



# Using Parameter Sweep in WaterTAP to Analyze New Water Treatment Technologies

Kinshuk Panda,<sup>1</sup> Bernard Knueven,<sup>1</sup> Ethan Young,<sup>1</sup> Sam Helman,<sup>1</sup> Jeffery Allen,<sup>1</sup> and Alexander V. Dudchenko<sup>2</sup>

*1 National Renewable Energy Laboratory*

*2 SLAC National Accelerator Laboratory*

**NREL is a national laboratory of the U.S. Department of Energy  
Office of Energy Efficiency & Renewable Energy  
Operated by the Alliance for Sustainable Energy, LLC**

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at [www.nrel.gov/publications](http://www.nrel.gov/publications).

Contract No. DE-AC36-08GO28308

**Technical Report**  
NREL/TP-2C00-90162  
September 2024



# Using Parameter Sweep in WaterTAP to Analyze New Water Treatment Technologies

Kinshuk Panda,<sup>1</sup> Bernard Knueven,<sup>1</sup> Ethan Young,<sup>1</sup> Sam Helman,<sup>1</sup> Jeffery Allen,<sup>1</sup> and Alexander V. Dudchenko<sup>2</sup>

*1 National Renewable Energy Laboratory*

*2 SLAC National Accelerator Laboratory*

## **Suggested Citation**

Panda, Kinshuk, Bernard Knueven, Ethan Young, Sam Helman, Jeffery Allen, and Alexander Dudchenko. 2024. *Using Parameter Sweep in WaterTAP to Analyze New Water Treatment Technologies*. Golden, CO: National Renewable Energy Laboratory. NREL/TP-2C00-90162. <https://www.nrel.gov/docs/fy24osti/90162.pdf>.

**NREL is a national laboratory of the U.S. Department of Energy  
Office of Energy Efficiency & Renewable Energy  
Operated by the Alliance for Sustainable Energy, LLC**

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at [www.nrel.gov/publications](http://www.nrel.gov/publications).

Contract No. DE-AC36-08GO28308

**Technical Report**  
NREL/TP-2C00-90162  
September 2024

National Renewable Energy Laboratory  
15013 Denver West Parkway  
Golden, CO 80401  
303-275-3000 • [www.nrel.gov](http://www.nrel.gov)

## NOTICE

This material is based upon work supported by the National Alliance for Water Innovation (NAWI), funded by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy (EERE), Advanced Manufacturing Office, under Funding Opportunity Announcement Number DE-FOA-0001905. M.S.M also acknowledges funding from the National Science Foundation under award number CBET-1554117 and U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under award number DE-EE0008397. A portion of this research was performed using computational resources sponsored by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy and located at the National Renewable Energy Laboratory. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. ChatGPT was used to assist in the preparation of the Introduction, Methods, and Results chapter of the manuscript. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at [www.nrel.gov/publications](http://www.nrel.gov/publications).

U.S. Department of Energy (DOE) reports produced after 1991 and a growing number of pre-1991 documents are available free via [www.OSTI.gov](http://www.OSTI.gov).

*Cover Photos by Dennis Schroeder: (clockwise, left to right) NREL 51934, NREL 45897, NREL 42160, NREL 45891, NREL 48097, NREL 46526.*

NREL prints on paper that contains recycled content.

## Executive Summary

We describe a powerful and generalized parameter sweep tool in this report that was originally developed to analyze the performance of existing and novel water treatment models being developed in WaterTAP (Beattie et al. 2021). Since WaterTAP is built upon IDAES (Lee et al. 2021; Lee et al. 2018; *The IDAES Process Systems Engineering Framework*) and Pyomo (Hart, Watson, and Woodruff 2011; Bynum et al. 2021), the parameter sweep tool can be used to systematically explore and debug the behavior of most Pyomo and IDAES numerical models.

In order to enable meaningful analyses, the parameter sweep tool has been designed with the following features:

- *Model flexibility*: The parameter sweep tool does not enforce any restrictions on the types of models that can be used with it. As long as a Pyomo model can be solved and the parameter is active and mutable, the tool only needs functions that describe how to run the model, the sweep parameters, and the output quantities of interest.
- *Flexible sampling*: The parameter sweep tool has inbuilt functions to generate samples from a random distribution or a multidimensional Euclidean space. Furthermore, the users have to ability to supply samples generated from a tool of their choice
- *Multiple sweep types*: A user can choose from one of 3 types of parameter sweeps depending on their needs.
- *Detailed outputs*: Outputs generated by the parameter sweep tool can be stored in detailed H5 file or user-friendly CSV files for post processing.
- *Parallel computing*: The parameter sweep supports shared and distributed memory parallel computing to enable the use of high performance computers (HPC) for large-scale analyses.
- *Modular*: The parameter sweep tool is self-contained and can easily be integrated within an outer-loop analysis or as desired by the user.
- *Ease of use*: The tool is well documented and a simple sweep can be easily executed by following the online documentation in a few lines of code.

We demonstrate the use of the parameter sweep tool on a simple water treatment system from the WaterTAP repository and show its parallel scaling performance on an Apple laptop and NREL's Eagle HPC. The parameter sweep tool is actively being used with models currently being developed within WaterTAP and we expect its use to grow beyond it to other IDAES and Pyomo models.

## Acknowledgments

This material is based upon work supported by the National Alliance for Water Innovation (NAWI), funded by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy (EERE), Advanced Manufacturing Office, under Funding Opportunity Announcement Number DE-FOA-0001905. M.S.M also acknowledges funding from the National Science Foundation under award number CBET-1554117 and U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under award number DE-EE0008397. A portion of this research was performed using computational resources sponsored by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy and located at the National Renewable Energy Laboratory. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. ChatGPT was used to assist in the preparation of the Introduction, Methods, and Results chapter of the manuscript. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

# Table of Contents

<b>Executive Summary</b> . . . . .	<b>iv</b>
<b>Acknowledgments</b> . . . . .	<b>v</b>
<b>1 Introduction</b> . . . . .	<b>1</b>
<b>2 Methods</b> . . . . .	<b>3</b>
2.1 Model Compatibility . . . . .	3
2.1.1 WaterTAP Models . . . . .	3
2.2 The Parameter Sweep Tool . . . . .	4
2.2.1 Sampling Types . . . . .	4
2.2.2 Sweep Types . . . . .	4
2.3 Parallel Manager . . . . .	6
<b>3 Results</b> . . . . .	<b>8</b>
3.1 Water Treatment Model . . . . .	8
3.2 Single-Variable Parameter Sweep . . . . .	8
3.3 Multi-Variable Parameter Sweep . . . . .	8
3.4 Differential Parameter Sweep . . . . .	10
3.5 Parallel Scalability . . . . .	10
<b>4 Conclusions</b> . . . . .	<b>13</b>
<b>References</b> . . . . .	<b>16</b>
<b>Appendix A Parameter Sweep Options</b> . . . . .	<b>17</b>
<b>Appendix B Generative AI Prompts and Responses</b> . . . . .	<b>19</b>

# List of Figures

Figure 1. Full treatment train for seawater desalination. The dashed boxes indicate a basic unit of operation and the icons represent the models (Bartholomew 2022). . . . .	3
Figure 2. Flow chart depicting the simplest parameter sweep algorithm. . . . .	5
Figure 3. Flow chart depicting the recursive parameter sweep algorithm that guarantees a specified number of samples. . . . .	6
Figure 4. Flow chart depicting the parameter sweep algorithm that enables differential sampling at a nominal value. . . . .	7
Figure 5. Diagram of RO with ERD flowsheet (Atia). . . . .	8
Figure 6. Variation in LCOW, specific energy consumption, and membrane area as a function of membrane water permeability. . . . .	9
Figure 7. Two variable parameter sweep that shows variation in LCOW over the swept domain. . . . .	10
Figure 8. Box plot from a stochastic value of innovation analysis using the differential parameter sweep. The narrower the box plot, the more consistent improvement in LCOW should be expected. The whiskers denote the best and worse cases. . . . .	11
Figure 9. Strong scaling results when using concurrent futures for running parameter sweep with on the RO with ERD flowsheet with 2500 samples. . . . .	12
Figure 10. Strong scaling results when using MPI for running parameter sweep on NREL’s Eagle HPC. Two flowsheets, RO with ERD and LSRRO, are considered for this analysis. Both exhibit linear scaling. . . . .	12

## List of Tables

Table 1. A list of sampling types currently supported by the WaterTAP parameter sweep tool. . . . .	4
Table A.1. Parameter Sweep inputs that leverage Pyomo’s configuration system . . . . .	17
Table A.2. Arguments to the <code>parameter_sweep</code> method. . . . .	18
Table A.3. Configuration options for <code>ParameterSweepWriter</code> class . . . . .	18

# 1 Introduction

Advances in computing and numerical modeling have enabled researchers to develop and rigorously assess novel technologies across multiple domains. These technologies often depend on several input parameters and their resulting impact on the output is not known ahead of time. This has given rise to the field of optimization, sensitivity analysis, and uncertainty quantification, where several techniques have been developed to quantify how the outputs change with a change in inputs (Nocedal and Wright 2006; Sobol 2001; Saltelli et al. 2007; Cacuci 2003; Smith 2013; Razavi and Gupta 2015). Parameter sweep is one way of performing optimization, sensitivity analysis, and uncertainty quantification where experiments are run repeatedly with a different set of input parameters (Youn and Kaiser 2010; Ino et al. 2014; Pacini, Mateos, and Garino 2013). These input parameter values may either be selected from a pre-determined multi-dimensional grid space, e.g., linear space, geometric space, log space, or sampled from a probability distribution, e.g., normal or uniform distribution.

Parameter sweep has, in particular, been very effective in identifying critical areas of innovation in emerging water treatment technologies (Dudchenko, Bartholomew, and Mauter 2021), including osmotically assisted reverse osmosis (OARO) (Bartholomew et al. 2017; Bartholomew, Siefert, and Mauter 2018), high-pressure reverse osmosis (HPRO) (Davenport et al. 2018; Davenport 2020), low-salt-rejection reverse osmosis (LSRRO) (Atia et al. 2023), cascading osmotically mediated reverse osmosis (COMRO) (Chen and Yip 2018; Atia, Yip, and Fthenakis 2021), mechanical vapor compression (Onishi et al. 2017), electrodialysis (Patel et al. 2024), and membrane distillation (Bartholomew et al. 2020). For example, Atia et al. (2023) demonstrated that increasing the maximum allowable pressure in LSRRO can reduce process levelized cost of producing water (LCOW) by 22%, and Bartholomew, Siefert, and Mauter (2018) used parameter sweep to quantify the impact of improving membrane structure, burst pressure, and component costs on minimizing the cost of OARO.

Some of the aforementioned work leveraged WaterTAP (Beattie et al. 2021; *WaterTAP development repository*; Bartholomew 2022) which is an open-source Python (Van Rossum and Drake 2009) package created to facilitate modeling and analysis of water treatment technologies. WaterTAP leverages powerful equation-oriented modeling methods to solve large and complicated models efficiently, thereby enabling the application of parameter sweep methodology to the analysis of emerging processes with a high number of dimensions and level of detail.

WaterTAP leverages the open-source Institute for the Design of Advanced Energy Systems (IDAES) Integrated Platform (Lee et al. 2018; Lee et al. 2021; *The IDAES Process Systems Engineering Framework*) which is a process modeling framework that allows modular construction of engineering process systems for a wide variety of industrial applications, including water treatment. Various components of a water treatment system can be modeled as unit models based on the IDAES framework. These unit models can then be combined into a process flowsheet to represent the operation of a water treatment train.

The equation oriented modeling and optimization of process systems within IDAES is enabled by Pyomo (Hart, Watson, and Woodruff 2011; Bynum et al. 2021) which is a python-based open-source high-level framework for modeling algebraic optimization problems. Pyomo allows a modeler to define the underlying mathematical problem, initialize, modify, and solve it with an external solver. It allows for a large optimization problem to be constructed using smaller sub-problem blocks which is very amenable to modeling processes within the IDAES framework. Pyomo provides the interface to multiple open and closed solvers, including open-source IPOPT (Wächter and Biegler 2006), that can be used to initialize and solve the numerical problem.

In this report, we describe a modular and flexible parameter sweep tool that was originally developed for analyzing WaterTAP models but can also be used with any Pyomo or IDAES model. This parameter sweep tool accepts functions that build, modify, and solve a Pyomo model, and can also leverage parallel processing for multiple parameter combinations at once. A user may perform sweeps for multi-objective optimization where a single parameter metric is changed and the whole process can be re-optimized for the target objective (e.g., minimize LCOW) and desired treatment target (e.g., feed water composition and water recovery). This ensures that, as parameters are changed, the impacts can be directly compared since the overall performance target of the process does not change.

The parameter sweep module has sampling tools that allow a user to sample the parameter space from either a probability distribution, a Latin hypercube, or a Euclidean space. The tool, furthermore, boasts features that allow a user to customize sweeps with hard-to-solve Pyomo problems, something that is often true for water treatment models.



The remainder of the report is organized as follows. Chapter 2 describes the parameter sweep tool in detail. Chapter 3 demonstrates how the parameter sweep tool can be used for analyzing WaterTAP models. Finally, Chapter 4 summarizes and highlights the capability of the parameter sweep tool and provides insight for future development.

## 2 Methods

We briefly describe the models compatible with the parameter sweep tool below before delving into the details in the subsequent section.

### 2.1 Model Compatibility

While the parameter sweep tool was originally intended for use with WaterTAP models, it is immensely flexible and compatible with any Pyomo or IDAES model. In fact, the unit tests that check the functionality of the tool employ simple problems constructed directly in Pyomo.

In order to conduct a successful parameter sweep, a Pyomo, IDAES, or WaterTAP model should be well-posed and have a feasible solution that can be obtained using a compatible optimizer such as IPOPT (Wächter and Biegler 2006). Any variable, expression, parameter, or objective in the aforementioned model that is active and mutable can be swept over using the parameter sweep tool regardless of the model's complexity. The parameter sweep expects some helper functions and corresponding keyword arguments that describe how to build the model, how to initialize it, the sweep parameters, which outputs to record, and how to solve or optimize the model.

#### 2.1.1 WaterTAP Models

As mentioned in Chapter 1, WaterTAP enables techno-economic assessment of a broad array of water treatment technologies. It contains a library of various technology models, e.g., ion-exchange, granulated activated carbon adsorption, reverse osmosis, etc., that can be combined to simulate specialized full treatment trains. Additionally, WaterTAP currently contains example flowsheets for conventional treatment trains that can be used as a baseline to compare emerging technologies.

WaterTAP technology models can be further broken down into basic units of operations that can be split into 3 major categories, viz., property models, unit models, and costing models. Property models describe the physical properties, e.g., density, temperature, viscosity, etc., and chemical properties, e.g., kinetic and equilibrium reactions, of water. They can also describe biological processes. Unit models describe the performance of the equipment such as efficiency, water recovery, and energy consumption. Lastly, costing models describe the capital and operational costs associated with the treatment system. These units of operations can be combined into a full treatment train as shown in Fig. 1. Users can implement additional models using the Pyomo and IDAES API. Finally, the flowsheet is optimized using the nonlinear optimizers, e.g., IPOPT (Wächter and Biegler 2006) with HSL solvers (*HSL(2013). A collection of Fortran codes for large scale scientific computation*).

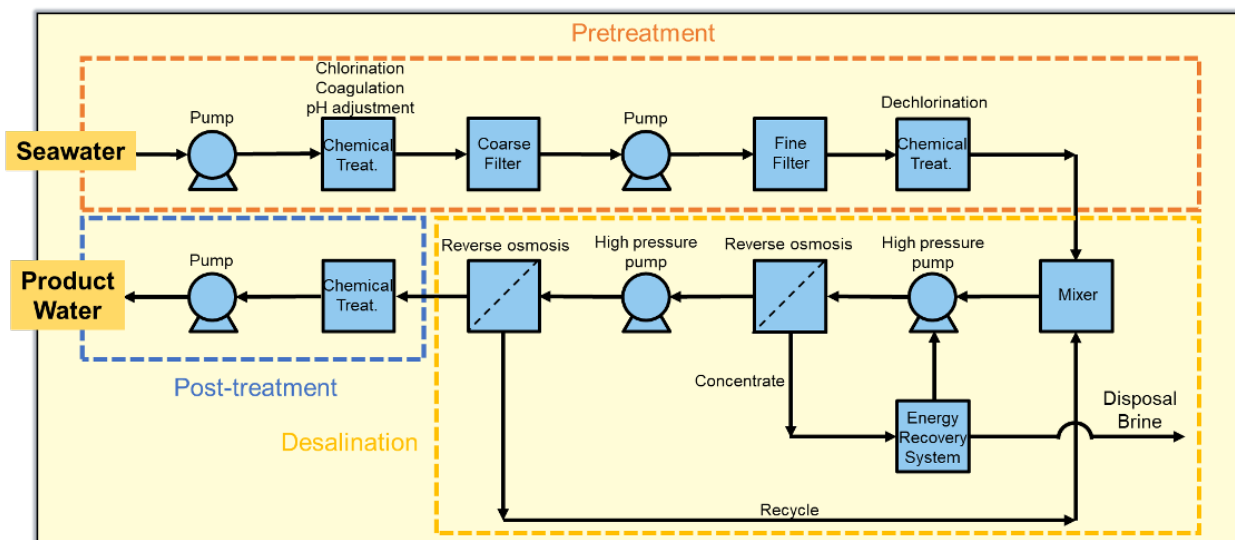


Figure 1. Full treatment train for seawater desalination. The dashed boxes indicate a basic unit of operation and the icons represent the models (Bartholomew 2022).

## 2.2 The Parameter Sweep Tool

Robustness and ease-of-use were especially important considerations while developing the parameter sweep tool. Specifically, it has been designed to be approachable to researchers and students with basic python programming experience while also offering advanced users the ability to add custom functions for looping that would make their analyses more meaningful. The user can invoke a parameter sweep for a flowsheet using one function call.

We discuss the two major concepts that dictate how the parameter sweep tool behaves during an analysis, viz, the sampling type and sweep type. The former dictates what types of samples are generated, while the latter dictates how a sweep is executed.

### 2.2.1 Sampling Types

To facilitate parametric sweeps, the parameter sweep tool enables "fixed" sampling and "random" sampling. Within the fixed sampling category, the parameter sweep allows the creation of evenly spaced samples on a linear or log scale. While, in random sampling, the user can sample input parameters from normal and uniform distributions. The parameter sweep tool uses NumPy (Harris et al. 2020) functions to generate these samples. In addition, users can employ Latin hypercube sampling (LHS) that is enabled through the IDAES API for efficient sampling. Lastly, users also have the ability to generate samples using the tool of their choice and supply them to the parameter sweep tool. Table 1 summarizes the different sampling techniques that are currently supported by the parameter sweep tool.

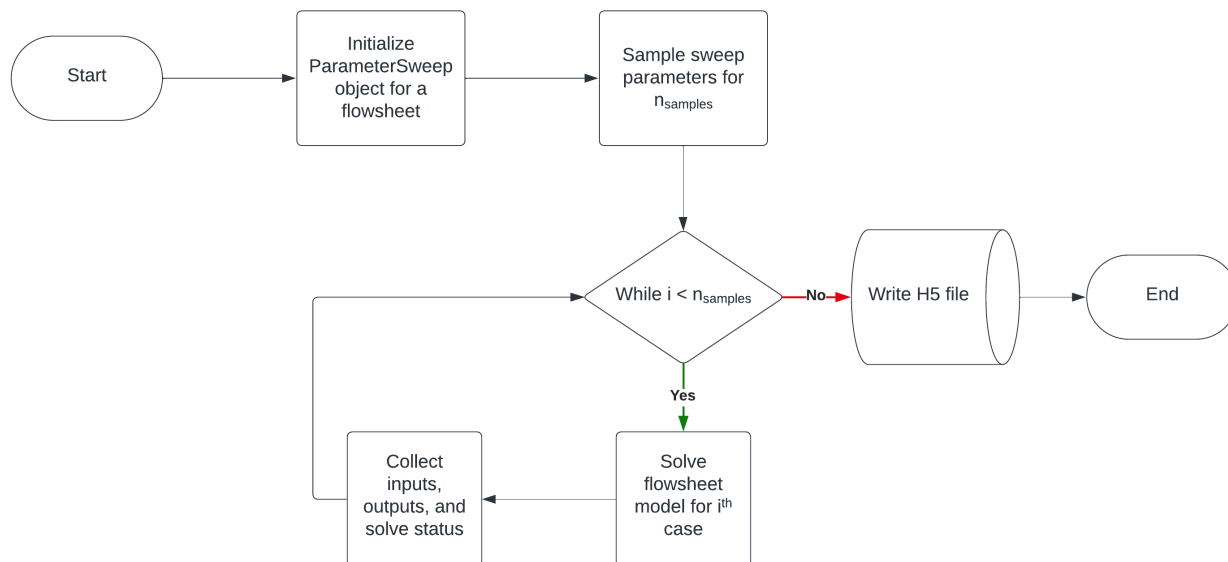
When studying the impact of multiple parameters on a quantity of interest, a user can pick any sampling type for the different parameters within fixed sampling or random sampling category. However, they cannot mix sampling types between the two categories, i.e., all the sampling types have to be either fixed or random. This is because of the way the samples are constructed for the parameter sweep. Fixed sampling uses an outer product to construct a mesh-grid of sampled points while random sampling samples the parameters from their respective distribution or hypercube and stacks them into a multi-dimensional array.

### 2.2.2 Sweep Types

The simplest analysis a user can perform is the basic parameter sweep shown in Figure 2. Here, the tool samples the sweep parameters, solves the flowsheet with the sampled parameter values, and finally records the outputs in an H5 or a CSV file. To get started with a basic analysis, the user needs a function that creates a python dictionary containing the sampling information for the sweep parameters. Next, they need to provide how many samples they want in their sweep, separate functions and keyword arguments for building the flowsheet model, initializing it, optimizing the model, and lastly, output H5 and/or CSV file name. The user can choose to specify the outputs they wish to record in the output file. This can be done by providing a function that points to the objects in the Pyomo model. If no output function is specified, the parameter sweep records all the variables, expressions, and objectives in the output file. Lastly, users can execute the parameter sweep in parallel by specifying a parallel backend and the number of parallel processes. This reduces the total compute times significantly for complex models and is discussed further in Section 2.3.

**Table 1. A list of sampling types currently supported by the WaterTAP parameter sweep tool.**

Category	Sampling Type	Class Name	Description
Fixed Sampling	Linear	LinearSample	Creates evenly spaced samples on a linear scale
	Geometric	GeomSample	Creates evenly spaced samples on a log scale through geometric progression
	Reverse geometric	ReverseGeomSample	Creates sampling on the log scale where more samples are created close to the upper limit.
	Pre-determined	PredeterminedFixedSample	Accepts an array or list of values to work within the fixed sampling framework
Random Sampling	Normal sampling	NormalSample	Samples from a normal distribution
	Uniform	UniformSample	Samples from a uniform distribution
	Latin Hypercube	LatinHypercubeSample	Generates samples from a Latin hypercube
	Pre-determined	PredeterminedRandomSample	Accepts an array or list of values to work within the random sampling framework



**Figure 2. Flow chart depicting the simplest parameter sweep algorithm.**

For complex or computationally expensive water treatment trains where certain model configurations generated as part of the parameter sweep may not be feasible or extremely hard to solve numerically, the parameter sweep provides additional tools to aid analysis. This includes the ability to reinitialize the model with a user specified function (including reading values from file), logging the model state, specify probe functions for inexpensively rejecting invalid configurations, and publish progress when using the graphical user interface (GUI). Keyword arguments that enable these features are described in Appendix A.

Alternatively, when investigating the impact of novel technologies where parameters are drawn from a probability distribution or when the bounds on the parameters are not clearly defined, a user can use the recursive parameter sweep tool, shown in Figure 3, to conduct statistically significant analysis. The recursive parameter tool runs the simple parameter sweep multiple times in a loop until the user-specified number of samples has been successfully solved or once the success rate for the recursive loop falls below 10%. All the features available with the simple parameter sweep tool are also available with the recursive parameter sweep.

Lastly, as has been mentioned before, a user may be interested in getting output sensitivity to the various parameters for stochastic value of innovation analyses. This is enabled by calling the differential parameter sweep function, shown in Figure 4, where the tool loops over the nominally sampled values, but at each nominal iteration, conducts another sweep on a parameter while holding other parameters at the nominal values for the iteration.

All three types of parameter sweep record the inputs, outputs, and whether the model was solved successfully or not in a nested Python results dictionary at each iteration. The differential parameter sweep type also stores mapping indices that links the nominal sweep values with their differential sweep values. Once the tool has swept over the parameters, the users then have the option to write the results dictionary to an H5 or a CSV file for post processing.

Outputs stored in the H5 file are more detailed than in the CSV output file. The entire python results dictionary, i.e., the input sweep parameters, model outputs, successful solve status, and for the differential sweep, the mapping indices, gets stored in the H5 file as groups and datasets. The input sweep parameters and output groups in the H5 file contain additional metadata corresponding to the Pyomo object. This includes the full name of the parameter, units, upper and lower bounds, and the actual values from the sweep. The successful solve status is a boolean list that states whether the flowsheet for a given parameter combination successfully solved or not. Mapping indices exist in the case of differential parameter sweep, this is a set of two integer arrays that keeps track of the nominal indices and their corresponding differential sweeps. The CSV output file, on the other hand, simply stores the input sweep parameters and output values in a dataframe-like structure.

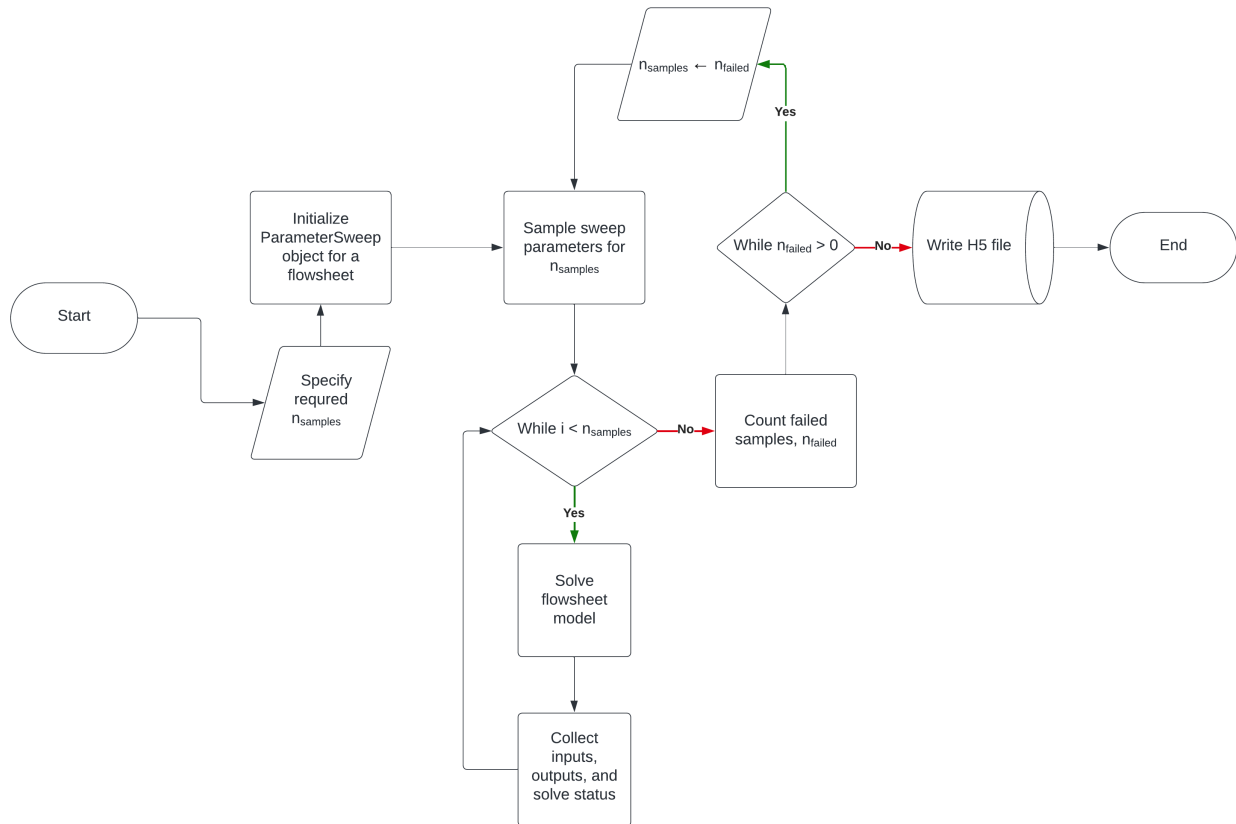


Figure 3. Flow chart depicting the recursive parameter sweep algorithm that guarantees a specified number of samples.

## 2.3 Parallel Manager

The parameter sweep tool can leverage multiple CPU cores for solving multiple parameter combinations in parallel. This is enabled through a parallel manager, which is an abstraction layer that allows the user to run their analysis on different types of computing resources, e.g., laptops and desktops, high performance computers (HPC), and the cloud. The parallel manager allows the user to run the parameter sweep in parallel using Message Passing Interface (MPI) (Dalcín, Paz, and Storti 2005), Python multiprocessing, concurrent futures, and Ray (Moritz et al. 2018), in addition to running in serial.

The user can specify which of these parallel backends and how many parallel processes they wish to use at run-time. The choice of the parallel backend depends on whether the parameter sweep will be run on a shared memory or distributed memory computer. Examples of shared memory systems include one's personal laptop or desktop, or one node on an HPC. All four parallel backends can be used on shared memory systems. Examples of a distributed memory systems include clusters and HPCs where the system memory is distributed across multiple processors or nodes. In the case of distributed memory systems only MPI and Ray Core can be used. Note that when using MPI, users must use `mpirun`, `mpiexec`, or any related job scheduler command, e.g., `srun`, `jsrun`, etc. The parallel manager and the parameter sweep tool is not designed to leverage hardware accelerators, e.g., GPUs.

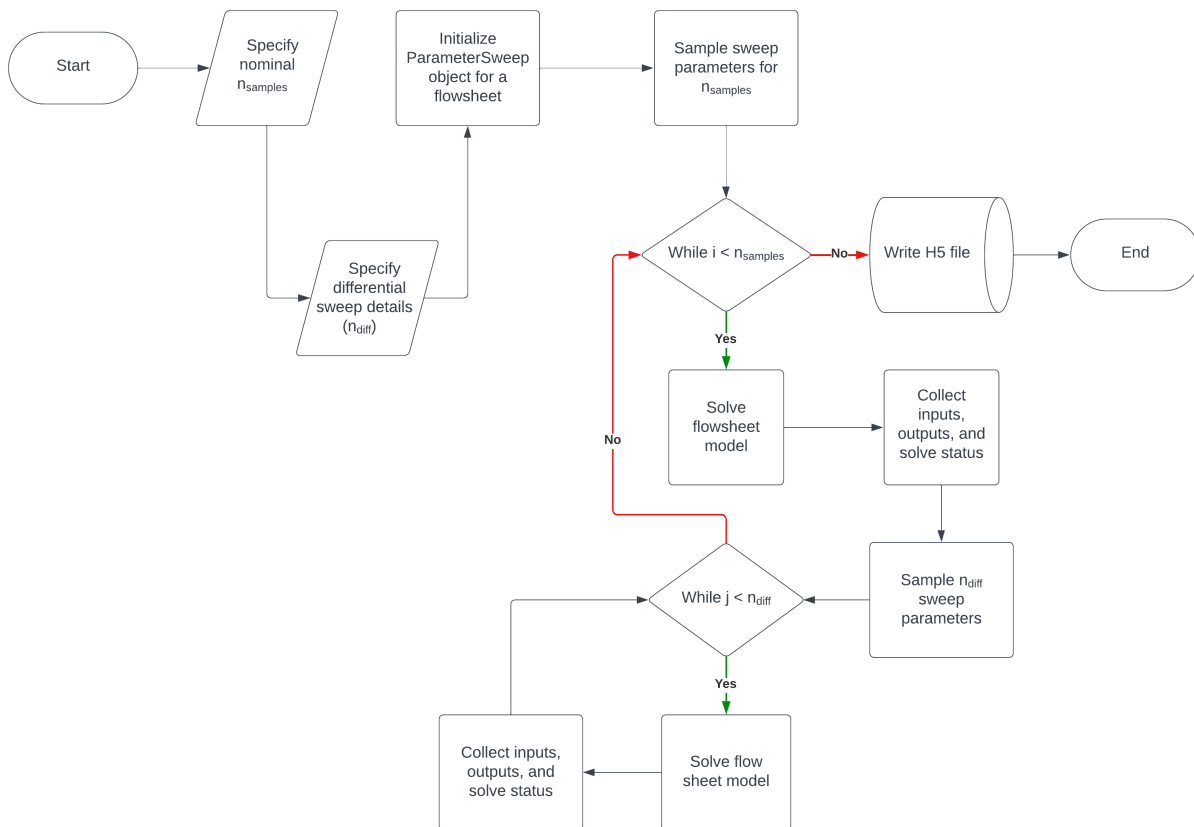


Figure 4. Flow chart depicting the parameter sweep algorithm that enables differential sampling at a nominal value.

## 3 Results

We demonstrate how the parameter sweep tool can be used to analyze an example flowsheet that models reverse osmosis (RO) with energy recovery device and is distributed with WaterTAP examples. Section 3.1 gives a brief description of the water treatment model, Sections 3.2 - 3.4 show the results of the different types of parameter sweeps, and lastly, Section 3.5 discusses parallel scalability.

### 3.1 Water Treatment Model

We modeled a single RO stage with an energy recovery device (ERD) shown in Figure 5 to demonstrate the capabilities of the parameter sweep tool. It includes a high pressure pump with a fixed outlet pressure connected to a steady state RO process model. A turbine-type isothermal ERD model is connected downstream of the RO model. The feed water is an NaCl brine solution and is modeled using NaCl property package within WaterTAP. Outputs that can be measured from this flowsheet include LCOW, product flow rate and concentration, volumetric recovery, membrane area, water recovery, and specific energy consumption (SEC).

### 3.2 Single-Variable Parameter Sweep

We begin by demonstrating how a single variable parameter sweep can be performed. Specifically, we look at the variation in LCOW, SEC, and membrane area as a function of the membrane water permeability coefficient. We varied the membrane water permeability between 0.36 – 3.6 LMH/bar using linear spacing and 100 sample points. The results are plotted in Figure 6. LCOW and SEC in Figures 6a and 6b, respectively, exhibit similar trends where there is a significant initial decrease in both metrics with an increase in water permeability, followed by a plateau where past permeability of 2.5 LMH/bar reduction in LCOW and SEC is minimal. The membrane area, plotted in Figure 6c, also decreased as the water permeability increased; recall that since the outlet pressure of the high-pressure pump and target recovery are fixed to 75 bar and 50%, respectively, the only free variable is the membrane area that must decrease. However, as evidenced by the flattening of the curves in all three cases, there are diminishing returns to further increase in water permeability. This is because of thermodynamic and mass transport limitations (concentration polarization) (Okamoto and Lienhard 2019).

### 3.3 Multi-Variable Parameter Sweep

Next, we conducted two two-dimensional sweeps. We varied the membrane water permeability coefficient and the membrane salt permeability coefficient in the first sweep. The second sweep studied the influence of water recovery and feed water salinity (NaCl concentration) on the LCOW for a given water and salt permeability. Both variables used linear spacing to create the mesh grid for the two sweeps. Membrane water permeability varied between 0.36 – 3.6 LMH/bar, while membrane salt permeability varied between 0.036 – 0.288 LMH. Each parameter is linearly sampled, containing 10 sample points each, and resulting in a 100-point mesh-grid. The results are plotted in Figure 7a. We can see that LCOW shows very little sensitivity to salt permeability, especially when compared to

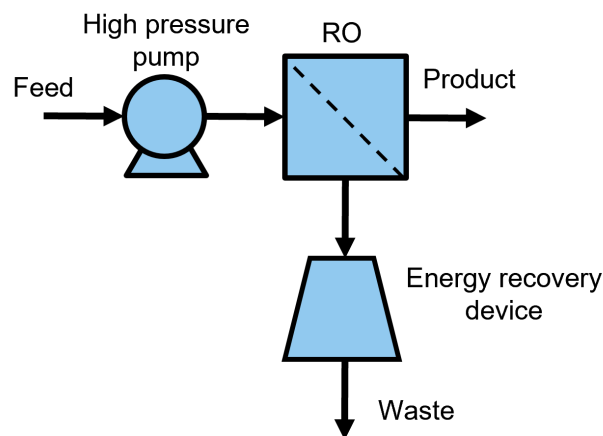


Figure 5. Diagram of RO with ERD flowsheet (Atia).

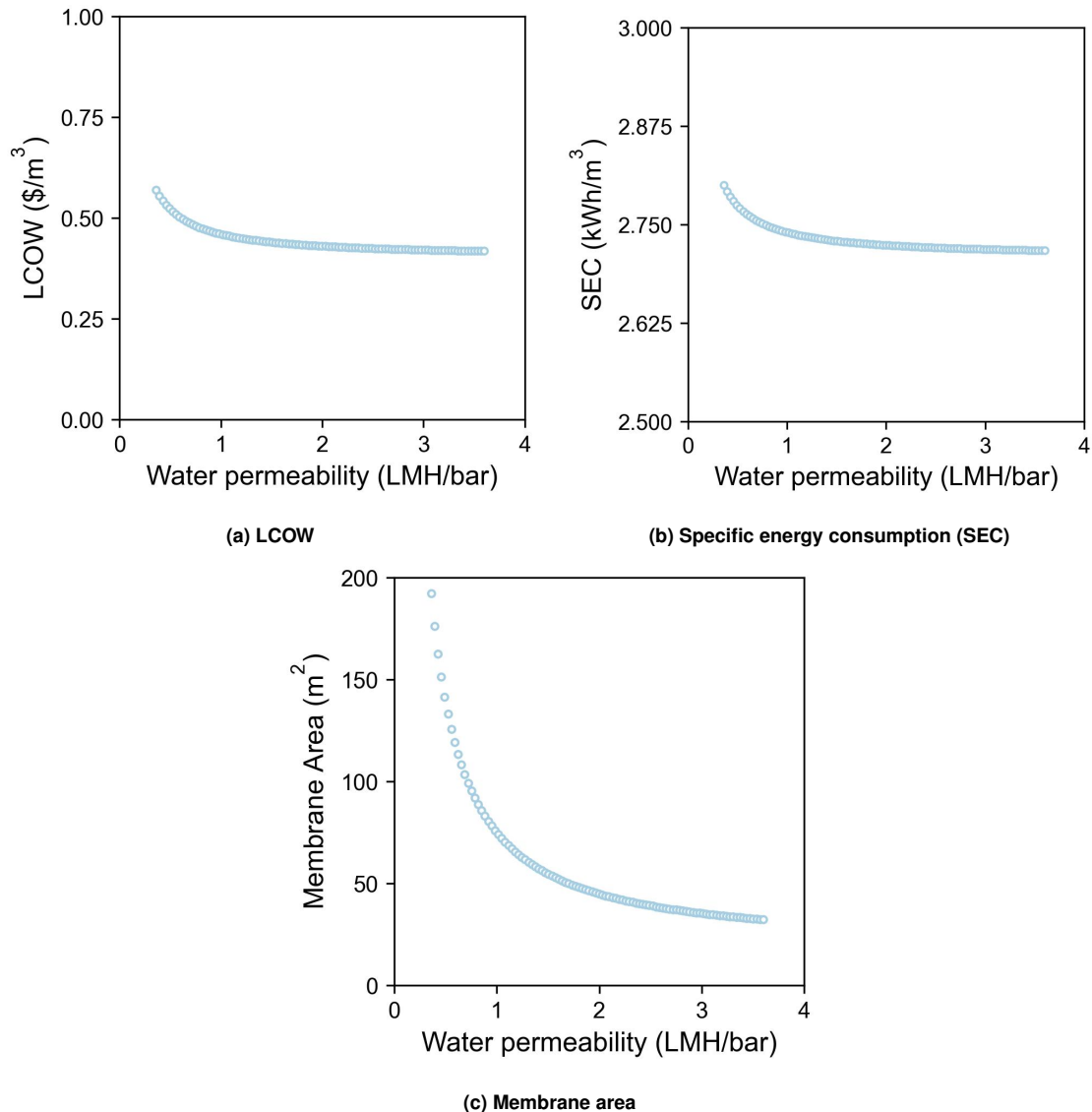


Figure 6. Variation in LCOW, specific energy consumption, and membrane area as a function of membrane water permeability.

water permeability. The increase in salt permeability increases salt transport and salt concentration on the permeate side of the membrane, reducing overall osmotic pressure difference and required operating pressure; however, the overall impact on the studied system is small. The water permeability shows a similar trend as described in the section above.

In the second sweep, water recovery is varied between 10% – 65% and feed water NaCl concentration is varied between 10 – 50 g/L. The impact of this variation on LCOW is shown in Figure 7b. We find that the LCOW decreases with an increase in water recovery. LCOW vs. water permeability exhibits a similar trend to Figure 6a where LCOW flattens with an increase in water recovery. This is due to overall costs being normalized by total produced water, and at low recovery, the overall system cost is high, while the quantity of product water is low, resulting in higher LCOW; as production increases, the cost decreases. At sufficiently high recoveries, the LCOW will increase due to higher energy and capital cost requirements to achieve the high recovery; this was not observed in the current sweep due to the range of explored water recoveries and limited maximum operating pressure of 65 bar, which prevented treatment of high salinity feed at recoveries above 50

LCOW increases with feed water NaCl concentration, which is to be expected as the RO system has to do more work in desalinating. Unlike the variation in LCOW with water recovery, the increase in LCOW with feed water salinity is



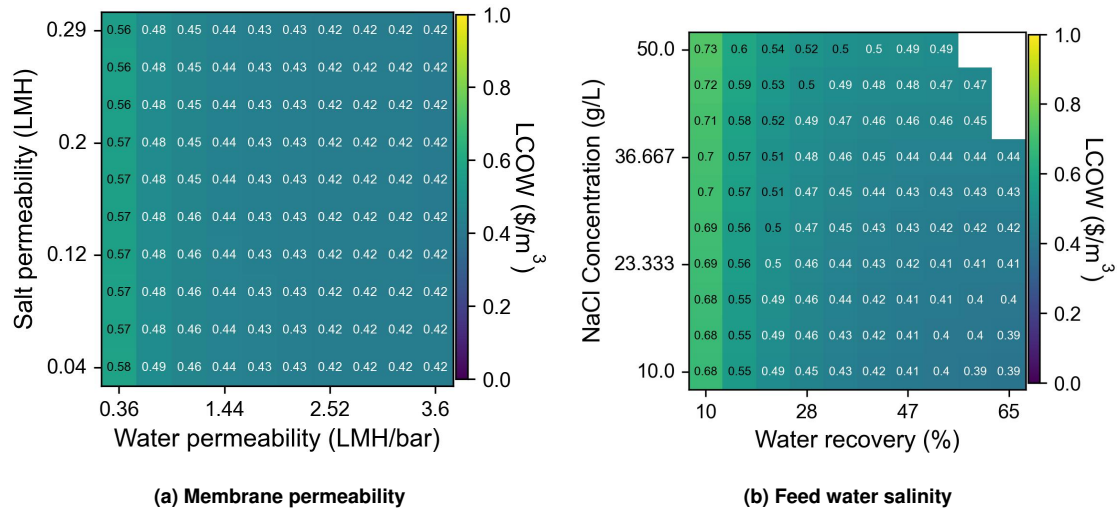


Figure 7. Two variable parameter sweep that shows variation in LCOW over the swept domain.

almost linear. Additionally, we see that LCOW is more sensitive to feed water salinity as compared to membrane salt permeability. The white space on the top right corner of Figure 7b represents sweep iterations that failed to converge to a solution because the osmotic pressure of the feed exceeded the maximum pressure allowed for the pump of 65 bar.

### 3.4 Differential Parameter Sweep

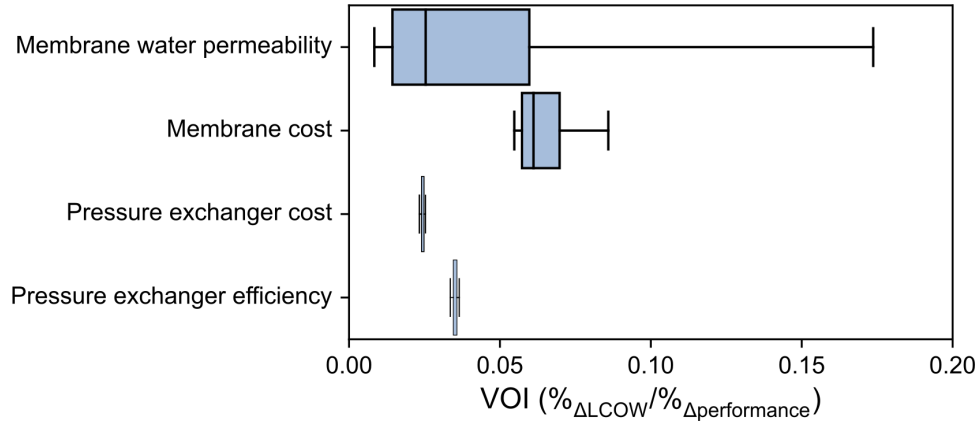
We ran the differential parameter sweep analysis to quantify and compare the value of innovation in various RO components. Specifically, we assess changes in which of the following parameters, viz., membrane water permeability, membrane cost, pressure exchanger efficiency, and pressure exchanger cost, will result in the greatest improvements in LCOW. Uniform sampling was used to vary the nominal values of all 4 parameters. Membrane water permeability was drawn from a uniform distribution  $\mathcal{U}(1.512, 7.56)$  LMH/bar, membrane cost from  $\mathcal{U}(10, 30)$   $\$/m^2$ , pressure exchanger cost from  $\mathcal{U}(250, 535)$   $\$/m^3$ , and pressure exchanger efficiency from  $\mathcal{U}(95, 99)\%$ . In the differential sweep step, the membrane water permeability was improved by 5%, the membrane cost and pressure exchanger costs were decreased by 5%, and the pressure exchanger efficiency was improved by 2%. The parameter sweep had 1000 nominal samples with one differential sample per each nominal value, thereby resulting in 2000 total samples.

Figure 8 plots the results of this stochastic value of innovation analysis. The box plots shows 5<sup>th</sup>, 25<sup>th</sup>, 50<sup>th</sup>, 75<sup>th</sup>, and 95<sup>th</sup> percentile change in LCOW per percentile change in parameter performance. The results show that in a small set of cases at (95<sup>th</sup> percentile) the membrane permeability would have highest increase in cost. This occurs only in scenarios where the initial membrane permeability is low ( $\sim 1$  LMH/bar), whereas most contemporary commercial membranes have higher water permeability. In almost all cases, decreasing membrane costs will reduce process costs by about 0.06% on median per percentile change in membrane cost ( $\sim 0.2\$/m^2$ ). A large spread in box plot, as in case of membrane water permeability, implies the cost improvement is largely dependent on the initial value. A narrower box plot indicates a more consistent improvement in process LCOW. For example, improving the cost and efficiency pressure exchanger always reduces cost regardless of their initial values.

### 3.5 Parallel Scalability

As was mentioned previously in Section 2.3, the parameter sweep tool is designed to be run in parallel on different computing resources. We evaluated the scaling performance of the parameter sweep for different types of computing systems that are representative of how the tool is being used currently. Specifically, we looked the scaling performance on a 2021 16" Apple MacBook Pro and on NREL's Eagle HPC.

The Apple MacBook Pro is used to demonstrate the performance of shared memory parallelism using Python's concurrent futures module. This laptop was equipped with the Apple M1 Max SoC that contains 10 physical cores (8



**Figure 8. Box plot from a stochastic value of innovation analysis using the differential parameter sweep. The narrower the box plot, the more consistent improvement in LCOW should be expected. The whiskers denote the best and worse cases.**

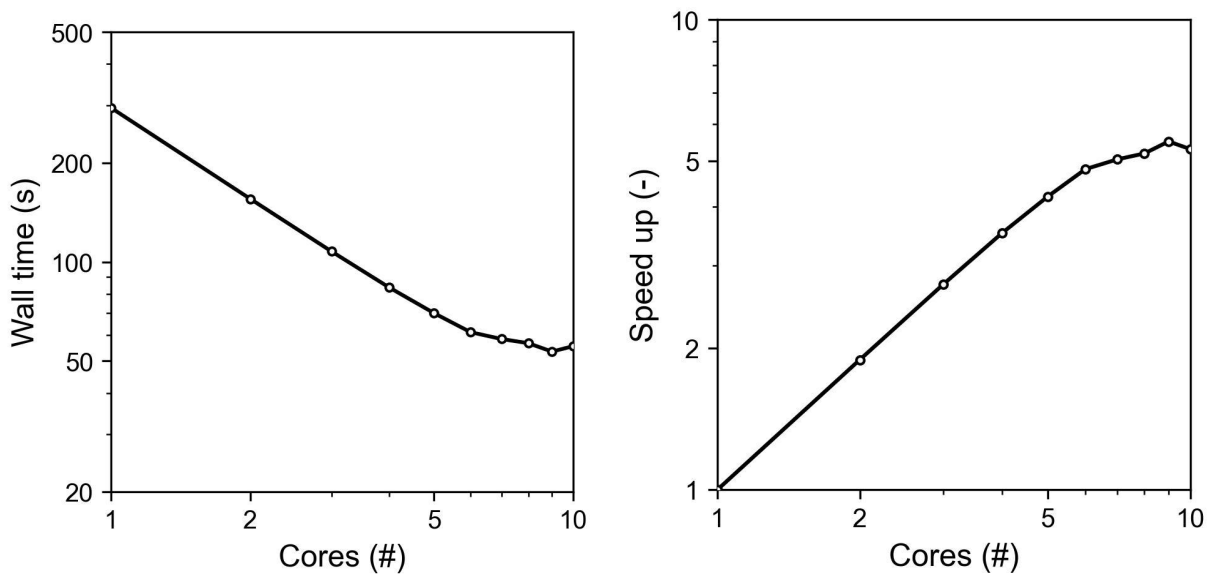
high-performance, 2 efficiency), 32 GB of RAM, and a 1TB SSD. We used the RO with ERD flowsheet described in Section 3.1 to test the parallel performance by sweeping over membrane water permeability, feed water salinity, and the energy recovery pressure exchanger costs. Random sampling was used to generate 2500 total samples. Samples for the membrane water permeability were drawn from a uniform distribution  $\mathcal{U}(0.36, 3.6)$  LMH/bar. Feed water salinity was also assumed to have a uniform distribution  $\mathcal{U}(10, 50)$  g/L, and the energy recovery pressure exchanger cost was assumed to have a normal distribution  $\mathcal{N}(535, 60^2)$   $\$/\text{m}^3$ . We conducted a strong scaling study using the simple parameter sweep described in Figure 2 and recorded the total solve time as a function of the number of processes used.

Figure 9 plots the results of the strong scaling study where 9a and 9b plot the total time elapsed and the computational speed up as a function of number of processes, respectively. We see that the scaling performance is almost linear for the first six cores before tapering off. Since there are parts of the code that are inherently serial, e.g., sample generation, result aggregation, file I/O, we should expect the slope of the speed up curve to be less than 1. The decrease in scalability at higher core counts should be expected because on a personal computer; multiple background processes are expected to run and interfere with the python processes resulting in some slowdowns. In addition, the difference between the high-performance and efficiency cores on the M1 Max can also affect the scaling performance.

Next we submit jobs on NREL’s Eagle HPC. Eagle is used to demonstrate the performance of distributed memory parallelism using MPI. Each compute node on Eagle is equipped with dual 3.0 GHz Intel Xeon Gold Skylake 6154 processors with a total of 36 physical processor cores and 96 GB of RAM. An InfiniBand network with 100 Gb/s bandwidth is used to connect the compute nodes and storage with each other. Eagle uses a Lustre files system for high-performance parallel I/O (National Renewable Energy Laboratory).

We use the RO with energy recovery devices flowsheet and a low-salt-rejection reverse osmosis (LSRRO) flowsheet described in (Atia et al. 2023) for this scalability analysis. The parameters in the sweep for RO with ERD were the same as before. However, we increased the total number of samples to 100,000. The LSRRO flowsheet contains a 2-stage model and two parameters are swept across. Fixed linear sampling was used to vary the feed concentration between 5 – 250  $\text{kg}/\text{m}^3$  and volumetric recovery rate between 30 – 90%. 100 samples were drawn for each parameter, resulting in a  $100 \times 100$  meshgrid, i.e., 10,000 total samples.

The strong scaling study on Eagle was conducted using up to 8 compute nodes or 288 processor cores. The results of these scaling studies are shown in Figure 10. As we can see the parameter sweep exhibits linear scaling. For the RO with ERD flowsheet, we observed a speed up of  $4.5\times$  when going from one compute node (36 cores) to five (180 cores). LSRRO exhibits a speed up of  $2.4\times$  when scaling up from 3 (108 cores) nodes to 8 (288 cores).



(a) Total Solve Time (b) Computational Speed-up  
 Figure 9. Strong scaling results when using concurrent futures for running parameter sweep with on the RO with ERD flowsheet with 2500 samples.

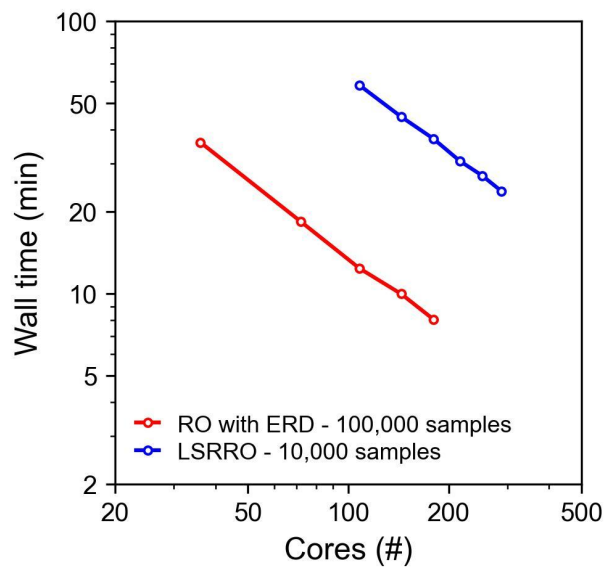


Figure 10. Strong scaling results when using MPI for running parameter sweep on NREL's Eagle HPC. Two flowsheets, RO with ERD and LSRRO, are considered for this analysis. Both exhibit linear scaling.

## 4 Conclusions

We present a powerful parameter sweep tool for water treatment analysis in this paper. This tool resides within WaterTAP and leverages the most advanced features of IDAES and Pyomo to conduct efficient and scalable numerical experiments of current and emerging technologies. Specifically, a user may use the parameter sweep tool for feasible configuration analyses, efficiency analyses, techno-economic analyses, sensitivity analyses, and even stochastic value of innovation analyses. Although, all examples shown here used WaterTAP models, the parameter sweep tool will also work with most IDAES and Pyomo models.

The parameter sweep is designed to be modular and three different types of parameter sweep have been implemented, viz., simple parameter sweep, recursive parameter sweep, and differential parameter sweep, as described in Figures 2, 3, and 4, respectively. Users can select one suitable for their specific analysis or combine them as needed in an outer-loop analysis. For analyses involving a design space exploration, a user may start by selecting the simple parameter sweep tool. If the user needs to generate a specific number of stochastic samples where some will fail, the user may use the recursive parameter sweep tool that attempts to successfully solve the desired number of samples. Lastly, if a user wishes to study the sensitivity of a parameter, they can use the differential parameter sweep tool. All parameter sweep types have several configuration options that enable the user to tailor the sweep run to their needs.

The parameter sweep tool is designed to be portable and leverage parallel processing. The parallel manager described in Section 2.3 enables the parameter sweep to leverage different parallel backends depending on their computing hardware. Section 3.5 shows that the parameter sweep tool scales well with simple and complex problems on both a personal computer and on an HPC. Such scalability opens the door to previously computationally intractable analyses.

The WaterTAP parameter sweep tool has already been used in scientific literature (Atia et al. 2023; Amusat et al. 2024) and its use is expected to continue. Some of the proposed future work includes continued support for existing and future WaterTAP analyses, integration with the IDAES diagnostic toolbox to help debug IDAES and Pyomo models, better sampling strategies, dynamic load balancing for more robust parallel performance, additional configuration options for sweeps, and tighter coupling with the WaterTAP GUI. The parameter sweep tool is under active development will continue to improve and better suit the needs of users and researchers. Users can access the parameter sweep tool at <https://github.com/watertap-org/parameter-sweep/tree/main> (watertap-org 2024).

## References

- Amusat, O. O., A. A. Atia, A. V. Dudchenko, and T. V. Bartholomew. 2024. “Modeling Framework for Cost Optimization of Process-Scale Desalination Systems with Mineral Scaling and Precipitation.” *ACS ES&T Engineering*, <https://doi.org/10.1021/acsestengg.3c00537>.
- Atia, A. *WaterTAP Introduction Tutorial*. <https://github.com/watertap-org/watertap/blob/0.6.0rc0/tutorials/introduction.ipynb>. Accessed: Apr 29, 2024.
- Atia, A. A., J. Allen, E. Young, B. Knueven, and T. V. Bartholomew. 2023. “Cost optimization of low-salt-rejection reverse osmosis.” *Desalination* 551:116407. ISSN: 0011-9164. <https://doi.org/10.1016/j.desal.2023.116407>. <https://www.sciencedirect.com/science/article/pii/S0011916423000395>.
- Atia, A. A., N. Y. Yip, and V. Fthenakis. 2021. “Pathways for minimal and zero liquid discharge with enhanced reverse osmosis technologies: Module-scale modeling and techno-economic assessment.” *Desalination* 509:115069. ISSN: 0011-9164. <https://doi.org/10.1016/j.desal.2021.115069>. <https://www.sciencedirect.com/science/article/pii/S0011916421001405>.
- Bartholomew, T. 2022. *NAWI Alliance Virtual Fall Meeting - NAWI Tools Deep Dive: WaterTAP*. <https://www.youtube.com/watch?v=Gi3fkzb51z4>. Accessed: May 25, 2023.
- Bartholomew, T. V., A. V. Dudchenko, N. S. Siefert, and M. S. Mauter. 2020. “Cost optimization of high recovery single stage gap membrane distillation.” *Journal of Membrane Science* 611:118370. ISSN: 0376-7388. <https://doi.org/10.1016/j.memsci.2020.118370>. <https://www.sciencedirect.com/science/article/pii/S0376738820309480>.
- Bartholomew, T. V., L. Mey, J. T. Arena, N. S. Siefert, and M. S. Mauter. 2017. “Osmotically assisted reverse osmosis for high salinity brine treatment.” *The Latest Advances and Opportunities in Forward Osmosis, Desalination* 421:3–11. ISSN: 0011-9164. <https://doi.org/10.1016/j.desal.2017.04.012>. <https://www.sciencedirect.com/science/article/pii/S0011916416319579>.
- Bartholomew, T. V., N. S. Siefert, and M. S. Mauter. 2018. “Cost Optimization of Osmotically Assisted Reverse Osmosis.” *Environmental Science & Technology* 52 (20): 11813–11821. <https://doi.org/10.1021/acs.est.8b02771>.
- Beattie, K. S., D. Gunter, B. Knueven, A. Lee, A. Ladshaw, M. Drouven, T. Bartholomew, et al. 2021. *WaterTAP v1.0.0*, May. <https://doi.org/10.11578/dc.20210527.7>. <https://www.osti.gov/biblio/1785311>.
- Bynum, M. L., G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff. 2021. *Pyomo—optimization modeling in python*. Third. Vol. 67. Springer Science & Business Media.
- Cacuci, D. 2003. *Sensitivity & Uncertainty Analysis, Volume 1: Theory*. v. 1. CRC Press. ISBN: 9780203498798.
- Chen, X., and N. Y. Yip. 2018. “Unlocking High-Salinity Desalination with Cascading Osmotically Mediated Reverse Osmosis: Energy and Operating Pressure Analysis.” *Environmental Science & Technology* 52 (4): 2242–2250. <https://doi.org/10.1021/acs.est.7b05774>.
- Dalcín, L., R. Paz, and M. Storti. 2005. “MPI for Python.” *Journal of Parallel and Distributed Computing* 65 (9): 1108–1115. ISSN: 0743-7315. <https://doi.org/10.1016/j.jpdc.2005.03.010>. <https://www.sciencedirect.com/science/article/pii/S0743731505000560>.
- Davenport, D. M. 2020. “Thin film composite membrane compaction in high-pressure reverse osmosis” [in English]. *Journal of membrane science* v. 610:pp. 118268–2020 v.610. <https://doi.org/10.1016/j.memsci.2020.118268>.
- Davenport, D. M., A. Deshmukh, J. R. Werber, and M. Elimelech. 2018. “High-Pressure Reverse Osmosis for Energy-Efficient Hypersaline Brine Desalination: Current Status, Design Considerations, and Research Needs.” *Environmental Science & Technology Letters* 5 (8): 467–475. <https://doi.org/10.1021/acs.estlett.8b00274>.
- Dudchenko, A. V., T. V. Bartholomew, and M. S. Mauter. 2021. “High-impact innovations for high-salinity membrane desalination.” *Proceedings of the National Academy of Sciences* 118 (37): e2022196118. <https://doi.org/doi:10.1073/pnas.2022196118>.

- Harris, C. R., K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, et al. 2020. “Array programming with NumPy.” *Nature* 585, no. 7825 (September): 357–362. <https://doi.org/10.1038/s41586-020-2649-2>.
- Hart, W. E., J.-P. Watson, and D. L. Woodruff. 2011. “Pyomo: modeling and solving mathematical programs in Python.” *Mathematical Programming Computation* 3, no. 3 (September): 219–260. ISSN: 1867-2957. <https://doi.org/10.1007/s12532-011-0026-8>.
- HSL(2013). *A collection of Fortran codes for large scale scientific computation*. <https://www.hsl.rl.ac.uk/>. Accessed: May 26, 2023.
- Ino, F., K. Shigeoka, T. Okuyama, M. Motokubota, and K. Hagihara. 2014. “A parallel scheme for accelerating parameter sweep applications on a GPU” [in eng]. *Concurrency and computation* 26 (2): 516–531. ISSN: 1532-0626.
- Lee, A., J. H. Ghouse, Q. Chen, J. C. Eslick, J. D. Sirola, I. E. Grossman, and D. C. Miller. 2018. “A Flexible Framework and Model Library for Process Simulation, Optimization and Control.” In *13th International Symposium on Process Systems Engineering (PSE 2018)*, edited by M. R. Eden, M. G. Ierapetritou, and G. P. Towler, 44:937–942. Computer Aided Chemical Engineering. Elsevier. <https://doi.org/10.1016/B978-0-444-64241-7.50151-8>. <https://www.sciencedirect.com/science/article/pii/B9780444642417501518>.
- Lee, A., J. H. Ghouse, J. C. Eslick, C. D. Laird, J. D. Sirola, M. A. Zamarripa, D. Gunter, et al. 2021. “The IDAES process modeling framework and model library—Flexibility for process simulation and optimization.” *Journal of Advanced Manufacturing and Processing* 3 (3): e10095. <https://doi.org/10.1002/amp2.10095>. <https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/amp2.10095>.
- Moritz, P., R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, et al. 2018. *Ray: A Distributed Framework for Emerging AI Applications*. arXiv: 1712.05889 [cs.DC].
- National Renewable Energy Laboratory. *Eagle Computing System*. <https://www.nrel.gov/hpc/eagle-system.html>. Accessed: Jul 12, 2023.
- Nocedal, J., and S. J. Wright. 2006. *Numerical optimization*. 2nd ed. New York: Springer. ISBN: 978-0-387-30303-1.
- Okamoto, Y., and J. H. Lienhard. 2019. “How RO membrane permeability and other performance factors affect process cost and energy use: A review.” *Desalination* 470:114064. ISSN: 0011-9164. <https://doi.org/10.1016/j.desal.2019.07.004>. <https://www.sciencedirect.com/science/article/pii/S0011916419305752>.
- Onishi, V. C., A. Carrero-Parreño, J. A. Reyes-Labarta, E. S. Fraga, and J. A. Caballero. 2017. “Desalination of shale gas produced water: A rigorous design approach for zero-liquid discharge evaporation systems.” *Journal of Cleaner Production* 140:1399–1414. ISSN: 0959-6526. <https://doi.org/10.1016/j.jclepro.2016.10.012>. <https://www.sciencedirect.com/science/article/pii/S0959652616316109>.
- Pacini, E., C. Mateos, and C. Garino. 2013. “Schedulers Based on Ant Colony Optimization for Parameter Sweep Experiments in Distributed Environments.” In *Handbook of Research on Computational Intelligence for Engineering, Science, and Business*, 410–448. USA. ISBN: 9781466625181.
- Patel, S. K., B. Lee, P. Westerhoff, and M. Elimelech. 2024. “The potential of electrodialysis as a cost-effective alternative to reverse osmosis for brackish water desalination.” *Water Research* 250:121009. ISSN: 0043-1354. <https://doi.org/10.1016/j.watres.2023.121009>. <https://www.sciencedirect.com/science/article/pii/S0043135423014495>.
- Razavi, S., and H. V. Gupta. 2015. “What do we mean by sensitivity analysis? The need for comprehensive characterization of “global” sensitivity in Earth and Environmental systems models.” *Water Resources Research* 51 (5): 3070–3092. <https://doi.org/10.1002/2014WR016527>. <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2014WR016527>.
- Saltelli, A., M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. 2007. *Global Sensitivity Analysis. The Primer*. England: John Wiley & Sons, Ltd. ISBN: 9780470725184. <https://doi.org/10.1002/9780470725184>. <https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470725184>.

Smith, R. C. 2013. *Uncertainty Quantification: Theory, Implementation, and Applications*. USA: Society for Industrial / Applied Mathematics. ISBN: 161197321X.

Sobol, I. 2001. “Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates.” The Second IMACS Seminar on Monte Carlo Methods, *Mathematics and Computers in Simulation* 55 (1): 271–280. ISSN: 0378-4754. [https://doi.org/10.1016/S0378-4754\(00\)00270-6](https://doi.org/10.1016/S0378-4754(00)00270-6). <https://www.sciencedirect.com/science/article/pii/S0378475400002706>.

*The IDAES Process Systems Engineering Framework*. <https://github.com/IDAES/idaes-pse>. Accessed May 18, 2023.

Van Rossum, G., and F. L. Drake. 2009. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace. ISBN: 1441412697.

Wächter, A., and L. T. Biegler. 2006. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming.” From <https://doi.org/10.1007/s10107-004-0559-y>, *Mathematical Programming* 106, no. 1 (March): 25–57. ISSN: 1436-4646. <https://doi.org/10.1007/s10107-004-0559-y>.

*WaterTAP development repository*. <https://github.com/watertap-org/watertap>. Accessed May 18, 2023.

watertap-org. 2024. *parameter-sweep*. Accessed June 3, 2024. <https://github.com/watertap-org/parameter-sweep/tree/main>.

Youn, C., and T. Kaiser. 2010. “Management of a parameter sweep for scientific applications on cluster environments.” *Concurrency and Computation: Practice and Experience* 22 (18): 2381–2400. <https://doi.org/10.1002/cpe.1563>. <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.1563>.

## Appendix A. Parameter Sweep Options

Table A.1 describes all the arguments and the keyword arguments that can be supplied to the 3 parameter sweep classes, viz., `ParameterSweep`, `RecursiveParameterSweep`, and `DifferentialParameterSweep`, described in Section 2.2.2. These keyword arguments are stored in the parameter sweep classes using Pyomo's configuration system. All three classes are derived from the private base class `_ParameterSweepBase`. This enables a consistent API with most of the keyword arguments are shared across the three classes. Differential parameter sweep expects a few additional options due to its added complexity.

All three parameter sweep classes contain a `parameter_sweep` method that actually executes the parameter sweep. The arguments and the keyword arguments for this method is consistent across three methods and are detailed in Table A.2.

**Table A.1. Parameter Sweep inputs that leverage Pyomo's configuration system**

<b>Inputs</b>	<b>Description</b>
<code>build_model</code>	Function for building the model
<code>build_model_kwargs</code>	Keyword argument for the model build function
<code>build_sweep_params</code>	Function for building the sweep parameters
<code>build_sweep_params_kwargs</code>	Keyword argument for the build sweep params function
<code>build_outputs</code>	Function for building outputs
<code>build_outputs_kwargs</code>	Keyword argument for the build outputs function
<code>index_global_combo_array</code>	Will add indexing to <code>global_combo_array</code> created while running the sweep, primarily used with differential parameter sweep tool
<code>initialize_before_sweep</code>	Initializing a model before every iteration
<code>initialize_function</code>	Function to initialize a flowsheet
<code>initialize_kwargs</code>	Keyword arguments for the initialization function
<code>log_model_states</code>	Enables logging of model states during serial execution
<code>number_of_subprocesses</code>	Number of processes to fan out to locally - ignored if running under MPI
<code>optimize_function</code>	Optimization function to be used in the parameter sweep
<code>optimize_kwargs</code>	Keyword argument for the optimization function
<code>parallel_backend</code>	Backend for parallelization
<code>probe_function</code>	Function to probe if a flowsheet configuration will work
<code>reinitialize_function</code>	Function to reinitialize a flowsheet
<code>reinitialize_kwargs</code>	Keyword arguments for the reinitialization function
<code>reinitialize_before_sweep</code>	Reinitializing a model before every iteration
<code>update_sweep_params_before_init</code>	Enables update of vars to sweep values before initialization (only enabled if <code>init_before_sweep=True</code> )
<b>GUI</b>	
<code>publish_progress</code>	Boolean to decide whether information about how many iterations of the parameter sweep have completed should be sent
<code>publish_address</code>	Address to which the parameter sweep progress will be sent
<b>For advanced users only</b>	
<code>custom_do_param_sweep</code>	Alternative implementation of the parameter sweep function in case the user is doing unique analyses or wishes to implement a different parallel implementation
<code>custom_do_param_sweep_kwargs</code>	Keyword arguments associated with the alternative parameter sweep function
<b>Specific to DifferentialParameterSweep</b>	
<code>build_differential_sweep_specs</code>	Function for building the differential sweep specification dictionary
<code>build_differential_sweep_specs_kwargs</code>	Keyword argument for the building differential sweep function
<code>differential_sweep_specs</code>	Python dictionary containing the specifications for the differential sweep.
<code>guarantee_solves</code>	Guarantee if a certain number of solves is expected
<code>num_diff_samples</code>	Number of samples for differential sweep



**Table A.2. Arguments to the `parameter_sweep` method.**

<b>Arguments</b>	<b>Description</b>
<code>build_model</code>	Function for building the model
<code>build_sweep_params</code>	Function for building the sweep parameters
<b>Keyword Arguments</b>	<b>Description</b>
<code>build_model_kwargs</code>	Keyword argument for the model build function
<code>build_sweep_params_*</code> <code>kwargs</code>	Keyword argument for the build sweep params function
<code>build_outputs</code>	Function for building outputs
<code>build_outputs_kwargs</code>	Keyword argument for the build outputs function
<code>seed</code>	Seed for the random sampling
<code>num_samples</code>	Number of samples for the random sampling type

Finally, the process of writing the results to file is handled by another class called `ParameterSweepWriter`. `ParameterSweepWriter` also uses Pyomo's configuration system to store user specified keyword arguments. All parameter sweep classes instantiate a `ParameterSweepWriter` object and the keyword arguments for the writer are specified when creating a parameter sweep object. The `ParameterSweepWriter` are specified in Table A.3 below.

**Table A.3. Configuration options for `ParameterSweepWriter` class**

<b>Inputs</b>	<b>Description</b>
<code>debugging_data_dir</code>	Directory path to output debugging data
<code>csv_results_file_name</code>	File path to the output CSV file
<code>h5_results_file_name</code>	File path to the output H5 file
<code>interpolate_nan_outputs</code>	Bool to decide whether to interpolate NaN outputs
<code>h5_parent_group_name</code>	Parent group name (container like objects, similar to a folder/directory in a file system) for parameter sweep outputs to be saved

## Appendix B. Generative AI Prompts and Responses

This appendix contains the user prompts and the corresponding ChatGPT outputs that were used to assist in the preparation of this technical report. The user prompts are italicized and the ChatGPT responses are provided as verbatim.

*Give me seminal references for sensitivity analysis*

Sure! Here are some seminal references on sensitivity analysis:

1. Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., ... & Tarantola, S. (2008). Global sensitivity analysis: The primer. John Wiley & Sons.
2. Morris, M. D. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2), 161-174.
3. Sobol', I. M. (1993). Sensitivity estimates for nonlinear mathematical models. *Mathematical Modeling and Computational Experiment*, 1(4), 407-414.
4. Saltelli, A., Tarantola, S., & Chan, K. P. (1999). A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41(1), 39-56.
5. Campolongo, F., Cariboni, J., & Saltelli, A. (2007). An effective screening design for sensitivity analysis of large models. *Environmental Modelling & Software*, 22(10), 1509-1518.
6. Iooss, B., & Prieur, C. (2019). Shapley effects for sensitivity analysis with dependent inputs: comparisons with Sobol' indices, numerical estimation and applications. *Journal of Statistical Computation and Simulation*, 89(7), 1300-1334.
7. Borgonovo, E. (2007). A new uncertainty importance measure. *Reliability Engineering & System Safety*, 92(6), 771-784.
8. Plischke, E., Eichfelder, G., & Boxberger, S. (2013). Derivative-free global optimization based on radial basis function interpolation and sensitivity analysis. *Journal of Global Optimization*, 55(4), 843-867.

These references cover various aspects of sensitivity analysis, including global sensitivity analysis, screening designs, importance measures, uncertainty analysis, and derivative-free optimization. They should provide a solid foundation for understanding and conducting sensitivity analysis.

*Give me seminal papers of parameter sweep*

Here are some seminal papers related to parameter sweep:

1. Forrest, S., Perelson, A. S., Allen, L., & Cherukuri, R. (1994). Self-nonsel self discrimination in a computer. In *Advances in Neural Information Processing Systems* (pp. 396-403).
2. Carbone, A., Forest, M. G., Gokhale, M. B., & Grama, A. Y. (2005). Adjoint sensitivity analysis for high performance computing: BlueGene/L case study. In *SC'05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing* (pp. 43-43).
3. Helbing, D., & Molnár, P. (1995). Social force model for pedestrian dynamics. *Physical review E*, 51(5), 4282.
4. Barnes, W. L., Dereux, A., & Ebbesen, T. W. (2003). Surface plasmon subwavelength optics. *Nature*, 424(6950), 824-830.
5. De Fabritiis, G., & Coveney, P. V. (2003). Coupling fluctuating hydrodynamics to molecular dynamics: Finite-size effects and synchronization. *Physical Review E*, 68(3), 036704.
6. Perelson, A. S., & Nelson, P. W. (1999). Mathematical analysis of HIV-1 dynamics in vivo. *SIAM Review*, 41(1), 3-44.
7. Gillespie, D. T. (1976). A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4), 403-434.
8. Clayton, R. W. (1982). An analysis of the accuracy of finite-difference schemes for hyperbolic IBVPs with different boundary conditions. *Journal of Computational Physics*, 46(1), 63-78.

These papers cover a variety of fields, including computer science, physics, biology, and computational chemistry. While not all of them explicitly focus on parameter sweep methods, they represent seminal works that utilize different simulation and computational techniques, some of which may involve parameter sweeps as part of their analysis or experimentation.

*Write an introduction on the importance of parameter sweep for sensitivity analysis*

## Introduction:

Parameter sweep is a powerful technique used in sensitivity analysis to investigate the behavior of a system or model across a range of input parameter values. It involves systematically varying the values of one or more input parameters and observing the corresponding changes in the output or response of the system. Parameter sweep analysis provides valuable insights into how changes in parameter values impact the overall behavior, performance, and robustness of a system or model.

Sensitivity analysis plays a crucial role in numerous scientific and engineering fields, including environmental modeling, climate science, economics, optimization, and computational biology, among others. It aims to understand and quantify the influence of input parameters on the model's outputs, enabling researchers and decision-makers to identify the most influential parameters and better understand the system's dynamics.

The importance of parameter sweep for sensitivity analysis lies in its ability to reveal the relationships and dependencies between input parameters and model outputs. By systematically varying parameter values over a defined range, researchers can observe the resulting changes in the model's behavior and assess how sensitive the model is to different input conditions. This information is vital for a range of purposes, such as identifying critical parameters, assessing model reliability, optimizing system performance, and making informed decisions.

Parameter sweep analysis allows researchers to explore the input parameter space comprehensively. Instead of relying on single-point estimates or limited scenarios, parameter sweep provides a more comprehensive understanding of the system's response to different parameter combinations. It helps in identifying non-linear relationships, threshold effects, and interactions between parameters that might not be apparent when considering individual parameters in isolation.

Furthermore, parameter sweep analysis facilitates the identification of influential parameters by quantifying their impact on the model output. Various sensitivity measures can be employed to assess the sensitivity of model outputs to changes in parameter values, such as global sensitivity indices, local sensitivity analyses, and regression-based approaches. These measures provide valuable information on the relative importance of different parameters, enabling researchers to prioritize their efforts and resources in further investigations and model refinements.

Additionally, parameter sweep analysis supports model validation and uncertainty quantification. By exploring the parameter space and comparing the model outputs to experimental or observed data, researchers can assess the model's capability to reproduce known phenomena. Sensitivity analysis using parameter sweep techniques can also help in characterizing the uncertainty associated with model predictions by quantifying the range of possible outcomes due to parameter variations.

In conclusion, parameter sweep analysis is a crucial tool in sensitivity analysis, enabling researchers to explore the relationships between input parameters and model outputs. It provides a comprehensive understanding of system behavior, helps identify influential parameters, supports model validation, and contributes to uncertainty quantification. By employing parameter sweep techniques, researchers can gain valuable insights into complex systems and make informed decisions based on a thorough understanding of parameter sensitivities.

### *What is sensitivity analysis*

Sensitivity analysis is a systematic technique used to assess the sensitivity or responsiveness of a system or model to changes in input parameters or assumptions. It aims to understand how variations in input values impact the outputs or outcomes of a model, simulation, or decision-making process.

The primary goal of sensitivity analysis is to quantify and rank the influence of individual parameters or groups of parameters on the model's outputs or performance. It helps identify which parameters have the most significant impact on the results, which are relatively insensitive, and which might interact with each other. Sensitivity analysis provides valuable insights into the relationships and dependencies between input parameters and model outputs, aiding in decision-making, model improvement, and uncertainty management.

Sensitivity analysis can be performed using various approaches, including local and global methods. Local sensitivity analysis focuses on understanding the impact of small perturbations or variations in individual parameters on the model output. It often involves calculating sensitivity indices, such as partial derivatives or elasticity coefficients, to quantify the change in the output resulting from a unit change in a specific parameter.

Global sensitivity analysis, on the other hand, aims to assess the combined or interactive effects of multiple parameters across a wide range of their possible values. It involves exploring the entire parameter space and analyzing the model's response to different parameter combinations. Global sensitivity analysis methods, such as variance-based methods (e.g., Sobol' indices) or regression-based methods (e.g., Morris method), provide insights into the relative importance of different parameters and their interactions.

Sensitivity analysis is valuable in several domains, including engineering, environmental modeling, finance, health sciences, and policy analysis. It helps in decision-making processes by identifying critical parameters and understanding the robustness of models or systems. It aids in model calibration and validation, as sensitivity analysis can reveal the most influential parameters to focus on during the model refinement process. Sensitivity analysis is also essential for uncertainty quantification, as it helps characterize the variability and uncertainty associated with model predictions.

In summary, sensitivity analysis is a fundamental tool for understanding and quantifying the impact of input parameter variations on model outputs. By systematically exploring the parameter space and assessing the sensitivity of the model, sensitivity analysis provides valuable insights for decision-making, model improvement, and uncertainty management.

#### *What is parameter sweep*

Parameter sweep, also known as parameter exploration or parameter scanning, is a technique used in computational modeling and simulation to systematically explore the behavior of a system or model by varying the values of input parameters over a specified range. It involves running the model multiple times, each time with different combinations of parameter values, and observing the corresponding outputs or results.

The purpose of parameter sweep is to understand how changes in input parameter values affect the behavior, performance, or outcomes of the model. By sweeping through a range of parameter values, researchers can analyze the sensitivity of the model to different parameter settings and explore the relationships between input parameters and model outputs. Parameter sweep allows for a comprehensive investigation of the system's response to various input conditions and helps identify important parameter values, optimal configurations, or critical regions in the parameter space.

Parameter sweep can be conducted in a structured or unstructured manner, depending on the goals and characteristics of the model. Structured parameter sweeps involve systematically varying parameters within predefined ranges or grids, ensuring a comprehensive exploration of the parameter space. Grid-based sampling methods, such as Latin hypercube sampling or full-factorial designs, are often used for structured parameter sweeps.

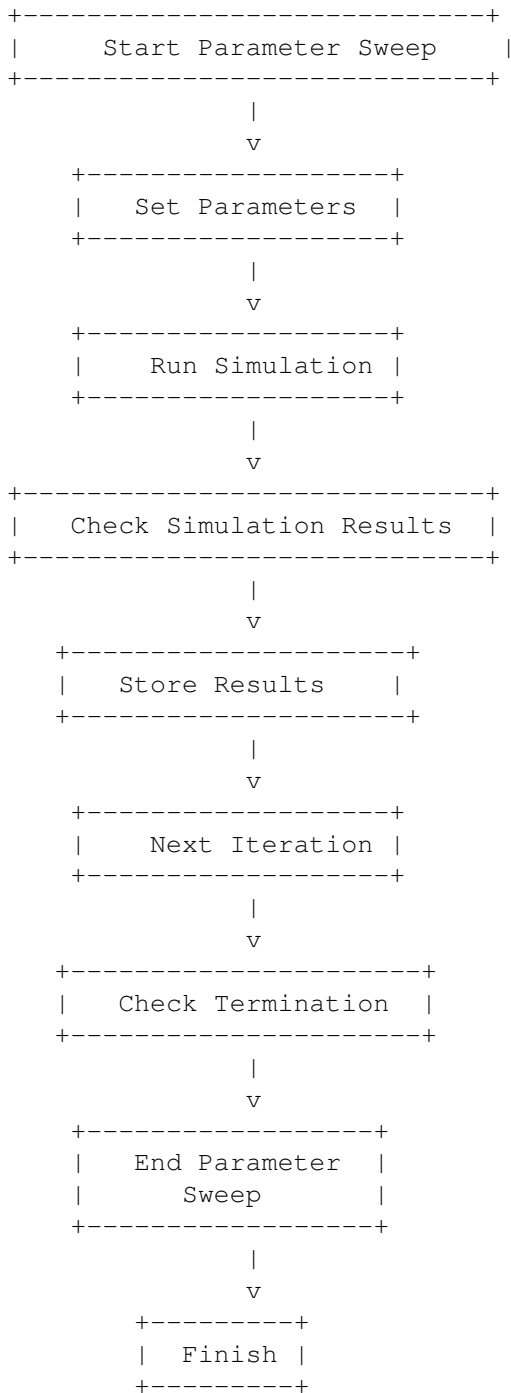
In contrast, unstructured parameter sweeps involve randomly or intelligently sampling the parameter space. Random sampling techniques, such as Monte Carlo simulations, randomly select parameter values from within specified ranges. Intelligent sampling methods, such as adaptive sampling or Latin hypercube sampling with optimization algorithms, aim to prioritize sampling regions of the parameter space that are likely to yield interesting or informative results.

The choice of parameter values and ranges in a parameter sweep is crucial and should be based on prior knowledge, expert judgment, available data, or sensitivity analysis results. It is important to consider both the plausible ranges of parameter values and their potential impact on the model's behavior or outputs.

Parameter sweep is a valuable technique in computational modeling and simulation, allowing researchers to gain insights into the sensitivity, robustness, and behavior of complex systems. It helps in understanding the impact of parameter variations, identifying influential parameters, optimizing system performance, and exploring the range of possible outcomes. By systematically exploring the parameter space, parameter sweep aids in improving model understanding, decision-making, and model validation processes.

#### *Create a flow chart for a parameter sweep tool*

Certainly! Here's a basic flow chart for a parameter sweep tool:



Explanation of the flow chart:

1. Start Parameter Sweep: The process begins here.
2. Set Parameters: Set the initial parameters for the simulation.
3. Run Simulation: Execute the simulation with the current parameter values.

4. Check Simulation Results: Analyze the results obtained from the simulation.
5. Store Results: Store or record the simulation results for further analysis or visualization.
6. Next Iteration: Move on to the next iteration or set of parameters.
7. Check Termination: Determine if the parameter sweep should continue based on predefined termination conditions (e.g., maximum number of iterations reached).
8. End Parameter Sweep: If the termination conditions are met, end the parameter sweep.
9. Finish: The process is complete.

This flow chart represents a basic outline of a parameter sweep tool. The specifics of each step may vary depending on the context and requirements of your particular tool or application.

*Write a paragraph on how a python parameter sweep software has been developed to simplify its use by professors, scientists, and graduate students while offering robustness in its analysis*

A Python parameter sweep software has been meticulously developed to provide a user-friendly experience for professors, scientists, and graduate students, simplifying the complex task of conducting parameter sweeps. The software offers an intuitive interface that allows users to easily set up and configure their desired parameters, eliminating the need for extensive coding knowledge. By abstracting the technical complexities, the software enables users to focus on their scientific objectives and analysis. Despite its user-friendly nature, the software ensures robustness in analysis by incorporating efficient algorithms and error handling mechanisms. It facilitates the execution of simulations or experiments with various parameter combinations, enabling users to explore a wide range of scenarios effortlessly. The software also provides comprehensive result analysis tools, allowing users to visualize and interpret the outcomes of their parameter sweeps. With its combination of user-friendly features and robust analysis capabilities, this Python parameter sweep software empowers researchers to efficiently and effectively explore and analyze complex systems.

*Write the conclusion section for a parameter sweep technical report*

The conclusion of a parameter sweep technical report should encapsulate the essential findings obtained from systematically varying the parameters in your study. Here's an example of how you might structure the conclusion section:

—

### **Conclusion:**

In conclusion, the comprehensive parameter sweep conducted in this study has provided valuable insights into the behavior and performance of our system under varying conditions. The key findings can be summarized as follows:

1. **Optimal Parameter Configurations:** Through the systematic exploration of a range of parameter values, we have identified specific configurations that result in optimal system performance. These configurations, elucidated through careful analysis, can serve as valuable guidelines for practitioners seeking to enhance the efficiency and effectiveness of similar systems.
2. **Sensitivity Analysis:** The sensitivity of the system to changes in certain parameters has been thoroughly examined. Our findings shed light on the parameters that significantly impact system outcomes, helping prioritize areas for further investigation or optimization.
3. **Trade-offs and Constraints:** The parameter sweep has revealed inherent trade-offs and constraints within the system. Understanding these trade-offs is crucial for making informed decisions in real-world applications where competing objectives or resource limitations may be present.

4. **Robustness and Generalizability:** By exploring a broad parameter space, we have gained insights into the robustness of the system across diverse conditions. This understanding is instrumental in assessing the generalizability of our findings and the applicability of the system in various scenarios.
5. **Limitations and Future Directions:** It is essential to acknowledge the limitations of our study, including potential simplifications and assumptions made during the parameter sweep. Future research should address these limitations and explore additional dimensions of the parameter space to further refine our understanding.

In summary, the outcomes of this parameter sweep not only contribute to a nuanced understanding of the system's behavior but also provide actionable recommendations for practitioners and researchers. This work sets the stage for continued exploration and refinement, paving the way for advancements in the field and the optimization of systems with practical implications.