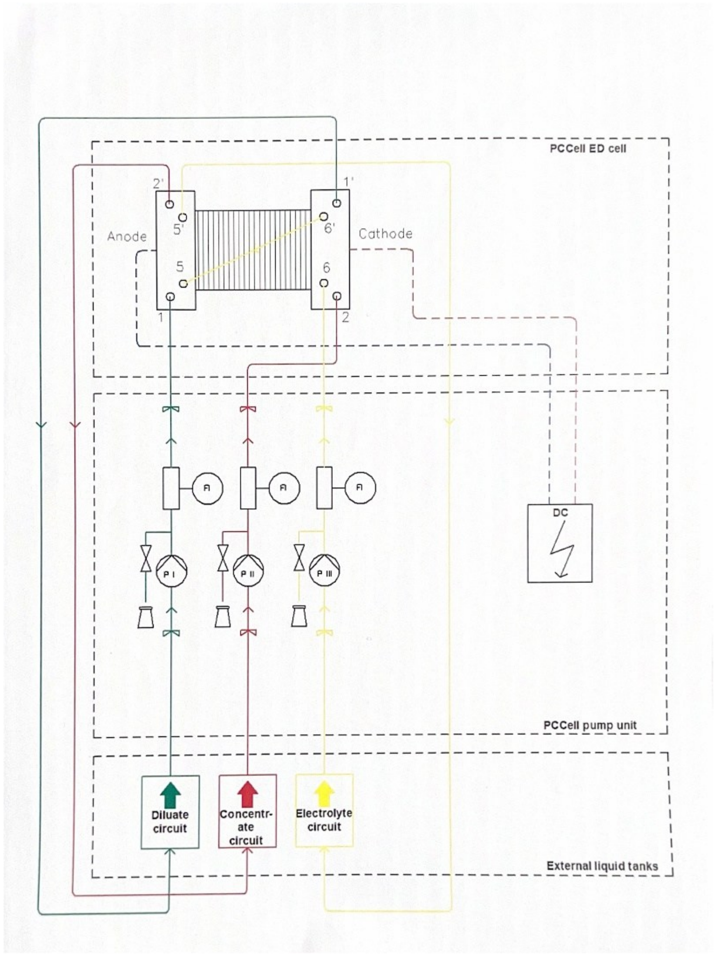


**Figure 1:** Bench Scale Electrodialysis Unit P&ID.

***Lab-Scale Testing with Synthetic Ammonium Salt Solutions***

The viability of electrodialysis as a post-treatment option following electrocoagulation was evaluated by testing its ability to remove ammonium from synthetic ammonium salt solutions at concentrations typical of blackwater. Two concentrations, 200 mg $NH_3$-N/L and 600 mg $NH_3$-N/L, were selected to reflect the lower and upper bounds commonly found in blackwater. The treatment goal was to reduce the dilute-phase ammonia concentration to below 10 mg N/L, in line with common discharge standards. More stringent discharge limits of 1 mg

N/L, adopted in some jurisdictions, were also considered to evaluate the system's potential to meet stricter regulatory thresholds. Concurrently, energy consumption was monitored to assess the feasibility of implementation in a continuous treatment process.

For the 200 mg $NH_3$-N/L dilute solution, a total of thirty batch runs were performed, with ammonia concentrations measured in both the dilute and concentrate streams before and after each run. The concentrate stream reached a final ammonia concentration of 4800 mg $NH_3$-N/L, showing a near-linear increase over successive runs (Figure 2).

Significant variation was observed in the final dilute-phase ammonia concentration due to changes in the conductivity cutoff criteria used to terminate each run. Initially, electrodialysis was stopped when the dilute conductivity reached 0.03 mS/cm, yielding a mean ammonia concentration of 1.92 ± 0.931 mg $NH_3$-N/L. Since this was above the stricter threshold of 1 mg N/L, the cutoff was lowered to 0.01 mS/cm for the next five runs, which resulted in a lower average ammonium concentration of 0.442 ± 0.154 mg $NH_3$-N/L, demonstrating the system's capability to meet ultra-low discharge standards.

As ammonia accumulated in the concentrate, the treatment time for each batch increased. To address this, the conductivity threshold was raised to 0.25 mS/cm for subsequent runs, resulting in an average ammonia concentration of 20.7 ± 0.424 mg $NH_3$-N/L, which was well above both target thresholds. The cutoff was then reduced to 0.1 mS/cm for the remaining 15 runs, yielding a mean final concentration of 8.75 ± 4.00 mg $NH_3$-N/L. These results demonstrate that electrodialysis can reliably reduce ammonium concentrations to below 10 mg N/L with moderate operation times. However, achieving ultra-low targets (<1 mg N/L) requires significantly longer runtimes and therefore would require a much larger unit for continuous treatment with the same energy input. A summary of concentration trends relative to conductivity cutoffs is provided in Figure 2.

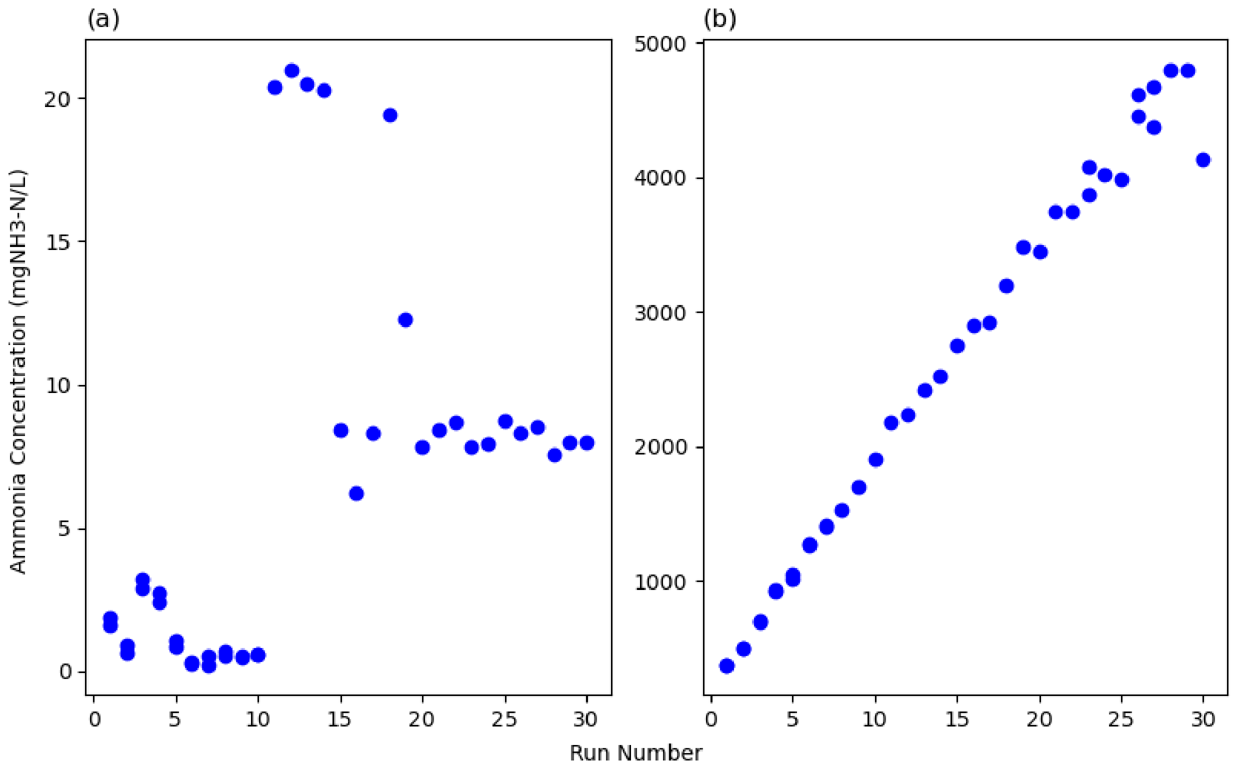


**Figure 2:** Low-concentration testing values after each run for a) Dilutes stream b) Concentrate stream.

For the high-concentration testing (600 mg $NH_3$-N/L), eight batch electrodialysis runs were conducted with a conductivity cutoff of 0.1 mS/cm to determine the treatment endpoint. The concentrate compartment reached a final ammonia concentration of 5059 mg $NH_3$-N/L, exhibiting an approximately linear increase in concentration with each successive run (Figure 3). Ammonia concentrations in both the dilute and concentrate streams were monitored throughout the experiments.

The final ammonia concentration in the treated dilute stream averaged 9.50 ± 1.62 mg $NH_3$-N/L. A slight upward trend in dilute-phase concentrations was observed over the course of the runs, possibly due to reduced ion removal efficiency as the concentrate solution became more saturated (Figure 3). This contrasts with the low-concentration trials, where treated dilute concentrations remained relatively stable.

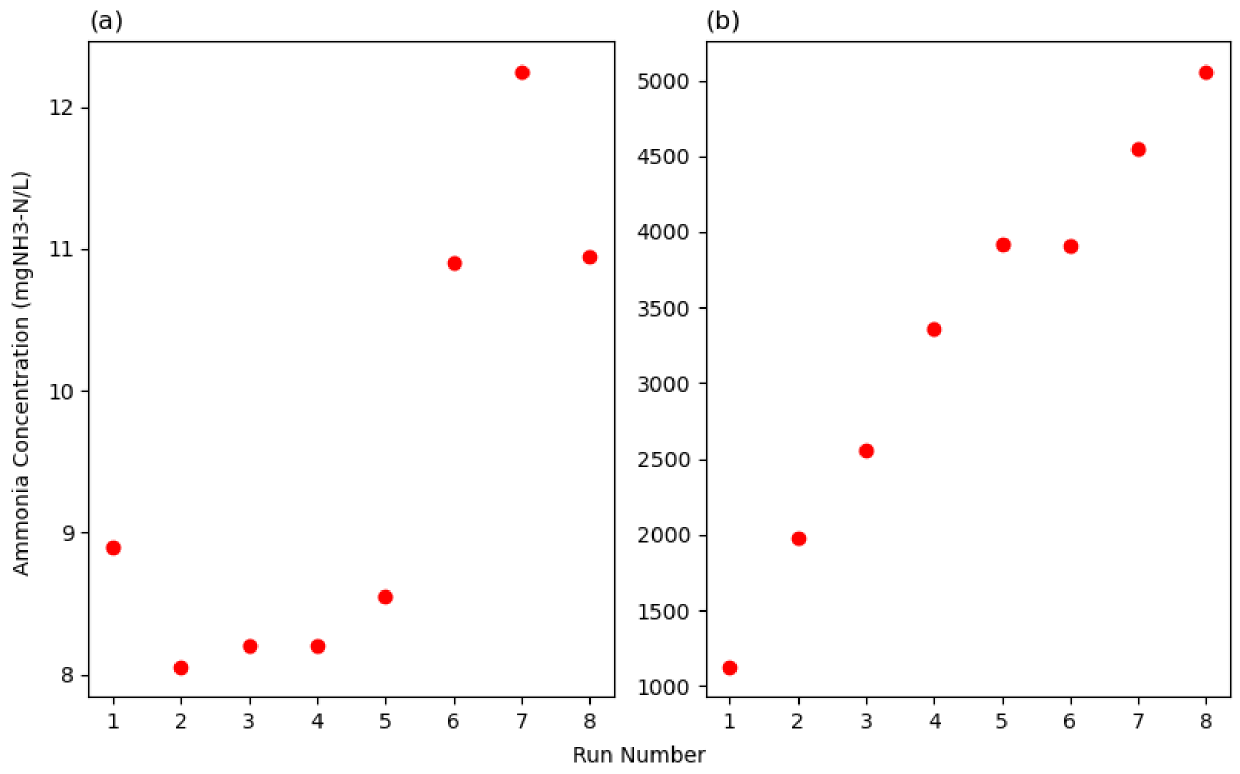


**Figure 3:** High-concentration testing values after each run for a) Dilutes stream b) Concentrate stream.

When comparing the low- and high-concentration experiments, both sets exhibited approximately linear increases in concentrate ammonia concentration up to ~5000 mg $NH_3$-N/L.

However, only the high-concentration tests displayed a noticeable increase in treated dilute concentrations across runs as seen in Figure 4.

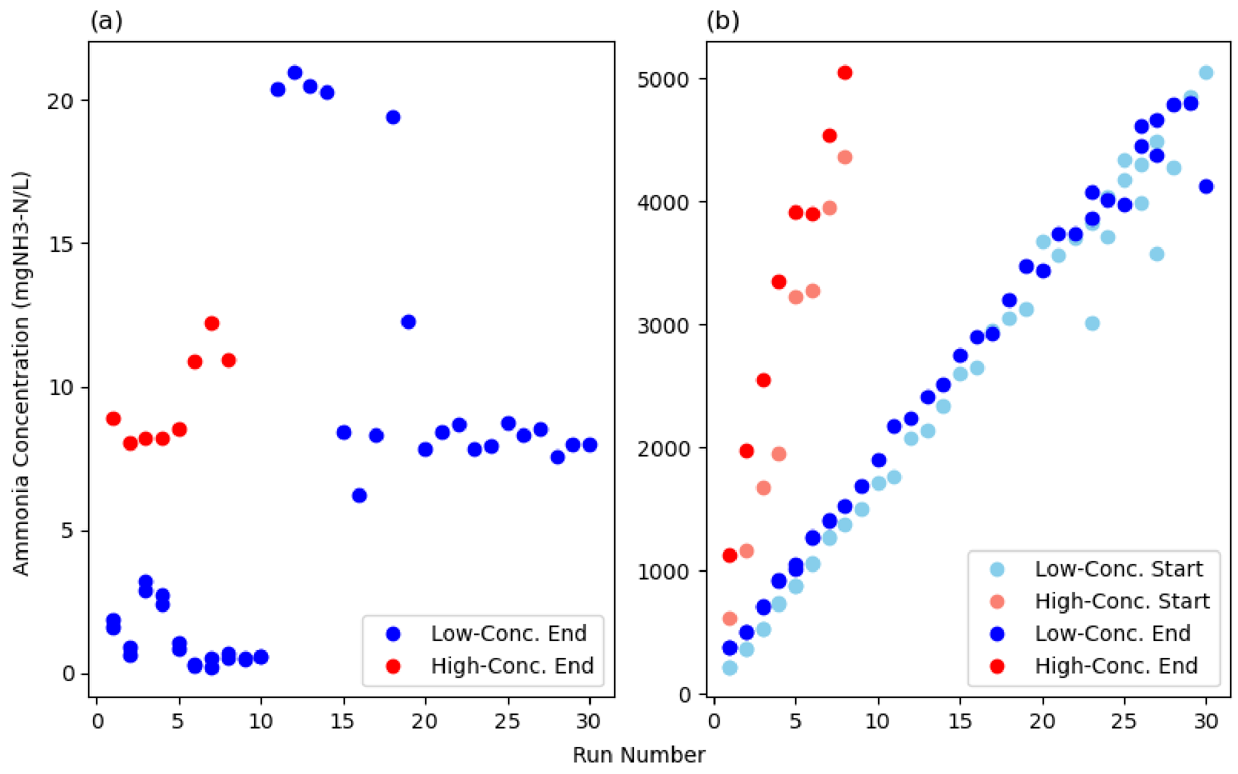


**Figure 4:** Low- and high-concentration testing values after each run for **a)** Dilutes stream **b)** Concentrate stream.

Energy consumption was monitored in real time by logging voltage and current data every 2 seconds. These readings were averaged to calculate the mean power input for each run. Total energy consumption per liter of dilute solution treated was determined by multiplying the average power by the runtime and dividing by the treated volume. The low-concentration runs consumed an average of 0.58 ± 0.12 Wh/L, while the high-concentration runs required slightly more energy at 0.75 ± 0.02 Wh/L. This increase was primarily due to longer treatment times needed to reach the 0.1 mS/cm cutoff under higher ammonium loading. The average runtime for high-concentration runs was 53.9 ± 0.4 minutes, which is 21.6 minutes longer than the low-concentration runs (32.3 ± 9.3 minutes). These results suggest that while electrodialysis remains viable at higher concentrations, increased residence time and energy input must be considered in system design.

Statistical analysis was conducted using the ttest_ind function from the SciPy library to compare the low-concentration and high-concentration experiments for two parameters: power consumption per liter of water treated and dilute end concentration. The null hypothesis, which assumed the means of the two experiments were equal, was tested for each parameter. The analysis revealed a statistically significant difference in power consumption between the two experiments, with a t-statistic of 5.31 and a p-value of 5.84e-6, indicating strong evidence against the null hypothesis. However, no significant difference was found for the dilute end concentration, as the t-statistic was -1.41 and the p-value was 0.16, leading to the failure to reject the null hypothesis. These tests demonstrated that electrodialysis is a viable post-treatment option for ammonia removal and concentration following electrocoagulation. However, the linear increase in ammonia concentration in the concentrate stream suggests that further enrichment is possible.

### ***Maximum Concentration***

To explore the upper limit of this concentration capability while still achieving effective treatment of the dilute stream, a maximum concentration test was conducted using the synthetic 600 mg $NH_3$-N/L solution. A conductivity cutoff of 0.1 mS/cm was applied unless the run exceeded 90 minutes, in which case the treatment was concluded due to time constraints. The 90-minute limit was first reached during the 11th run and was subsequently applied to all remaining runs. Ammonia concentration in the concentrate was observed to plateau at approximately 7500 mg $NH_3$-N/L, as shown in Figure 5.

During the first 10 runs, the average final ammonia concentration in the dilute stream was 8.91 ± 1.35 mg $NH_3$-N/L. However, after implementation of the 90-minute cutoff, ammonia began to accumulate in the dilute stream, gradually increasing to nearly 20 mg $NH_3$-N/L. This increase mirrored a concurrent rise in final dilute conductivity (Figure 5), suggesting diminished ion removal as concentration polarization occurred leading to higher membrane resistance.

Energy consumption also increased steadily over the first 10 runs but stabilized at 1.37 ± 0.03 Wh/L once the 90-minute limit was consistently reached (Figure 5). These results suggest that the rise in energy demand was primarily driven by longer treatment durations rather than by efficiency losses due to concentration polarization and increased membrane resistance. This has important implications for system design, as it indicates that energy efficiency may be preserved even at higher ammonia loadings, provided runtime constraints are managed appropriately.

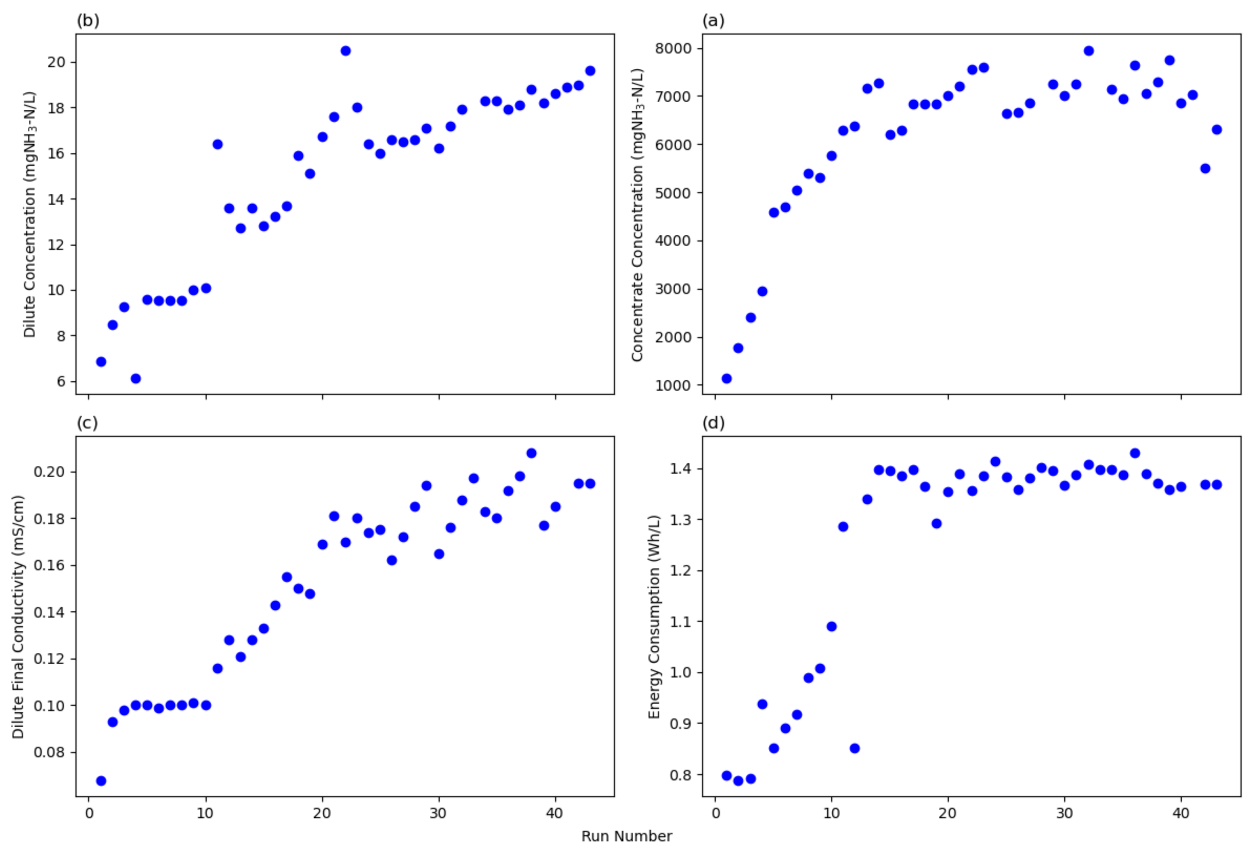


**Figure 5:** Maximum concentration testing values after each run for **a)** Dilutes stream **b)** Concentrate stream **c)** Dilute conductivity **d)** Energy consumption per liter of water treated.

### *Electrodialysis Reversal*

To further investigate operational robustness and fouling mitigation in ED systems, electrodialysis reversal (EDR) was tested as an alternative configuration. Since membrane fouling is a common concern in electrodialysis systems, electrodialysis reversal was tested to evaluate whether periodic polarity switching could improve treatment performance when processing blackwater. Due to mechanical limitations, including increased head loss caused by additional EDR tubing and valves, the system flow rate was reduced to 75 L/hr for these tests.

An initial baseline electrodialysis test without reversal was conducted and compared to an EDR trial under otherwise identical conditions. Reversal was manually implemented at the 45-

minute mark of the 90-minute test, and each run was terminated either when the dilute stream conductivity reached 0.1 mS/cm or upon reaching the 90-minute limit.

Both ED and EDR were effective at concentrating ammonia in the concentrate stream, achieving final concentrations greater than 7000 mg $NH_3$-N/L. ED achieved a maximum concentrate concentration of 8310 mg $NH_3$-N/L, whereas EDR reached 7094 mg $NH_3$-N/L (Figure 6). In the dilute stream, ED yielded a mean final ammonia concentration of 19.5 ± 8.48 mg $NH_3$-N/L, compared to 24.1 ± 19.3 mg $NH_3$-N/L for EDR.

To assess statistical significance, a two-sample t-test was performed on the dilute-phase concentrations. The null hypothesis, that there was no difference in the mean value of ED and EDR, was not rejected, with a t-statistic of -0.848 and a p-value of 0.403. Similarly, energy consumption was analyzed, showing average values of 1.22 ± 0.14 Wh/L for ED and 1.14 ± 0.16 Wh/L for EDR. Again, no statistically significant difference was observed (t-statistic = 1.38, p-value = 0.177).

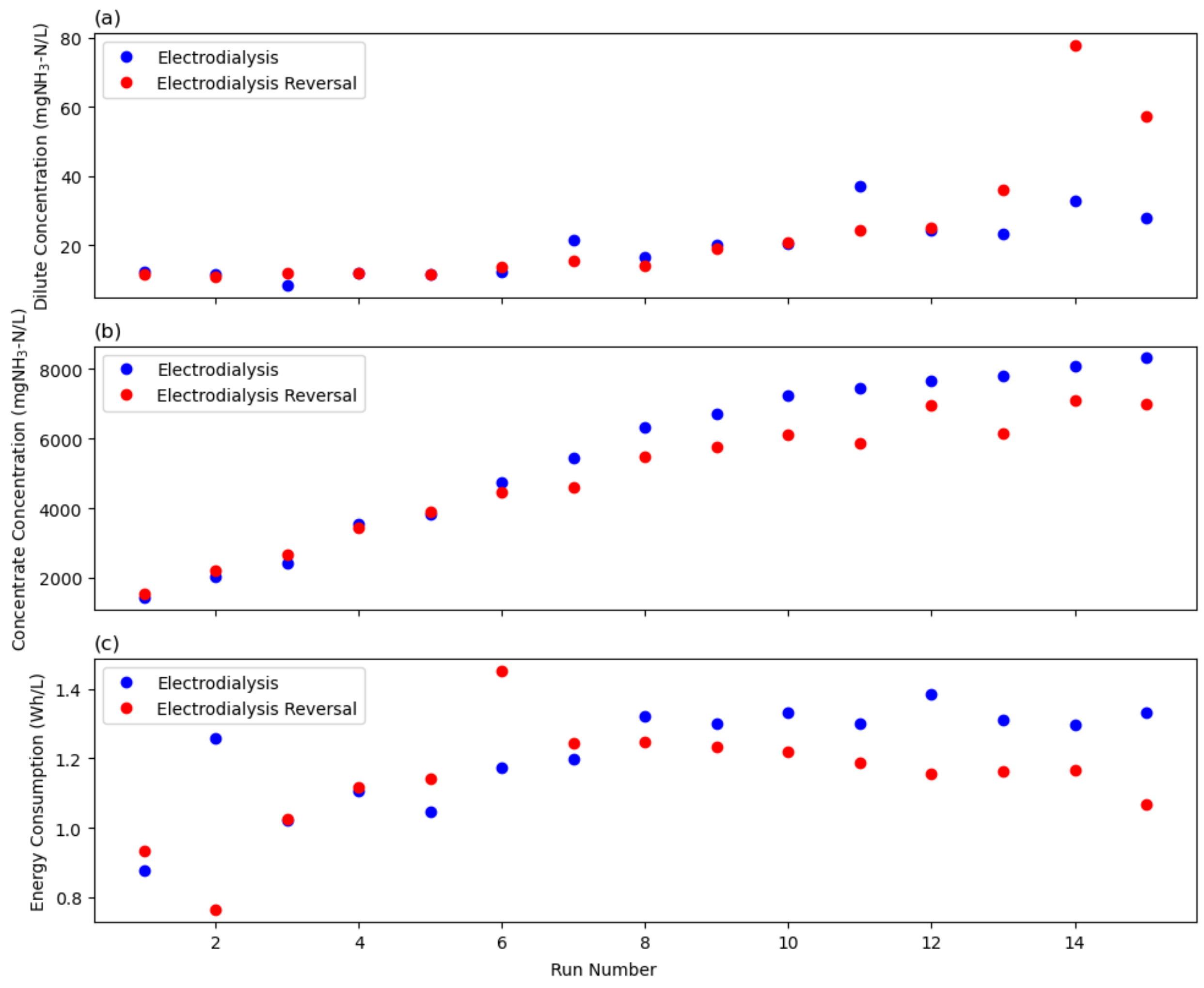


**Figure 6:** Comparison of ED and EDR performance for **a)** Dilute concentration after each run **b)** Concentrate ammonia concentration after each run **c)** Energy consumption per L of water treated for each run.

While EDR successfully maintained treatment performance, the lack of statistically significant improvement in energy efficiency or treated water quality, combined with the added mechanical and electrical complexity, makes it a less practical option for scale-up. The reversal process introduces additional valves, control algorithms, and plumbing requirements that increase both capital and maintenance costs. Given these tradeoffs and the absence of measurable performance enhancement, EDR was deemed impractical for this application and will not be considered for scaling to a pilot system.

***Sensitivity Analysis***

To support pilot-scale system design, a basic sensitivity analysis was conducted to identify which operational parameters most significantly influenced energy consumption during electrodialysis. Four parameters were tested: maximum current, maximum voltage, dilute flow rate, and concentrate flow rate. Each was independently halved and doubled from baseline to evaluate its effect on energy demand, with all other conditions held constant. A dilute-phase conductivity cutoff of 0.1 mS/cm was used for all tests. The baseline values were set at 22 volts, 0.6 amps, and 150 L/h for both dilute and concentrate flow rates. The synthetic wastewater used had an influent ammonia concentration of 300 mg $NH_3$-N/L in the dilute stream and 700 mg $NH_3$-N/L in the concentrate stream. Both solutions were replaced before each run to maintain consistency.

Among the tested variables, maximum current had the largest impact on energy consumption. Halving the current reduced energy use by 0.13 Wh/L, while doubling it increased energy consumption by 0.23 Wh/L (Figure 7). The next most sensitive parameter was maximum voltage, with deviations of -0.15 and +0.11 Wh/L for 0.5× and 2× the baseline, respectively. Dilute flow rate adjustments resulted in energy changes from +0.18 to -0.01 Wh/L, and concentrate flow rate had the smallest effect, varying energy use from +0.05 to +0.01 Wh/L.

These results show that electrical parameters (voltage and current) exert a greater influence on energy efficiency than hydraulic parameters (flow rates). This is advantageous for scaling, as electrical adjustments can be implemented dynamically through programmable controllers, whereas modifying flow rates may require mechanical changes. Moreover, the lower sensitivity of energy consumption to flow rate changes indicates that the ED system can accommodate upstream variability from processes like electrocoagulation, which are more sensitive to hydraulic shifts. This flexibility supports ED's integration as a robust post-treatment step capable of automated control in decentralized or variable-load treatment systems.

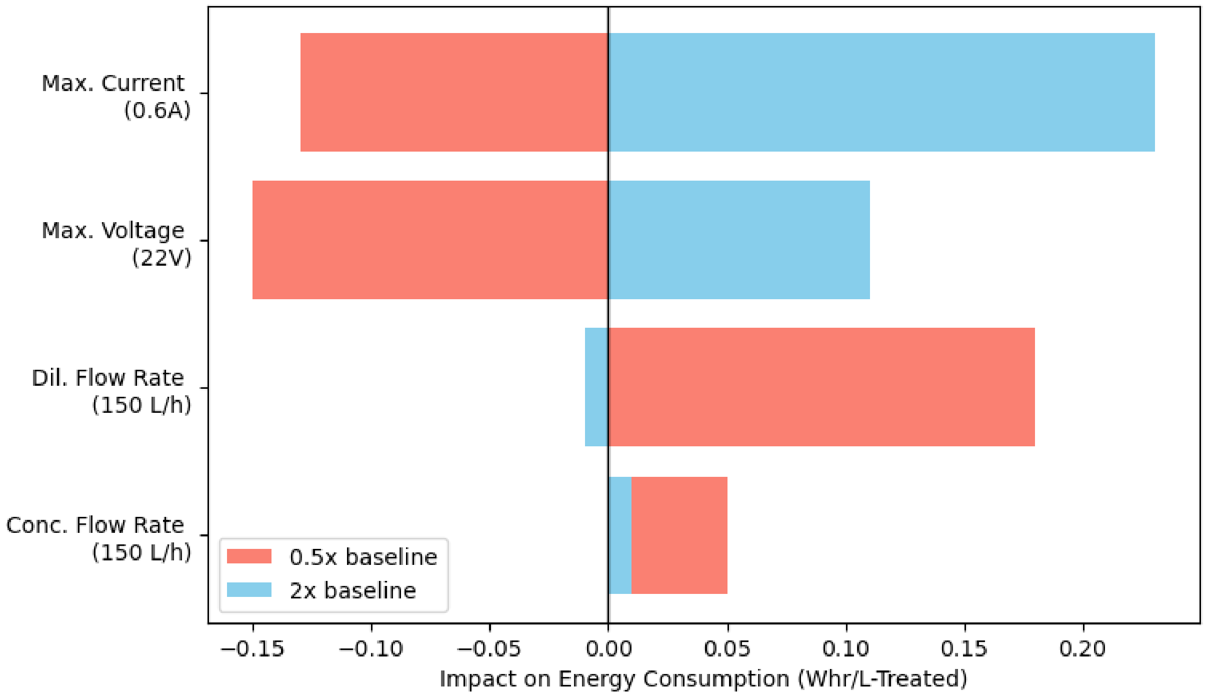


**Figure 7:** ED sensitivity analysis for scale up considerations.

***Sizing for Pilot Scale***

To determine the required membrane area for a continuous-flow pilot-scale ED system, a graphical mass transfer analysis was performed based on lab-scale performance data. The lab-scale unit had an effective membrane area of 2 m$^2$, and conductivity measurements were taken both before and after treatment across time to produce representative pre- and post-unit conductivity curves. These conductivity values served as a proxy for ion concentration and removal, given the direct correlation between conductivity and ammonium content in the synthetic solution.

These curves were used to perform a McCabe-Thiele-like stepping method, a technique commonly applied in distillation modeling, to approximate the effective membrane area required to achieve a target final conductivity. The procedure involved vertically stepping from the pre-treatment curve down to the post-treatment curve and then horizontally stepping back to the pre-treatment curve, continuing this pattern until the target dilute conductivity of 0.1 mS/cm was reached. This conductivity threshold was selected because it consistently resulted in final ammonia concentrations below 10 mg $NH_3$-N/L in previous testing.

Each vertical step represented the performance of one ED segment with the same removal efficiency as the lab-scale system. Therefore, multiplying the number of steps by the lab-scale membrane area yielded the required continuous membrane area. As shown in Figure 8, between 9 and 11 steps were needed depending on the applied maximum current, resulting in a total estimated membrane area of 18 to 22 $m^2$.

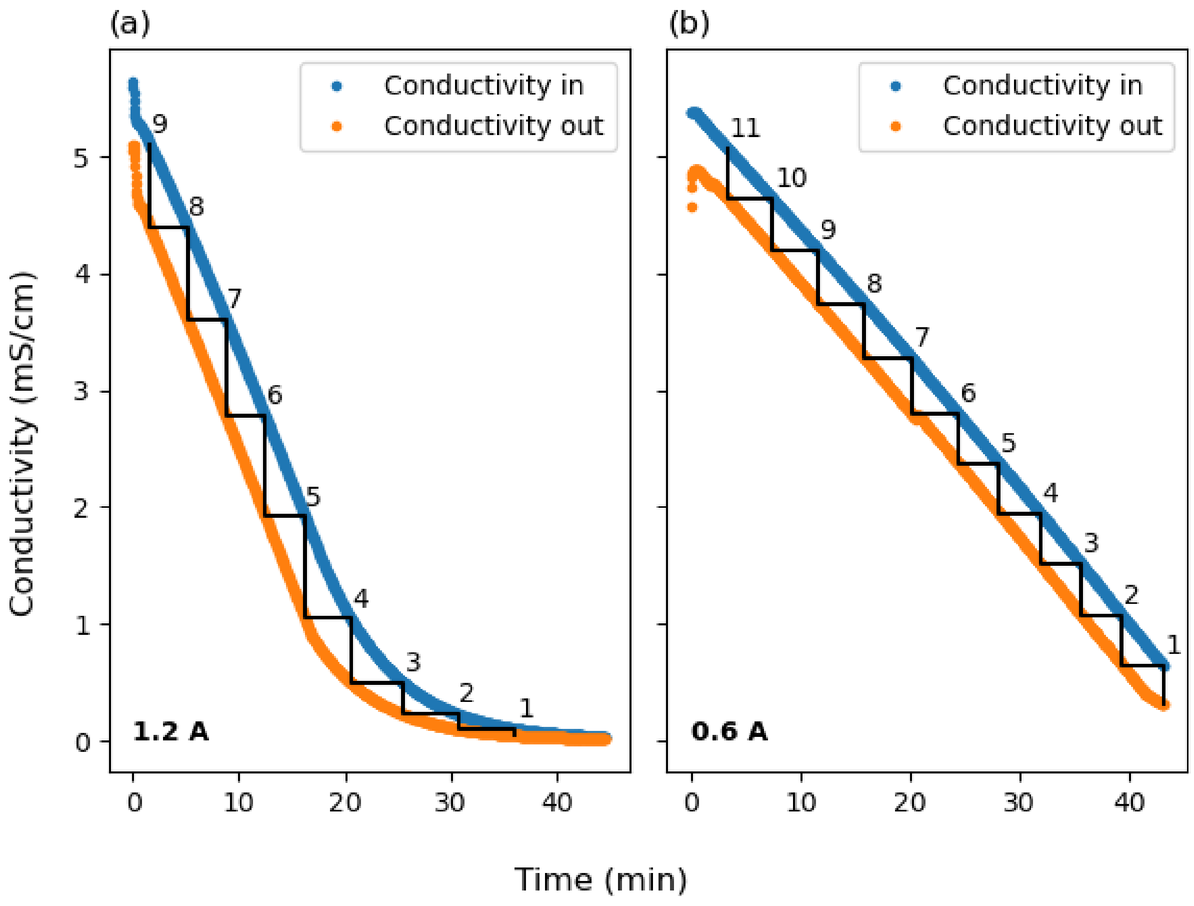


**Figure 8:** Sizing of the pilot scale ED stack using stepping with currents **a)** 0.6 amps **b)** 1.2 amps.

PILOT SCALE TESTING

***Materials and methods***

To prepare the sewage sludge, 9.46 liters (2.5 gallons) of primary settling sludge from the East Lansing Wastewater Treatment Plant was diluted with 198.7 liters (52.5 gallons) of tap water in a 208-liter (55-gallon) barrel, achieving a 1:20 dilution ratio to reach parameter concentrations viable for electrocoagulation. To increase ionic strength and more closely match the concentrations in latrine blackwater, 60 grams of NaCl and 60 grams of $NH_4Cl$ were added. The mixture was blended using a mechanical agitator (Dayton Electric, Model 32V142, Niles, IL) and kept in suspension during electrocoagulation treatment.

Water quality parameters including COD, TN, TP, and $NH_3$-N were measured using United States Environmental Protection Agency (USEPA)-approved HACH standard methods. Nutrient concentrations were determined colorimetrically using a HACH DR3900 spectrophotometer, turbidity was measured using a HACH 2100Q portable turbidimeter, and pH was recorded using a HACH IntelliCAL PHC201 pH probe. Total solids (TS) and total suspended solids (TSS) were quantified via gravimetric analysis. Sampling points included the initial blackwater, clarified effluent, filtered effluent, electrodialysis-treated effluent, and the electrodialysis concentrate stream.

The pilot treatment system was constructed to fit within a TRICON shipping container (2.44 m × 2.44 m × 1.96 m). The electrocoagulation reactor, fabricated from PVC, had a volume of 15 L and housed two iron plate electrodes measuring 44.45 cm × 20.32 cm × 0.64 cm, spaced 2.54 cm apart. Electrodes were connected to a titanium basket, and blackwater was pumped to the equalization tank using an electric pump (Edson Model 25200, New Bedford, MA).

The EC reactor was gravity-fed at 1.51 L/min from a 37.9-liter PVC equalization tank, which received blackwater pumped at 18.9 L/min from the mixing barrel. Voltage was controlled with current setpoints of 30 and 36 A (maximum 30 V), and polarity was reversed every 5 minutes to minimize sludge accumulation on electrode surfaces.

An aluminum sludge separator was connected to the EC reactor via PVC unions and featured three chambers separated by steel dividers. EC effluent and sludge entered the center chamber and distributed to adjacent compartments. A Teflon-coated baffle near the sludge inlet promoted flow and reduced clogging. Separated sludge exited via a flexible hose into a collection bucket, while clarified liquid overflowed through a PVC pipe.

Following sludge separation, clarified EC effluent flowed into an incline plate separator (SPC-5, M.W. Watermark, Holland, MI). The effluent was then stored in one of three intermediate storage tanks, which fed the electrodialysis unit (PCCell GmbH, ED1000H, Saarland, Germany). One tank supplied the dilute stream, another supplied the concentrate stream, and the third provided an electrolyte stream containing 600 mg/L $Na_2SO_4$, which was circulated through the electrode rinse compartments. Both the concentrate and electrolyte streams were continuously recycled, while the dilute stream was treated and discharged.

For the sewage sludge, electrodialysis testing was performed by collecting 8 liters of EC effluent post-clarification and 7 µm filtration, which was used as the dilute stream in the same batch-style configuration as the lab-scale ED tests.

### *Sewage sludge*

Sewage sludge testing was conducted over a six-month period to evaluate the long-term viability and operational robustness of the pilot-scale treatment system. Treatment performance was assessed at each major stage of the system for key parameters: chemical oxygen demand, total phosphorus, ammonia-nitrogen, total suspended solids, and pH, as summarized in Table 1. To confirm the accuracy of the sewage sludge formulation, total solids (TS) were measured at $2.06 \pm 0.64$ g/L, which aligns with reported ranges for latrine blackwater.

**Table 1:** Average COD, TP, $NH_3$-N, and pH values for each major stage of the combined system.

| | Blackwater | EC Treated | EC Filtered | ED Treated |
|---|---|---|---|---|
| COD (mg/L) | 2035.7 ± 830.30 | 310.6 ± 93.22 | 206.2 ± 54.22 | 75.3 ± 25.79 |
| TP (mg/L) | 43.3 ± 24.91 | 3.3 ± 2.95 | 1.5 ± 1.66 | 1.0 ± 0.73 |
| TN (mg/L) | 135.6 ± nan | nan ± nan | 20.6 ± nan | nan ± nan |
| $NH_3$-N (mg/L) | 134.8 ± 46.81 | 122.5 ± 18.88 | 95.0 ± 24.30 | 3.9 ± 1.26 |
| TSS (mg/L) | 884.1 ± 613.27 | 158.5 ± 211.88 | 27.0 ± 24.01 | 11.8 ± 15.25 |
| pH | 7.1 ± 0.23 | 7.5 ± 0.37 | 7.7 ± 0.61 | 7.0 ± 0.70 |

The system was able to achieve ammonia concentrations below 10 mg $NH_3$-N/L having a mean concentration post ED treatment of 3.9 ± 1.36 mg $NH_3$-N/L. Additionally, COD, TP, TSS, and pH were all also below discharge standards having values of 75.3 ± 25.79, 1.0 ± 0.73, and 11.8 ± 15.25 mg/L, respectively, as well as a pH of 7.0 ± 0.7. Most COD, TP, and TSS removal occurred during electrocoagulation, as shown in Figure 9. In contrast, ammonia was not significantly removed during electrocoagulation or filtration removing only 31.6%, but was effectively treated by electrodialysis, which achieved a 95.9% reduction of the remaining $NH_3$-N. Removal efficiencies for each stage of the treatment process can be seen in Table 2.

**Table 2:** Removal efficiencies of COD, TP, $NH_3$-N, and TSS between each stage of treatment as well as the overall removal efficiency.

| | COD (%) | TP (%) | $NH_3$ (%) | TSS (%) |
|---|---|---|---|---|
| Blackwater → EC Treated | 84.7 ± 41.0 | 92.3 ± 57.9 | 9.2 ± 37.4 | 82.1 ± 73.4 |
| EC Treated → EC Filtered | 33.6 ± 34.7 | 56.3 ± 101.2 | 22.4 ± 25.1 | 82.9 ± 134.6 |
| EC Filtered → ED Treated | 63.5 ± 29.1 | 34.4 ± 124.1 | 95.9 ± 25.6 | 56.4 ± 105.1 |
| Total | 96.3 ± 51.3 | 97.8 ± 93.1 | 97.1 ± 46.1 | 98.7 ± 144.8 |

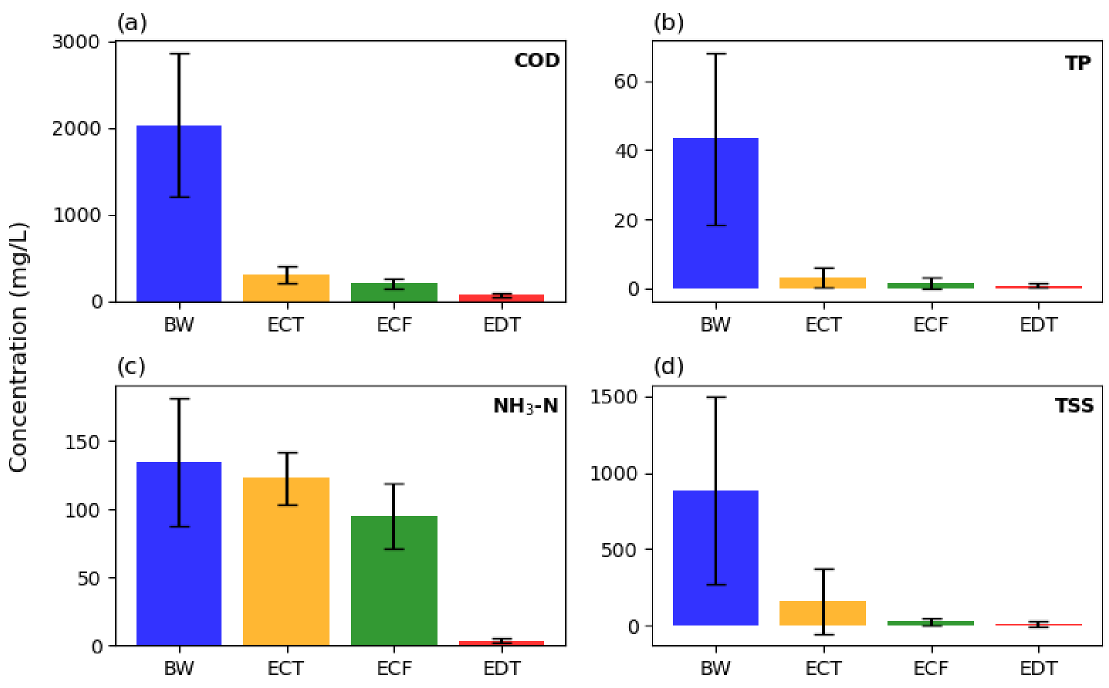


**Figure 9:** Treatment performance for key parameters at each stage of the treatment process **a)** COD **b)** TP **c)** $NH_3$-N **d)** TSS.

The treatment performance for the EC current setpoints of 30 and 36 amps were analyzed to investigate the change in treatment performance and energy demand. A comparison of the treatments can be seen in Figure 10 for COD, TP, $NH_3$-N, and pH. There appears to be a slight improvement in treatment performance at the current of 36 amps, but when a Welch's t-test was performed with a significance level of 0.05 there was no statistically significant difference in the mean values for every water and parameter except for blackwater pH which had a t-statistic of 0.653 and p-value of 0.017. For the energy consumption of the 30 and 36 amp testing, the mean values were 842 ± 166 and 762 ± 178 Wh/L treated respectively.

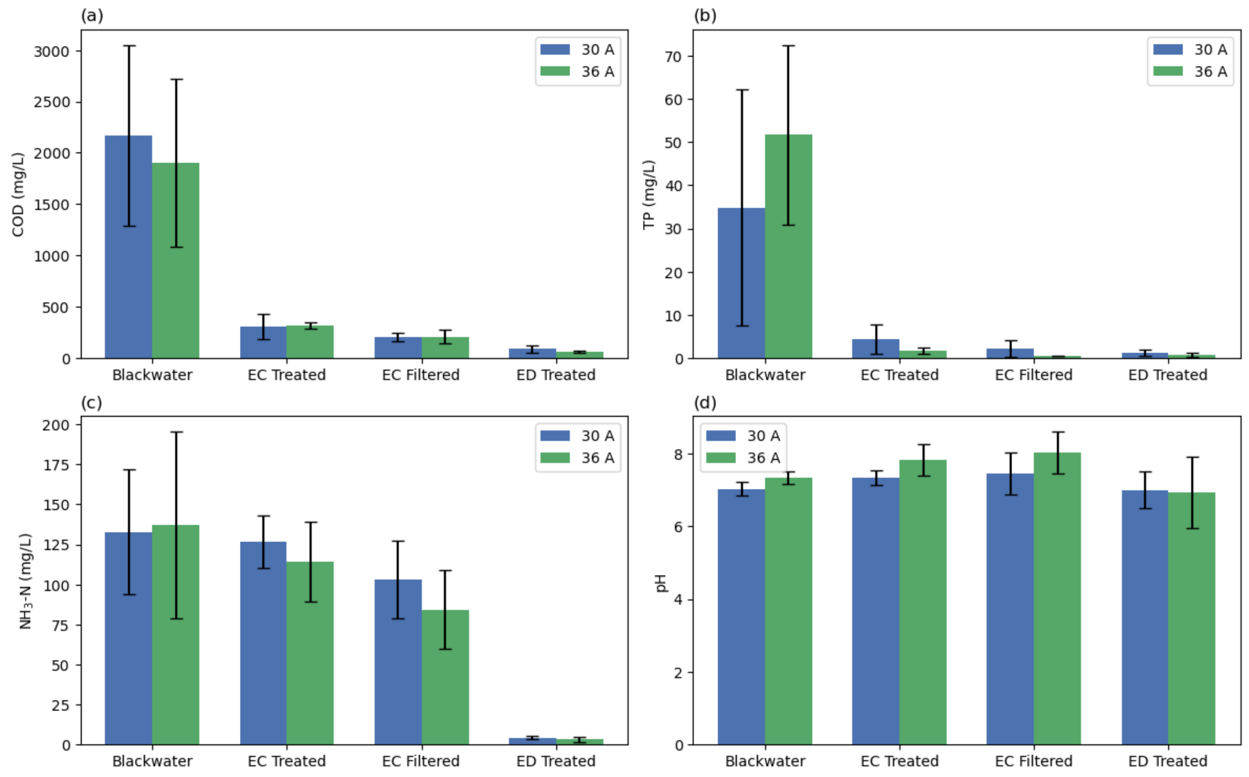


**Figure 10:** Comparison of treatment parameters for EC current setpoints of 30 and 36 amps. **a)** COD in mg/L **b)** TP in mg/L **c)** NH3-N in mg/L **d)** pH.

***Camp Shelby***

A field demonstration of the treatment system was conducted at Camp Shelby Joint Forces Training Center in Mississippi over a three-week period in June 2024, using blackwater collected from portable toilet units (port-o-johns) on site. The collected waste was diluted with tap water to a target COD concentration of approximately 2500 mg/L, representative of typical high-strength blackwater conditions. As part of the dilution process, the blackwater was passed through a grinding pump to homogenize solids and reduce particle size, improving downstream flow and minimizing the risk of clogging in the electrocoagulation system. This approach ensured representative loading while protecting mechanical components and optimizing treatment performance.

This wastewater was used to evaluate the treatment performance and operational robustness of the full-scale system under field conditions, following the same electrocoagulation protocol as previously established in East Lansing. The key deviation from the protocol was the integration of the continuous electrodialysis unit (PCCell ED1000H) directly inline with the system, enabling real-time treatment rather than batch-mode operation.

The ED unit was operated in current-controlled mode with a maximum current of 1.2 A and a voltage limit of 22 V. This configuration allowed for continuous separation of ammonia from the EC effluent, simulating a more practical deployment scenario for decentralized wastewater treatment.

Operational challenges encountered during the Camp Shelby demonstration impacted system performance and treatment efficiency. Damage to internal wiring during transport prevented the electrocoagulation unit from reversing polarity, a critical step for minimizing electrode fouling and maintaining consistent performance. Without polarity reversal, rapid electrode degradation and fouling occurred, reducing the overall effectiveness of the EC process.

In addition, the high pH of the port-o-john blackwater adversely affected ammonium removal by the electrodialysis unit. Under basic conditions, a significant portion of ammonium, $NH_4^+$, is converted to uncharged ammonia, $NH_3$, which cannot be transported across ion-exchange membranes by electrodialysis. This chemical shift diminished the effectiveness of ED treatment, as reflected in the lower removal efficiencies shown in Table 3 and the reduction in treatment performance illustrated in Figure 11.

To mitigate these issues, the system's flow rate setpoint was reduced to 0.76 L/min, increasing the hydraulic retention time and allowing more contact time between water and electrodes. Simultaneously, the electrocoagulation current was increased to 36 A, maximizing the electrical load to compensate for reduced system performance.

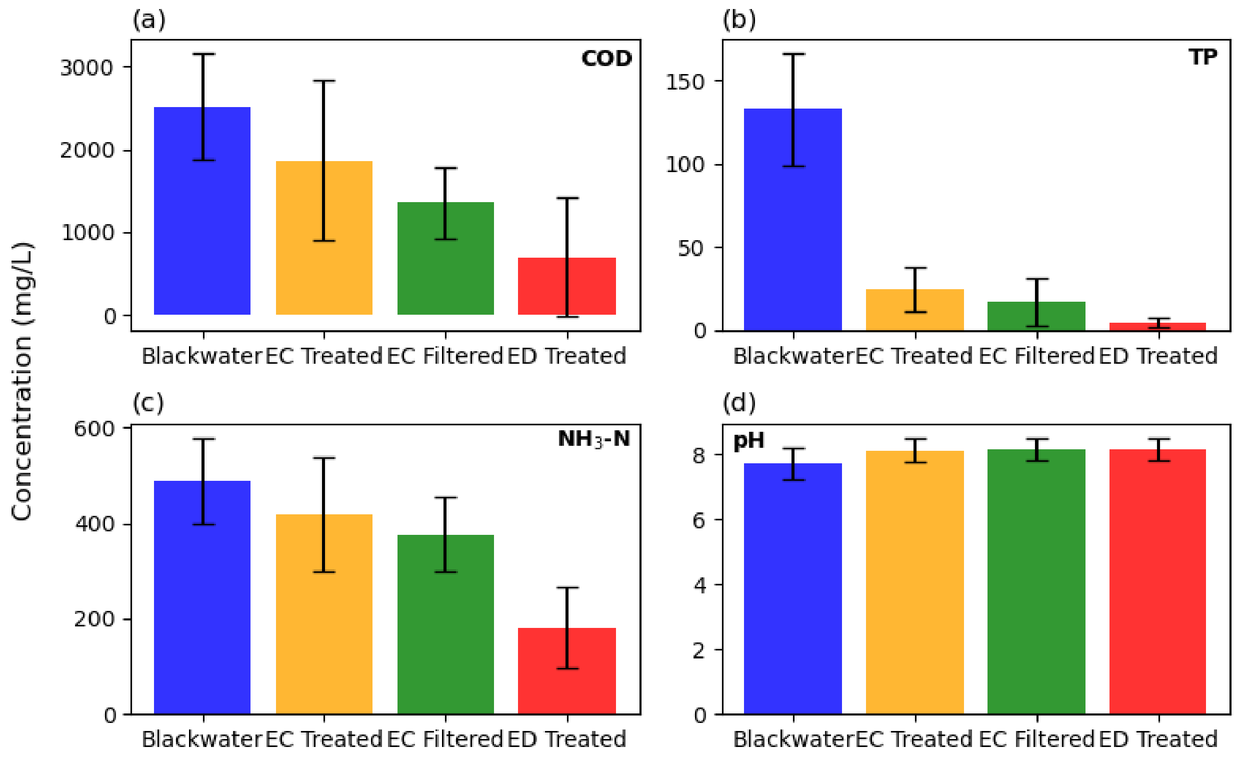


**Figure 11:** Treatment performance for key parameters at each stage of the treatment process for testing done in Camp Shelby **a)** COD **b)** TP **c)** $NH_3$-N **d)** pH.

Despite the operational limitations encountered during the Camp Shelby demonstration, the system achieved greater than 60% average removal for chemical oxygen demand, total phosphorus, and ammonia. In particular, TP removal reached 96.5%, highlighting the effectiveness of electrocoagulation in precipitating phosphorus, as well as electrodialysis in removing residual soluble contaminants not captured through coagulation or sedimentation.

Significant variation in treatment performance was observed over the course of the demonstration, primarily due to the polarity reversal issue and its gradual identification and resolution. As the underlying electrical fault was addressed and adjustments were made to flow rate and current, treatment efficiency improved.

To better illustrate the system's capacity under constrained conditions, a subset of the dataset, collected after the polarity issue was identified, can be used to reflect true system performance. This filtered dataset provides a clearer view of treatment potential even in the absence of full operational functionality and demonstrates the resilience of the combined EC–ED system under field conditions.

**Table 3:** Removal efficiencies of COD, TP, and $NH_3$-N between each stage of treatment as well as the overall removal efficiency for testing at Camp Shelby.

| | COD (%) | TP (%) | $NH_3$-N (%) |
|---|---|---|---|
| Blackwater → EC Treated | 26.0 ± 46.2 | 81.2 ± 27.4 | 14.5 ± 30.3 |
| EC Treated → EC Filtered | 27.2 ± 56.7 | 30.7 ± 78.7 | 9.9 ± 33.9 |
| EC Filtered → ED Treated | 49.1 ± 61.7 | 73.2 ± 85.0 | 52.1 ± 30.6 |
| Total | 72.6 ± 77.7 | 96.5 ± 70.4 | 63.1 ± 31.6 |

A focused analysis was conducted using a subset of data collected during the period prior to electrode damage when the electrocoagulation system was operating with the functional polarity. As shown in Figure 12, improved treatment performance was evident: chemical oxygen demand (COD) and total phosphorus (TP) concentrations were consistently lower throughout the treatment compared to the full dataset. A modest improvement was also observed immediately following EC treatment, indicating enhanced coagulation and settling performance during this operational window.

As with previous tests, electrocoagulation alone was not effective in removing ammonia-nitrogen ($NH_3$-N), with no significant reduction observed until electrodialysis treatment. However, ED performance continued to be limited by the high pH of the wastewater, which remained above acceptable discharge standards and inhibited the removal of uncharged ammonia.

Despite this limitation, ED was still effective at ion removal, as evidenced by substantial reductions in COD and TP concentrations following ED treatment. This suggests that the system maintained partial functionality for non-ammonium ions, even under suboptimal pH conditions, and highlights the importance of pH control for maximizing ammonium removal in field deployments.

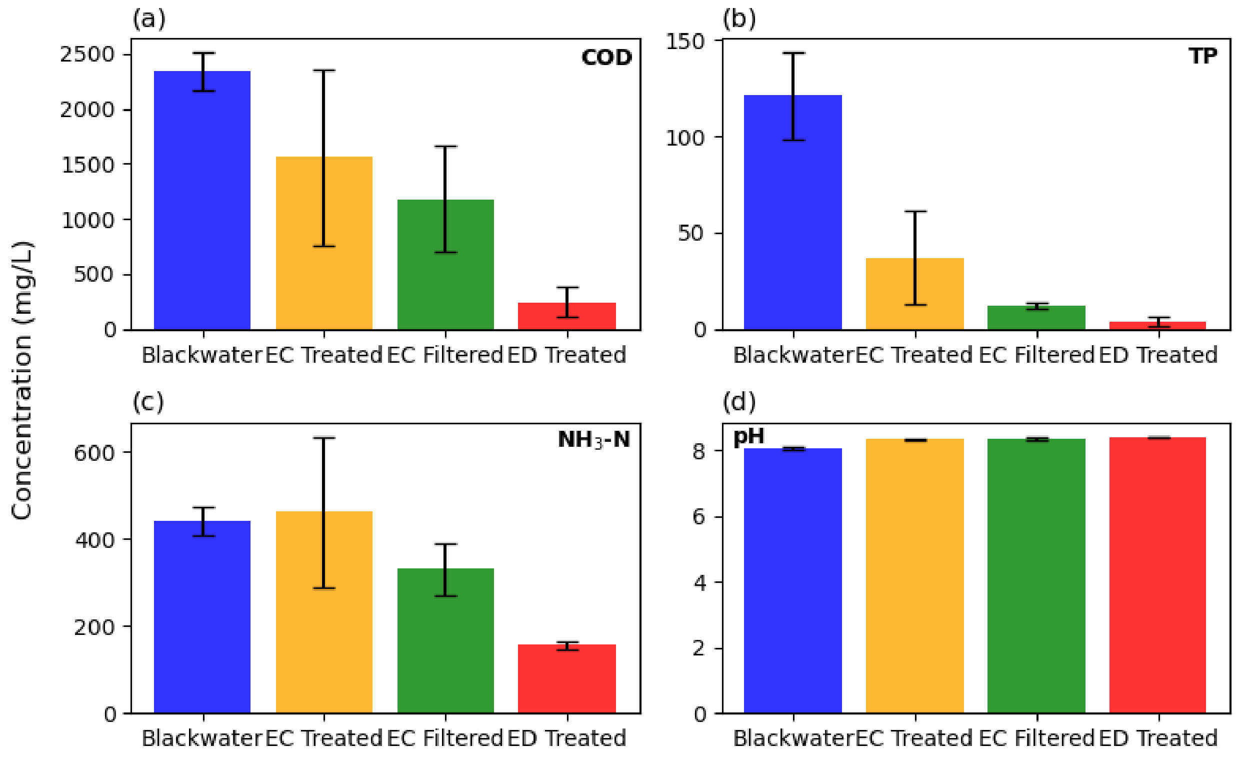


**Figure 12:** Subset of the treatment performance for key parameters at each stage of the treatment process for optimal treatment days at Camp Shelby **a)** COD **b)** TP **c)** $NH_3$-N **d)** pH.

Even when analyzing the subset of data from the operational period prior to severe electrode degradation, the absence of polarity reversal resulted in suboptimal EC performance, as shown in Table 4. This highlights the critical role of polarity switching in facilitating the consistent release of ferrous and ferric ions, which are essential for destabilizing colloidal particles and promoting floc formation through charge neutralization and adsorption.

Without polarity reversal, removal efficiencies for most parameters remained below 70%, underscoring the decline in EC effectiveness due to electrode passivation and fouling. While total phosphorus and ammonia removal efficiencies were slightly improved relative to the full dataset, this was primarily due to the continued contribution of electrodialysis downstream.

Notably, COD removal showed a marked improvement, reaching 89.5% ± 50.2, despite the instability in system operation. This high variability suggests that treatment efficiency was highly dependent on transient conditions such as influent quality and electrode condition during each run.

**Table 4:** Isolated data removal efficiencies of key parameters: COD, TP, and $NH_3$-N for each treatment.

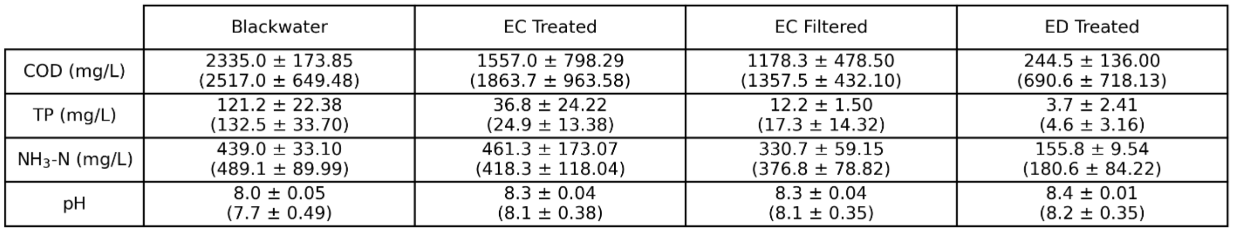
| | COD (%) | TP (%) | $NH_3$-N (%) |
|---|---|---|---|
| Blackwater → EC Treated | 33.3 ± 35.0 | 69.6 ± 27.2 | -5.1 ± 40.1 |
| EC Treated → EC Filtered | 24.3 ± 59.8 | 67.0 ± 65.9 | 28.3 ± 39.6 |
| EC Filtered → ED Treated | 79.3 ± 42.2 | 69.7 ± 23.4 | 52.9 ± 18.1 |
| Total | 89.5 ± 50.2 | 97.0 ± 65.9 | 64.5 ± 6.3 |

An analysis of both the isolated dataset and the full demonstration, (**Table 5**), confirms that influent pH plays a critical role in the effective removal of ammonia during ED treatment. In the subset of pre-damage data, the $NH_3$-N concentration in the ED-treated effluent remained elevated at 155.8 ± 9.54 mg/L, well above typical discharge limits, emphasizing the impact of high pH on ammonium speciation and membrane transport.

While chemical oxygen demand in the isolated dataset averaged 244.5 ± 136 mg/L, narrowly meeting the U.S. Environmental Protection Agency (EPA) effluent guideline of 250 mg/L, the overall demonstration failed to meet this requirement. Total phosphorus in the isolated dataset averaged 3.7 ± 2.41 mg/L, exceeding the commonly cited discharge threshold of 2 mg/L, further indicating incomplete treatment under compromised operating conditions (Owusu-Ansah et al. 2015).

In addition to reduced treatment efficacy, the system's energy demand increased significantly. Due to the lower flow rate, elevated current setpoint, and reduced electrode efficiency from fouling, the energy consumption for the field demonstration rose to 18.16 ± 10.17 Wh/L, more than double the energy demand observed during sewage sludge testing. This underscores the importance of optimizing operating conditions and system resilience in field deployments to ensure both treatment performance and energy efficiency.

**Table 5:** Comparison of the subset data to the full test data (parenthesis) after each treatment unit of the port-o-john water.



| | Blackwater | EC Treated | EC Filtered | ED Treated |
|---|---|---|---|---|
| COD (mg/L) | 2335.0 ± 173.85<br>(2517.0 ± 649.48) | 1557.0 ± 798.29<br>(1863.7 ± 963.58) | 1178.3 ± 478.50<br>(1357.5 ± 432.10) | 244.5 ± 136.00<br>(690.6 ± 718.13) |
| TP (mg/L) | 121.2 ± 22.38<br>(132.5 ± 33.70) | 36.8 ± 24.22<br>(24.9 ± 13.38) | 12.2 ± 1.50<br>(17.3 ± 14.32) | 3.7 ± 2.41<br>(4.6 ± 3.16) |
| $NH_3$-N (mg/L) | 439.0 ± 33.10<br>(489.1 ± 89.99) | 461.3 ± 173.07<br>(418.3 ± 118.04) | 330.7 ± 59.15<br>(376.8 ± 78.82) | 155.8 ± 9.54<br>(180.6 ± 84.22) |
| pH | 8.0 ± 0.05<br>(7.7 ± 0.49) | 8.3 ± 0.04<br>(8.1 ± 0.38) | 8.3 ± 0.04<br>(8.1 ± 0.35) | 8.4 ± 0.01<br>(8.2 ± 0.35) |

***Latrine Blackwater***

Blackwater was collected from a portable latrine located at the Michigan State University Dairy Farm and stored in a 3,000-gallon flexible onion tank. The waste stream was delivered from the latrine to the storage tank via a grinding pump, which helped to homogenize solids and reduce particle size before treatment. The blackwater was later transferred in 250-gallon Uline tote tanks to the Michigan State University Anaerobic Digestion Research and Education Center for testing.

The treatment system was evaluated using the same general protocol as the sewage sludge tests, with one key difference: the ED1000H continuous-flow electrodialysis unit was operated inline rather than batch ED testing. For this experiment, the flow rate setpoint was maintained at 0.95 L/min, and the electrocoagulation current was set to 35 A.

Treatment performance was evaluated at each major process stage, though due to the minor impact of filtration relative to EC and ED, data from the EC-filtered and ED-treated stages were combined for analysis. A summary of the average effluent concentrations at each stage is provided in Table 6.

The EC–ED process successfully met all U.S. EPA effluent guidelines for wastewater discharge, except for those related to ammonia and total nitrogen. The average post-treatment values for COD, TP, turbidity, TSS, and pH were 200.67 ± 40.72 mg/L, 0.96 ± 0.27 mg/L, 6.06 ± 1.14 NTU, 13.4 ± 2.59 mg/L, and 8.32 ± 0.04, respectively, relative to their corresponding EPA standards of 250 mg/L, 2 mg/L, 75 NTU, 50 mg/L and pH value ranging from 6-9 (Owusu-Ansah et al. 2015).

However, TN and $NH_3$-N concentrations remained significantly above discharge thresholds of 50 and 10 mg/L (Owusu-Ansah et al. 2015), with average values of 135.33 ± 56.05 mg/L and 78.59 ± 7.5 mg/L, respectively. These elevated nitrogen levels indicate a continued challenge in ammonium removal, likely due to the persistent high pH of the influent and the limitations of ED for uncharged ammonia species.

**Table 6:** Latrine blackwater mean treatment parameters at each process unit.

| | COD (mg/L) | TP (mg/L) | $NH_3$-N (mg/L) | TN (mg/L) | Turbidity (NTU) | TSS (mg/L) | pH |
|---|---|---|---|---|---|---|---|
| Blackwater | 1696.67 ± 214.82 | 22.2 ± 0.73 | 261.0 ± 27.94 | 429.25 ± 53.91 | 185.5 ± 13.55 | 183.67 ± 135.05 | 8.53 ± 0.03 |
| EC Treated | 358.0 ± 30.68 | 2.26 ± 0.52 | 250.0 ± 13.16 | 480.33 ± 141.71 | 5.22 ± 1.28 | 25.28 ± 14.75 | 8.69 ± 0.02 |
| ED Treated | 200.67 ± 40.72 | 0.96 ± 0.27 | 78.58 ± 7.5 | 135.33 ± 56.05 | 6.06 ± 1.14 | 13.4 ± 2.59 | 8.32 ± 0.04 |

The key parameters of COD, TN, NH3-N, and TP were plotted in Figure 13 to illustrate the treatment performance. There was a substantial removal of COD and TP during electrocoagulation while the TN and NH3-N values remained high. Electrodialysis showed to further reduce COD and TP while also reducing the TN and $NH_3$-N concentrations. The nitrogen and ammonia concentrations remained high relative to the sewage sludge testing as the latrine blackwater had a high pH of 8.53 ± 0.03.

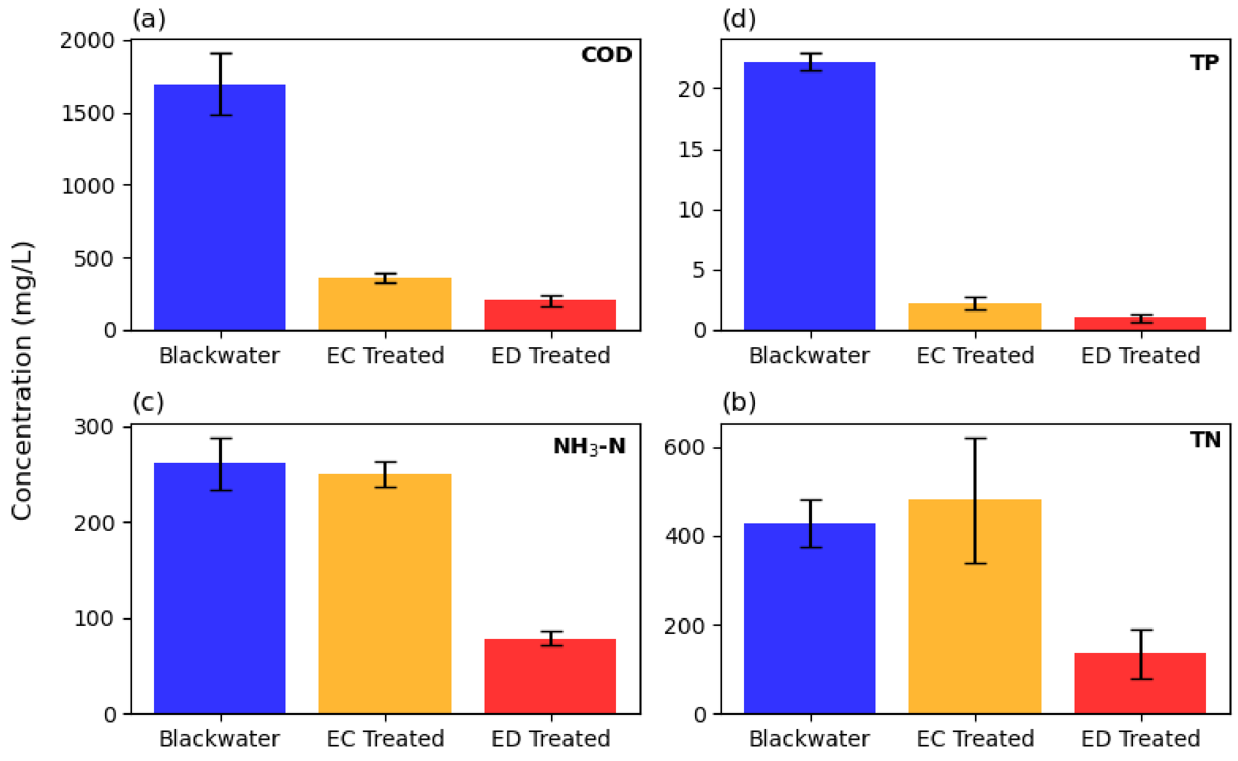


**Figure 13:** Treatment performance for key parameters at each stage of the treatment process for testing of the latrine blackwater **a)** COD **b)** TP **c)** $NH_3$-N **d)** TN.

These trends are further illustrated in **Table 7**, which presents the percent removal of each key parameter by process stage, as well as the overall system removal. High average removal efficiencies were achieved for COD, TP, turbidity, and TSS, with values of 88.2 ± 21.1%, 95.7 ± 27.4%, 96.7 ± 19.6%, and 92.7 ± 70.5%, respectively. In contrast, TN and $NH_3$-N showed moderately lower removal efficiencies, averaging 68.5 ± 29.6% and 69.9 ± 10.0%, respectively. This contrast in treatment performance further shows the need for pH reduction before electrodialysis for high removal efficiencies of ammonia and total nitrogen allowing the process to meet discharge standards.

**Table 7:** Removal efficiencies for the latrine blackwater testing.

| | COD (%) | TP (%) | $NH_3$-N (%) | TN (%) | Turbidity (%) | TSS (%) |
|---|---|---|---|---|---|---|
| Blackwater → EC Treated | 78.9 ± 12.8 | 89.8 ± 4.1 | 4.2 ± 11.8 | -11.9 ± 35.3 | 97.2 ± 7.3 | 86.2 ± 74.0 |
| EC Treated → ED Treated | 43.9 ± 14.2 | 57.5 ± 26.1 | 68.6 ± 6.1 | 71.8 ± 31.7 | -16.1 ± 32.9 | 47.0 ± 59.2 |
| Total | 88.2 ± 21.1 | 95.7 ± 27.4 | 69.9 ± 10.0 | 68.5 ± 29.6 | 96.7 ± 19.6 | 92.7 ± 70.5 |

The energy consumption of the process for the latrine wastewater were slightly higher than observed in the sewage sludge testing, being 12.49 ± 2.23 Wh/L treated. A comparison of the mean energy consumption and observed flow rate for each water tested can be seen in Table 8.

**Table 8:** Observed flow rate and energy consumption comparison for each blackwater tested.

| | Flow Rate (L/min) | Energy (Wh/L) |
|---|---|---|
| Sewage sludge | 1.61 ± 0.69 | 9.07 ± 4.20 |
| Camp Shelby | 0.73 ± 0.35 | 18.16 ± 10.17 |
| Latrine | 1.03 ± 0.13 | 12.49 ± 2.23 |

# CHAPTER 3: LIFE CYCLE AND TECHNO-ECONOMIC ANALYSIS

## MATERIALS AND METHODS

### *Mass and energy balance*

Mass and energy balance analyses were carried out for three different blackwater sources of sewage sludge blackwater, port-o-john blackwater, and latrine blackwater based on the system flowchart (Figure 14). The mass balance includes the following flows in (kg/day) with the treatment flow rates. All flow data and characteristic data were obtained from the previous sections. Following the mass balance analysis, an energy balance was conducted for each blackwater. Energy input data includes energy consumption for EC and ED. Energy output data include energy generation from PV. All energy data were obtained from the pilot operation. The mass and energy balance analysis determine the energy demand per day or the energy demand/m$^3$ treated blackwater for each blackwater.

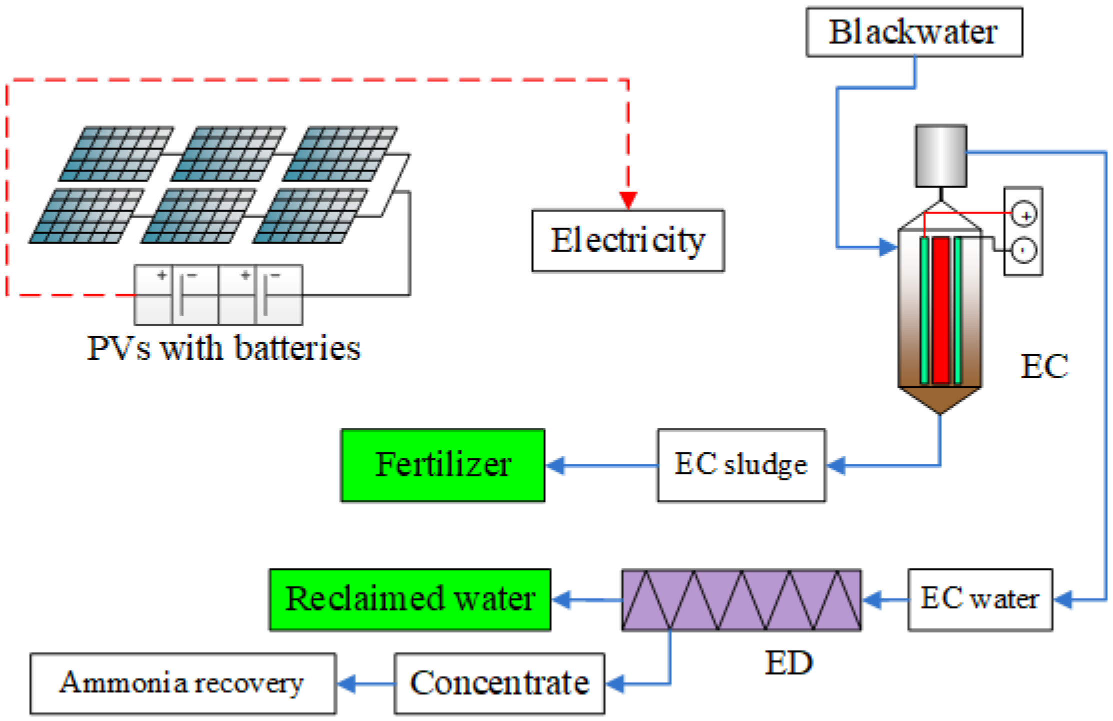


**Figure 14:** The flowchart of the studied PV-EC-ED system.

***Life cycle impact assessment (LCIA)***

With the detailed mass and energy balance analysis, an LCIA was carried out to evaluate the environmental impacts of the EC-ED on the treatment of each blackwater. A conventional treatment practice of activated sludge treatment of blackwater was used as the control. The boundary of the LCIA is from the source of the blackwater to the end products of the EC-ED process including treated water, renewable energy, and fertilizer (Figure 14). Two impact categories related to carbon emission and water quality were chosen for the life cycle impact assessment: Global Warming Potential (GWP) and Water Eutrophication Potential (WEP). The data generated from the mass and energy balance was used to establish a life cycle inventory. All emission factors for individual compounds are listed in Table 9. The EPA Tool for Reduction and Assessment of Chemicals and Other Environmental Impacts (TRACI) version 2.1 was used for the LCIA. To calculate the impact for each category being considered, the substance mass from each emission source is multiplied by the listed characterization factors. Summing the total emissions within each impact category results in the total impact score for each category. Contribution analysis was performed to elucidate the influences of different treatment combinations on each impact category.

**Table 9:** Parameters for life cycle impact analysis.

| | Item | Value | Unit |
|---|---|---|---|
| GWP | $N_2O$ emission factor | 0.005 | g N emitted as $N_2O$/g TN in the food waste or wastewater |
| | Molecular weight conversion of $N_2O$ per $N_2$ | 1.5714 | |
| | GWP factor of $N_2O$ emission | 298 | Kg $CO_2$-e/kg $N_2O$ |
| | GWP factor of natural gas electricity | 0.491 | Kg $CO_2$-e/kWh |
| WEP | WEP factor of TN | 0.9864 | Kg N-eq/kg TN |
| | WEP factor of TP | 7.29 | Kg N-eq/kg TP |
| | WEP factor of COD | 0.05 | Kg N-eq/kg COD |

***Economic analysis***

The economic assessment is important to determine the viability of the real-world application. Capital Expenditure (CapEx), Operational Expenditure (OpEx), and cost-saving are the parameters to assess the economic performance of different treatment combinations. The CapEx (Table 10) and OpEx data were collected from the system fabrication and the demonstration operation (Table 10). The current electricity cost of \$0.2/kWh-e is used to calculate energy cost. The Modified Accelerated Cost Recovery System (MACRS) was used to calculate the annual depreciation of CapEx. The MACRS annual depreciation rates are 0.100, 0.188, 0.144, 0.115, 0.092, 0.074, 0.066, 0.066, 0.065, 0.065, 0.033, and 0.033 (after 10 years). Twenty years was set as the lifetime for individual treatment combinations. Annual inflation of 3.2% was set for OpEx. The tax rate is 35%. The net cash flow based on depreciated CapEx and inflated OpEx was conducted to determine the treatment cost.

**Table 10:** Capital cost of individual units.

| Unit | The cost |
|---|---|
| **PV unit** | **\$21,820** |
| 23 $m^2$ PV panel and batteries | \$4,900 |
| 4 batteries | \$9,600 |
| Control panel and software | \$2,500 |
| Unit installation (20% of the capital cost) | \$4,820 |
| | |
| **EC unit** | **\$29,160** |
| EC reactor with electrodes and valves | \$3,500 |
| EC sludge separator with electrodes and valves | \$3,500 |
| Inclining clarifier | \$10,800 |
| Pumps (feeding pump) | \$4,500 |
| Control panel and software | \$2,000 |
| Unit installation (20% of the capital cost) | \$4,860 |
| | |
| **ED unit** | **\$18,500** |
| ED unit with valves and flow meters | \$16,000 |
| Power unit | \$500 |

Table 10 (cont'd)

| | |
|---|---|
| Control panel and software | \$2,000 |
| | |
| Tricon unit | \$5,000 |
| **Total Cost** | **\$74,480** |

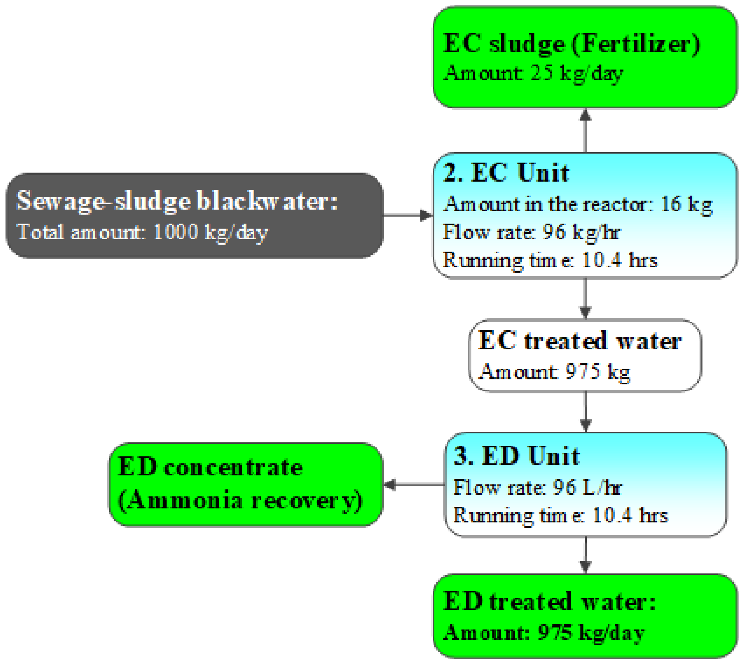
## RESULTS AND DISCUSSION

### ***Mass and energy balance***

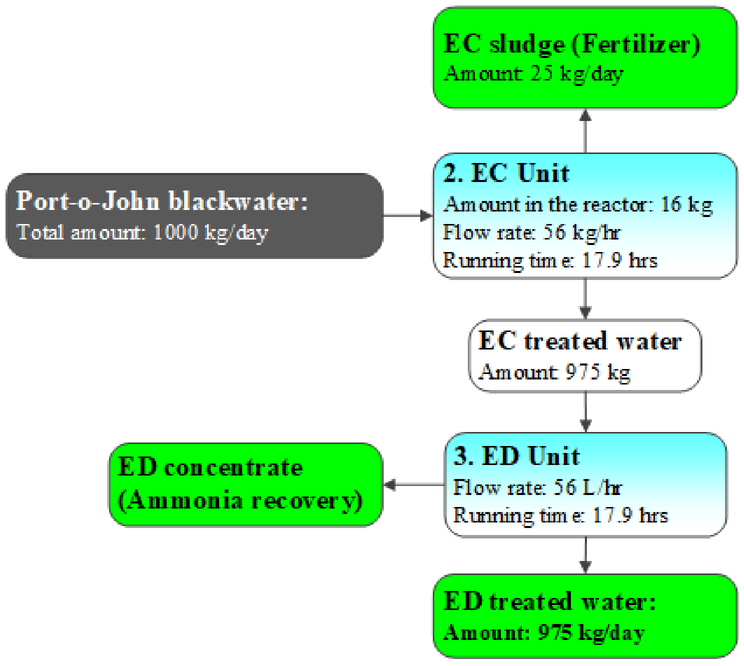
To evaluate the performance and adaptability of the electrochemical-electrodialysis (EC-ED) strategy across a spectrum of high-strength wastewaters, we analyzed mass flows and water quality transformations for 1000 kg of three distinct blackwater types: (a) sewage sludge blackwater, (b) port-o-john blackwater, and (c) real latrine toilet blackwater. Figure 15 a–c presents the mass flow balance for each stream. The treatment performance was presented in Table 8. According to the mass balance data (Figure 15), all three treatments generate the same amount of treated water (975 kg/day) and sludge (256 kg/day). However, the flowrates of individual treatments are different at 96, 56, and 62 kg/hr for sewage sludge blackwater, port-o-john blackwater, and latrine blackwater, respectively, which reflects the hydraulic response of the EC-ED process to differences of blackwater strengths and compositions.

The sewage sludge blackwater was characterized by moderate COD (2035.7 ± 830.3 mg/L), $NH_3$-N (134.8 ± 46.8 mg/L), and TSS (884.1 ± 613.3 mg/L). The EC step substantially reduced suspended solids (TSS from 884.1 to 158.5 mg/L), enabling effective downstream filtration and electrodialysis. By the end of treatment, COD dropped by ~96% (to 75.3 ± 25.8 mg/L), and $NH_3$-N decreased by ~97% (to 3.9 ± 1.3 mg/L). While ammonia reduction appears significant, it is important to recognize that this stream was highly diluted and consistent in composition, lacking many of the physical and chemical interferences found in real wastewaters. Port-o-John blackwater exhibited the highest pollutant loading, with COD at 2335.0 ± 173.9

mg/L, TP at 121.2 ± 22.4 mg/L, and $NH_3$-N at 439.0 ± 33.1 mg/L. Its low water content and high solids load significantly impacted flow rate. The flow rate of the treatment (56 kg/hr) was the slowest among three treatment. Despite EC reducing COD to 1557.0 ± 798.3 mg/L and $NH_3$-N to 461.3 ± 173.1 mg/L, downstream filtration and ED only brought final values down to 244.5 ± 136.0 mg/L and 155.8 ± 9.5 mg/L, respectively. These reductions were proportionally lower compared to the other streams due to the influent concentrations. Notably, ammonia remained relatively high in the ED-treated effluent, suggesting limitations in separation efficiency that were influenced by pH and chemical additives common in portable sanitation systems. The latrine blackwater presented intermediate characteristics with COD at 1696.7 ± 214.8 mg/L and $NH_3$-N at 261.0 ± 27.9 mg/L, but the highest TN at 429.3 ± 53.9 mg/L. EC treatment reduced COD to 358.0 ± 30.7 mg/L and TSS from 183.7 to 25.3 mg/L, facilitating smoother downstream separation. ED further reduced COD to 200.7 ± 40.7 mg/L and $NH_3$-N to 78.6 ± 7.5 mg/L. even though the COD and NH3-N removal efficiencies are notable given the variability and complexity of real toilet waste, the final water quality was not as good as that from the treatment of sewage sludge blackwater.



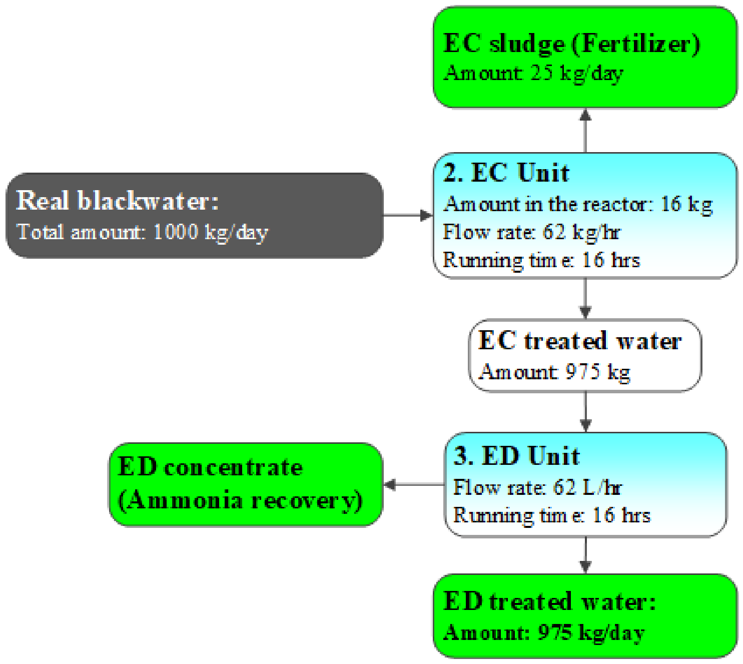
(a)



(b)

**Figure 15:** Mass balance of the EC-ED process on different blackwater. (a) blackwater made from sewage sludge (b) blackwater from a port-o-John latrine (c) real blackwater from toilet.

Figure 15 (cont'd)



(c)

Following the detailed mass balance and flowrate comparison, an energy balance was conducted to evaluate the feasibility of the EC-ED process from a sustainability perspective. The system was powered by solar energy, with an average collection capacity of 22 kWh/day on sunny days in East Lansing, MI. Energy consumption and recovery were assessed across the same three blackwater sources. Despite the differences in treatment performance and flowrates for different blackwater sources, the system maintained net-positive energy output in all cases. However, the magnitude of net energy gain varied significantly, reflecting the influence of wastewater strength on energy efficiency. Sewage sludge blackwater among the three blackwater shows the best energy balance. With an energy input of only 8.5 kWh/day, the system yielded a net energy output of 13.5 kWh/day (13.5 kWh per cubic meter of blackwater treated). The high treated flowrate allowed efficient use of energy, demonstrating the system's baseline potential under controlled conditions. The port-o-john blackwater had the lowest treated flowrate (56 kg/hr) due to high solids loading and suspected chemical interference, which led to a higher energy demand of 9.5 kWh/day. The resulting net energy output was 12.5 kWh/day (12.5 kWh/$m^3$)—lower than the sewage sludge blackwater treatment, but still clearly positive. The latrine blackwater represented a real blackwater stream, required the highest energy input of 12.5 kWh/day. Although the flowrate (62 kg/hr) was higher than port-o-john blackwater, the variability in solids and nitrogen content, along with higher pH, likely contributed to greater energy demand during electrochemical operations. The net energy output was 9.5 kWh/day, or 9.5 kWh per cubic meter, the lowest among the three. Nonetheless, the treatment process remained energy-positive, reinforcing the system's adaptability even under more challenging and realistic wastewater conditions.

The comparison between mass and energy balances reveals a clear trade-off that as blackwater strength and complexity increase, treated flowrate decreases and energy input rises, reducing overall energy efficiency. Nevertheless, across all three source waters, the system delivered surplus energy on sunny days—a critical requirement for sustainable decentralized treatment systems.

**Table 11:** Energy balance of the system[a,b].

| Blackwater source | Energy input (kWh-e/day)[c] | Energy output (kWh-e/day) | Net energy output (kWh-e/day) | Net energy output (kWh-e/$m^3$ blackwater) |
|---|---|---|---|---|
| Sewage sludge blackwater | -8.5 | 22 | 13.5 | 13.5 |
| Port-o-John blackwater | -9.5 | 22 | 12.5 | 12.5 |
| Latrine blackwater | -12.5 | 22 | 9.5 | 9.5 |

a. The positive numbers are energy outputs, and the negative numbers are energy inputs. The energy consumption is based on the average during a year-round operation and only considers the sunny days.
b. All data are collected from the demonstration operation.
c. The recorded energy consumptions of the EC-ED process on sewage sludge blackwater, Port-o-John blackwater, and latrine blackwater were 8.5, 9.5, and 12.5 Wh/kg water.

Table 11 (cont'd)

d. The solar panels can collect 22 kWh/day of electricity on a sunny day in the summer at East Lansing, MI.

***Life cycle impact assessment of the EC-ED on three different blackwaters***

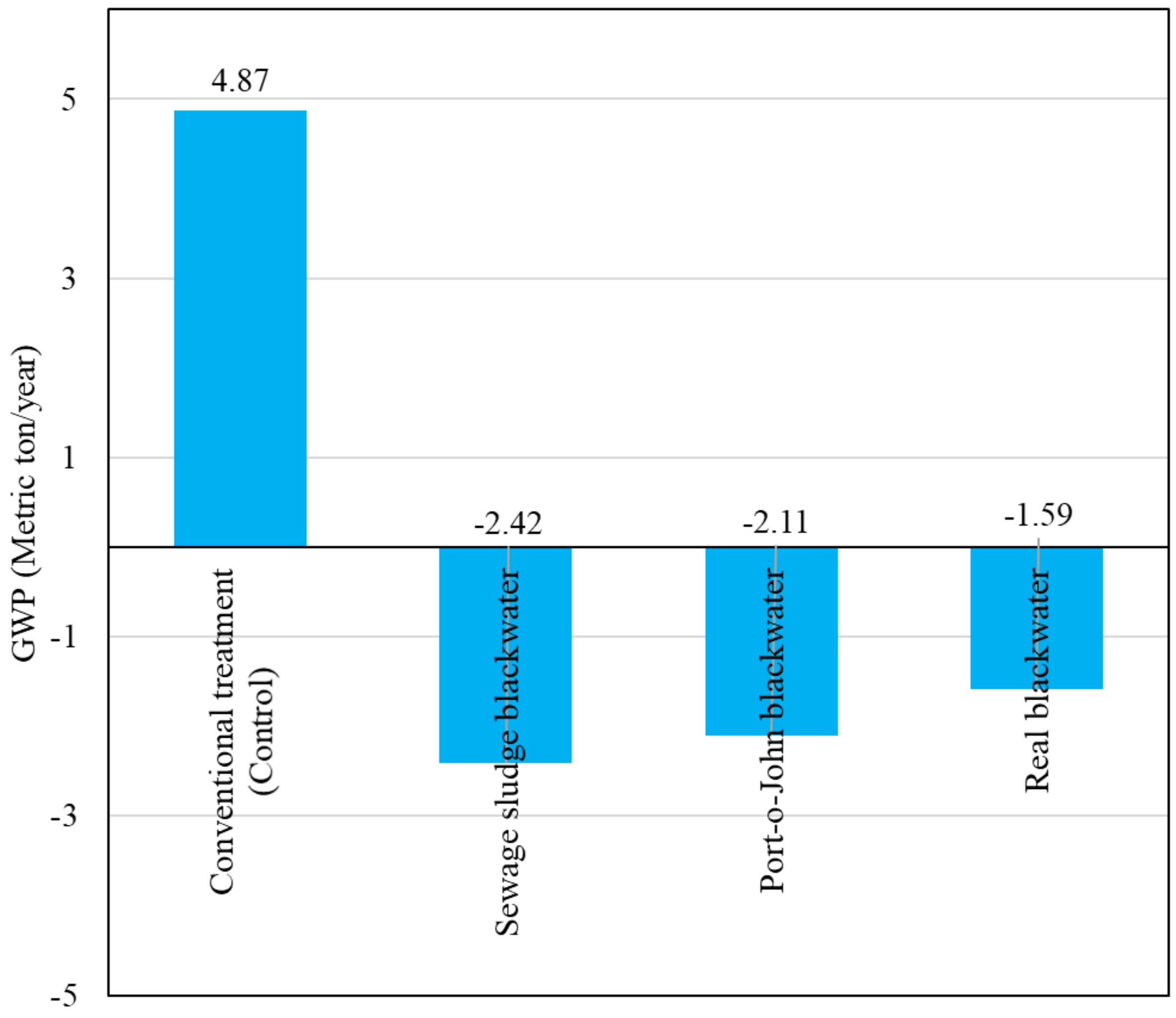
Building upon the mass and energy balance results, a life cycle impact assessment was conducted to evaluate the environmental performance of the EC-ED system relative to a conventional activated sludge (CAS) treatment system, used here as a control. Two impact categories were considered: Global Warming Potential (GWP) and Water Eutrophication Potential (WEP). The CAS control represents a typical wastewater treatment pathway relying on grid electricity and biological nitrogen removal, which is typical for relatively large-scale operations. In contrast, the EC-ED system is solar-powered and deployed, and also able to handle variation of wastewater strength.

The conventional activated sludge system exhibited a significant GWP of 4.87 metric tons $CO_2$-eq/year, reflecting its high reliance on fossil-derived grid electricity and energy-intensive aeration and sludge processing. In comparison, all EC-ED configurations demonstrated net-negative GWP values: -2.42, -2.11, and -1.59 metric tons $CO_2$-eq/year for sewage sludge blackwater, port-o-john blackwater, and latrine blackwater, respectively. These reductions were directly enabled by the system's solar-powered operation and energy-positive performance described earlier. Although energy consumption varied slightly among the EC-ED cases (from 8.5 to 12.5 kWh/day), all systems leveraged solar energy with a daily yield of 22 kWh, achieving clear climate benefits over CAS. The magnitude of GWP reduction correlated with treated flowrate and energy efficiency, with the sewage sludge blackwater treatment outperforming due to smoother hydraulic throughput and lower treatment burden.

Unlike GWP, the WEP analysis revealed a trade-off, where the EC-ED processes, particularly those treating high-strength wastewaters, produced higher nutrient-related impacts

(5.33, 71.18, and 54.83 kg N eq/year for sewage sludge blackwater, port-o-john blackwater, and latrine blackwater) than the conventional system (7.6 kg N eq/year). While sewage sludge blackwater performed comparably to the CAS control, the other two blackwaters exhibited much higher eutrophication potentials. These elevated WEP values are mainly caused by incomplete nitrogen removal, particularly residual $NH_3$-N in the ED-treated effluents (e.g., $155.8 \pm 9.5$ mg/L for PJ-BW and $78.6 \pm 7.5$ mg/L for TBW). Unlike CAS, which uses biological nitrification–denitrification to remove nitrogen, the EC-ED system in its current configuration focuses on ammonia separation but not complete capture or polishing. This performance gap underscores the need for integrated nutrient recovery or polishing steps in EC-ED systems, especially for blackwater streams with high nitrogen content or low flow rates that limit separation efficiency. ED unit operation needs to be further optimized to improve NH3 removal efficiency from port-o-john and latrine blackwater.

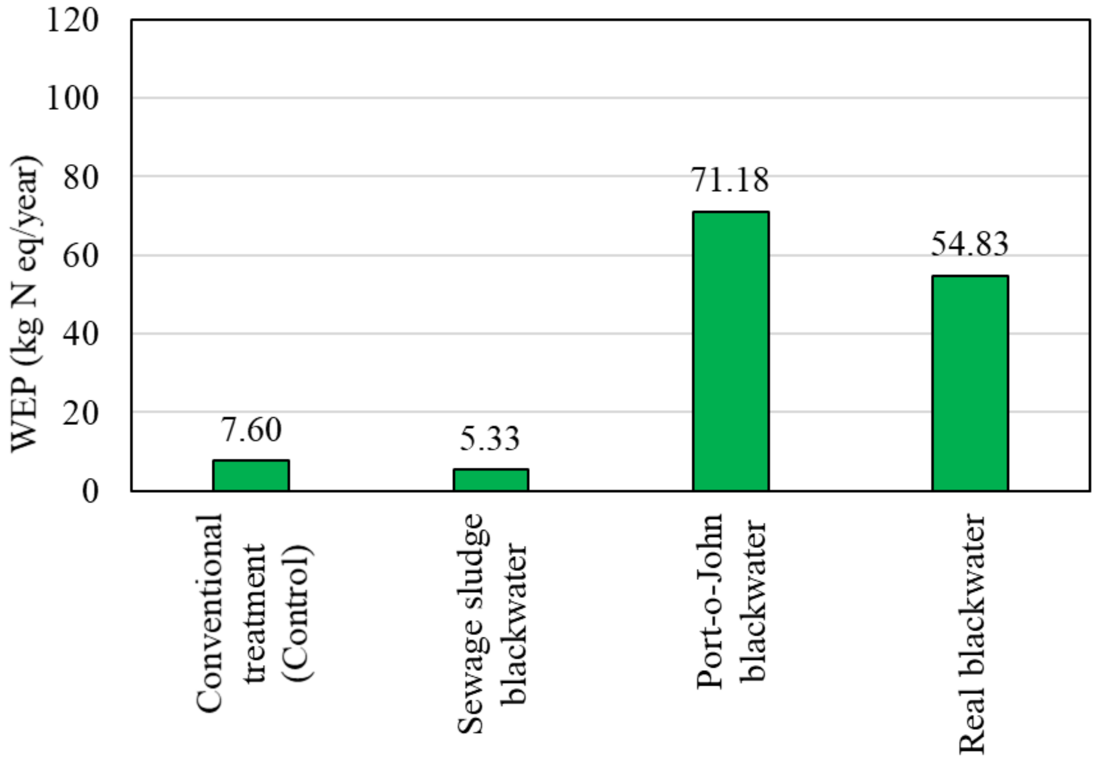
Overall, the EC-ED approach represents a low-carbon, decentralized alternative to conventional wastewater treatment, especially in sunny, off-grid environments. The EC-ED system achieves net-negative GWP across all blackwater types when operated on solar power, making it a promising climate mitigation tool in decentralized settings. However, nutrient leakage remains a concern in EC-ED, particularly for real and concentrated wastewaters, where ammonia separation is incomplete and effluent polishing is absent. Its integration into a circular sanitation framework will depend on further development of nutrient recovery strategies to complement its energy and climate advantages.



(a)

**Figure 16:** Contribution analysis of GWP and WEP for different combinations. (a) Global warming potential (GWP) (b) Water eutrophication potential (WEP)

Figure 16 (cont'd)



(b)

***Economic analysis of the EC-ED treatment***

An economic assessment was conducted to evaluate the financial viability of deploying the EC-ED system for decentralized blackwater treatment. The evaluation included capital and operational expenditures, maintenance, labor, and energy savings over a 20-year project lifetime. Table 12 summarizes the economic performance across three blackwater streams.

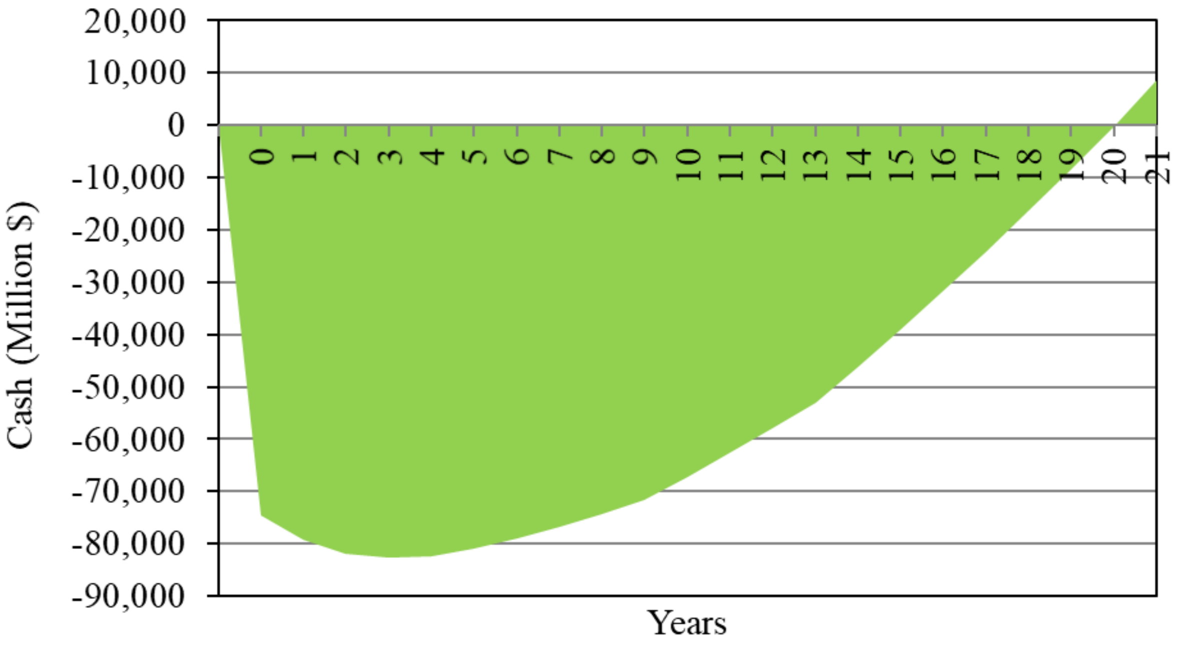
The capital cost (CapEx) for all scenarios was fixed at $74,480, reflecting identical system design and equipment specifications. Operational expenditures (OpEx), however, varied slightly due to differences in energy use and treatment throughput: $7,464/year, $7,537/year, and $7,756/year for sewage sludge blackwater, port-o-john blackwater, and latrine blackwater, respectively. The higher OpEx for the latrine blackwater treatment reflects its greater energy demand (12.5 kWh/day) and slightly lower throughput, which increases the cost per volume treated. While the port-o-john blackwater treatment had a slightly lower energy input (9.5 kWh/day), its lower treated flowrate (56 kg/hr) also contributed to higher unit costs. Maintenance was set at $1,000/year across all scenarios, based on pilot-scale experience involving periodic replacement of EC electrodes. Labor costs were uniform at $8,450/year, assuming one hour of system monitoring and feeding per day at a loaded rate of $32.50/hour. These costs reflect the minimal but essential input required for decentralized operation. Because the system is solar-powered and energy-positive on sunny days, net energy generation was monetized as a cost saving based on Michigan's electricity market price ($0.20/kWh). Annual energy savings were: $986/year, $913/year, and $694/year for sewage sludge blackwater, port-o-john blackwater, and latrine blackwater. The trend reflects earlier energy balance results, sewage sludge blackwater had the highest net energy output and thus the greatest cost savings, while latrine blackwater's higher energy demand and lower flowrate resulted in more modest savings. Although these savings are not enough to offset total OpEx, they do contribute meaningfully to lowering the treatment cost. The final treatment cost, calculated over a 20-year system life, varied narrowly across sources: $39.6/m$^3$, $39.8/m$^3$, and $40.4/m$^3$ for sewage sludge blackwater, port-o-john blackwater, and latrine blackwater, respectively. These values represent high-end costs relative to centralized treatment, but they are within range for decentralized and off-grid sanitation systems, especially when accounting for the environmental and energy co-benefits discussed previously.

When interpreted alongside energy and environmental metrics, the economic data suggest that the EC-ED system is technically and economically viable for decentralized applications, with treatment costs below $41/m$^3$ even under real-world blackwater conditions. Energy-positive operation contributes modest but real cost savings, particularly in areas with high electricity prices or limited grid access. Labor and maintenance costs dominate the annual expenditure, highlighting the importance of automation, local training, or shared operation models in rural or remote deployments. While the EC-ED system does not yet reach the economies of scale of conventional infrastructure, its ability to deliver climate-positive, energy-generating treatment in compact, modular form offers significant value for decentralized, circular sanitation solutions, particularly in small or off-grid communities.

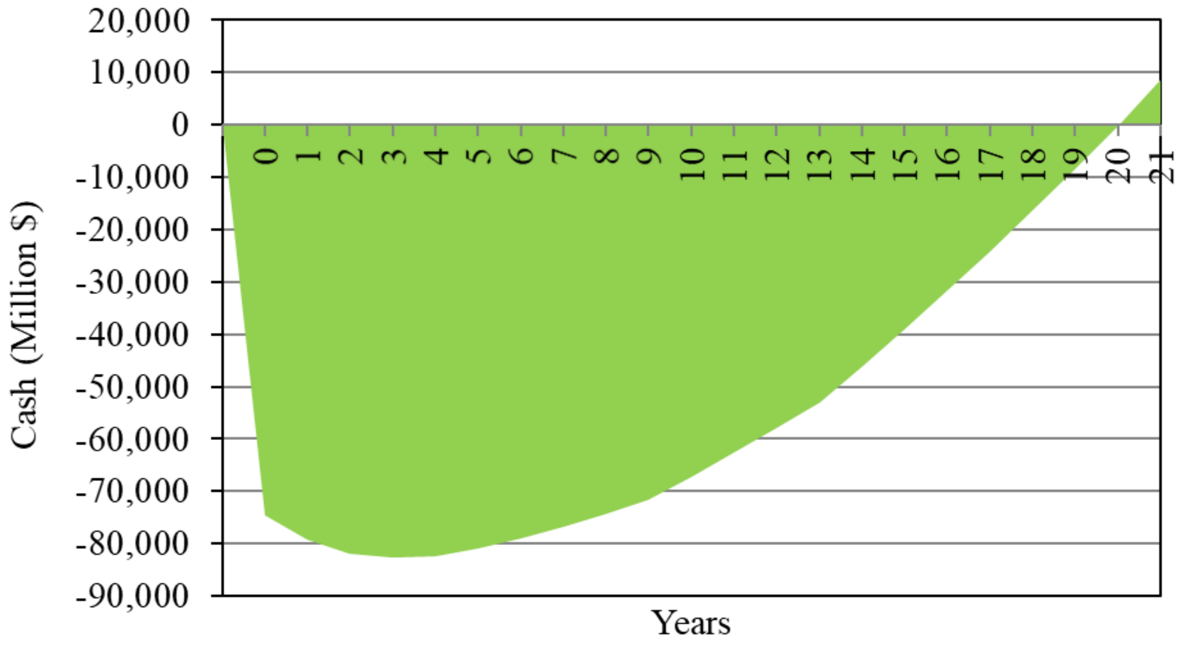
**Table 12:** Economic performance of the EC-ED process on different blackwater.

| | Sewage sludge blackwater | Port-o-John blackwater | Latrine blackwater |
|---|---|---|---|
| **Capital expenditure (CapEx) ($)** | 74,480 | 74,480 | 74,480 |
| **Operational expenditure (OpEx) ($/year)** | 8,464 | 8,537 | 8,756 |
| Maintenance ($/year) [a] | 1,000 | 1,000 | 1,000 |
| Labor cost ($/year) [b] | 8,450 | 8,450 | 8,450 |
| Energy saving ($/year) [c] | -986 | -913 | -694 |
| **Treatment cost ($/m$^3$ treated water)** [d] | 39.6 | 39.8 | 40.4 |

a. The maintenance cost is based on the demonstration operation to replace EC electrodes.
b. It requires 1 hour/working day to feed the system and check the operation based on the pilot operation. The hourly payment for the operator is $25/hour with a 30% fringe benefit.
c. The cost of energy demand is assigned as positive numbers, and the cost of energy generation is assigned as negative numbers. The energy cost is $0.20/kWh-e based on the market price of electricity in Michigan in 2024.
d. The treatment cost is calculated based on 20 years of lifetime for the EC-ED processes (Figure 17).



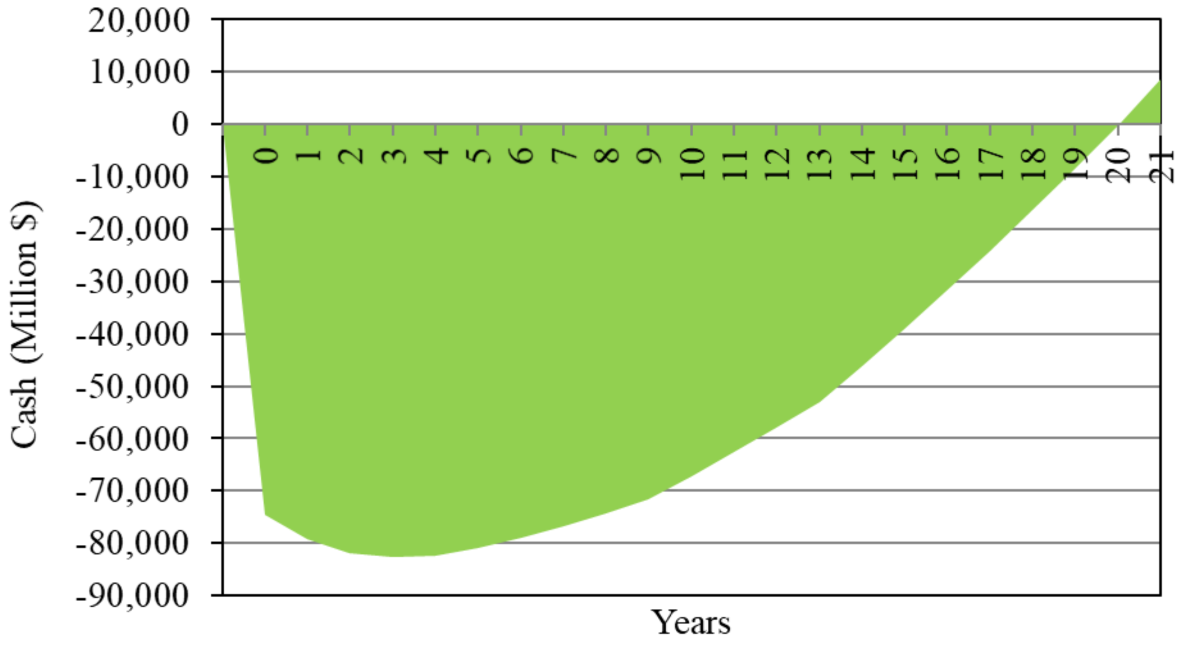
(a)



(b)

**Figure 17:** Treatment cost analysis (a) Sewage sludge blackwater (b) Port-o-John blackwater, and (c) Latrine blackwater.

Figure 17 (cont'd)



(c)

# CHAPTER 4: CONCLUSIONS AND FUTURE WORK

Lab scale testing was performed to determine that electrodialysis is a viable electrocoagulation posttreatment option. It was able to reduce $NH_3$-N concentrations below 1 mg/L with a relatively low energy consumption of ~1.3 Wh/L treated. It was able to achieve this while also concentrating ammonia to ~7500 mg$NH_3$-N/L. ED reversal was determined to add unnecessary complication and therefore was determined to not be beneficial for implementation in pilot-scale testing.

A combined electrocoagulation and electrodialysis process was used to treat three high-strength blackwaters: sewage sludge, port-o-john, and latrine. The results showed that high removal rates of COD, TP, $NH_3$-N, TSS, and turbidity (>95%) are possible using this process. Treatment performance was found to decrease with high pollutant loading with $NH_3$-N and TN being the most strongly affected parameters especially for high pH waters. Treatment was performed with energy demands from 9.07 to 18.16 Wh/L.

LCA and TEA showed that treatment of all three waters had a negative GWP ranging from -2.42 to -1.59 metric tons $CO_2$-eq/year. The WEP ranged from 5.33 to 71.18 kg N eq/year. The total capital cost of the process is \$74,480 with operating costs ranging from \$39.6/m$^3$ to \$40.4/m$^3$ treated water.

Future work should focus on optimizing and improving the robustness of the process to handle a variety of high pollutant wastewaters. Emphasis should be placed on the processes ability to tolerate high pH wastewaters and to determine if the decreased $NH_3$-N and TN removal resulted from the high pH. Multi-objective optimization of the water quality, energy demand, and EC-ED parameters should be performed to determine the ideal relationship between EC and ED. Additionally, future work should be done to further study and purify the ammonia concentrate to isolate and upgrade the ammonia quality. Water recycling technologies such as reverse osmosis should also be studied for their viability of implementation into this process. Biological treatment performance should also be studied to determine the processes' ability to prevent biological contamination.

## REFERENCES

Aburto Vazquez, Gustavo. 2023. “An Electrochemical Process for Decentralized Treatment of High-Strength Wastewater.” M.S., Michigan State University, United States -- Michigan.

Ahangarnokolaei, M. A., B. Ayati, and H. Ganjidoust. 2021. “Simultaneous and Sequential Combination of Electrocoagulation and Ozonation by Al and Fe Electrodes for DirectBlue71 Treatment in a New Reactor: Synergistic Effect and Kinetics Study.” *Chemosphere* 285:131424. doi: 10.1016/j.chemosphere.2021.131424.

Akbari, Somaye, Farshid Ghanbari, and Mahsa Moradi. 2016. “Bisphenol A Degradation in Aqueous Solutions by Electrogenerated Ferrous Ion Activated Ozone, Hydrogen Peroxide and Persulfate: Applying Low Current Density for Oxidation Mechanism.” *Chemical Engineering Journal* 294:298–307. doi: 10.1016/j.cej.2016.02.106.

Anon. 2020. “New Frontiers from Removal to Recycling of Nitrogen and Phosphorus from Wastewater in the Circular Economy.” *Bioresource Technology* 300:122673. doi: 10.1016/j.biortech.2019.122673.

Anon. n.d.-a. “Haber-Bosch Process.” Retrieved January 6, 2025 (http://large.stanford.edu/courses/2022/ph240/jimenez2/).

Anon. n.d.-b. “Sanitation.” Retrieved October 24, 2024 (https://www.who.int/news-room/fact-sheets/detail/sanitation).

Asaithambi, P., Modepalli Susree, R. Saravanathamizhan, and Manickam Matheswaran. 2012. “Ozone Assisted Electrocoagulation for the Treatment of Distillery Effluent.” *Desalination* 297:1–7. doi: 10.1016/j.desal.2012.04.011.

Bernal-Martínez, Lina A., Carlos Barrera-Díaz, Reyna Natividad, and Manuel A. Rodrigo. 2013. “Effect of the Continuous and Pulse *in Situ* Iron Addition onto the Performance of an Integrated Electrochemical–Ozone Reactor for Wastewater Treatment.” *Fuel* 110:133–40. doi: 10.1016/j.fuel.2012.11.067.

Butler, Erick, Yung-Tse Hung, Ruth Yu-Li Yeh, and Mohammed Suleiman Al Ahmad. 2011. “Electrocoagulation in Wastewater Treatment.” *Water* 3(2):495–525. doi: 10.3390/w3020495.

Chen, Tse-Lun, Li-Heng Chen, Yupo J. Lin, Chang-Ping Yu, Hwong-wen Ma, and Pen-Chi Chiang. 2021. “Advanced Ammonia Nitrogen Removal and Recovery Technology Using Electrokinetic and Stripping Process towards a Sustainable Nitrogen Cycle: A Review.” *Journal of Cleaner Production* 309:127369. doi: 10.1016/j.jclepro.2021.127369.

Das, Pranjal P., Mukesh Sharma, and Mihir K. Purkait. 2022. “Recent Progress on Electrocoagulation Process for Wastewater Treatment: A Review.” *Separation and Purification Technology* 292:121058. doi: 10.1016/j.seppur.2022.121058.

Gardoni, Davide, A. Vailati, and Roberto Canziani. 2012. “Decay of Ozone in Water: A Review.” *Ozone Science and Engineering* 34:233. doi: 10.1080/01919512.2012.686354.

Geetha Varma, V., Swetti Jha, L. Himesh Karthik Raju, R. Lalith Kishore, and V. Ranjith. 2022. “A Review on Decentralized Wastewater Treatment Systems in India.” *Chemosphere* 300:134462. doi: 10.1016/j.chemosphere.2022.134462.

Gurreri, Luigi, Alessandro Tamburini, Andrea Cipollina, and Giorgio Micale. 2020. “Electrodialysis Applications in Wastewater Treatment for Environmental Protection and Resources Recovery: A Systematic Review on Progress and Perspectives.” *Membranes* 10(7):146. doi: 10.3390/membranes10070146.

Hakizimana, Jean Nepo, Bouchaib Gourich, Mohammed Chafi, Youssef Stiriba, Christophe Vial, Patrick Drogui, and Jamal Naja. 2017. “Electrocoagulation Process in Water Treatment: A Review of Electrocoagulation Modeling Approaches.” *Desalination* 404:1–21. doi: 10.1016/j.desal.2016.10.011.

Kinidi, Lennevey, Ivy Ai Wei Tan, Noraziah Binti Abdul Wahab, Khairul Fikri Bin Tamrin, Cirilo Nolasco Hipolito, and Shanti Faridah Salleh. 2018. “Recent Development in Ammonia Stripping Process for Industrial Wastewater Treatment.” *International Journal of Chemical Engineering* 2018(1):3181087. doi: 10.1155/2018/3181087.

Liu, Junguo, Hong Yang, Simon N. Gosling, Matti Kummu, Martina Flörke, Stephan Pfister, Naota Hanasaki, Yoshihide Wada, Xinxin Zhang, Chunmiao Zheng, Joseph Alcamo, and Taikan Oki. 2017. “Water Scarcity Assessments in the Past, Present, and Future.” *Earth’s Future* 5(6):545–59. doi: 10.1002/2016EF000518.

Massoud, May A., Akram Tarhini, and Joumana A. Nasr. 2009. “Decentralized Approaches to Wastewater Treatment and Management: Applicability in Developing Countries.” *Journal of Environmental Management* 90(1):652–59. doi: 10.1016/j.jenvman.2008.07.001.

Meng, Jizhong, Xiaoxiao Shi, Shun Wang, Zhenhu Hu, Derya Y. Koseoglu-Imer, Piet N. L. Lens, and Xinmin Zhan. 2024. “Application of Electrodialysis Technology in Nutrient Recovery from Wastewater: A Review.” *Journal of Water Process Engineering* 65:105855. doi: 10.1016/j.jwpe.2024.105855.

Mohammadi, Rubaba, Walter Tang, and Mika Sillanpää. 2021. “A Systematic Review and Statistical Analysis of Nutrient Recovery from Municipal Wastewater by Electrodialysis.” *Desalination* 498:114626. doi: 10.1016/j.desal.2020.114626.

Mollah, M. Yousuf A., Robert Schennach, Jose R. Parga, and David L. Cocke. 2001. “Electrocoagulation (EC) — Science and Applications.” *Journal of Hazardous Materials* 84(1):29–41. doi: 10.1016/S0304-3894(01)00176-5.

Muzioreva, Happison, Trynos Gumbo, Neema Kavishe, Thembani Moyo, and Innocent Musonda. 2022. “Decentralized Wastewater System Practices in Developing Countries: A Systematic Review.” *Utilities Policy* 79:101442. doi: 10.1016/j.jup.2022.101442.

Owusu-Ansah, Emmanuel De-Graft Johnson, Angelina Sampson, Samuel K. Amponsah, Robert C. Abaidoo, and Tine Hald. 2015. "Performance, Compliance and Reliability of Waste Stabilization Pond: Effluent Discharge Quality and Environmental Protection Agency Standards in Ghana." *Research Journal of Applied Sciences, Engineering and Technology* 10(11):1293–1302. doi: 10.19026/rjaset.10.1825.

Robles, Ángel, Daniel Aguado, Ramón Barat, Luis Borrás, Alberto Bouzas, Juan Bautista Giménez, Nuria Martí, Josep Ribes, María Victoria Ruano, Joaquín Serralta, José Ferrer, and Aurora Seco. 2020a. "New Frontiers from Removal to Recycling of Nitrogen and Phosphorus from Wastewater in the Circular Economy." *Bioresource Technology* 300:122673. doi: 10.1016/j.biortech.2019.122673.

Robles, Ángel, Daniel Aguado, Ramón Barat, Luis Borrás, Alberto Bouzas, Juan Bautista Giménez, Nuria Martí, Josep Ribes, María Victoria Ruano, Joaquín Serralta, José Ferrer, and Aurora Seco. 2020b. "New Frontiers from Removal to Recycling of Nitrogen and Phosphorus from Wastewater in the Circular Economy." *Bioresource Technology* 300:122673. doi: 10.1016/j.biortech.2019.122673.

Shahedi, A., A. K. Darban, F. Taghipour, and A. Jamshidi-Zanjani. 2020. "A Review on Industrial Wastewater Treatment via Electrocoagulation Processes." *Current Opinion in Electrochemistry* 22:154–69. doi: 10.1016/j.coelec.2020.05.009.

Smoczynskia, Lech, Siawomir Kalinowskia, Harsha Ratnaweerab, Marta Kosobuckaa, Mihaela Trifescua, and Krystyna Pieczulis-Smoczynskaa. 2017. "Electrocoagulation of Municipal Wastewater – a Pilot-Scale Test." *Desalination and Water Treatment* 72:162–68. doi: 10.5004/dwt.2017.20654.

Tahreen, Amina, Mohammed Saedi Jami, and Fathilah Ali. 2020. "Role of Electrocoagulation in Wastewater Treatment: A Developmental Review." *Journal of Water Process Engineering* 37:101440. doi: 10.1016/j.jwpe.2020.101440.

Thomas, Benjamin D., Annaliese Marks, Blake Smerigan, Gustavo Aburto-Vazquez, Sibel Uludag-Demirer, James S. Dusenbury, and Wei Liao. 2024. "Life Cycle Impact and Economic Assessment of Decentralized Strategies to Treat Source-Separated Wastewater." *Journal of Water Process Engineering* 64:105550. doi: 10.1016/j.jwpe.2024.105550.

Tzanakakis, Vasileios A., Nikolaos V. Paranychianakis, and Andreas N. Angelakis. 2020. "Water Supply and Water Scarcity." *Water* 12(9):2347. doi: 10.3390/w12092347.

Villaseñor-Basulto, Deborah L., Alain Picos-Benítez, Martin Pacheco-Alvarez, Tzayam Pérez, Erick R. Bandala, and Juan M. Peralta-Hernández. 2022. "Tannery Wastewater Treatment Using Combined Electrocoagulation and Electro-Fenton Processes." *Journal of Environmental Chemical Engineering* 10(2):107290. doi: 10.1016/j.jece.2022.107290.

Ward, Andrew J., Kimmo Arola, Emma Thompson Brewster, Chirag M. Mehta, and Damien J. Batstone. 2018. "Nutrient Recovery from Wastewater through Pilot Scale Electrodialysis." *Water Research* 135:57–65. doi: 10.1016/j.watres.2018.02.021.

Zhang, Xiaoyuan, and Yu Liu. 2021. “Circular Economy-Driven Ammonium Recovery from Municipal Wastewater: State of the Art, Challenges and Solutions Forward.” *Bioresource Technology* 334:125231. doi: 10.1016/j.biortech.2021.125231.

# APPENDIX A: INLINE OZONE TREATMENT OF ELECTROCOAGULATION

## Materials and Methods

Initial ozone testing was performed in a 3L PVC reactor. Ozone was supplied at 8g/hr by a MP-8000 Light Duty Water Ozone Generator from A2Z Ozone Inc. The ozone unit was connected via PTFE tubing and was bubbled into the reactor using an Underwater Treasures Cylindrical Airstone. Testing was first performed on port-o-john water. Various levels of ferric chloride were added to represent the iron released during electrocoagulation. The COD, NH3, and pH were measured to determine the effect that ozone had on EC treated blackwater.

Pilot scale testing was performed with ozonation at various concentrations as well as locations of implementation. The ozone units used were both from A2Z Ozone Inc being the MP-8000 as well as a SP-Series Swimming Pool Ozone generator with output up to 16g/hr. The ozone was implemented inline of the EC reactor, in the dilute tank, and in both locations simultaneously. The same tubing and air stones as the lab-scale tests were used. The system was operated with the same parameters as the latrine testing.

## Initial Tests

An initial test with no added ferric chloride showed ozone implementation increased the pH of the blackwater gradually overtime as seen in Figure 18. Once $FeCl_3$ was added, an initial increase in pH was observed followed by a stagnation a sharp decline in pH. The rate at which the pH changed was found to depend on the concentration of ferric chloride present in the solution with a shorter stagnation and steeper change being observed at higher concentrations of ferric chloride.

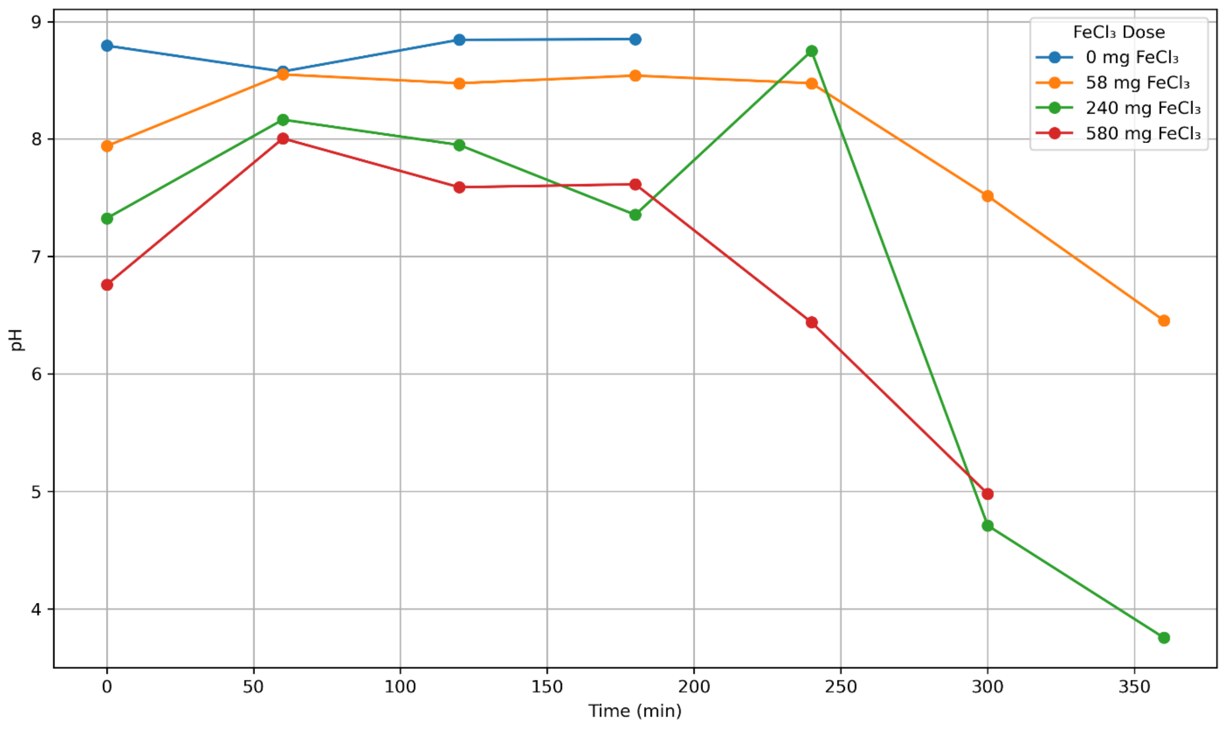


**Figure 18:** Change in pH over time for various levels of ferric chloride added blackwater.

When comparing the 8, 16, and 25 g/hr ozone levels, no significant decrease in pH is observed after EC or ED as seen in Figure 19. Additionally, no definitive trend was observed for the concentrations of COD or $NH_3$-N either as seen in

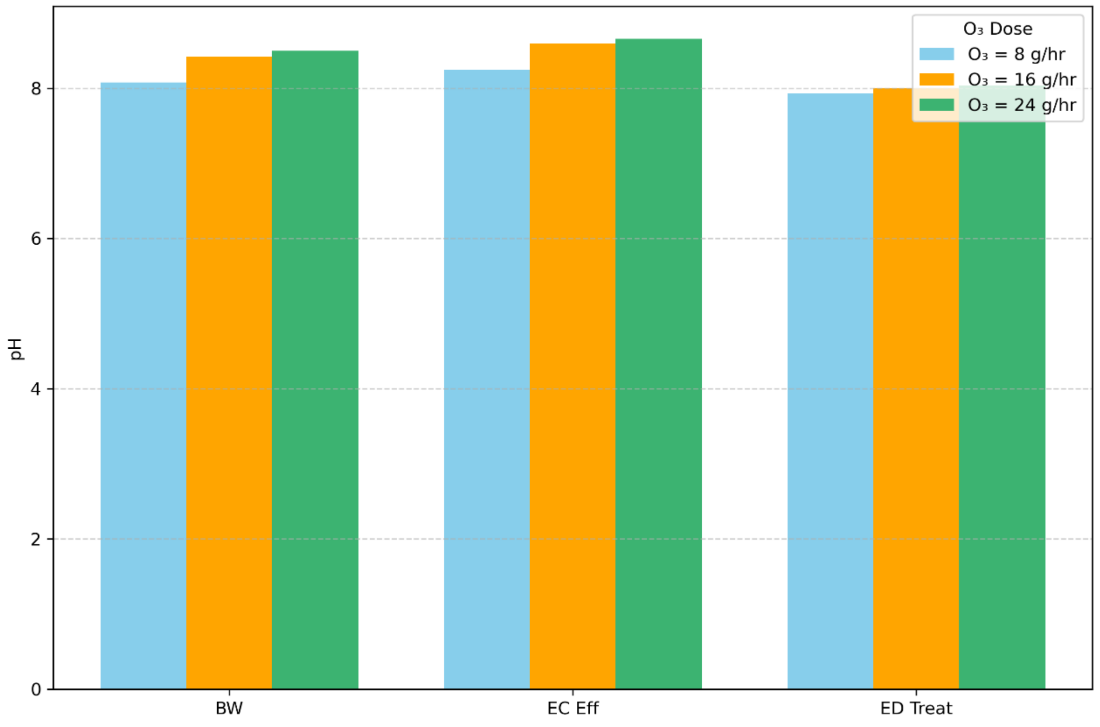


**Figure 19:** pH at each process unit for different ozone doses.

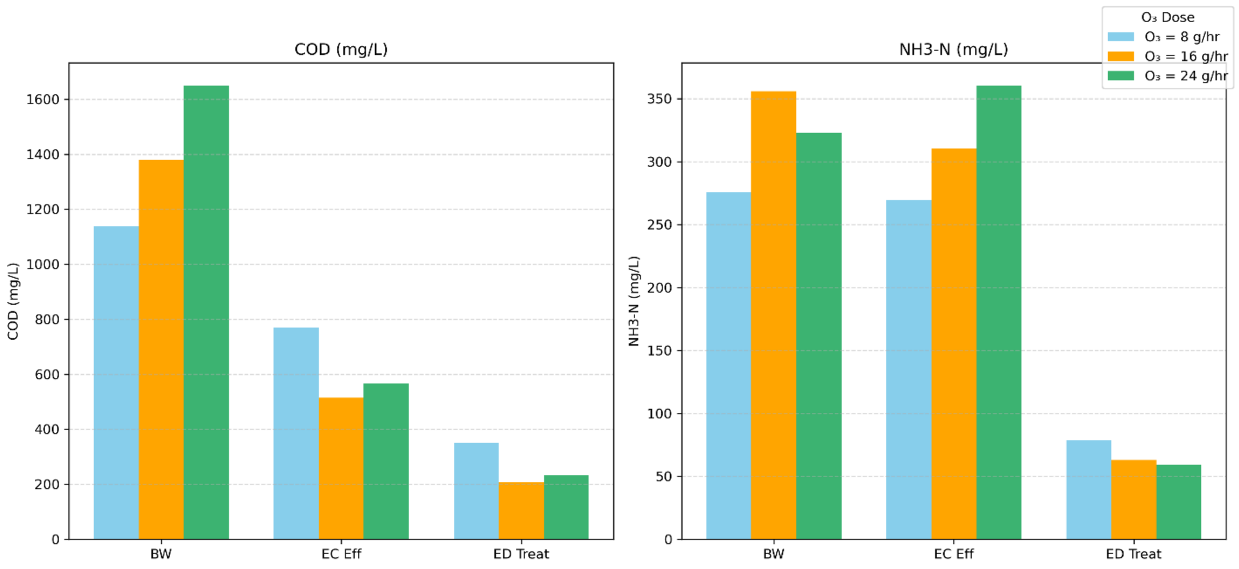


**Figure 20:** COD and $NH_3$-N concentrations for different ozone doses.

# APPENDIX B: PYTHON CODE

## 75 L/hr testing to determine EDR viability

In [2]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.multivariate.manova import MANOVA
from scipy import stats
```

In [3]:

```
ED = pd.read_excel('MetaDataSheet v1.xlsx', sheet_name = 2, index_col = 'Run Number', na_values = 0, usecols
= 'A:J')
ED['label'] = 'ED'
```

In [4]:

```
EDR = pd.read_excel('MetaDataSheet v1.xlsx', sheet_name = 3, index_col = 'Run Number', na_values = 0,
usecols = 'A:J')
EDR['label'] = 'EDR'
```

In [5]:

```
#Create subplot
fig, axs = plt.subplots(3, 1, sharex = True, figsize = (10,8))

axs = axs.flatten()

# Create each individual plot
axs[2].plot(ED.index, ED['Energy per batch (Whr/L)'],'bo')
axs[2].plot(EDR.index, EDR['Energy per batch (Whr/L)'],'ro')
axs[2].set_ylabel('Energy Consumption (Wh/L)')
axs[2].set_title('(c)', loc = 'left')

axs[0].plot(ED.index, ED['DE'],'bo')
axs[0].plot(EDR.index, EDR['DE'],'ro')
axs[0].set_ylabel('Dilute Concentration (mgNH$_3$-N/L)')
axs[0].set_title('(a)', loc = 'left')

axs[1].plot(ED.index, ED['CE'],'bo')
axs[1].plot(EDR.index, EDR['CE'],'ro')
axs[1].set_ylabel('Concentrate Concentration (mgNH$_3$-N/L)')
axs[1].set_title('(b)', loc = 'left')

#create legends
legendED = 'Electrodialysis'
legendEDR = 'Electrodialysis Reversal'

for i in range(3):
    axs[i].legend([legendED, legendEDR])

# Create shared x and y labels
fig.text(0.5,0,'Run Number', ha='center', va = 'center')
```

```
plt.tight_layout()

fig.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/Comparison of Electrodialysis and
Electrodialysis Reversal', bbox_inches="tight")
```

## Statistical Analysis

In [7]:

```
data = pd.concat([ED, EDR], axis = 0)
data = data.rename(columns = {'Energy per batch (Whr/L)': 'Energy', 'Total Time (mins)': 'Time'})
```

In [8]:

```
print('Max ED CE:', ED['CE'].max())
print('Mean ED DE:', ED['DE'].mean(), '+-', ED['DE'].std())
print('Max EDR CE:', EDR['CE'].max())
print('Mean EDR DE:', EDR['DE'].mean(), '+-', EDR['DE'].std())
print('Mean ED Energy Consumption:', ED['Energy per batch (Whr/L)'].mean(), '+-', ED['Energy per batch
(Whr/L)'].std())
print('Mean EDR Energy Consumption:', EDR['Energy per batch (Whr/L)'].mean(), '+-', EDR['Energy per
batch (Whr/L)'].std())
Max ED CE: 8310.0
Mean ED DE: 19.489399999999996 +- 8.484597960001235
Max EDR CE: 7094.0
Mean EDR DE: 24.106666666666666 +- 19.293626507464165
Mean ED Energy Consumption: 1.2174011266 +- 0.14498314668939563
Mean EDR Energy Consumption: 1.1412789934666667 +- 0.15613209084387208
```

In [9]:

```
# Individual Anova analysis

# Dilute end
model = smf.ols('DE ~ C(label)', data = data).fit()
anova_dilute = sm.stats.anova_lm(model,typ=2)

# Concentrate end
model = smf.ols('CE ~ C(label)', data = data).fit()
anova_conc = sm.stats.anova_lm(model,typ=2)

# Time
model = smf.ols('Time ~ C(label)', data = data).fit()
anova_time = sm.stats.anova_lm(model,typ=2)

# Energy
model = smf.ols('Energy ~ C(label)', data = data).fit()
anova_energy = sm.stats.anova_lm(model,typ=2)

# Results
print('Dilute')
print(anova_dilute)
print("\n")
print('Conc')
print(anova_conc)
print("\n")
print('Time')
print(anova_time)
print("\n")
print('Energy')
print(anova_energy)
Dilute
           sum_sq    df        F    PR(>F)
C(label)   159.893636   1.0  0.719865  0.403384
Residual  6219.253969  28.0       NaN       NaN


Conc
           sum_sq    df        F    PR(>F)
C(label)  3.186715e+06   1.0  0.717544  0.404133
Residual  1.243520e+08  28.0       NaN       NaN


Time
          sum_sq    df      F    PR(>F)
C(label)   217.89075   1.0  2.4289  0.130349
Residual  2511.81232  28.0     NaN       NaN


Energy
         sum_sq    df        F    PR(>F)
C(label)  0.043459   1.0  1.914621  0.177383
Residual  0.635563  28.0       NaN       NaN
```

In [10]:

```
t_stat, p_value = stats.ttest_ind(ED['DE'], EDR['DE'])

print(f"T-statistic: {t_stat:.4f}, P-value: {p_value}")
T-statistic: -0.8484, P-value: 0.40338359989660777
```

In [11]:

```
t_stat, p_value = stats.ttest_ind(ED['CE'], EDR['CE'])

print(f"T-statistic: {t_stat:.4f}, P-value: {p_value}")
T-statistic: 0.8471, P-value: 0.40413318368032325
```

In [12]:

```
t_stat, p_value = stats.ttest_ind(ED['Total Time (mins)'], EDR['Total Time (mins)'])

print(f"T-statistic: {t_stat:.4f}, P-value: {p_value}")
T-statistic: -1.5585, P-value: 0.13034890250331604
```

In [13]:

```
t_stat, p_value = stats.ttest_ind(ED['Energy per batch (Whr/L)'], EDR['Energy per batch (Whr/L)'])

print(f"T-statistic: {t_stat:.4f}, P-value: {p_value}")
T-statistic: 1.3837, P-value: 0.1773834615296145
```

## Initial testing of the EC system with Sewage Sludge

In [2]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

```
from scipy import stats
                                                                                                    In [3]:
Blackwater = pd.read_excel('MetaDataSheet v1.xlsx', sheet_name = 6, index_col = 'Date', na_values = ('nm',0),
usecols = 'A:M')
Blackwater['Current (A)'] = 36
Blackwater['Current (A)'][7:] = 30
In [4]:
ECTreated = pd.read_excel('MetaDataSheet v2.xlsx', sheet_name = 7, index_col = 'Date', na_values = ('nm',0))
ECTreated['Current (A)'] = 36
ECTreated['Current (A)'][7:] = 30

In [5]:
ECFiltered = pd.read_excel('MetaDataSheet v2.xlsx', sheet_name = 8, index_col = 'Date', na_values = ('nm',0,
'undermeasuring range', 'neg value'))
ECFiltered = ECFiltered.drop('2024-04-22')
ECFiltered['Current (A)'] = 36
ECFiltered['Current (A)'][5:] = 30

In [6]:
EDTreated = pd.read_excel('MetaDataSheet v1.xlsx', sheet_name = 9, index_col = 'Date', na_values = ('nm',0))
EDTreated['Current (A)'] = 36
EDTreated['Current (A)'][5:] = 30
In [7]:
EDConcentrate = pd.read_excel('MetaDataSheet v1.xlsx', sheet_name = 10, index_col = 'Date', na_values =
('nm',0))
EDConcentrate['Current (A)'] = 36
EDConcentrate['Current (A)'][5:] = 30
In [8]:
# Create Dataframes
parameters = ['COD (mg/L)', 'TP (mg/L)', 'TN (mg/L)', 'NH3-N (mg/L)', 'TSS (mg/L)', 'pH']
waters = ['Blackwater', 'EC Treated', 'EC Filtered', 'ED Treated']

aveBlackwater = []
aveECTreated = []
aveECFiltered = []
aveEDTreated = []

stdBlackwater = []
stdECTreated = []
stdECFiltered = []
stdEDTreated = []

for parameter in parameters:
  aveBlackwater.append(Blackwater[parameter].mean())
  aveECTreated.append(ECTreated[parameter].mean())
  aveECFiltered.append(ECFiltered[parameter].mean())
  aveEDTreated.append(EDTreated[parameter].mean())

  stdBlackwater.append(Blackwater[parameter].std())
  stdECTreated.append(ECTreated[parameter].std())
  stdECFiltered.append(ECFiltered[parameter].std())
  stdEDTreated.append(EDTreated[parameter].std())


Averages = {
  'Blackwater': aveBlackwater,
  'EC Treated': aveECTreated,
  'EC Filtered': aveECFiltered,
  'ED Treated': aveEDTreated
}

# Create a dictionary to store the formatted values (Mean ± SD)
summary_data = {
  'Parameter': parameters,
  'Blackwater': [f"{mean:.1f} ± {std:.2f}" for mean, std in zip(aveBlackwater, stdBlackwater)],
  'EC Treated': [f"{mean:.1f} ± {std:.2f}" for mean, std in zip(aveECTreated, stdECTreated)],
  'EC Filtered': [f"{mean:.1f} ± {std:.2f}" for mean, std in zip(aveECFiltered, stdECFiltered)],
  'ED Treated': [f"{mean:.1f} ± {std:.2f}" for mean, std in zip(aveEDTreated, stdEDTreated)]
}

# Convert dictionary to DataFrame
summary_df = pd.DataFrame(summary_data)

# Replace "NH3-N (mg/L)" with subscripted version in the Parameter column
summary_df_latex = summary_df.copy()
summary_df_latex['Parameter'] = summary_df_latex['Parameter'].replace({
  'NH3-N (mg/L)': r'NH$_3$-N (mg/L)'
})

# Set figure size based on number of rows/columns
fig_height = 1 + 0.5 * len(summary_df_latex)
fig_width = 2 + 1.5 * (len(summary_df_latex.columns) - 1)  # minus 1 because 'Parameter' becomes index

# Set 'Parameter' as index again
table_df = summary_df_latex.set_index("Parameter")

# Plot the table
fig, ax = plt.subplots(figsize=(fig_width, fig_height))
ax.axis('off')
ax.axis('tight')

table = ax.table(
  cellText=table_df.values,
  rowLabels=table_df.index,
  colLabels=table_df.columns,  # Exclude 'Parameter'
  cellLoc='center',
  rowLoc='center',
  loc='center'
)

# Format appearance
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.2)

# Save figure
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis Figures/summary_table_flipped.png", dpi=300,
bbox_inches='tight')
plt.show()
```

```
In [9]:
# Create Dataframes split based on current setpoint

BW36 = Blackwater[Blackwater['Current (A)'] == 36]
BW30 = Blackwater[Blackwater['Current (A)'] == 30]
ECT36 = ECTreated[ECTreated['Current (A)'] == 36]
ECT30 = ECTreated[ECTreated['Current (A)'] == 30]
ECF36 = ECFiltered[ECFiltered['Current (A)'] == 36]
ECF30 = ECFiltered[ECFiltered['Current (A)'] == 30]
EDT36 = EDTreated[EDTreated['Current (A)'] == 36]
EDT30 = EDTreated[EDTreated['Current (A)'] == 30]

parameters = ['COD (mg/L)', 'TP (mg/L)', 'TN (mg/L)', 'NH3-N (mg/L)', 'TSS (mg/L)', 'pH']
waterssplit = ['BW36','BW30', 'ECT36', 'ECT30', 'ECF36', 'ECF30', 'EDT36','EDT30']

# Create mapping of names to DataFrames
waters_dict = {
    'Blackwater 36A': BW36,
    'Blackwater 30A': BW30,
    'EC Treated 36A': ECT36,
    'EC Treated 30A': ECT30,
    'EC Filtered 36A': ECF36,
    'EC Filtered 30A': ECF30,
    'ED Treated 36A': EDT36,
    'ED Treated 30A': EDT30
}

# Store results in a dictionary
summary_data = {}

for water_name, df in waters_dict.items():
    values = []
    for parameter in parameters:
        mean_val = df[parameter].mean()
        std_val = df[parameter].std()
        values.append(f"{mean_val:.1f} ± {std_val:.2f}")
    summary_data[water_name] = values

# Convert dictionary to DataFrame
split_df = pd.DataFrame(summary_data, index=parameters)

split_df_flipped = split_df.transpose()

# Replace column name with subscripted NH3
split_df_flipped.columns = [
    r'COD (mg/L)',
    r'TP (mg/L)',
    r'TN (mg/L)',
    r'NH$_3$-N (mg/L)',
    r'TSS (mg/L)',
    r'pH'
]

# Set figure size based on flipped dimensions
fig_height = 1 + 0.5 * len(split_df_flipped)
```

```
fig_width = 2 + 1.5 * len(split_df_flipped.columns)

fig, ax = plt.subplots(figsize=(fig_width, fig_height))
ax.axis('off')
ax.axis('tight')

# Create the table
table = ax.table(
    cellText=split_df_flipped.values,
    rowLabels=split_df_flipped.index,
    colLabels=split_df_flipped.columns,
    cellLoc='center',
    rowLoc='center',
    loc='center'
)

# Style
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.2)

# Save flipped version
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis
Figures/parameter_summary_by_current_FLIPPED.png", dpi=300, bbox_inches='tight')
plt.show()

In [10]:
# Define waters and parameters to compare
waters = ['Blackwater', 'EC Treated', 'EC Filtered', 'ED Treated']
parameters = ['COD (mg/L)', 'TP (mg/L)', 'NH3-N (mg/L)', 'pH']

# Map water names to their 30A and 36A DataFrames
water_data = {
    'Blackwater': (BW30, BW36),
    'EC Treated': (ECT30, ECT36),
    'EC Filtered': (ECF30, ECF36),
    'ED Treated': (EDT30, EDT36)
}

# Set up plot
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
axes = axes.flatten()
colors = ['#4C72B0', '#55A868']  # 30A and 36A

for i, param in enumerate(parameters):
    ax = axes[i]
    means_30, stds_30 = [], []
    means_36, stds_36 = [], []

    for key in waters:
        df30, df36 = water_data[key]
        means_30.append(df30[param].mean())
        stds_30.append(df30[param].std())
        means_36.append(df36[param].mean())
        stds_36.append(df36[param].std())
    x = np.arange(len(waters))
    bar_width = 0.35

    # Plot with error bars
    ax.bar(x - bar_width/2, means_30, bar_width, yerr=stds_30, capsize=4, label='30 A', color=colors[0])
    ax.bar(x + bar_width/2, means_36, bar_width, yerr=stds_36, capsize=4, label='36 A', color=colors[1])

    # LaTeX format for NH3 as a subscript
    if param == 'NH3-N (mg/L)':
        param_title = r'NH$_3$-N (mg/L)'
    else:
        param_title = param

    ax.set_title(f"({chr(97 + i)})", loc='left', fontsize=12)  # a=97 in ASCII
    ax.set_xticks(x)
    ax.set_xticklabels(waters)
    ax.set_ylabel(param_title)
    ax.legend()


# Tidy layout
plt.tight_layout()
plt.subplots_adjust(top=0.90)

fig.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/Initial EC Current Comp',
bbox_inches="tight")

In [11]:# Create Dataframes
parameters = ['COD (mg/L)', 'TP (mg/L)', 'NH3-N (mg/L)', 'TSS (mg/L)']
waters = ['Blackwater', 'EC Treated', 'EC Filtered', 'ED Treated']

aveBlackwater = []
aveECTreated = []
aveECFiltered = []
aveEDTreated = []

stdBlackwater = []
stdECTreated = []
stdECFiltered = []
stdEDTreated = []

for parameter in parameters:
    aveBlackwater.append(Blackwater[parameter].mean())
    aveECTreated.append(ECTreated[parameter].mean())
    aveECFiltered.append(ECFiltered[parameter].mean())
    aveEDTreated.append(EDTreated[parameter].mean())

    stdBlackwater.append(Blackwater[parameter].std())
    stdECTreated.append(ECTreated[parameter].std())
    stdECFiltered.append(ECFiltered[parameter].std())
    stdEDTreated.append(EDTreated[parameter].std())


Averages = {
    'Blackwater': aveBlackwater,
    'EC Treated': aveECTreated,
    'EC Filtered': aveECFiltered,
    'ED Treated': aveEDTreated
}
MeanTable = pd.DataFrame({'Parameter': parameters, 'Blackwater': aveBlackwater, 'EC Treated':
aveECTreated, 'EC Filtered': aveECFiltered, 'ED Treated': aveEDTreated})
MeanTable = MeanTable.set_index('Parameter')
MeanTable = MeanTable.transpose()


StdTable = pd.DataFrame({'Parameter': parameters, 'Blackwater': stdBlackwater, 'EC Treated': stdECTreated,
'EC Filtered': stdECFiltered, 'ED Treated': stdEDTreated})
StdTable = StdTable.set_index('Parameter')
StdTable = StdTable.transpose()

# Create Plot for average parameters
colors = ['blue', 'orange','green','red']

fig, ax = plt.subplots(2,2, figsize = (8,5))

ax[0, 0].bar(waters, MeanTable['COD (mg/L)'], yerr=StdTable['COD (mg/L)'], color = colors, capsize=5,
alpha=0.8)
ax[0, 0].set_title("(a)", loc = 'left')
ax[0, 0].text(3.1, 2700, 'COD', weight = 'bold')

ax[0, 1].bar(waters, MeanTable['TP (mg/L)'], yerr=StdTable['TP (mg/L)'], color = colors, capsize=5,
alpha=0.8)
ax[0, 1].set_title("(b)", loc = 'left')
ax[0, 1].text(3.25, 63, 'TP', weight = 'bold')

ax[1, 0].bar(waters, MeanTable['NH3-N (mg/L)'], yerr=StdTable['NH3-N (mg/L)'], color = colors, capsize=5,
alpha=0.8)
ax[1, 0].set_title("(c)", loc = 'left')
ax[1, 0].text(2.9, 170, 'NH$_3$-N', weight = 'bold')

ax[1, 1].bar(waters, MeanTable['TSS (mg/L)'], yerr=StdTable['TSS (mg/L)'], color = colors, capsize=5,
alpha=0.8)
ax[1, 1].set_title("(d)", loc = 'left')
ax[1, 1].text(3.15, 1400, 'TSS', weight = 'bold')

# Add some text for labels, title and custom x-axis tick labels, etc.
fig.supylabel('Concentration (mg/L)')
plt.tight_layout()

fig.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/Initial EC Parameters',
bbox_inches="tight")
```

**Statistical Analysis**

In [13]:

```
Blackwater['label'] = 'Blackwater'
ECTreated['label'] = 'ECTreated'
ECFiltered['label'] = 'ECFiltered'
EDTreated['label'] = 'EDTreated'
```

```
data = pd.concat([Blackwater, ECTreated, ECFiltered, EDTreated], axis=0)
data = data.rename(columns = {'COD (mg/L)': 'COD', 'TP (mg/L)': 'TP', 'NH3-N (mg/L)': 'NH3'})
In [14]:
# Statistical Analysis

data_BW_EC = data[data['label'].isin(['Blackwater', 'ECTreated'])]
data_BW_ECF = data[data['label'].isin(['Blackwater', 'ECFiltered'])]
data_BW_ED = data[data['label'].isin(['Blackwater', 'EDTreated'])]
data_EC_ECF = data[data['label'].isin(['ECTreated', 'ECFiltered'])]
data_EC_ED = data[data['label'].isin(['ECTreated', 'EDTreated'])]
data_ECF_ED = data[data['label'].isin(['ECFiltered', 'EDTreated'])]

data_frames = [data_BW_EC, data_BW_ECF, data_BW_ED, data_EC_ECF, data_EC_ED, data_ECF_ED]
Variables = ['COD', 'TP', 'NH3', 'pH']
anovas = []

# Power ANOVA Analysis
for variable in Variables:
    for frame in data_frames:
        model = smf.ols(f'{variable} ~ C(label)', data = frame).fit()
        anova = sm.stats.anova_lm(model,typ=2)
        anovas.append(f'Variable: {variable}, Waters: {frame['label'].unique()}')
        anovas.append(anova)
        anovas.append(' ')
In [15]:
parameters = ['COD (mg/L)', 'TP (mg/L)', 'NH3-N (mg/L)', 'TSS (mg/L)']
Variables = ['COD', 'TP', 'NH$_3$-N', 'TSS']
reduction = []
std = []

for param in parameters:
    for i in range(len(MeanTable)-1):
        per = (MeanTable[param][i] - MeanTable[param][i+1]) / MeanTable[param][i]*100
        reduction.append(per)
        prop = np.abs(1/MeanTable[param][i]) * ( (StdTable[param][i])**2 + (StdTable[param][i+1])**2)**0.5 *
100
        std.append(prop)

# Generate data
var, w = [], []
for v in Variables:
    for i in range(len(waters) - 1):  # Pairwise transitions
        var.append(v)
        w.append(f'{waters[i]} → {waters[i+1]}')

# Create DataFrame
df = pd.DataFrame({
    'Parameters': var,
    'Waters': w,
    '% Reduction': reduction,
    '% Error': std
})

# Pivot the table to restructure it
df_pivot = df.pivot(index='Waters', columns="Parameters", values=["% Reduction", "% Error"])
```

```
# Flatten the multi-index column names
df_pivot.columns = [f"{metric} {param}" for metric, param in df_pivot.columns]

# Combine % Reduction and % Error into a single formatted column
for col in Variables:
    df_pivot[col] = df_pivot[f'% Reduction {col}'].apply(lambda x: f'{x:.1f}') + " ± " + df_pivot[f'% Error
{col}'].apply(lambda x: f'{x:.1f}')

# Drop the separate % Error and % Reduction columns
df_pivot = df_pivot[Variables]

# Calculate total reduction (assuming we have the actual initial and final values)
initial_values = {"COD": MeanTable['COD (mg/L)'][0], "TP": MeanTable['TP (mg/L)'][0], "NH$_3$-N":
MeanTable['NH3-N (mg/L)'][0], "TSS": MeanTable['TSS (mg/L)'][0]}  # Replace with real initial values
final_values = {"COD": MeanTable['COD (mg/L)'][3], "TP": MeanTable['TP (mg/L)'][3], "NH$_3$-N":
MeanTable['NH3-N (mg/L)'][3], "TSS": MeanTable['TSS (mg/L)'][3]}    # Replace with real final values
initial_errors = {"COD": StdTable['COD (mg/L)'][0], "TP": StdTable['TP (mg/L)'][0], "NH$_3$-N":
StdTable['NH3-N (mg/L)'][0], "TSS": StdTable['TSS (mg/L)'][0]}     # Replace with real error values
final_errors = {"COD": StdTable['COD (mg/L)'][3], "TP": StdTable['TP (mg/L)'][3], "NH$_3$-N":
StdTable['NH3-N (mg/L)'][3], "TSS": StdTable['TSS (mg/L)'][3]}      # Replace with real error values

total_reduction = {}
total_error = {}

for param in Variables:
    # Compute total reduction
    reduction_val = ((initial_values[param] - final_values[param]) / initial_values[param]) * 100
    total_reduction[param] = f"{reduction_val:.1f}"

    # Error propagation formula for percent reduction:
    error_val = reduction_val * np.sqrt(
        (initial_errors[param] / initial_values[param])**2 +
        (final_errors[param] / final_values[param])**2
    )
    total_error[param] = f"{error_val:.1f}"

# Format total reduction row
total_row = {var: f"{total_reduction[var]} ± {total_error[var]}" for var in Variables}


# Append total row to DataFrame
df_pivot = pd.concat([df_pivot, pd.DataFrame([total_row])])
df_pivot.rename(index={0: "Total"}, inplace=True)

# Example: New desired order of the rows
new_order = ['Blackwater → EC Treated', 'EC Treated → EC Filtered', 'EC Filtered → ED Treated', 'Total']

df_pivot.columns = [
    'COD (%)', 'TP (%)', r'NH$_3$ (%)', 'TSS (%)'
]
# Reorder the DataFrame rows
df_pivot = df_pivot.reindex(new_order)

# Set figure size based on the number of rows
```

```
fig_height = 1 + 0.5 * len(df_pivot)
fig, ax = plt.subplots(figsize=(8, fig_height))

# Turn off axes
ax.axis('off')
ax.axis('tight')

# Create the table figure
table = ax.table(
    cellText=df_pivot.values,
    rowLabels=df_pivot.index,
    colLabels=df_pivot.columns,
    cellLoc='center',
    loc='center'
)

# Optional formatting
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.2)  # width, height scaling

# Save figure as high-resolution PNG
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis Figures/removal_efficiency_summary.png",
dpi=300, bbox_inches='tight')
plt.show()

In [16]:
waters = ['Blackwater', 'EC Treated', 'EC Filtered', 'ED Treated']
parameters = ['COD (mg/L)', 'TP (mg/L)', 'NH3-N (mg/L)', 'TSS (mg/L)', 'pH']

# Map water names to their 30A and 36A DataFrames
water_data = {
    'Blackwater': (BW30, BW36),
    'EC Treated': (ECT30, ECT36),
    'EC Filtered': (ECF30, ECF36),
    'ED Treated': (EDT30, EDT36)
}

# Store results
ttest_result = []

for water in waters:
    df30, df36 = water_data[water]
    for param in parameters:
        group30 = df30[param].dropna()
        group36 = df36[param].dropna()

        t_stat, p_value = stats.ttest_ind(group30, group36, equal_var=False)  # Welch's t-test
        ttest_result.append({
            'Water': water,
            'Parameter': param,
            'T-statistic': round(t_stat, 4),
            'P-value': round(p_value, 4)
        })
```

```
# Convert to DataFrame
ttest_df = pd.DataFrame(ttest_result)

# Add significance flag
ttest_df["Significant (p < 0.05)"] = ttest_df["P-value"] < 0.05

# Set figure size based on content
fig_height = 1 + 0.4 * len(ttest_df)
fig_width = 2 + 1.5 * len(ttest_df.columns)

# Create figure and axis
fig, ax = plt.subplots(figsize=(fig_width, fig_height))
ax.axis('off')
ax.axis('tight')

# Create the table
table = ax.table(
   cellText=ttest_df.values,
   colLabels=ttest_df.columns,
   cellLoc='center',
   loc='center'
)

# Format the table
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.2)

# Save the figure
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis Figures/ttest_table_results.png", dpi=300,
bbox_inches='tight')
plt.show()

In [17]:
t_stat, p_value = stats.ttest_ind(group30, group36, equal_var=False)  # Welch's t-test
```

**75 L/hr testing to determine EDR viability**

```
In [2]:
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.multivariate.manova import MANOVA
from scipy import stats
In [3]:
ED = pd.read_excel('MetaDataSheet v1.xlsx', sheet_name = 2, index_col = 'Run Number',
na_values = 0, usecols = 'A:J')
ED['label'] = 'ED'
In [4]:
EDR = pd.read_excel('MetaDataSheet v1.xlsx', sheet_name = 3, index_col = 'Run Number',
na_values = 0, usecols = 'A:J')
```

```
EDR['label'] = 'EDR'
In [5]:
#Create subplot
fig, axs = plt.subplots(3, 1, sharex = True, figsize = (10,8))

axs = axs.flatten()

# Create each individual plot
axs[2].plot(ED.index, ED['Energy per batch (Whr/L)'],'bo')
axs[2].plot(EDR.index, EDR['Energy per batch (Whr/L)'],'ro')
axs[2].set_ylabel('Energy Consumption (Wh/L)')
axs[2].set_title('(c)', loc = 'left')

axs[0].plot(ED.index, ED['DE'],'bo')
axs[0].plot(EDR.index, EDR['DE'],'ro')
axs[0].set_ylabel('Dilute Concentration (mgNH$_3$-N/L)')
axs[0].set_title('(a)', loc = 'left')

axs[1].plot(ED.index, ED['CE'],'bo')
axs[1].plot(EDR.index, EDR['CE'],'ro')
axs[1].set_ylabel('Concentrate Concentration (mgNH$_3$-N/L)')
axs[1].set_title('(b)', loc = 'left')

#create legends
legendED = 'Electrodialysis'
legendEDR = 'Electrodialysis Reversal'

for i in range(3):
    axs[i].legend([legendED, legendEDR])

# Create shared x and y labels
fig.text(0.5,0,'Run Number', ha='center', va = 'center')


plt.tight_layout()

fig.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/Comparison of
Electrodialysis and Electrodialysis Reversal', bbox_inches="tight")
```

**Statistical Analysis**

```
In [7]:
data = pd.concat([ED, EDR], axis = 0)
data = data.rename(columns = {'Energy per batch (Whr/L)': 'Energy', 'Total Time (mins)':
'Time'})
In [8]:
print('Max ED CE:', ED['CE'].max())
print('Mean ED DE:', ED['DE'].mean(), '+-', ED['DE'].std())
print('Max EDR CE:', EDR['CE'].max())
print('Mean EDR DE:', EDR['DE'].mean(), '+-', EDR['DE'].std())
print('Mean ED Energy Consumption:', ED['Energy per batch (Whr/L)'].mean(), '+-', ED['Energy
per batch (Whr/L)'].std())
print('Mean EDR Energy Consumption:', EDR['Energy per batch (Whr/L)'].mean(), '+-',
EDR['Energy per batch (Whr/L)'].std())
Max ED CE: 8310.0
Mean ED DE: 19.489399999999996 +- 8.484597960001235
Max EDR CE: 7094.0
Mean EDR DE: 24.106666666666666 +- 19.293626507464165
Mean ED Energy Consumption: 1.2174011266 +- 0.14498314668939563
Mean EDR Energy Consumption: 1.1412789934666667 +- 0.15613209084387208
In [9]:
# Individual Anova analysis

# Dilute end
model = smf.ols('DE ~ C(label)', data = data).fit()
anova_dilute = sm.stats.anova_lm(model,typ=2)

# Concentrate end
model = smf.ols('CE ~ C(label)', data = data).fit()
anova_conc = sm.stats.anova_lm(model,typ=2)

# Time
model = smf.ols('Time ~ C(label)', data = data).fit()
anova_time = sm.stats.anova_lm(model,typ=2)

# Energy
model = smf.ols('Energy ~ C(label)', data = data).fit()
anova_energy = sm.stats.anova_lm(model,typ=2)

# Results
print('Dilute')
print(anova_dilute)
print("\n")
print('Conc')
print(anova_conc)
print("\n")
print('Time')
print(anova_time)
print("\n")
print('Energy')
print(anova_energy)
Dilute
          sum_sq    df         F    PR(>F)
```

```
C(label)   159.893636   1.0  0.719865  0.403384
Residual  6219.253969  28.0       NaN       NaN


Conc
              sum_sq    df         F    PR(>F)
C(label)  3.186715e+06   1.0  0.717544  0.404133
Residual  1.243520e+08  28.0       NaN       NaN


Time
             sum_sq    df       F    PR(>F)
C(label)   217.89075   1.0  2.4289  0.130349
Residual  2511.81232  28.0     NaN       NaN


Energy
            sum_sq    df         F    PR(>F)
C(label)  0.043459   1.0  1.914621  0.177383
Residual  0.635563  28.0       NaN       NaN
```

In [10]:

```
t_stat, p_value = stats.ttest_ind(ED['DE'], EDR['DE'])

print(f"T-statistic: {t_stat:.4f}, P-value: {p_value}")
T-statistic: -0.8484, P-value: 0.40338359989660777
```

In [11]:

```
t_stat, p_value = stats.ttest_ind(ED['CE'], EDR['CE'])

print(f"T-statistic: {t_stat:.4f}, P-value: {p_value}")
T-statistic: 0.8471, P-value: 0.40413318368032325
```

In [12]:

```
t_stat, p_value = stats.ttest_ind(ED['Total Time (mins)'], EDR['Total Time (mins)'])

print(f"T-statistic: {t_stat:.4f}, P-value: {p_value}")
T-statistic: -1.5585, P-value: 0.13034890250331604
```

In [13]:

```
t_stat, p_value = stats.ttest_ind(ED['Energy per batch (Whr/L)'], EDR['Energy per batch
(Whr/L)'])

print(f"T-statistic: {t_stat:.4f}, P-value: {p_value}")
T-statistic: 1.3837, P-value: 0.1773834615296145
```

In [ ]:

**Maximum Concentration Testing**

In [2]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
from scipy.optimize import curve_fit
import statsmodels.api as sm
import statsmodels.formula.api as smf
In [3]:
MaxConc = pd.read_excel('MetaDataSheet v1.xlsx', sheet_name = 4, index_col = 'Run Number',
na_values = 0, usecols = 'A:S')
In [4]:
#Create subplot
fig, axs = plt.subplots(2, 2, sharex = True, figsize = (12,8))

axs = axs.flatten()

# Create each individual plot
axs[0].plot(MaxConc.index, MaxConc['DEA'],'bo')
axs[0].set_ylabel('Dilute Concentration (mgNH$_3$-N/L)')
axs[0].set_title('(b)', loc = 'left')

axs[1].plot(MaxConc.index, MaxConc['CEA'],'bo')
axs[1].set_ylabel('Concentrate Concentration (mgNH$_3$-N/L)')
axs[1].set_title('(a)', loc = 'left')

axs[2].plot(MaxConc.index, MaxConc['Dilute Final Conductivity (mS/cm)'],'bo')
axs[2].set_ylabel('Dilute Final Conductivity (mS/cm)')
axs[2].set_title('(c)', loc = 'left')

axs[3].plot(MaxConc.index, MaxConc['Energy per batch (Whr/L)'],'bo')
axs[3].set_ylabel('Energy Consumption (Wh/L)')
axs[3].set_title('(d)', loc = 'left')

# Create shared x and y labels
fig.text(0.5,0,'Run Number', ha='center', va = 'center')

plt.tight_layout()

fig.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/Maximum Concentration
Testing', bbox_inches="tight")
```

**Statistical Analysis**

```
In [6]:
print('Dilute Concentration runs 1-10:', MaxConc['DEA'][0:10].mean(), '+-',
MaxConc['DEA'][0:10].std())
print('Dilute Concentration runs 11-43:', MaxConc['DEA'][10:].mean(), '+-',
MaxConc['DEA'][10:].std())
```

```
print('Energy Consumption runs 1-10:', MaxConc['Energy per batch (Whr/L)'][0:10].mean(), '+-',
MaxConc['Energy per batch (Whr/L)'][0:10].std())
print('Energy Consumption runs 11-43:', MaxConc['Energy per batch (Whr/L)'][10:].mean(), '+-',
MaxConc['Energy per batch (Whr/L)'][10:].std())
Dilute Concentration runs 1-10: 8.911999999999999 +- 1.3493356389967128
Dilute Concentration runs 11-43: 16.75 +- 2.062726038444616
Energy Consumption runs 1-10: 0.9060044178250001 +- 0.10240844078872025
Energy Consumption runs 11-43: 1.3600033727945313 +- 0.09750556367770684
In [7]:
# Curve Fitting

def func(x, a, b):
   return a + b * np.log(x)

MaxConc = MaxConc.dropna()
p0 = [1,1 ]

vals, cov = curve_fit(func, MaxConc.index, MaxConc['CEA'], p0)

fit = func(MaxConc.index, vals[0], vals[1])

plt.plot(MaxConc.index, MaxConc['CEA'], 'ob')
plt.plot(MaxConc.index, fit, '--r')

print('Parameter Values:', vals)

# Calculate R-squared

y_mean = np.mean(MaxConc['CEA'])

TSS = np.sum((MaxConc['CEA'] - y_mean)**2)

RSS = np.sum((MaxConc['CEA'] - fit)**2)

R_squared = 1 - (RSS / TSS)



print("R-squared:", R_squared)
Parameter Values: [1612.89493975 1642.78093447]
R-squared: 0.8072340872120218
```

**ED Optimization**

```
In [2]:
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
In [3]:
Opt = pd.read_excel('MetaDataSheet v1.xlsx', sheet_name = 'Optimization', index_col =
'Variable', na_values = 0, usecols = 'A:U')
In [4]:
# Sample sensitivity data (variable impact on outcome)
variables = ["Conc Flow Rate", "Dil Flow Rate", "Voltage", "Current"]

x05 = []
x2 = []
for var in variables:

   x05.append(Opt['Energy per batch (Whr/L)'][f'0.5x {var}'] - Opt['Energy per batch
(Whr/L)']['1x baseline'])

   x2.append(Opt['Energy per batch (Whr/L)'][f'2x {var}'] - Opt['Energy per batch (Whr/L)']['1x
baseline'])

energy = {
   '0.5x Baseline' : x05,
   '2x Baseline' : x2
}

# Sorting by impact
variables = ["Conc. Flow Rate \n (150 L/h)", "Dil. Flow Rate \n (150 L/h)", "Max. Voltage \n
(22V)", "Max. Current \n (0.6A)"]
df = pd.DataFrame({"Variable": variables, "Low": x05, "High": x2})
df = df.set_index("Variable")

# Plotting
fig, ax = plt.subplots(figsize=(8, 5))
ax.barh(df.index, df["Low"], color="salmon", label="0.5x baseline")
ax.barh(df.index, df["High"], color="skyblue", label="2x baseline")


# Vertical reference line at zero
ax.axvline(x=0, color='black', linewidth=1)

# Labels and title
plt.xlabel("Impact on Energy Consumption (Whr/L-Treated)")
plt.legend()
plt.show()

fig.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/Electrodialysis Sensitivity',
bbox_inches="tight")
```

**ED Stepping**

In [2]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

In [3]:

```
# Read in data
Data = pd.read_excel('6-14-23 Continuous Current.xlsx', sheet_name = 1, skiprows=60, 
usecols='J:L')
Data = Data.dropna(how = 'all')
Data.columns = ['Time', 'Cond in', 'Cond out']
```

In [4]:

```
# Extract necessary data
time = Data["Time"].values - Data["Time"].values[0]
cond_in = Data["Cond in"].values
cond_out = Data["Cond out"].values

# Find the index where cond_in is closest to 0.1
start_index = np.argmin(np.abs(cond_in - 0.1))

# Initialize step transition lists
step_time = []
step_cond = []

# Step up and left to reach the highest Cond out value
max_cond_out = np.max(cond_out)
current_time = time[start_index]
current_cond = cond_out[start_index]
```

In [5]:

```
while current_cond < max_cond_out:

    # Move up to the next highest cond_out value
    next_cond = cond_in[np.argmin(np.abs(cond_out - current_cond))]

    # Move left by decreasing time slightly
    current_time = time[np.argmin(np.abs(cond_out - current_cond))]  # Adjust step size as 
needed

    step_time.append(current_time)  # Keep the same time for vertical step
    step_cond.append(current_cond)  # Move up

    step_time.append(current_time)  # Keep the same time for vertical step
    step_cond.append(next_cond)  # Move up

    next_time = time[np.argmin(np.abs(cond_out - next_cond))]
    current_cond = next_cond  # Update current condition
In [6]:
# Plot the original data
fig = plt.figure()

plt.plot(time, cond_in, ".", label="Cond in")
plt.plot(time, cond_out, ".", label="Cond out")

# Plot the stepping transition
plt.plot(step_time, step_cond, 'k-', label="Step Transition")

# Labels and legend
plt.xlabel("Time")
plt.ylabel("Conductivity")


step_count = 1
for i in range(len(step_cond)):
    if i%2 == 1:
        plt.text(step_time[i]+0.1, step_cond[i]+0.1, str(step_count))
        step_count += 1

#plt.axhline(0.1, color = 'black')
plt.legend()
plt.show()

fig.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/ED Stepped 12A',
bbox_inches="tight")

In [7]:
# Store the first runs data in new variables
time1 = time
cond_in1 = cond_in
cond_out1 = cond_out
step_time1 = step_time
step_cond1 = step_cond
In [8]:
# Read second set of data
Df = pd.read_excel('6-14-23 Continuous run 1.xlsx', skiprows=575, usecols='J:L')
Df = Df.dropna(how = 'all')
Df.columns = ['Time', 'Cond in', 'Cond out']
In [9]:
# Extract necessary data
```

```
time = Df["Time"].values - Df["Time"].values[0]
cond_in = Df["Cond in"].values
cond_out = Df["Cond out"].values

# Find the index where cond_in is closest to 0.1
start_index = np.argmin(np.abs(cond_in - 0.1))

# Initialize step transition lists
step_time = []
step_cond = []

# Step up and left to reach the highest Cond out value
max_cond_out = np.max(cond_out)
current_time = time[start_index]
current_cond = cond_out[start_index]
In [10]:
while current_cond < max_cond_out:

   # Move up to the next highest cond_out value
   next_cond = cond_in[np.argmin(np.abs(cond_out - current_cond))]

   # Move left by decreasing time slightly
   current_time = time[np.argmin(np.abs(cond_out - current_cond))]  # Adjust step size as
needed

   step_time.append(current_time)  # Keep the same time for vertical step
   step_cond.append(current_cond)  # Move up

   step_time.append(current_time)  # Keep the same time for vertical step
   step_cond.append(next_cond)  # Move up

   next_time = time[np.argmin(np.abs(cond_out - next_cond))]
   current_cond = next_cond  # Update current condition
In [11]:
# Plot the original data
fig = plt.figure()

plt.plot(time, cond_in, ".", label="Cond in")
plt.plot(time, cond_out, ".", label="Cond out")

# Plot the stepping transition
plt.plot(step_time, step_cond, 'k-', label="Step Transition")

# Labels and legend
plt.xlabel("Time")
plt.ylabel("Conductivity")
plt.title("Electrodialysis Sizing (I = 0.6A)")

step_count = 1
for i in range(len(step_cond)):
   if i%2 == 1:
      plt.text(step_time[i]+0.1, step_cond[i]+0.1, str(step_count))
      step_count += 1

#plt.axhline(0.1, color = 'black')
plt.legend()
plt.show()

fig.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/ED Stepped 06A',
bbox_inches="tight")

In [12]:
#Create subplot with both runs stepped
f , ax = plt.subplots(1,2, tight_layout = True, sharey=True)
ax[0].plot(time1, cond_in1, ".", label="Cond in")
ax[0].plot(time1, cond_out1, ".", label="Cond out")

# Plot the stepping transition
ax[0].plot(step_time1, step_cond1, 'k-', label="Step Transition")

ax[0].set_title('(a)', loc = 'left')

step_count = 1
for i in range(len(step_cond1)):
   if i%2 == 1:
      ax[0].text(step_time1[i]+0.1, step_cond1[i]+0.1, str(step_count))
      step_count += 1
ax[0].text(0, 0, '0.6 A', weight = 'bold')

ax[1].plot(time, cond_in, ".", label="Cond in")
ax[1].plot(time, cond_out, ".", label="Cond out")

#create legends
legin = 'Conductivity in'
legout = 'Conductivity out'

for i in range(2):
   ax[i].legend([legin, legout])

# Plot the stepping transition
ax[1].plot(step_time, step_cond, 'k-', label="Step Transition")
```

```
ax[1].set_title('(b)', loc = 'left')

step_count = 1
for i in range(len(step_cond)):
    if i%2 == 1:
        ax[1].text(step_time[i]+0.1, step_cond[i]+0.1, str(step_count))
        step_count += 1
ax[1].text(0, 0, '1.2 A', weight = 'bold')

f.supxlabel('Time (min)')
f.supylabel('Conductivity (mS/cm)')

f.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/ED Stepped Combined',
bbox_inches="tight")
```

## Port-o-Potty testing at MSU

```
                                                                                      In [2]:
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
                                                                                      In [3]:
data = pd.read_excel('MetadataSheet v1.xlsx', sheet_name = 15, index_col = 'Date', na_values = ('NM',0))
                                                                                      In [4]:
BWmask = data['Sample'] == 'BW'
Blackwater = data[BWmask]
                                                                                      In [5]:
ECmask = data['Sample'] == 'EC Eff'
ECEffluent = data[ECmask]
                                                                                      In [6]:
EDmask = data['Sample'] == 'ED Treat'
EDTreated = data[EDmask]
                                                                                      In [7]:
# Create Plot for average parameters
parameters = ['COD (mg/L)', 'NH3-N (mg/L)', 'Turbidity (NTU)', 'pH']

aveBlackwater = []
aveECEffluent = []
aveEDTreated = []

for parameter in parameters:
    aveBlackwater.append(Blackwater[parameter].mean())
    aveECEffluent.append(ECEffluent[parameter].mean())
    aveEDTreated.append(EDTreated[parameter].mean())

Averages = {
    'Blackwater': aveBlackwater,
    'EC Treated': aveECEffluent,
    'ED Treated': aveEDTreated
}
```

```
x = np.arange(len(parameters))  # the Sample locations
width = 0.25  # the width of the bars
multiplier = 0

fig, ax = plt.subplots(layout='constrained')

for attribute, measurement in Averages.items():
    offset = width * multiplier
    rects = ax.bar(x + offset, measurement, width, label=attribute)
    ax.bar_label(rects, padding=1)
    multiplier += 1

# Add some text for Samples, title and custom x-axis tick Samples, etc.
ax.set_yscale('log')
ax.set_ylabel('Difference in Energy Consumption from Baseline (Whr/L)')
ax.set_title('Effect of Different Parameters on Energy Consumption')
ax.set_xticks(x + width, parameters)
ax.legend()

plt.show()
```

In [8]:

```
# Create Dataframes
waters = ['Blackwater', 'EC Treated', 'ED Treated']

aveBlackwater = []
aveECEffluent = []
aveEDTreated = []

stdBlackwater = []
stdECEffluent = []
stdEDTreated = []

for parameter in parameters:
    aveBlackwater.append(Blackwater[parameter].mean())
    aveECEffluent.append(ECEffluent[parameter].mean())
    aveEDTreated.append(EDTreated[parameter].mean())

    stdBlackwater.append(Blackwater[parameter].std())
    stdECEffluent.append(ECEffluent[parameter].std())
    stdEDTreated.append(EDTreated[parameter].std())
```

In [9]:

```
MeanTable = pd.DataFrame({'Parameter': parameters, 'Blackwater': aveBlackwater, 'EC Treated':
aveECEffluent, 'ED Treated': aveEDTreated})
MeanTable = MeanTable.set_index('Parameter')
MeanTable = MeanTable.transpose()


StdTable = pd.DataFrame({'Parameter': parameters, 'Blackwater': stdBlackwater, 'EC Treated': stdECEffluent,
'ED Treated': stdEDTreated})
StdTable = StdTable.set_index('Parameter')
StdTable = StdTable.transpose()

# Round values to 2 decimal places and format as "Mean ± Std"
```

```
RoundedMeanTable = MeanTable.round(2)
RoundedStdTable = StdTable.round(2)

# Create the formatted table with "Mean ± Std"
CombinedTable = RoundedMeanTable.astype(str) + " ± " + RoundedStdTable.astype(str)

# Display the formatted table as a figure
fig, ax = plt.subplots(figsize=(8, 3))  # Adjust figure size as needed
ax.axis('tight')
ax.axis('off')

# Create the table
table = ax.table(cellText=CombinedTable.values,
            colLabels=CombinedTable.columns,
            rowLabels=CombinedTable.index,
            cellLoc='center',
            loc='center')

# Adjust the font size
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.2)  # Adjust scaling if necessary

# Save the table as an image
plt.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/PP_mean_Table.png',
bbox_inches="tight")
plt.show()
```

In [10]:

```
# Create Plot for average parameters
colors = ['blue', 'orange','red']

fig, ax = plt.subplots(2,2, figsize = (8,5))

ax[0, 0].bar(waters, MeanTable['COD (mg/L)'], yerr=StdTable['COD (mg/L)'], color = colors, capsize=5,
alpha=0.8)
ax[0, 0].set_title("(a)  COD")

ax[0, 1].bar(waters, MeanTable['Turbidity (NTU)'], yerr=StdTable['Turbidity (NTU)'], color = colors,
capsize=5, alpha=0.8)
ax[0, 1].set_title("(b)  Turbidity (NTU)")

ax[1, 0].bar(waters, MeanTable['NH3-N (mg/L)'], yerr=StdTable['NH3-N (mg/L)'], color = colors, capsize=5,
alpha=0.8)
ax[1, 0].set_title("(c)  NH3-N")

ax[1, 1].bar(waters, MeanTable['pH'], yerr=StdTable['pH'], color = colors, capsize=5, alpha=0.8)
ax[1, 1].set_title("(d)  pH")

# Add some text for labels, title and custom x-axis tick labels, etc.
fig.supylabel('Concentration (mg/L)')
plt.tight_layout()

fig.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/Port-o-potty', bbox_inches="tight")
```

## Statistical Analysis

In [12]:

```
# Statistical Analysis

data = data.rename(columns = {'COD (mg/L)': 'COD', 'NH3-N (mg/L)': 'NH3'})

data_BW_EC = data[data['Sample'].isin(['BW', 'EC Eff'])]
data_BW_ED = data[data['Sample'].isin(['BW', 'ED Treat'])]
data_EC_ED = data[data['Sample'].isin(['EC Eff', 'ED Treat'])]

data_frames = [data_BW_EC, data_BW_ED, data_EC_ED]
Variables = ['COD', 'NH3', 'pH']
anovas = []

# Power ANOVA Analysis
for variable in Variables:
  for frame in data_frames:
    model = smf.ols(f'{variable} ~ C(Sample)', data = frame).fit()
    anova = sm.stats.anova_lm(model,typ=2)
    anovas.append(f'Variable: {variable}, Waters: {frame['Sample'].unique()}')
    anovas.append(anova)
    anovas.append(' ')
```

In [13]:

```
reduction = []
std = []
Variables = ['COD', 'NH3', 'Turbidity', 'pH']

for param in parameters[0:3]:
  for i in range(len(MeanTable)-1):
    per = (MeanTable[param][i] - MeanTable[param][i+1]) / MeanTable[param][i]*100
    reduction.append(per)
    prop = np.abs(1/MeanTable[param][i]) * ( (StdTable[param][i])**2 + (StdTable[param][i+1])**2)**0.5 *
100
    std.append(prop)

# Generate data
var, w = [], []
for v in Variables[0:3]:
  for i in range(len(waters) - 1):  # Pairwise transitions
    var.append(v)
    w.append(f'{waters[i]} -> {waters[i+1]}')

# Create DataFrame
df = pd.DataFrame({
  'Parameters': var,
  'Waters': w,
  '% Reduction': reduction,
  '% Error': std
})

# Pivot the table to restructure it
df_pivot = df.pivot(index='Waters', columns="Parameters", values=["% Reduction", "% Error"])
```

```
# Flatten the multi-index column names
df_pivot.columns = [f"{metric} {param}" for metric, param in df_pivot.columns]

# Combine % Reduction and % Error into a single formatted column
for col in Variables[0:3]:
   df_pivot[col] = df_pivot[f'% Reduction {col}'].apply(lambda x: f'{x:.1f}') + " ± " + df_pivot[f'% Error
{col}'].apply(lambda x: f'{x:.1f}')

# Drop the separate % Error and % Reduction columns
df_pivot = df_pivot[Variables[0:3]]

# Calculate total reduction (assuming we have the actual initial and final values)
initial_values = {"COD": MeanTable['COD (mg/L)'][0], "Turbidity": MeanTable['Turbidity (NTU)'][0], "NH3":
MeanTable['NH3-N (mg/L)'][0]}  # Replace with real initial values
final_values = {"COD": MeanTable['COD (mg/L)'][2], "Turbidity": MeanTable['Turbidity (NTU)'][2], "NH3":
MeanTable['NH3-N (mg/L)'][2]}    # Replace with real final values
initial_errors = {"COD": StdTable['COD (mg/L)'][0], "Turbidity": StdTable['Turbidity (NTU)'][0], "NH3":
StdTable['NH3-N (mg/L)'][0]}     # Replace with real error values
final_errors = {"COD": StdTable['COD (mg/L)'][2], "Turbidity": StdTable['Turbidity (NTU)'][2], "NH3":
StdTable['NH3-N (mg/L)'][2]}      # Replace with real error values

total_reduction = {}
total_error = {}

for param in Variables[0:3]:
   # Compute total reduction
   reduction_val = ((initial_values[param] - final_values[param]) / initial_values[param]) * 100
   total_reduction[param] = f"{reduction_val:.1f}"

   # Error propagation formula for percent reduction:
   error_val = reduction_val * np.sqrt(
      (initial_errors[param] / initial_values[param])**2 +
      (final_errors[param] / final_values[param])**2
   )
   total_error[param] = f"{error_val:.1f}"

# Format total reduction row
total_row = {var: f"{total_reduction[var]} ± {total_error[var]}" for var in Variables[0:3]}


# Append total row to DataFrame
df_pivot = pd.concat([df_pivot, pd.DataFrame([total_row])])
df_pivot.rename(index={0: "Total"}, inplace=True)

import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
In [2]:
data = pd.read_excel('MetadataSheet v3.xlsx', sheet_name = 'Real blackwater', usecols = 'A:H')
data = data.melt(id_vars=["Water", "Parameter"], var_name="Date", value_name="Value")
data = data.pivot(index=["Date", "Water"], columns="Parameter", values="Value")
In [3]:
water_frames = {water: data.xs(water, level="Water") for water in
data.index.get_level_values("Water").unique()}
In [4]:
Blackwater = water_frames['BW']
In [5]:
ECEffluent = water_frames['EC']
In [6]:
EDTreated = water_frames['ED']
In [7]:
# Create Dataframes
parameters = ['COD (mg/L)', 'TP (mg/L)', 'NH3-N (mg/L)', 'TN (mg/L)', 'Turbidity (NTU)', 'TSS
(mg/mL)', 'pH']
waters = ['Blackwater', 'EC Treated', 'ED Treated']

aveBlackwater = []
aveECEffluent = []
aveEDTreated = []

stdBlackwater = []
stdECEffluent = []
stdEDTreated = []

for parameter in parameters:
    aveBlackwater.append(Blackwater[parameter].mean())
    aveECEffluent.append(ECEffluent[parameter].mean())
    aveEDTreated.append(EDTreated[parameter].mean())

    stdBlackwater.append(Blackwater[parameter].std())
    stdECEffluent.append(ECEffluent[parameter].std())
    stdEDTreated.append(EDTreated[parameter].std())
In [8]:
MeanTable = pd.DataFrame({'Parameter': parameters, 'Blackwater': aveBlackwater, 'EC
Treated': aveECEffluent, 'ED Treated': aveEDTreated})
MeanTable = MeanTable.set_index('Parameter')
MeanTable = MeanTable.transpose()


StdTable = pd.DataFrame({'Parameter': parameters, 'Blackwater': stdBlackwater, 'EC Treated':
stdECEffluent, 'ED Treated': stdEDTreated})
StdTable = StdTable.set_index('Parameter')
StdTable = StdTable.transpose()

# Round values to 2 decimal places and format as "Mean ± Std"
RoundedMeanTable = MeanTable.round(2)
```

```
RoundedStdTable = StdTable.round(2)

# Create the formatted table with "Mean ± Std"
CombinedTable = RoundedMeanTable.astype(str) + " ± " + RoundedStdTable.astype(str)

CombinedTable.columns = [
   r'COD (mg/L)',
   r'TP (mg/L)',
   r'NH$_3$-N (mg/L)',
   r'TN (mg/L)',
   r'Turbidity (NTU)',
   r'TSS (mg/L)',
   r'pH'
]

# Display the formatted table as a figure
fig, ax = plt.subplots(figsize=(8, 3))  # Adjust figure size as needed
ax.axis('tight')
ax.axis('off')

# Create the table
table = ax.table(cellText=CombinedTable.values,
              colLabels=CombinedTable.columns,
              rowLabels=CombinedTable.index,
              cellLoc='center',
              loc='center')

# Adjust the font size
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.5, 1.5)  # Adjust scaling if necessary

# Save the table as an image
plt.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/BW_mean_Table.png',
bbox_inches="tight")
plt.show()

In [9]:
# Create Plot for average parameters
colors = ['blue', 'orange','red']

fig, ax = plt.subplots(2,2, figsize = (8,5))

ax[0, 0].bar(waters, MeanTable['COD (mg/L)'], yerr=StdTable['COD (mg/L)'], color = colors,
capsize=5, alpha=0.8)
ax[0, 0].set_title("(a)", loc = 'left')
```

```
ax[1, 1].bar(waters, MeanTable['TN (mg/L)'], yerr=StdTable['TN (mg/L)'], color = colors,
capsize=5, alpha=0.8)
ax[1, 1].set_title("(b)", loc = 'left')

ax[1, 0].bar(waters, MeanTable['NH3-N (mg/L)'], yerr=StdTable['NH3-N (mg/L)'], color =
colors, capsize=5, alpha=0.8)
ax[1, 0].set_title("(c)", loc = 'left')

ax[0, 1].bar(waters, MeanTable['TP (mg/L)'], yerr=StdTable['TP (mg/L)'], color = colors,
capsize=5, alpha=0.8)
ax[0, 1].set_title("(d)", loc = 'left')

ax[0, 0].text(2.18, 1850, 'COD', weight = 'bold')
ax[0, 1].text(2.3, 21.5, 'TP', weight = 'bold')
ax[1, 0].text(2, 270, 'NH$_3$-N', weight = 'bold')
ax[1, 1].text(2.3, 600, 'TN', weight = 'bold')

# Add some text for labels, title and custom x-axis tick labels, etc.
fig.supylabel('Concentration (mg/L)')
plt.tight_layout()

fig.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/Real Blackwater',
bbox_inches="tight")
```

**Analysis**

```
In [11]:
reduction = []
std = []
Variables = ['COD', 'TP', 'NH3', 'TN', 'Turbidity', 'TSS']

for param in parameters[0:6]:
    for i in range(len(MeanTable)-1):
        per = (MeanTable[param][i] - MeanTable[param][i+1]) / MeanTable[param][i]*100
        reduction.append(per)
        prop = np.abs(1/MeanTable[param][i]) * ( (StdTable[param][i])**2 +
(StdTable[param][i+1])**2)**0.5 * 100
        std.append(prop)

# Generate data
var, w = [], []
for v in Variables[0:6]:
    for i in range(len(waters) - 1):  # Pairwise transitions
        var.append(v)
        w.append(f'{waters[i]} → {waters[i+1]}')
```

```
# Create DataFrame
df = pd.DataFrame({
    'Parameters': var,
    'Waters': w,
    '% Reduction': reduction,
    '% Error': std
})

# Pivot the table to restructure it
df_pivot = df.pivot(index='Waters', columns="Parameters", values=["% Reduction", "% Error"])

# Flatten the multi-index column names
df_pivot.columns = [f"{metric} {param}" for metric, param in df_pivot.columns]

# Combine % Reduction and % Error into a single formatted column
for col in Variables[0:6]:
    df_pivot[col] = df_pivot[f'% Reduction {col}'].apply(lambda x: f'{x:.1f}') + " ± " +
df_pivot[f'% Error {col}'].apply(lambda x: f'{x:.1f}')

# Drop the separate % Error and % Reduction columns
df_pivot = df_pivot[Variables[0:6]]

# Calculate total reduction (assuming we have the actual initial and final values)
initial_values = {"COD": MeanTable['COD (mg/L)'][0], "TN": MeanTable['TN (mg/L)'][0],
"NH3": MeanTable['NH3-N (mg/L)'][0], 'TP': MeanTable['TP (mg/L)'][0], 'TSS':
MeanTable['TSS (mg/mL)'][0], 'Turbidity': MeanTable['Turbidity (NTU)'][0]}  # Replace with
real initial values
final_values = {"COD": MeanTable['COD (mg/L)'][2], "TN": MeanTable['TN (mg/L)'][2],
"NH3": MeanTable['NH3-N (mg/L)'][2], 'TP': MeanTable['TP (mg/L)'][2], 'TSS':
MeanTable['TSS (mg/mL)'][2], 'Turbidity': MeanTable['Turbidity (NTU)'][2]}    # Replace with
real final values
initial_errors = {"COD": StdTable['COD (mg/L)'][0], "TN": StdTable['TN (mg/L)'][0], "NH3":
StdTable['NH3-N (mg/L)'][0], 'TP': StdTable['TP (mg/L)'][0], 'TSS': StdTable['TSS
(mg/mL)'][0], 'Turbidity': StdTable['Turbidity (NTU)'][0]}    # Replace with real error values
final_errors = {"COD": StdTable['COD (mg/L)'][2], "TN": StdTable['TN (mg/L)'][2], "NH3":
StdTable['NH3-N (mg/L)'][2], 'TP': StdTable['TP (mg/L)'][2], 'TSS': StdTable['TSS
(mg/mL)'][2], 'Turbidity': StdTable['Turbidity (NTU)'][2]}     # Replace with real error values

total_reduction = {}
total_error = {}

for param in Variables[0:6]:
    # Compute total reduction
    reduction_val = ((initial_values[param] - final_values[param]) / initial_values[param]) * 100
    total_reduction[param] = f"{reduction_val:.1f}"
    # Error propagation formula for percent reduction:
    error_val = reduction_val * np.sqrt(
        (initial_errors[param] / initial_values[param])**2 +
        (final_errors[param] / final_values[param])**2
    )
    total_error[param] = f"{error_val:.1f}"

# Format total reduction row
total_row = {var: f"{total_reduction[var]} ± {total_error[var]}" for var in Variables[0:6]}


# Append total row to DataFrame
df_pivot = pd.concat([df_pivot, pd.DataFrame([total_row])])
df_pivot.rename(index={0: "Total"}, inplace=True)
# Display the final DataFrame
df_pivot

# Optional: Format column labels with NH₃ subscript
df_pivot.columns = [
    r'COD (%)',
    r'TP (%)',
    r'NH$_3$-N (%)',
    r'TN (%)',
    r'Turbidity (%)',
    r'TSS (%)'
]

# Optional: Reorder rows if needed
desired_order = [
    'Blackwater → EC Treated',
    'EC Treated → ED Treated',
    'Total'
]
df_pivot = df_pivot.reindex(desired_order)

# Set figure size based on table size
fig_height = 1 + 0.6 * len(df_pivot)
fig_width = 3 + 1.2 * len(df_pivot.columns)

fig, ax = plt.subplots(figsize=(fig_width, fig_height))
ax.axis('off')
ax.axis('tight')

# Create the table
table = ax.table(
    cellText=df_pivot.values,
    rowLabels=df_pivot.index,
    colLabels=df_pivot.columns,
    cellLoc='center',
    rowLoc='center',
    loc='center'
)

# Styling
table.auto_set_font_size(False)
table.set_fontsize(9.5)
table.scale(1, 1.4)

# Save the figure
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis
Figures/BW_reduction_percent_table.png", dpi=300, bbox_inches="tight")
plt.show()
```

**Camp Shelby testing**

```
In [2]:
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
In [3]:
Blackwater = pd.read_excel('MetaDataSheet v1.xlsx', sheet_name = 11, index_col = 'Sample',
na_values = ('nm',0))
Blackwater = Blackwater.drop('Truck')
In [4]:
ECTreated = pd.read_excel('MetaDataSheet v1.xlsx', sheet_name = 12, index_col = 'Date',
na_values = ('nm',0))
In [5]:
ECFiltered = pd.read_excel('MetaDataSheet v1.xlsx', sheet_name = 13, index_col = 'Date',
na_values = ('nm',0, 'undermeasuring range', 'neg value'))
In [6]:
EDTreated = pd.read_excel('MetaDataSheet v1.xlsx', sheet_name = 14, index_col = 'Date',
na_values = ('nm',0))
In [7]:
# Create Plot for average parameters
parameters = ['COD (mg/L)', 'TP (mg/L)', 'NH3-N (mg/L)', 'pH']

aveBlackwater = []
aveECTreated = []
aveECFiltered = []
aveEDTreated = []
```

```
for parameter in parameters:
    aveBlackwater.append(Blackwater[parameter].mean())
    aveECTreated.append(ECTreated[parameter].mean())
    aveECFiltered.append(ECFiltered[parameter].mean())
    aveEDTreated.append(EDTreated[parameter].mean())

Averages = {
    'Blackwater': aveBlackwater,
    'EC Treated': aveECTreated,
    'EC Filtered': aveECFiltered,
    'ED Treated': aveEDTreated
}

x = np.arange(len(parameters))  # the label locations
width = 0.25  # the width of the bars
multiplier = 0

fig, ax = plt.subplots(layout='constrained')

for attribute, measurement in Averages.items():
    offset = width * multiplier
    rects = ax.bar(x + offset, measurement, width, label=attribute)
    ax.bar_label(rects, padding=1)
    multiplier += 1

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_yscale('log')
ax.set_ylabel('Parameter Concentration in (mg/L)')
ax.set_title('Water Treatment of Combined EC and ED System')
ax.set_xticks(x + width, parameters)
ax.legend()

plt.show()

In [8]:
# Create Dataframes
parameters = ['COD (mg/L)', 'TP (mg/L)', 'NH3-N (mg/L)', 'pH']
waters = ['Blackwater', 'EC Treated', 'EC Filtered', 'ED Treated']

aveBlackwater = []
aveECTreated = []
aveECFiltered = []
aveEDTreated = []

stdBlackwater = []
```

```
stdECTreated = []
stdECFiltered = []
stdEDTreated = []

for parameter in parameters:
    aveBlackwater.append(Blackwater[parameter].mean())
    aveECTreated.append(ECTreated[parameter].mean())
    aveECFiltered.append(ECFiltered[parameter].mean())
    aveEDTreated.append(EDTreated[parameter].mean())

    stdBlackwater.append(Blackwater[parameter].std())
    stdECTreated.append(ECTreated[parameter].std())
    stdECFiltered.append(ECFiltered[parameter].std())
    stdEDTreated.append(EDTreated[parameter].std())

Averages = {
    'Blackwater': aveBlackwater,
    'EC Treated': aveECTreated,
    'EC Filtered': aveECFiltered,
    'ED Treated': aveEDTreated
}
In [9]:
MeanTable = pd.DataFrame({'Parameter': parameters, 'Blackwater': aveBlackwater, 'EC
Treated': aveECTreated, 'EC Filtered': aveECFiltered, 'ED Treated': aveEDTreated})
MeanTable = MeanTable.set_index('Parameter')
MeanTable = MeanTable.transpose()


StdTable = pd.DataFrame({'Parameter': parameters, 'Blackwater': stdBlackwater, 'EC Treated':
stdECTreated, 'EC Filtered': stdECFiltered, 'ED Treated': stdEDTreated})
StdTable = StdTable.set_index('Parameter')
StdTable = StdTable.transpose()
In [10]:
# Create Dataframes
parameters = ['COD (mg/L)', 'TP (mg/L)', 'NH3-N (mg/L)', 'pH']
waters = ['Blackwater', 'EC Treated', 'EC Filtered', 'ED Treated']

aveBlackwaterSpliced = []
aveECTreatedSpliced = []
aveECFilteredSpliced = []
aveEDTreatedSpliced = []

stdBlackwaterSpliced = []
stdECTreatedSpliced = []
stdECFilteredSpliced = []
stdEDTreatedSpliced = []
```

```
for parameter in parameters:
    aveBlackwaterSpliced.append(Blackwater[parameter][3:6].mean())
    aveECTreatedSpliced.append(ECTreated[parameter][3:6].mean())
    aveECFilteredSpliced.append(ECFiltered[parameter][3:6].mean())
    aveEDTreatedSpliced.append(EDTreated[parameter][3:6].mean())

    stdBlackwaterSpliced.append(Blackwater[parameter][3:6].std())
    stdECTreatedSpliced.append(ECTreated[parameter][3:6].std())
    stdECFilteredSpliced.append(ECFiltered[parameter][3:6].std())
    stdEDTreatedSpliced.append(EDTreated[parameter][3:6].std())

AveragesSpliced = {
    'Blackwater': aveBlackwaterSpliced,
    'EC Treated': aveECTreatedSpliced,
    'EC Filtered': aveECFilteredSpliced,
    'ED Treated': aveEDTreatedSpliced
}

# Create a dictionary to store the formatted values (Mean ± SD)
summary_data = {
    'Parameter': parameters,
    'Blackwater': [f"{mean:.1f} ± {std:.2f}" for mean, std in zip(aveBlackwater, stdBlackwater)],
    'EC Treated': [f"{mean:.1f} ± {std:.2f}" for mean, std in zip(aveECTreated, stdECTreated)],
    'EC Filtered': [f"{mean:.1f} ± {std:.2f}" for mean, std in zip(aveECFiltered, stdECFiltered)],
    'ED Treated': [f"{mean:.1f} ± {std:.2f}" for mean, std in zip(aveEDTreated, stdEDTreated)]
}

# Convert dictionary to DataFrame
summary_df = pd.DataFrame(summary_data)

summary_data_full = {
    'Parameter': parameters,
    'Blackwater': [f"{mean:.1f} ± {std:.2f}" for mean, std in zip(aveBlackwater, stdBlackwater)],
    'EC Treated': [f"{mean:.1f} ± {std:.2f}" for mean, std in zip(aveECTreated, stdECTreated)],
    'EC Filtered': [f"{mean:.1f} ± {std:.2f}" for mean, std in zip(aveECFiltered, stdECFiltered)],
    'ED Treated': [f"{mean:.1f} ± {std:.2f}" for mean, std in zip(aveEDTreated, stdEDTreated)]
}

# Create the DataFrame
summary_df_full = pd.DataFrame(summary_data_full)

# Replace NH3 with subscript
summary_df_full['Parameter'] = summary_df_full['Parameter'].replace({
    'NH3-N (mg/L)': r'NH$_3$-N (mg/L)'
})
```

```
# Set index
summary_df_full = summary_df_full.set_index('Parameter')

# Set dynamic figure size
fig_height = 1 + 0.5 * len(summary_df_full)
fig_width = 2 + 1.5 * len(summary_df_full.columns)

# Plot the table
fig, ax = plt.subplots(figsize=(fig_width, fig_height))
ax.axis('off')
ax.axis('tight')

table = ax.table(
    cellText=summary_df_full.values,
    rowLabels=summary_df_full.index,
    colLabels=summary_df_full.columns,
    cellLoc='center',
    rowLoc='center',
    loc='center'
)

# Style
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.2)

# Save the table figure
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis
Figures/Camp_Shelby_summary_table.png", dpi=300, bbox_inches='tight')
plt.show()

In [11]:
MeanTableSpliced = pd.DataFrame({'Parameter': parameters, 'Blackwater':
aveBlackwaterSpliced, 'EC Treated': aveECTreatedSpliced, 'EC Filtered': aveECFilteredSpliced,
'ED Treated': aveEDTreatedSpliced})
MeanTableSpliced = MeanTableSpliced.set_index('Parameter')
MeanTableSpliced = MeanTableSpliced.transpose()


StdTableSpliced = pd.DataFrame({'Parameter': parameters, 'Blackwater': stdBlackwaterSpliced,
'EC Treated': stdECTreatedSpliced, 'EC Filtered': stdECFilteredSpliced, 'ED Treated':
stdEDTreatedSpliced})
StdTableSpliced = StdTableSpliced.set_index('Parameter')
StdTableSpliced = StdTableSpliced.transpose()
```

```
# Format each value as "mean ± std"
summary_formatted = MeanTableSpliced.copy()

for col in summary_formatted.columns:
    summary_formatted[col] = MeanTableSpliced[col].apply(lambda x: f"{x:.1f}") + " ± " +
StdTableSpliced[col].apply(lambda x: f"{x:.2f}")

# Optional: Make NH3 subscript
summary_formatted.columns = [
    r'COD (mg/L)',
    r'TP (mg/L)',
    r'NH$_3$-N (mg/L)',
    r'pH'
]

# Set figure size dynamically
fig_height = 1 + 0.5 * len(summary_formatted)
fig_width = 2 + 1.8 * len(summary_formatted.columns)

# Create the figure
fig, ax = plt.subplots(figsize=(fig_width, fig_height))
ax.axis('off')
ax.axis('tight')

# Create the table
table = ax.table(
    cellText=summary_formatted.values,
    rowLabels=summary_formatted.index,
    colLabels=summary_formatted.columns,
    cellLoc='center',
    rowLoc='center',
    loc='center'
)

# Style
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.2)

# Save the figure
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis
Figures/spliced_parameter_summary_table.png", dpi=300, bbox_inches='tight')
plt.show()

In [12]:
#  Function to format as:
```

```
# Spliced ± SD
# (Full ± SD)
def format_stacked(spliced_mean, spliced_std, full_mean, full_std):
    return f"{spliced_mean:.1f} ± {spliced_std:.2f}\n({full_mean:.1f} ± {full_std:.2f})"

# Build table data
stacked_data = {
    'Blackwater': [
        format_stacked(s, ss, f, fs)
        for s, ss, f, fs in zip(aveBlackwaterSpliced, stdBlackwaterSpliced, aveBlackwater,
stdBlackwater)
    ],
    'EC Treated': [
        format_stacked(s, ss, f, fs)
        for s, ss, f, fs in zip(aveECTreatedSpliced, stdECTreatedSpliced, aveECTreated,
stdECTreated)
    ],
    'EC Filtered': [
        format_stacked(s, ss, f, fs)
        for s, ss, f, fs in zip(aveECFilteredSpliced, stdECFilteredSpliced, aveECFiltered,
stdECFiltered)
    ],
    'ED Treated': [
        format_stacked(s, ss, f, fs)
        for s, ss, f, fs in zip(aveEDTreatedSpliced, stdEDTreatedSpliced, aveEDTreated,
stdEDTreated)
    ]
}

# Create DataFrame
stacked_df = pd.DataFrame(stacked_data, index=parameters)

# Subscript NH₃
stacked_df.index = [
    r'COD (mg/L)',
    r'TP (mg/L)',
    r'NH$_3$-N (mg/L)',
    r'pH'
]

# Set figure size with extra row height
fig_height = 1 + .86 * len(stacked_df)
fig_width = 2 + 1.7 * len(stacked_df.columns)

fig, ax = plt.subplots(figsize=(fig_width, fig_height))
ax.axis('off')
ax.axis('tight')

# Create the table with line breaks handled
table = ax.table(
    cellText=stacked_df.values,
    rowLabels=stacked_df.index,
    colLabels=stacked_df.columns,
    cellLoc='center',
    rowLoc='center',
    loc='center'
)

# Style the table
table.auto_set_font_size(False)
table.set_fontsize(9.5)
table.scale(1.2, 2)  # Slightly increase row height for better spacing

# Save the figure
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis
Figures/Camp_Shelby_comparison_table.png", dpi=300, bbox_inches='tight')
plt.show()

In [13]:
# Create Plot for average parameters
colors = ['blue', 'orange','green','red']

fig, ax = plt.subplots(2,2, figsize = (8,5))

ax[0, 0].bar(waters, MeanTable['COD (mg/L)'], yerr=StdTable['COD (mg/L)'], color = colors,
capsize=5, alpha=0.8)
ax[0, 0].set_title("(a)", loc = 'left')
ax[0, 0].text(3.1, 3000, 'COD', weight = 'bold')

ax[0, 1].bar(waters, MeanTable['TP (mg/L)'], yerr=StdTable['TP (mg/L)'], color = colors,
capsize=5, alpha=0.8)
ax[0, 1].set_title("(b)", loc = 'left')
ax[0, 1].text(3.25, 160, 'TP', weight = 'bold')

ax[1, 0].bar(waters, MeanTable['NH3-N (mg/L)'], yerr=StdTable['NH3-N (mg/L)'], color =
colors, capsize=5, alpha=0.8)
ax[1, 0].set_title("(c)", loc = 'left')
ax[1, 0].text(2.9, 560, 'NH$_3$-N', weight = 'bold')

ax[1, 1].bar(waters, MeanTable['pH'], yerr=StdTable['pH'], color = colors, capsize=5, alpha=0.8)
ax[1, 1].set_title("(d)", loc = 'left')
ax[1, 1].text(-0.5, 8.2, 'pH', weight = 'bold')
```

```
# Add some text for labels, title and custom x-axis tick labels, etc.
fig.supylabel('Concentration (mg/L)')
plt.tight_layout()

fig.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/Camp Shelby Total Data',
bbox_inches="tight")

In [14]:
# Create Plot for average parameters
colors = ['blue', 'orange','green','red']

fig, ax = plt.subplots(2,2, figsize = (8,5))

ax[0, 0].bar(waters, MeanTableSpliced['COD (mg/L)'], yerr=StdTableSpliced['COD (mg/L)'],
color = colors, capsize=5, alpha=0.8)
ax[0, 0].set_title("(a)", loc = 'left')

ax[0, 1].bar(waters, MeanTableSpliced['TP (mg/L)'], yerr=StdTableSpliced['TP (mg/L)'], color =
colors, capsize=5, alpha=0.8)
ax[0, 1].set_title("(b)", loc = 'left')

ax[1, 0].bar(waters, MeanTableSpliced['NH3-N (mg/L)'], yerr=StdTableSpliced['NH3-N
(mg/L)'], color = colors, capsize=5, alpha=0.8)
ax[1, 0].set_title("(c)", loc = 'left')

ax[1, 1].bar(waters, MeanTableSpliced['pH'], yerr=StdTableSpliced['pH'], color = colors,
capsize=5, alpha=0.8)
ax[1, 1].set_title("(d)", loc = 'left')

ax[0, 0].text(3.1, 2400, 'COD', weight = 'bold')
ax[0, 1].text(3.25, 138, 'TP', weight = 'bold')
ax[1, 0].text(2.9, 610, 'NH$_3$-N', weight = 'bold')
ax[1, 1].text(-0.5, 8.2, 'pH', weight = 'bold')

# Add some text for labels, title and custom x-axis tick labels, etc.
fig.supylabel('Concentration (mg/L)')
plt.tight_layout()

fig.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/Camp Shelby EC
Parameters Spliced', bbox_inches="tight")
```

**Statistical Analysis**

```
In [16]:
Blackwater['label'] = 'Blackwater'
ECTreated['label'] = 'ECTreated'
```

```
ECFiltered['label'] = 'ECFiltered'
EDTreated['label'] = 'EDTreated'

data = pd.concat([Blackwater, ECTreated, ECFiltered, EDTreated], axis=0)
data = data.rename(columns = {'COD (mg/L)': 'COD', 'TP (mg/L)': 'TP', 'NH3-N (mg/L)':
'NH3'})
In [17]:
# Statistical Analysis

data_BW_EC = data[data['label'].isin(['Blackwater', 'ECTreated'])]
data_BW_ECF = data[data['label'].isin(['Blackwater', 'ECFiltered'])]
data_BW_ED = data[data['label'].isin(['Blackwater', 'EDTreated'])]
data_EC_ECF = data[data['label'].isin(['ECTreated', 'ECFiltered'])]
data_EC_ED = data[data['label'].isin(['ECTreated', 'EDTreated'])]
data_ECF_ED = data[data['label'].isin(['ECFiltered', 'EDTreated'])]

data_frames = [data_BW_EC, data_BW_ECF, data_BW_ED, data_EC_ECF, data_EC_ED,
data_ECF_ED]
Variables = ['COD', 'TP', 'NH3', 'pH']
anovas = []

# Power ANOVA Analysis
for variable in Variables:
   for frame in data_frames:
      model = smf.ols(f'{variable} ~ C(label)', data = frame).fit()
      anova = sm.stats.anova_lm(model,typ=2)
      anovas.append(f'Variable: {variable}, Waters: {frame['label'].unique()}')
      anovas.append(anova)
      anovas.append(' ')

In [18]:
reduction = []
std = []
Variables = ['COD', 'TP', 'NH3', 'pH']

for param in parameters[0:3]:
   for i in range(len(MeanTable)-1):
      per = (MeanTable[param][i] - MeanTable[param][i+1]) / MeanTable[param][i]*100
      reduction.append(per)
      prop = np.abs(1/MeanTable[param][i]) * ( (StdTable[param][i])**2 +
(StdTable[param][i+1])**2)**0.5 * 100
      std.append(prop)

# Generate data
var, w = [], []
for v in Variables[0:3]:
    for i in range(len(waters) - 1):  # Pairwise transitions
        var.append(v)
        w.append(f'{waters[i]} → {waters[i+1]}')

# Create DataFrame
df = pd.DataFrame({
    'Parameters': var,
    'Waters': w,
    '% Reduction': reduction,
    '% Error': std
})

# Pivot the table to restructure it
df_pivot = df.pivot(index='Waters', columns="Parameters", values=["% Reduction", "% Error"])

# Flatten the multi-index column names
df_pivot.columns = [f"{metric} {param}" for metric, param in df_pivot.columns]

# Combine % Reduction and % Error into a single formatted column
for col in Variables[0:3]:
    df_pivot[col] = df_pivot[f'% Reduction {col}'].apply(lambda x: f'{x:.1f}') + " ± " +
df_pivot[f'% Error {col}'].apply(lambda x: f'{x:.1f}')

# Drop the separate % Error and % Reduction columns
df_pivot = df_pivot[Variables[0:3]]

# Calculate total reduction (assuming we have the actual initial and final values)
initial_values = {"COD": MeanTable['COD (mg/L)'][0], "TP": MeanTable['TP (mg/L)'][0],
"NH3": MeanTable['NH3-N (mg/L)'][0]}  # Replace with real initial values
final_values = {"COD": MeanTable['COD (mg/L)'][3], "TP": MeanTable['TP (mg/L)'][3],
"NH3": MeanTable['NH3-N (mg/L)'][3]}     # Replace with real final values
initial_errors = {"COD": StdTable['COD (mg/L)'][0], "TP": StdTable['TP (mg/L)'][0], "NH3":
StdTable['NH3-N (mg/L)'][0]}      # Replace with real error values
final_errors = {"COD": StdTable['COD (mg/L)'][3], "TP": StdTable['TP (mg/L)'][3], "NH3":
StdTable['NH3-N (mg/L)'][3]}       # Replace with real error values

total_reduction = {}
total_error = {}

for param in Variables[0:3]:
    # Compute total reduction
    reduction_val = ((initial_values[param] - final_values[param]) / initial_values[param]) * 100
    total_reduction[param] = f"{reduction_val:.1f}"

    # Error propagation formula for percent reduction:
    error_val = reduction_val * np.sqrt(
        (initial_errors[param] / initial_values[param])**2 +
        (final_errors[param] / final_values[param])**2
    )
    total_error[param] = f"{error_val:.1f}"

# Format total reduction row
total_row = {var: f"{total_reduction[var]} ± {total_error[var]}" for var in Variables[0:3]}


# Append total row to DataFrame
df_pivot = pd.concat([df_pivot, pd.DataFrame([total_row])])
df_pivot.rename(index={0: "Total"}, inplace=True)

# Optional: Fix NH3 column label to include subscript
df_pivot.columns = [
    'COD (%)', 'TP (%)', r'NH$_3$-N (%)'
]

# Reorder the rows
new_order = ['Blackwater → EC Treated', 'EC Treated → EC Filtered', 'EC Filtered → ED
Treated', 'Total']
df_pivot = df_pivot.reindex(new_order)

# Set figure size dynamically based on table shape
fig_height = 1 + 0.5 * len(df_pivot)
fig_width = 2 + 1.8 * len(df_pivot.columns)

# Create the plot
fig, ax = plt.subplots(figsize=(fig_width, fig_height))
ax.axis('off')
ax.axis('tight')

# Create the table
table = ax.table(
    cellText=df_pivot.values,
    rowLabels=df_pivot.index,
    colLabels=df_pivot.columns,
    cellLoc='center',
    rowLoc='center',
    loc='center'
)

# Style the table
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.2)
```

```
# Save the figure
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis
Figures/Camp_Shelby_reduction_summary_table.png", dpi=300, bbox_inches='tight')
plt.show()

In [19]:
reduction = []
std = []
Variables = ['COD', 'TP', 'NH3', 'pH']

for param in parameters[0:3]:
    for i in range(len(MeanTable)-1):
        per = (MeanTableSpliced[param][i] - MeanTableSpliced[param][i+1]) /
MeanTableSpliced[param][i]*100
        reduction.append(per)
        prop = np.abs(1/MeanTableSpliced[param][i]) * ( (StdTableSpliced[param][i])**2 +
(StdTableSpliced[param][i+1])**2)**0.5 * 100
        std.append(prop)

# Generate data
var, w = [], []
for v in Variables[0:3]:
    for i in range(len(waters) - 1):  # Pairwise transitions
        var.append(v)
        w.append(f'{waters[i]} → {waters[i+1]}')

# Create DataFrame
dfSpliced = pd.DataFrame({
    'Parameters': var,
    'Waters': w,
    '% Reduction': reduction,
    '% Error': std
})

# Pivot the table to restructure it
df_pivotSpliced = dfSpliced.pivot(index='Waters', columns="Parameters", values=["%
Reduction", "% Error"])

# Flatten the multi-index column names
df_pivotSpliced.columns = [f"{metric} {param}" for metric, param in df_pivotSpliced.columns]

# Combine % Reduction and % Error into a single formatted column
for col in Variables[0:3]:
    df_pivotSpliced[col] = df_pivotSpliced[f'% Reduction {col}'].apply(lambda x: f'{x:.1f}') + "
± " + df_pivotSpliced[f'% Error {col}'].apply(lambda x: f'{x:.1f}')
```

```
# Drop the separate % Error and % Reduction columns
df_pivotSpliced = df_pivotSpliced[Variables[0:3]]

# Calculate total reduction (assuming we have the actual initial and final values)
initial_valuesSpliced = {"COD": MeanTableSpliced['COD (mg/L)'][0], "TP":
MeanTableSpliced['TP (mg/L)'][0], "NH3": MeanTableSpliced['NH3-N (mg/L)'][0]}  # Replace
with real initial values
final_valuesSpliced = {"COD": MeanTableSpliced['COD (mg/L)'][3], "TP":
MeanTableSpliced['TP (mg/L)'][3], "NH3": MeanTableSpliced['NH3-N (mg/L)'][3]}    #
Replace with real final values
initial_errorsSpliced = {"COD": StdTableSpliced['COD (mg/L)'][0], "TP": StdTableSpliced['TP
(mg/L)'][0], "NH3": StdTableSpliced['NH3-N (mg/L)'][0]}     # Replace with real error values
final_errorsSpliced = {"COD": StdTableSpliced['COD (mg/L)'][3], "TP": StdTableSpliced['TP
(mg/L)'][3], "NH3": StdTableSpliced['NH3-N (mg/L)'][3]}      # Replace with real error values

total_reductionSpliced = {}
total_errorSpliced = {}

for param in Variables[0:3]:
   # Compute total reduction
   reduction_valSpliced = ((initial_valuesSpliced[param] - final_valuesSpliced[param]) /
initial_valuesSpliced[param]) * 100
   total_reductionSpliced[param] = f"{reduction_valSpliced:.1f}"

   # Error propagation formula for percent reduction:
   error_valSpliced = reduction_valSpliced * np.sqrt(
      (initial_errorsSpliced[param] / initial_valuesSpliced[param])**2 +
      (final_errorsSpliced[param] / final_valuesSpliced[param])**2
   )
   total_errorSpliced[param] = f"{error_valSpliced:.1f}"

# Format total reduction row
total_rowSpliced = {var: f"{total_reductionSpliced[var]} ± {total_errorSpliced[var]}" for var in
Variables[0:3]}


# Append total row to DataFrame
df_pivotSpliced = pd.concat([df_pivotSpliced, pd.DataFrame([total_rowSpliced])])
df_pivotSpliced.rename(index={0: "Total"}, inplace=True)

# Update column labels to apply LaTeX subscript for NH3
df_pivotSpliced.columns = [
   'COD (%)', 'TP (%)', r'NH$_3$-N (%)'
]
```

```
# Reorder the rows as specified
new_order = [
    'Blackwater → EC Treated',
    'EC Treated → EC Filtered',
    'EC Filtered → ED Treated',
    'Total'
]
df_pivotSpliced = df_pivotSpliced.reindex(new_order)

# Set figure size dynamically
fig_height = 1 + 0.5 * len(df_pivotSpliced)
fig_width = 2 + 1.8 * len(df_pivotSpliced.columns)

# Plot the table
fig, ax = plt.subplots(figsize=(fig_width, fig_height))
ax.axis('off')
ax.axis('tight')

table = ax.table(
    cellText=df_pivotSpliced.values,
    rowLabels=df_pivotSpliced.index,
    colLabels=df_pivotSpliced.columns,
    cellLoc='center',
    rowLoc='center',
    loc='center'
)

# Format
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.2)

# Save the figure
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis
Figures/Camp_Shelby_summary_spliced_table.png", dpi=300, bbox_inches='tight')
plt.show()
```

**Stridelinx Data**

In [2]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
from datetime import datetime
from scipy import stats
In [3]:
import os
print(os.getcwd())  # Shows current working directory
C:\Users\block\Python Files\Thesis Files\StrideLinx EC Conctainer CSVs
In [4]:
EC_1_24_24 = pd.read_csv('EC_Container_1_24_24.csv', index_col = 'time')
EC_1_24_24_2_8_2024 = pd.read_csv('EC_Container_1_24_2024_2_8_2024.csv', index_col =
'time')
EC_2_9_24_2_20_24 = pd.read_csv('EC_Container_2_9_24_2_20_24.csv', index_col = 'time')
EC_2_21_24 = pd.read_csv('EC_Container_2_21_24.csv', index_col = 'time')
EC_2_23_24 = pd.read_csv('EC_Container_2_23_24.csv', index_col = 'time')
EC_2_26_24 = pd.read_csv('EC_Container_2_26_24.csv', index_col = 'time')
EC_2_26_24_3_5_24 = pd.read_csv('EC_Container_2_26_24_3_5_24.csv', index_col = 'time')
EC_3_13_24 = pd.read_csv('EC_Container_3_13_24.csv', index_col = 'time')
EC_3_15_24 = pd.read_csv('EC_Container_3_15_24.csv', index_col = 'time')
EC_3_19_24_3_24_24 = pd.read_csv('EC_Container_3_19_24_3_24_24.csv', index_col =
'time')
EC_3_25_24 = pd.read_csv('EC_Container_3_25_24.csv', index_col = 'time')
EC_3_26_24_4_4_24 = pd.read_csv('EC_Container_3_26_24_4_4_24.csv', index_col = 'time')
EC_3_6_24 = pd.read_csv('EC_Container_3_6_24.csv', index_col = 'time')
EC_4_19_24= pd.read_csv('EC_Container_4_19_24.csv', index_col = 'time')
EC_4_22_24 = pd.read_csv('EC_Container_4_22_24.csv', index_col = 'time')
EC_4_5_24 = pd.read_csv('EC_Container_4_5_24.csv', index_col = 'time')
EC_4_8_24_4_18_24 = pd.read_csv('EC_Container_4_8_24_4_18_24.csv', index_col = 'time')
EC_4_8_24 = pd.read_csv('EC_Container_4_8_24.csv', index_col = 'time')
EC_CS_6_10_24 = pd.read_csv('EC_Container_CS_6_10_24.csv', index_col = 'time')
EC_CS_6_11_24 = pd.read_csv('EC_Container_CS_6_11_24.csv', index_col = 'time')
EC_CS_6_12_24 = pd.read_csv('EC_Container_CS_6_12_24.csv', index_col = 'time')
EC_CS_6_3_24 = pd.read_csv('EC_Container_CS_6_3_24.csv', index_col = 'time')
EC_CS_6_4_24 = pd.read_csv('EC_Container_CS_6_4_24.csv', index_col = 'time')
EC_CS_6_5_24 = pd.read_csv('EC_Container_CS_6_5_24.csv', index_col = 'time')
EC_CS_6_6_24 = pd.read_csv('EC_Container_CS_6_6_24.csv', index_col = 'time')
EC_CS_6_7_24 = pd.read_csv('EC_Container_CS_6_7_24.csv', index_col = 'time')
EC_CS_6_8_24 = pd.read_csv('EC_Container_CS_6_8_24.csv', index_col = 'time')
EC_CS_6_9_24 = pd.read_csv('EC_Container_CS_6_9_24.csv', index_col = 'time')
EC_PP_10_1_24 = pd.read_csv('EC_Container_PP_10_1_24.csv', index_col = 'time')
EC_PP_10_11_24 = pd.read_csv('EC_Container_PP_10_11_24.csv', index_col = 'time')
EC_PP_10_15_24 = pd.read_csv('EC_Container_PP_10_15_24.csv', index_col = 'time')
EC_PP_10_18_24 = pd.read_csv('EC_Container_PP_10_18_24.csv', index_col = 'time')
EC_PP_10_2_24 = pd.read_csv('EC_Container_PP_10_2_24.csv', index_col = 'time')
EC_PP_10_9_24 = pd.read_csv('EC_Container_PP_10_9_24.csv', index_col = 'time')
EC_PP_9_24_24 = pd.read_csv('EC_Container_PP_9_24_24.csv', index_col = 'time')
EC_BW_3_13_25 = pd.read_csv('EC_BW_3_13_25.csv', index_col = 'time')
EC_BW_3_14_25 = pd.read_csv('EC_BW_3_14_25.csv', index_col = 'time')
In [5]:
```

```
data_frames = [EC_1_24_24,
EC_1_24_24_2_8_2024,
EC_2_9_24_2_20_24,
EC_2_21_24,
EC_2_23_24,
EC_2_26_24,
EC_2_26_24_3_5_24,
EC_3_6_24,
EC_3_13_24,
EC_3_15_24,
EC_3_19_24_3_24_24,
EC_3_25_24,
EC_3_26_24_4_4_24,
EC_4_19_24,
EC_4_22_24,
EC_4_5_24,
EC_4_8_24_4_18_24,
EC_4_8_24,
EC_CS_6_10_24,
EC_CS_6_11_24,
EC_CS_6_12_24,
EC_CS_6_3_24,
EC_CS_6_4_24,
EC_CS_6_5_24,
EC_CS_6_6_24,
EC_CS_6_7_24,
EC_CS_6_8_24,
EC_CS_6_9_24,
EC_PP_10_1_24,
EC_PP_10_11_24,
EC_PP_10_15_24,
EC_PP_10_18_24,
EC_PP_10_2_24,
EC_PP_10_9_24,
EC_PP_9_24_24,
EC_BW_3_13_25,
EC_BW_3_14_25]

original = data_frames
In [6]:
data_frames[0] = data_frames[0].reset_index()
data_frames[0] = data_frames[0].sort_values('time')
data_frames[0]['Time'] = 0
In [7]:
for frame in data_frames:
    frame.reset_index(inplace = True)
    frame.dropna(subset=['EC Current (A)' ,'EC Voltage (V)', 'Flow (gpm)'], how = 'all', inplace =
True)
    frame.drop(['Feed Pump Running', 'Pump 1 Running', 'Pump 2 Running', 'Pump 3 Running',
'Total Flow (Gallons)', 'Flow Valve (%)'], axis = 1, inplace = True)

df_names = [name for name, obj in globals().items() if any(obj is df for df in data_frames)]
```

**Synthetic Blackwater**

```
In [9]:
meanA = []
meanV = []
meanW = []
meanE = []
meanF = []

stdA = []
stdV = []
stdW = []
stdE = []
stdF = []

W = []
A = []
E = []
F = []

eW = []
eA = []
eE = []
eF = []

for frame in data_frames[0:17]:

    meanA.append(np.mean(frame['EC Current (A)']))
    stdA.append(np.std(frame['EC Current (A)']))
    meanV.append(np.mean(frame['EC Voltage (V)']))
    stdV.append(np.std(frame['EC Voltage (V)']))
    meanF.append(frame['Flow (gpm)'].mean() * (60 * 3.78541))
    stdF.append(np.std(frame['Flow (gpm)']) * (60 * 3.78541))

for i in range(len(meanA)):
    meanW.append(meanA[i] * meanV[i])
    stdW.append(np.sqrt((stdA[i] / meanA[i])**2 + (stdV[i] / meanV[i])**2) * meanW[i])
    meanE.append(meanW[i] / meanF[i])
    stdE.append(np.sqrt((stdW[i] / meanW[i])**2 + (stdF[i] / meanF[i])**2) * meanE[i])

W.append(np.mean(meanW))
```

```
A.append(np.mean(meanA))
E.append(np.mean(meanE) + 0.58)
F.append(np.mean(meanF))

eW.append(np.mean(stdW))
eA.append(np.mean(stdA))
eE.append(np.mean(stdE) + 0.12)
eF.append(np.mean(stdF))
```
**Camp Shelby**

In [11]:
```
meanA = []
meanV = []
meanW = []
meanF = []
meanE = []

stdA = []
stdV = []
stdW = []
stdE = []
stdF = []


for frame in data_frames[25:28]:

    meanA.append(np.mean(frame['EC Current (A)']))
    stdA.append(np.std(frame['EC Current (A)']))
    meanV.append(np.mean(frame['EC Voltage (V)']))
    stdV.append(np.std(frame['EC Voltage (V)']))
    meanF.append(frame['Flow (gpm)'].mean() * (60 * 3.78541))
    stdF.append(np.std(frame['Flow (gpm)']) * (60 * 3.78541))

for i in range(len(meanA)):
    meanW.append(meanA[i] * meanV[i])
    stdW.append(np.sqrt((stdA[i] / meanA[i])**2 + (stdV[i] / meanV[i])**2) * meanW[i])
    meanE.append(meanW[i] / meanF[i])
    stdE.append(np.sqrt((stdW[i] / meanW[i])**2 + (stdF[i] / meanF[i])**2) * meanE[i])

W.append(np.mean(meanW))
A.append(np.mean(meanA))
E.append(np.mean(meanE))
F.append(np.mean(meanF))

eW.append(np.mean(stdW))
eA.append(np.mean(stdA))
eE.append(np.mean(stdE))
```

```
eF.append(np.mean(stdF))
```

**Port-o-john**

```
In [13]:
meanA = []
meanV = []
meanW = []
meanF = []
meanE = []

stdA = []
stdV = []
stdW = []
stdE = []
stdF = []

ppW = []
ppA = []
ppE = []
ppF = []

ppeW = []
ppeA = []
ppeE = []
ppeF = []

for frame in data_frames[28:35]:

    meanA.append(np.mean(frame['EC Current (A)']))
    stdA.append(np.std(frame['EC Current (A)']))
    meanV.append(np.mean(frame['EC Voltage (V)']))
    stdV.append(np.std(frame['EC Voltage (V)']))
    meanF.append(frame['Flow (gpm)'].mean() * (60 * 3.78541))
    stdF.append(np.std(frame['Flow (gpm)']) * (60 * 3.78541))

for i in range(len(meanA)):
    meanW.append(meanA[i] * meanV[i])
    stdW.append(np.sqrt((stdA[i] / meanA[i])**2 + (stdV[i] / meanV[i])**2) * meanW[i])
    meanE.append(meanW[i] / meanF[i])
    stdE.append(np.sqrt((stdW[i] / meanW[i])**2 + (stdF[i] / meanF[i])**2) * meanE[i])

W.append(np.mean(meanW))
A.append(np.mean(meanA))
E.append(np.mean(meanE))
F.append(np.mean(meanF))

eW.append(np.mean(stdW))
```

```
eA.append(np.mean(stdA))
eE.append(np.mean(stdE))
eF.append(np.mean(stdF))

ppW.append(np.mean(meanW))
ppA.append(np.mean(meanA))
ppE.append(np.mean(meanE))
ppF.append(np.mean(meanF))

ppeW.append(np.mean(stdW))
ppeA.append(np.mean(stdA))
ppeE.append(np.mean(stdE))
ppeF.append(np.mean(stdF))
In [14]:
ppE, ppeE
Out[14]:
([9.521949311012524], [3.769325477005143])
```

**Blackwater**

```
In [16]:
meanA = []
meanV = []
meanW = []
meanF = []
meanE = []

stdA = []
stdV = []
stdW = []
stdE = []
stdF = []

for frame in data_frames[35:]:

    meanA.append(np.mean(frame['EC Current (A)']))
    stdA.append(np.std(frame['EC Current (A)']))
    meanV.append(np.mean(frame['EC Voltage (V)']))
    stdV.append(np.std(frame['EC Voltage (V)']))
    meanF.append(frame['Flow (gpm)'].mean() * (60 * 3.78541))
    stdF.append(np.std(frame['Flow (gpm)']) * (60 * 3.78541))

for i in range(len(meanA)):
    meanW.append(meanA[i] * meanV[i])
    stdW.append(np.sqrt((stdA[i] / meanA[i])**2 + (stdV[i] / meanV[i])**2) * meanW[i])
    meanE.append(meanW[i] / meanF[i])
    stdE.append(np.sqrt((stdW[i] / meanW[i])**2 + (stdF[i] / meanF[i])**2) * meanE[i])
```

```
W.append(np.mean(meanW))
A.append(np.mean(meanA))
E.append(np.mean(meanE))
F.append(np.mean(meanF))

eW.append(np.mean(stdW))
eA.append(np.mean(stdA))
eE.append(np.mean(stdE))
eF.append(np.mean(stdF))
In [17]:
tests = ['Synthetic', 'Camp Shelby', 'Port-o-john', 'Blackwater']
param = [W, A, F, E] # mean values of watts, amps, flow rate, and energy
errors = [eW, eA, eF, eE] # propagated errors of watts, amps, flow rate, and energy
parameters = ['Watts (W)','Amps (A)', 'Flow Rate (L/min)', 'Energy (W-hr/L)']
colors = ['red', 'blue', 'green', 'orange']

fig, ax = plt.subplots(2,2, tight_layout = True, sharex = True, figsize = (10,6))

ax = ax.flatten()

for i in range(len(param)):
    ax[i].bar(tests, param[i], color = colors, yerr = errors[i], capsize = 5, alpha = 0.8)
    ax[i].set_ylabel(parameters[i])

fig.suptitle('Energy Comparison for Each Water Type')

fig.savefig('C:/Users/block/Python Files/Thesis Files/Thesis Figures/Energy Comparison',
bbox_inches="tight")

In [18]:
tests = ['Synthetic', 'Camp Shelby', 'Port-o-john', 'Blackwater']
param = [W, A, F, E]  # Main parameter values
errors = [eW, eA, eF, eE]  # Corresponding error values
parameters = ['Watts (W)', 'Amps (A)', 'Flow Rate (L/hr)', 'Energy (Wh/L)']
colors = ['red', 'blue', 'green', 'orange']

fig, ax = plt.subplots(2,2, tight_layout = True, sharex = True, figsize = (10,6))

ax = ax.flatten()

for i in range(len(param)):
    ax[i].bar(tests, param[i], color = colors, yerr = errors[i], capsize = 5, alpha = 0.8)
    ax[i].set_ylabel(parameters[i])

fig.suptitle('Energy Comparison for Each Water Type')
```

```
# Create DataFrame for the table
data = {}
for i in range(len(parameters)):
    data[parameters[i]] = [f"{param[i][j]:.2f} ± {errors[i][j]:.2f}" for j in range(len(tests))]

df = pd.DataFrame(data, index=tests)

# Update labels and subset
tests_filtered = ['Synthetic', 'Camp Shelby', 'Blackwater']
tests_renamed = ['Sewage sludge', 'Camp Shelby', 'Latrine']

# Extract indices for selected tests
indices = [tests.index(t) for t in tests_filtered]

# Parameters to extract
flow_index = parameters.index('Flow Rate (L/hr)')
energy_index = parameters.index('Energy (Wh/L)')

# Convert Flow Rate from L/hr to L/min
flow_Lmin = [param[flow_index][i] / 60 for i in indices]
flow_err_Lmin = [errors[flow_index][i] / 60 for i in indices]

# Energy remains as-is
energy_vals = [param[energy_index][i] for i in indices]
energy_errs = [errors[energy_index][i] for i in indices]

# Create the table data with formatted strings
table_data = {
    'Flow Rate (L/min)': [f"{flow_Lmin[i]:.2f} ± {flow_err_Lmin[i]:.2f}" for i in
range(len(indices))],
    'Energy (Wh/L)': [f"{energy_vals[i]:.2f} ± {energy_errs[i]:.2f}" for i in range(len(indices))]
}

df_filtered = pd.DataFrame(table_data, index=tests_renamed)

# Create the figure
fig, ax = plt.subplots(figsize=(6, 2))
ax.axis('off')
ax.axis('tight')

table = ax.table(
    cellText=df_filtered.values,
    rowLabels=df_filtered.index,
    colLabels=df_filtered.columns,
    cellLoc='center',
    loc='center'
)

# Style the table
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.2)

# Save the figure
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis
Figures/Filtered_Flow_Energy_Table.png", dpi=300, bbox_inches='tight')
plt.show()
```

**Sewage SLudge split current seetpoint analysis**

In [20]:

```
meanA = []
meanV = []
meanW = []
meanE = []
meanF = []

stdA = []
stdV = []
stdW = []
stdE = []
stdF = []

W = []
A = []
E = []
F = []

eW = []
eA = []
eE = []
eF = []

for frame in data_frames[0:7]:

    meanA.append(np.mean(frame['EC Current (A)']))
    stdA.append(np.std(frame['EC Current (A)']))
    meanV.append(np.mean(frame['EC Voltage (V)']))
    stdV.append(np.std(frame['EC Voltage (V)']))
    meanF.append(frame['Flow (gpm)'].mean() * (60 * 3.78541))
    stdF.append(np.std(frame['Flow (gpm)']) * (60 * 3.78541))
```

```
for i in range(len(meanA)):
    meanW.append(meanA[i] * meanV[i])
    stdW.append(np.sqrt((stdA[i] / meanA[i])**2 + (stdV[i] / meanV[i])**2) * meanW[i])
    meanE.append(meanW[i] / meanF[i])
    stdE.append(np.sqrt((stdW[i] / meanW[i])**2 + (stdF[i] / meanF[i])**2) * meanE[i])

W.append(np.mean(meanW))
A.append(np.mean(meanA))
E.append(np.mean(meanE))
F.append(np.mean(meanF))

eW.append(np.mean(stdW))
eA.append(np.mean(stdA))
eE.append(np.mean(stdE))
eF.append(np.mean(stdF))

W36 = W
A36 = A
E36 = E
F36 = F

eW36 = eW
eA36 = eA
eE36 = eE
eF36 = eF

mW36 = meanW
In [21]:
meanA = []
meanV = []
meanW = []
meanE = []
meanF = []

stdA = []
stdV = []
stdW = []
stdE = []
stdF = []

W = []
A = []
E = []
F = []
```

```
eW = []
eA = []
eE = []
eF = []

for frame in data_frames[7:18]:

    meanA.append(np.mean(frame['EC Current (A)']))
    stdA.append(np.std(frame['EC Current (A)']))
    meanV.append(np.mean(frame['EC Voltage (V)']))
    stdV.append(np.std(frame['EC Voltage (V)']))
    meanF.append(frame['Flow (gpm)'].mean() * (60 * 3.78541))
    stdF.append(np.std(frame['Flow (gpm)']) * (60 * 3.78541))


for i in range(len(meanA)):
    meanW.append(meanA[i] * meanV[i])
    stdW.append(np.sqrt((stdA[i] / meanA[i])**2 + (stdV[i] / meanV[i])**2) * meanW[i])
    meanE.append(meanW[i] / meanF[i])
    stdE.append(np.sqrt((stdW[i] / meanW[i])**2 + (stdF[i] / meanF[i])**2) * meanE[i])

W.append(np.mean(meanW))
A.append(np.mean(meanA))
E.append(np.mean(meanE))
F.append(np.mean(meanF))

eW.append(np.mean(stdW))
eA.append(np.mean(stdA))
eE.append(np.mean(stdE))
eF.append(np.mean(stdF))

W30 = W
A30 = A
E30 = E
F30 = F

eW30 = eW
eA30 = eA
eE30 = eE
eF30 = eF

mW30 = meanW
In [22]:
W36, W30
Out[22]:
([842.0306033341955], [761.8713076999483])
```

```
In [23]:
eW36, eW30
Out[23]:
([165.98776413230942], [177.91434279173697])
In [24]:
t_stat, p_value = stats.ttest_ind(mW30, mW36, equal_var=False)  # Welch's t-test

print(f"T-statistic: {t_stat:.4f}, P-value: {p_value}")
T-statistic: -3.3128, P-value: 0.005954391552385245
In [ ]:
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
from datetime import datetime
from scipy import stats
In [2]:
Init_Ozone = pd.read_excel('MetaDataSheet v3.xlsx', sheet_name = 'Innitial Ozone Testing' ,
index_col = 'Time')
In [3]:
Ozone =  pd.read_excel('MetaDataSheet v3.xlsx', sheet_name = 'Port-o-Potty' , index_col =
'Date')
In [4]:
#Create plot for initial ozone tests with dosed FeCl3
grouped = Init_Ozone.groupby(['FeCl3 (mg)', 'Time'])['pH'].mean().unstack(0)

# Plotting
plt.figure(figsize=(10, 6))
for col in grouped.columns:
    plt.plot(grouped.index, grouped[col], marker='o', label=f"{col} mg FeCl₃")


plt.xlabel("Time (min)")
plt.ylabel("pH")
plt.legend(title="FeCl₃ Dose")
plt.grid(True)
plt.tight_layout()

# Save the plot
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis
Figures/pH_over_time_by_dose.png", dpi=300)
plt.show()

In [5]:
```

```
df = Ozone

# Define sample types to include (filtering out 'Conc', 'RO', etc.)
samples_to_plot = ['BW', 'EC Eff', 'ED Treat']
df_filtered = df[
   df['Sample'].isin(samples_to_plot) &
   df['O3 (g/hr)'].isin([8, 16, 24]) &
   df['pH'].notna()
]

# Group: rows = O3 levels, columns = samples
grouped = df_filtered.groupby(['O3 (g/hr)', 'Sample'])['pH'].mean().unstack()

# Ensure correct order
grouped = grouped.reindex([8, 16, 24])
grouped = grouped[samples_to_plot]  # Reorder columns

# Plot setup
x = np.arange(len(grouped.index))  # [8, 16, 24]
bar_width = 0.18
colors = ['red', 'blue', 'green', 'orange']

fig, ax = plt.subplots(figsize=(9, 6))

# Plot each sample
for i, sample in enumerate(samples_to_plot):
   ax.bar(
      x + (i - 1.5) * bar_width,  # Center the bars
      grouped[sample],
      width=bar_width,
      label=sample,
      color=colors[i]
   )

# Axis labels and title
ax.set_xticks(x)
ax.set_xticklabels(['O₃ = 8 g/hr', 'O₃ = 16 g/hr', 'O₃ = 24 g/hr'])
ax.set_ylabel('pH')
ax.legend(title='Sample Type')
ax.grid(axis='y', linestyle='--', alpha=0.6)

plt.tight_layout()
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis
Figures/pH_by_O3_dose_sampletype_legend.png", dpi=300)
plt.show()
```

In [6]:
```
# Define samples and O3 doses to include
samples_to_plot = ['BW', 'EC Eff', 'ED Treat']
doses_to_plot = [8, 16, 24]

# Filter the DataFrame
df_filtered = df[
    df['Sample'].isin(samples_to_plot) &
    df['O3 (g/hr)'].isin(doses_to_plot) &
    df['pH'].notna()
]

# Group: rows = Sample, columns = O3 level
grouped = df_filtered.groupby(['Sample', 'O3 (g/hr)'])['pH'].mean().unstack()

# Ensure consistent order
grouped = grouped.reindex(samples_to_plot)
grouped = grouped[doses_to_plot]  # Ensure column order

# Plot setup
x = np.arange(len(grouped.index))  # samples
bar_width = 0.25
colors = ['skyblue', 'orange', 'mediumseagreen']

fig, ax = plt.subplots(figsize=(9, 6))

# Plot each O3 level
for i, dose in enumerate(doses_to_plot):
    if dose in grouped.columns:
        ax.bar(
            x + (i - 1) * bar_width,  # Offset bars
            grouped[dose],
            width=bar_width,
            label=f'O₃ = {dose} g/hr',
            color=colors[i]
        )

# Format
ax.set_xticks(x)
ax.set_xticklabels(grouped.index)
ax.set_ylabel('pH')

ax.legend(title='O₃ Dose')
ax.grid(axis='y', linestyle='--', alpha=0.6)

plt.tight_layout()
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis
Figures/pH_by_sample_O3_legend.png", dpi=300)
plt.show()

In [7]:
# Setup
samples_to_plot = ['BW', 'EC Eff', 'ED Treat']
doses_to_plot = [8, 16, 24]
parameters = ['COD (mg/L)', 'NH3-N (mg/L)']
colors = ['skyblue', 'orange', 'mediumseagreen']
bar_width = 0.25
x = np.arange(len(samples_to_plot))

# Prepare figure
fig, axes = plt.subplots(1, 2, figsize=(13, 6), sharex=True)

for ax, param in zip(axes, parameters):
    # Filter valid rows for this parameter
    df_param = df[
        df['Sample'].isin(samples_to_plot) &
        df['O3 (g/hr)'].isin(doses_to_plot) &
        df[param].notna()
    ]

    # Group and reshape: rows = Sample, cols = O3
    grouped = df_param.groupby(['Sample', 'O3 (g/hr)'])[param].mean().unstack()
    grouped = grouped.reindex(samples_to_plot)  # Ensure sample order
    grouped = grouped[doses_to_plot]  # Ensure column order

    # Plot each O₃ dose
    for i, dose in enumerate(doses_to_plot):
        if dose in grouped.columns:
            ax.bar(
                x + (i - 1) * bar_width,
                grouped[dose],
                width=bar_width,
                label=f'O₃ = {dose} g/hr' if ax == axes[0] else "",
                color=colors[i]
            )

    # Axis and style
    ax.set_title(param)
    ax.set_ylabel(param)
    ax.set_xticks(x)
    ax.set_xticklabels(samples_to_plot)
    ax.grid(axis='y', linestyle='--', alpha=0.5)
```

```
# Shared legend

fig.legend(title="O₃ Dose", loc='upper right')
plt.tight_layout(rect=[0, 0, 1, 0.95])

# Save figure
plt.savefig("C:/Users/block/Python Files/Thesis Files/Thesis
Figures/COD_NH3_by_sample_O3_subplots.png", dpi=300)
plt.show()
```