Pacific Northwest
NATIONAL LABORATORY

PNNL-31781

NRAP-Open-IAM: FutureGen2 Component Models

Development and Testing

August 2021

Diana H Bacon

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062;
ph: (865) 576-8401
fax: (865) 576-5728
email: reports@adonis.osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
email: orders@ntis.gov <https://www.ntis.gov/about>
Online ordering: http://www.ntis.gov

PNNL-31781

NRAP-Open-IAM: FutureGen2 Component Models

Development and Testing

August 2021

Diana H Bacon

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Abstract

This report describes the development and testing of three component models for NRAP-Open-IAM, the National Risk Assessment Partnership's open-source integrated assessment model. The FutureGen2 Lookup Table Reservoir component model is based on interpolation of data from a set of lookup tables. The lookup tables contain pressures and saturations predicted by multiphase flow simulations performed with a reservoir simulator. The FutureGen2 Above Zone Monitoring Interval (AZMI) component is a surrogate model that can be used to estimate the impact that carbon dioxide (CO_2) and brine leaks from the CO_2 storage reservoir at the FutureGen 2.0 site might have had on overlying aquifers or monitoring units were a leak to occur. The model estimates the size of "impact plumes" according to five metrics: pH, Total Dissolved Solids (TDS), pressure, dissolved CO_2 and temperature. The FutureGen2 Aquifer component is similar, but is limited to four metrics: pH, Total Dissolved Solids (TDS), pressure, and dissolved CO_2. The input parameters for each model are the same, but the Aquifer component is applicable to depths between 100 m and 700 m and the AZMI component is applicable from depths between 700 m and 1050 m.

Acknowledgments

The research presented in this article was completed as part of the National Risk Assessment Partnership (NRAP). Funding for this project was provided to Pacific Northwest National Laboratory (PNNL) under DOE contract number DE-AC0576RL01830 by the U.S. Department of Energy's (DOE's) Office of Fossil Energy. A portion of the modeling research was performed using PNNL's Research Computing cluster. Many thanks to Paige Morkner, Andrew Bean and Kelly Rose of the National Energy Technology Laboratory for curating and making public the FutureGen 2.0 Subsurface Technical Dataset.

Acronyms and Abbreviations

CCUS	Carbon capture, utilization, and storage
EDX	Energy Data Exchange
GCV	The generalized cross validation score of the model after the final linear fit
GRSQ	An R2-like score based on the GCV
MSE	The mean squared error of the model after the final linear fit
NETL	National Energy Technology Laboratory
NRAP	National Risk Assessment Partnership
ROM	Reduced Order Model
RSQ	The generalized R2 of the model after the final linear fit
SDWA	Safe Drinking Water Act
US DOE	United States Department of Energy
USEPA	United States Environmental Protection Agency

Contents

Figures

Tables

1.0 Introduction

Carbon capture, utilization, and storage (CCUS) technologies are being developed, both domestically and internationally, for their potential to mitigate environmental impacts associated with atmospheric release of carbon dioxide (CO_2) from anthropogenic sources, such as power production from fossil fuels and other large industrial sources. Over the last decade, the United States Department of Energy (US DOE) has invested millions of dollars developing carbon capture technologies and demonstrating safe and secure geologic carbon storage via a number of pilot-scale projects sited throughout the country (NETL 2015). To date, these projects have stored more than 16 million tonnes of CO_2 (NETL 2018).

Within the US, CO_2 injection activities are overseen by the US Environmental Protection Agency (EPA) following regulations (the Class VI Rule) promulgated under the Safe Drinking Water Act (SDWA) (USEPA 2010). The Class VI regulations are designed to protect underground sources of drinking water (USDWs), and include strict requirements for site characterization, CO_2 injection well construction, injection operations, site monitoring, financial liability, and record keeping/reporting. Key elements of the Class VI permitting process include delineating an Area of Review (AoR) and defining an appropriate Post-Injection Site Care (PISC) period for the project, both of which require simulated CO_2 saturations and pressure distributions from computational models. The models are based on site-specific data and are updated periodically during the lifetime of the project to evaluate reservoir performance and evolution of the storage system.

Despite the sophistication of today's multi-physics reactive transport codes, significant uncertainty exists in predicting the performance of geologic storage reservoirs. Challenges associated with developing greenfield sites include the inherit difficulty in scaling a few point source measurements of geological structure and reservoir permeability derived from characterization of borehole samples throughout the extensive area likely to be impacted by a commercial-scale CO_2 injection, a lack of site-specific data on the behavior of supercritical CO_2 in the reservoir being evaluated, and understanding changes in the transport behavior of carbon dioxide caused by changes in pressure and/or temperature and the buoyant nature of CO_2 over the long time scales required for geologic sequestration to have long-term benefit to atmospheric CO_2 levels. Additionally, the computational resources required to run high fidelity simulations limits their usefulness in performing sensitivity analysis for uncertainty reduction.

To help address this need, the US DOE established the National Risk Assessment Partnership (NRAP), an initiative across five US DOE national laboratories with the goal of developing defensible, science-based methodologies and platforms for quantifying risks amidst system uncertainty. In 2017, the NRAP team released a set of ten tools (i.e., the NRAP Toolset) that can be used to estimate risks associated with carbon sequestration (https://edx.netl.doe.gov/nrap/). The toolset adopts a stochastic approach in which includes uncertainties in storage reservoirs, leakage scenarios, and shallow groundwater impacts. It is derived from detailed physics and chemistry simulation results that are used to train more computationally efficient models, referred to here as reduced-order models (ROMs), for each component of the system. These tools can be used to help regulators and operators define the AoR and better understand the expected sizes and longevity of changes in water quality caused by CO_2 and brine leakage from a storage reservoir into drinking water aquifers.

This report details the development and testing of surrogate models that can be used to estimate the impact that carbon dioxide (CO_2) and brine leaks from the CO_2 storage reservoir at

the FutureGen 2.0 site might have on overlying aquifers or monitoring units. The FutureGen2 Lookup Table Reservoir component model is a ROM based on interpolation of data from a set of lookup tables. The lookup tables are based on multiphase flow simulations performed with a reservoir simulator. Each row of the lookup table is related to a particular set of model input parameters and contains pressures and saturations at selected time steps for a particular vertical layer of the full-physics model. The FutureGen2 Above Zone Monitoring Interval (AZMI) model estimates the size of "impact plumes" where changes in the values of five metrics exceed detectable thresholds: pH, Total Dissolved Solids (TDS), pressure, dissolved CO_2 and temperature. The FutureGen2 Aquifer model is similar, but is limited to four metrics: pH, Total Dissolved Solids (TDS), pressure, and dissolved CO_2. The aquifer component model is based on isothermal simulations because the model used to train the surrogate model, STOMP-CO2-R, required prohibitively small times steps for the nonisothermal phase transition from supercritical to liquid CO_2 at shallower depths.

2.0 FutureGen2 Lookup Table Reservoir Component

The FutureGen2 Lookup Table Reservoir component model is a ROM based on interpolation of data from a set of lookup tables. The lookup tables are based on multiphase flow simulations performed with a reservoir simulator. Each row of the lookup table is related to a particular set of model input parameters and contains pressures and saturations at selected time steps for a particular vertical layer of the full-physics model.

This study expands on the reservoir model developed by Zhang et al. (2014) used for the sensitivity analysis performed for the FutureGen 2.0 Site. Division of stratigraphic layers into 31 computational model layers is provided in Table 1 of the main paper. Four horizontal injection wells were screened within the layer named MtSimon11.

2.1 Input parameters

For the purposes of this application, additional simulations were conducted with this reservoir model to create the lookup tables specific to the site to be used in system-scale modeling. In the sensitivity analysis reported by Zhang et al. (2014), 11 distinct parameters for each of the 31 model layers were investigated relative to a reference base-case (i.e., the most representative input values based on the characterization data available). They identified that the horizontal permeability of the injection layer was the most sensitive parameter for injectivity. In our model, leakage risk through a legacy well is driven by pressure and saturation in the reservoir. We performed additional simulations varying permeability of each reservoir model layer independently with uncertainty ranges shown in Figure 1.

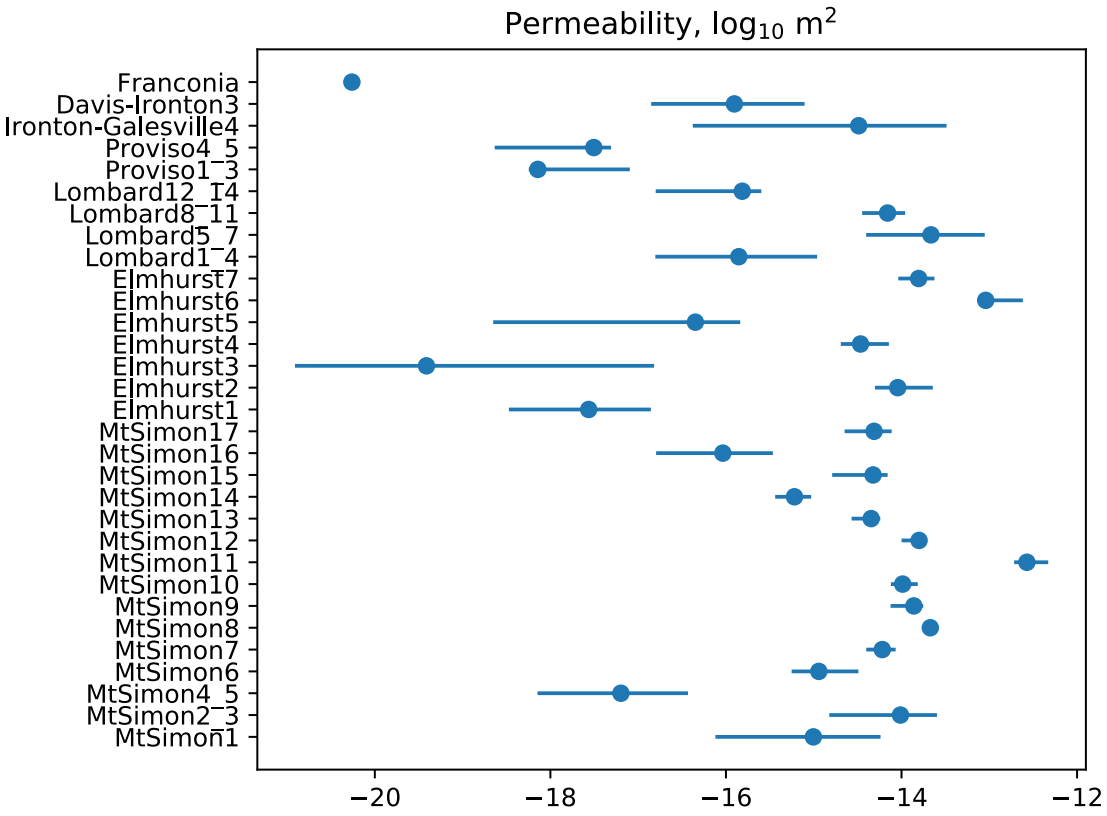

Figure 1. Range of values for reservoir model layer permeability.

2.2 Parameter Sampling

To create an NRAP-Open-IAM lookup table reservoir component, ideally the minimum and maximum value for each model input parameter must be specified, and a set of simulations performed with sampled values for each input parameter. The original ELAN log and core data was revisited to calculate the minimum and maximum permeability for each of the 31 model layers (Figure 1). The current analysis assumed that the permeability values were lognormally distributed; average values were calculated using the log of the permeabilities. The average permeability values were assumed to be the most representative input values based on the characterization data available. The minimum value was calculated using the lower 30th percentile and the maximum value determined using the upper 70th percentile of the log of the values within each vertical layer, roughly equal to one standard deviation about the mean.

2.3 STOMP Simulations

One thousand STOMP-CO2 simulation runs were performed to capture the effect of variability in permeability at each layer of the injection reservoir. Predicted pressures (Figure 2) and separate-phase CO_2 distribution (Figure 3) at the end of the injection period are shown for a single run with the average permeability values. While most of the CO_2 stays within Mount Simon, vertical migration results in small amounts of CO_2 in the overlying Elmhurst formation.

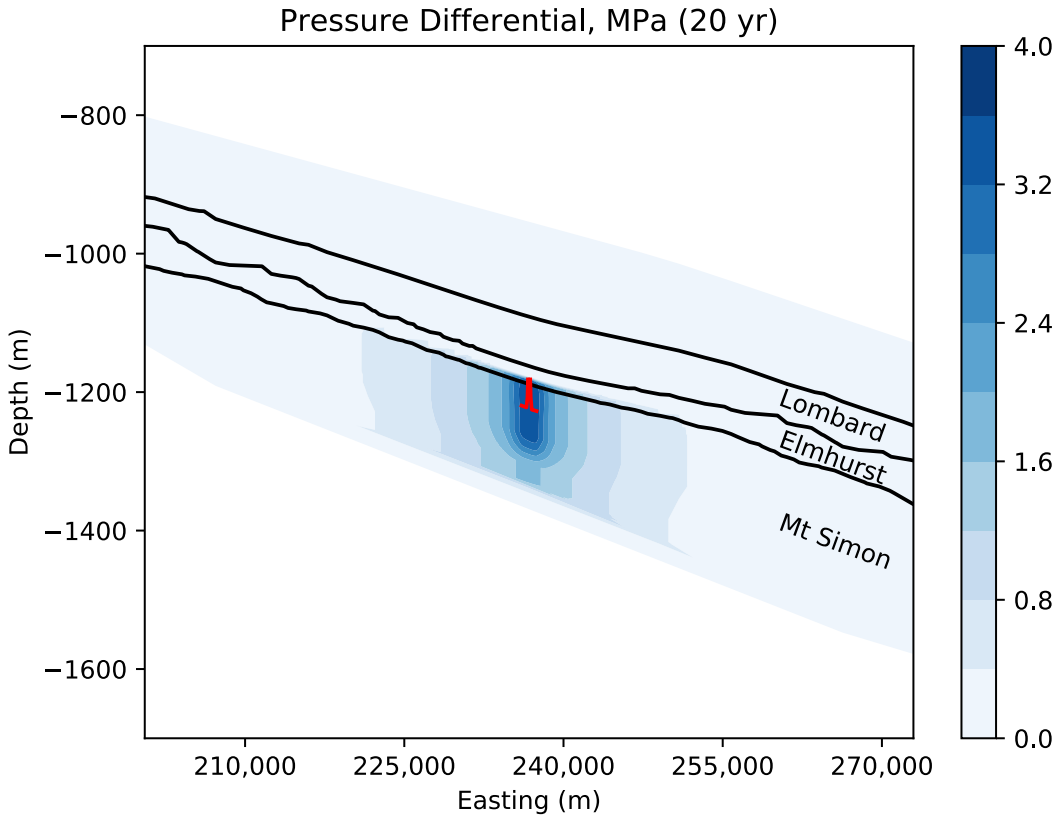

Figure 2. 2D cutaway view of the predicted pressure differential results in the reservoir at the end of injection period (20 yr). Screened portion of horizontal injection wells #1 and #4 are shown in red.

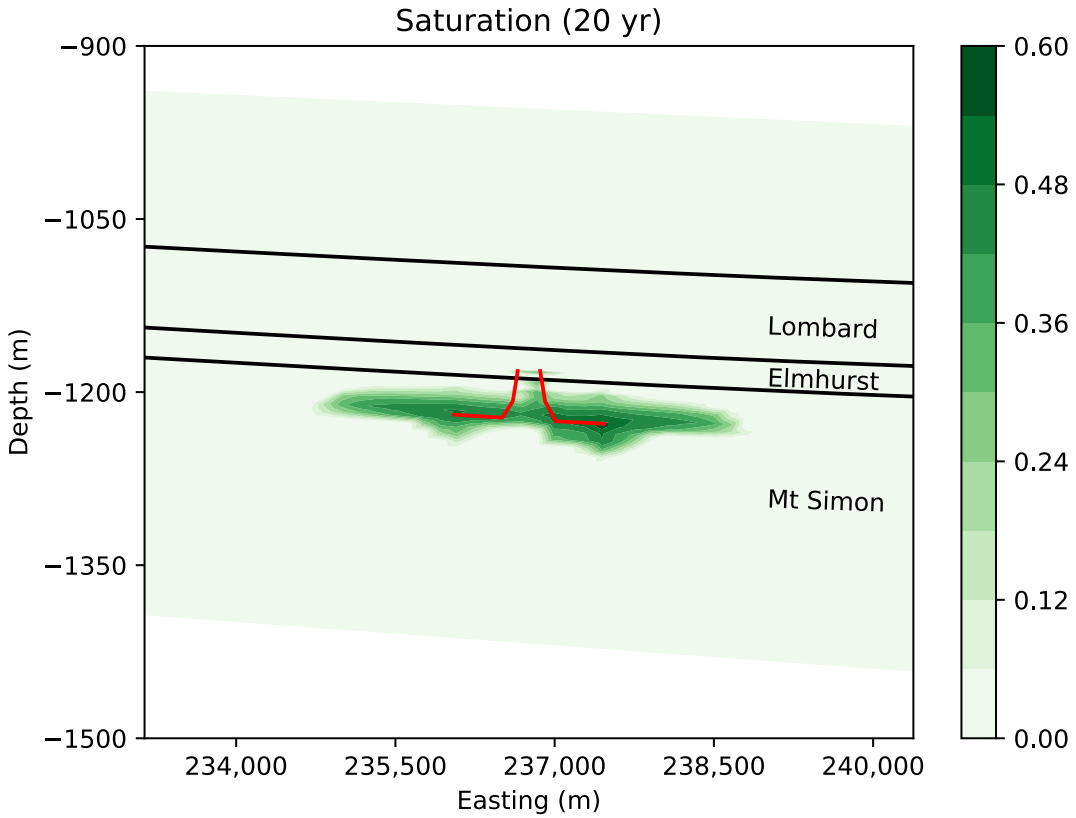

Figure 3. 2D cutaway view of the predicted gas saturation results in the reservoir at the end of injection period (20 yr). Screened portion of horizontal injection wells #1 and #4 are shown in red.

The pressure differentials (i.e., post-injection pressure in the injection formation minus the pre-injection reservoir pressure) for the end of the injection period—when the highest reservoir pressures were observed—are shown in Figure 4 and compared to the extent of the separate-phase CO_2 plume as shown in Figure 5. The plan view represents the extent of the CO_2 plume within the injection layer that is used for developing the lookup tables for the NRAP-Open-IAM model. The separate-phase CO_2 plume is confined to an area 4 km square around the injection wells. The reservoir model grid was approximately 166 km square, but the figure depicting gas saturations displays a 7.2 km square area to show more detail. The footprint of the pressure impact due to the injection operation extends beyond the separate-phase CO_2 plume to an area of approximately 56 km in diameter around the injection wells. The effect of the four injection wells on the distribution of supercritical CO_2 plume near the injection well can also be observed. Max saturation is 60% for the MtSimon11 layer of the model because, as shown in Figure 3, some CO_2 has migrated upward from injection layer.

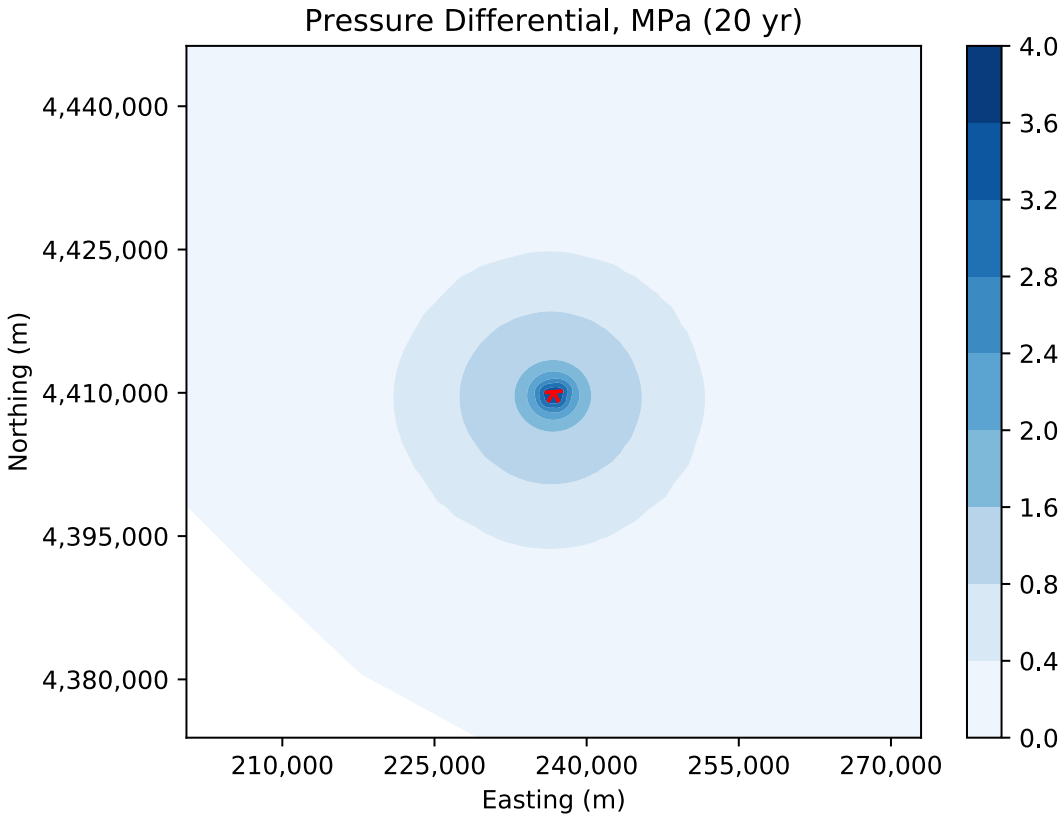

Figure 4. 2D areal view of predicted pressure differential in reservoir MtSimon11 layer at the end of the injection period (shows 72-km square area). Screened portion of horizontal injection wells #1 through #4 are shown in red.

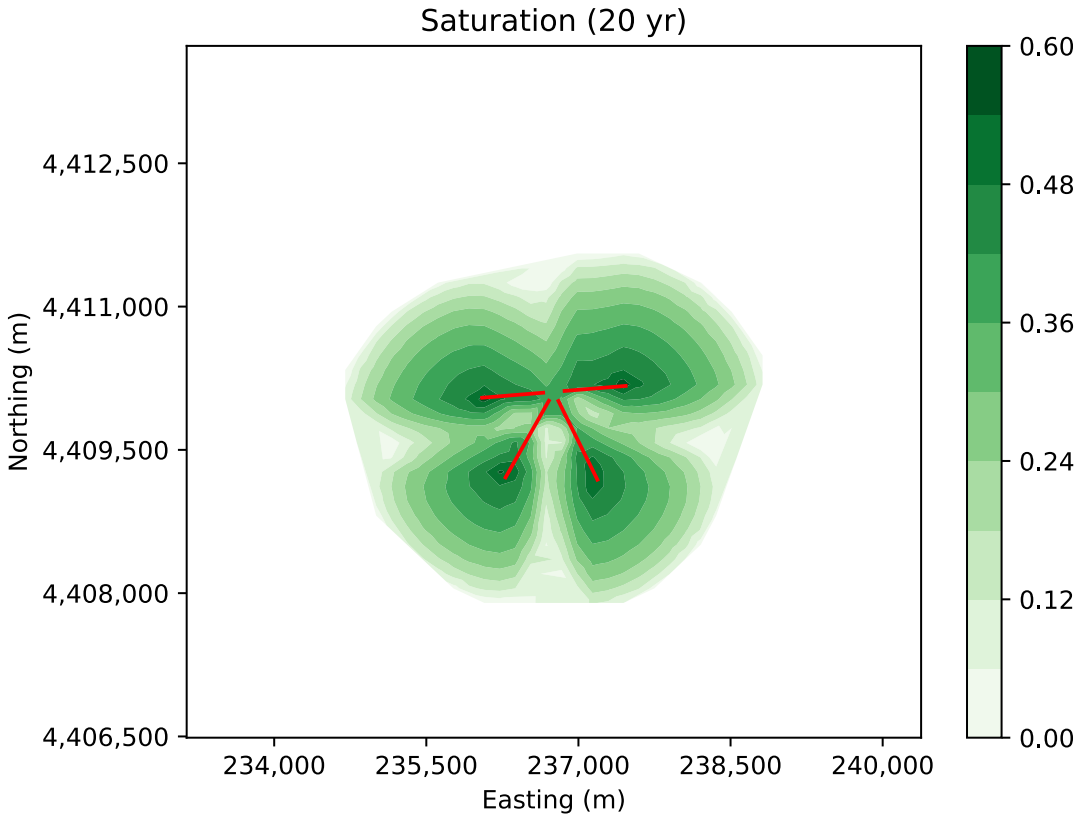

Figure 5. 2D areal view of predicted gas saturation in reservoir in MtSimon11 layer at the end of the injection period (shows 7.2-km square area). Screened portion of horizontal injection wells #1 through #4 are shown in red.

Pressure and free-phase CO_2 saturation simulation results for every year during the injection period and every five years for the 50-year post-injection period were output during these runs to create the lookup table format required by NRAP-Open-IAM. The STOMP-CO2 simulation output consists of text files that contain the grid coordinates and output variables at a given time step.

2.4 Lookup Tables

The pressure and saturation values for each horizontal grid location and time within a single model layer were converted to lookup tables for each run for the reservoir component of the NRAP-Open-IAM. For this application, the injection layer (MtSimon11) was selected to develop the necessary lookup tables.

The lookup table files consist of 1008 comma-delimited text files (.csv) with the grid coordinates and pressure, saturation, and salinity values vs. time. In addition, a file named parameters_and_filenames.csv contains the permeability values for each model layer for each of the 1008 simulations. Finally, a file named time_points.csv contains the simulation output times in years.

The python script for converting STOMP output to a NRAP-Open-IAM lookup table, plot_to_openiam.py, is included in Appendix A.1.

2.5 Testing

Given coordinates x and y and time as input, the FutureGen2 reservoir lookup table should produce pressure and saturation that match the results output by STOMP. Saturations and pressures were read at each grid location from a STOMP plot file at the end of a 20-year injection period. NRAP-Open-IAM output at the same time and grid locations were compared and show one-to-one correspondence (Figure 6). The python script to generate these plots is given in Appendix A.2.

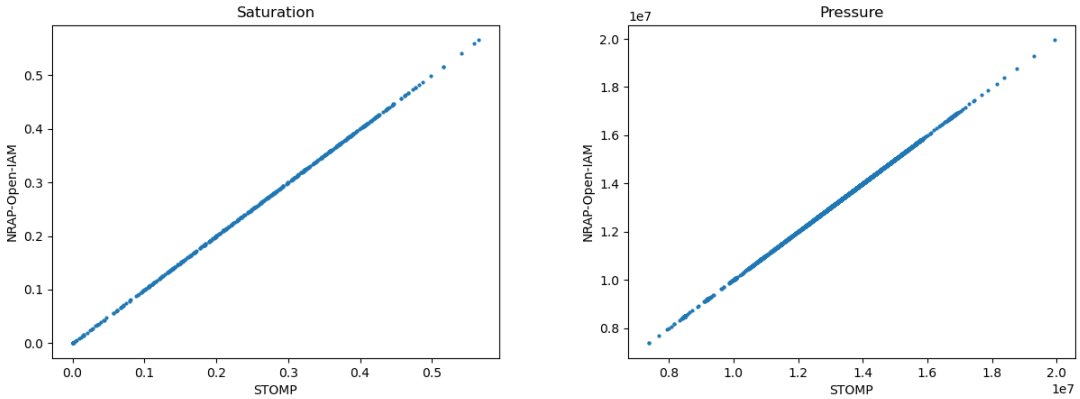

Figure 6. Comparison of STOMP and NRAP-Open-IAM FutureGen2 reservoir lookup table component output for saturation and pressure.

For a single run, a lookup table component can linearly interpolate between over space and time. To demonstrate that the interpolation is working correctly, pressure and saturation at known grid locations and times are compared to space and time-interpolated values at a particular grid location (Figure 7). The python script to generate these plots is given in Appendix A.3.

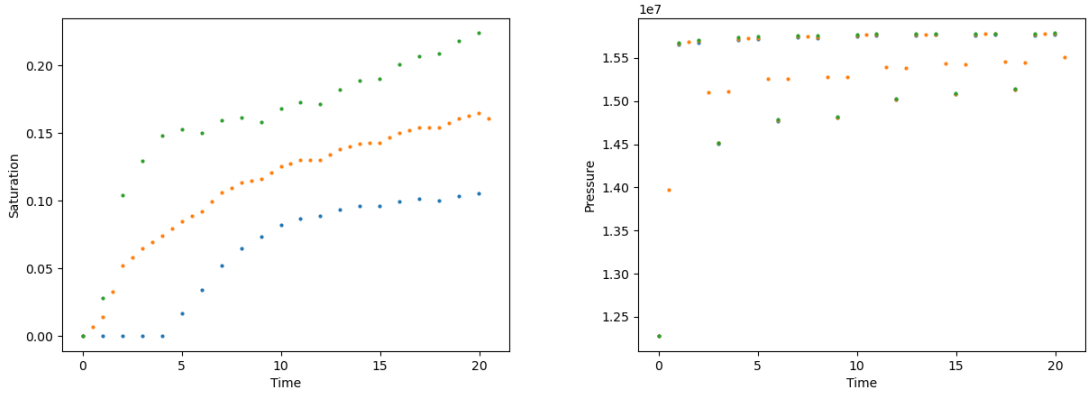

Figure 7. Comparison of known and interpolated values from the FutureGen2 reservoir lookup table component output for saturation and pressure. Blue (x=236681.772) and green (x=236834.172) markers are known values and orange markers (x=236757.972) are interpolated values (all at y=4409421.84).

3.0 FutureGen2 Aquifer and AZMI Components

3.1 Background Data

Based on a model constructed for the original monitoring design at the FutureGen 2.0 site (Vermeul et al. 2016), there are three high-permeability units between the injection reservoir (Mt. Simon) and the lowermost USDW (St. Peter Sandstone): the Ironton-Galesville, the Potosi, and the New Richmond. The aquifer properties used in this previous modeling effort (Table 1) inspired the current work.

Table 1. Hydraulic parameters used in permit application monitoring model (Vermeul et al. 2016).

Model Layer	Thickness (m)	Bottom Depth (m bgs)	Horizontal Permeability (log10 m^2)	Anisotropy (log10 Kh/Kv)	Porosity
St Peter	61.6	-591.9	-11.92	0.30	0.18
New Richmond	31.1	-741.9	-12.48	0.30	0.132
Potosi	84.1	-936.3	-11.05	1.00	0.038
Ironton-Galesville	33.2	-1043.9	-13.39	0.30	0.118

A fluid sample collected from the St. Peter Sandstone during installation of the stratigraphic well resulted in a laboratory-measured TDS value of 3,400 mg/L and field parameter values of 7.91 and 5,910 µS/cm for pH and electrical conductivity, respectively. Because the total dissolved solids measured within this zone was below the upper regulatory limit of 10,000 mg/L for potable aquifers the St. Peter Sandstone was considered to be the lowermost federal USDW for the purposes of the UIC permit application (FutureGen Industrial Alliance 2013b). Salinity measurements from the surficial aquifer (Groschen et al. 2000) and the St. Peter, Ironton, and Mt. Simon (FutureGen Industrial Alliance 2013b) indicate that salinity increases logarithmically with depth, so this relationship was used to interpolate salinity at intermediate depths (Figure 8).

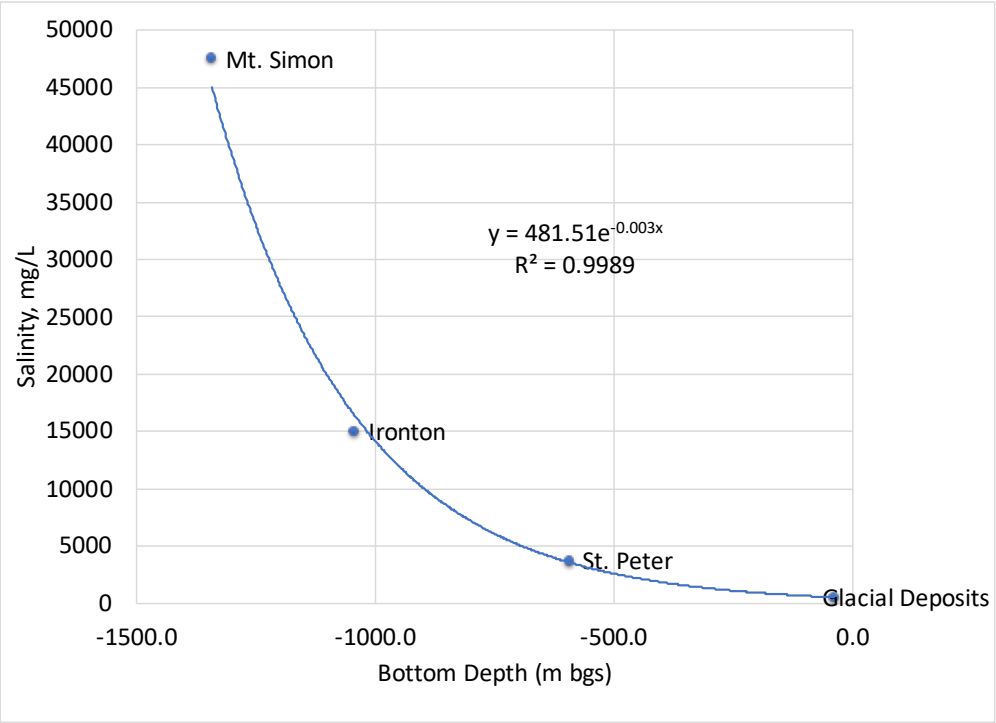

Figure 8. Increase in salinity with depth at the FutureGen 2.0 site.

The initial conditions in the reservoir (Mt. Simon) and USDW (St. Peter) are based on observed values at depth in the stratigraphic well (FutureGen Industrial Alliance 2013c) and are listed in Table 2. The St. Peter is an order of magnitude less saline than the Mt. Simon.

Table 2. Summary of reservoir and USDW initial conditions.

Parameter	Reference Depth (m GS)	Value
Aqueous Saturation		1.0
Mt. Simon Pressure	1,230	12.343 MPa
St. Peter Pressure	533	4.9510 MPa
Mt. Simon Temperature	1,190	35.9 °C
Temperature Gradient		0.0122 °C/m
Mt. Simon Salinity	1342	47,500 ppm
Ironton-Galesville Salinity	1044	15,000 ppm
St. Peter Salinity	592	3,700 mg/L

As indicated in Figure 9, the St. Peter is underpressured relative to a uniform hydrostatic pressure gradient.

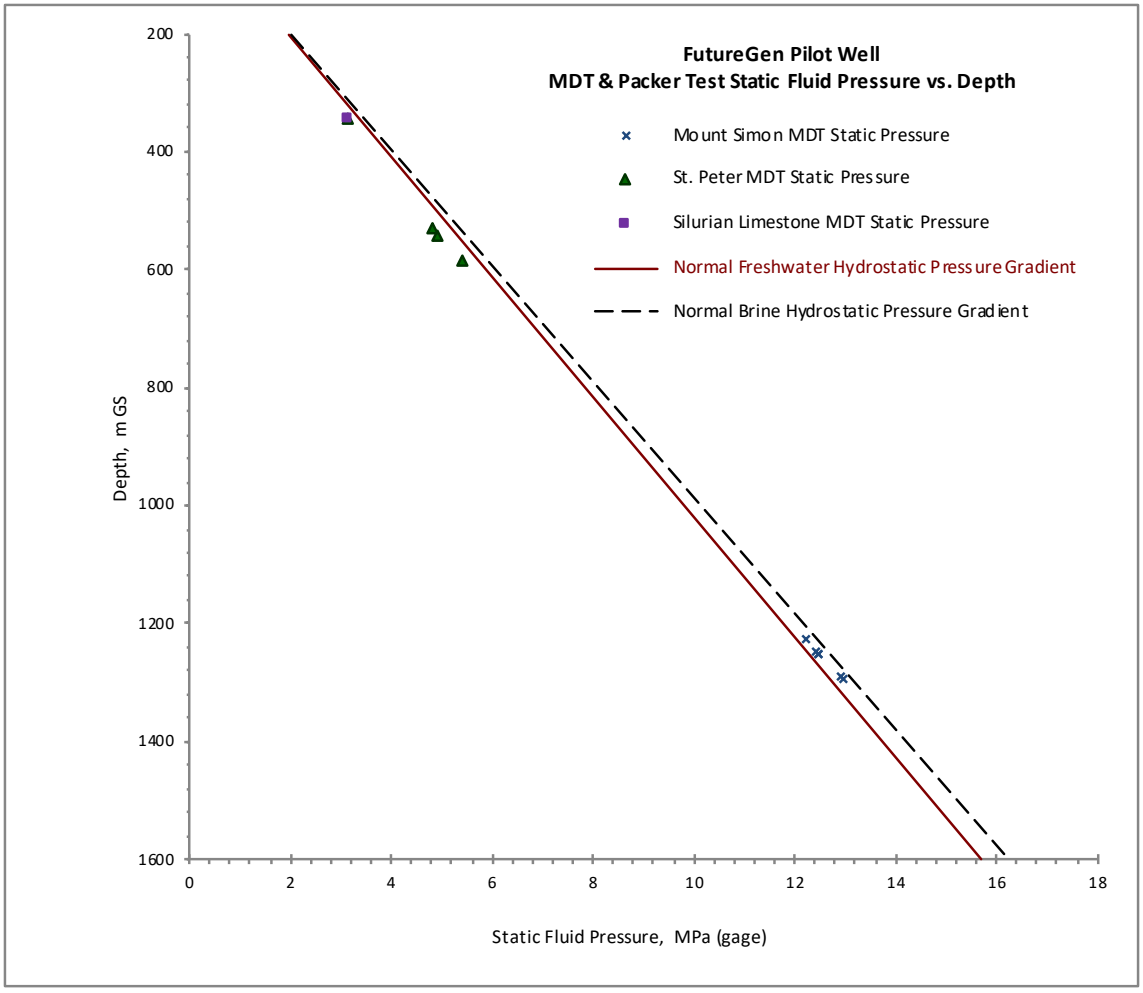

Figure 9. FutureGen pilot (stratigraphic) well formation vs. hydrostatic pressure difference (FutureGen Industrial Alliance 2013c).

3.2 Input Parameters

The AZMI and aquifer components were trained on a range of hydraulic properties (Table 3) so they could be applicable to all four of these high permeability zones. The AZMI simulations were performed with depths ranging from 700 m to 1050 m, whereas the aquifer simulations were performed at shallower depths ranging from 100 m to 700 m. The AZMI and aquifer simulations were performed using a wide range of CO_2 and brine leakage rates, ranging from $1x10^{-9}$ kg/s to 30 kg/s, so that the resulting AZMI Component model could be used with any of the wellbore models currently included in NRAP-Open-IAM.

Table 3. Ranges of input parameters for the Aquifer and AZMI components.

Aquifer Parameters	min	max
Thickness (m)	30	90
Depth (m bgs)		
AZMI	-1050	-700
Aquifer	-700	-100
Porosity	0.02	0.2
Horizontal Permeability (log_{10} m^2)	-14	-11
Anisotropy (log_{10} Kh/Kv)	0	3
Calcite (solid volume fraction)	0	1
Leakage Parameters	**min**	**max**
CO_2 Rate (log_{10} kg/s)	-9	1.5
Brine Rate (log_{10} kg/s)	-9	1.5

3.3 Parameter Sampling

Model input parameters for each simulation run were selected using Latin Hypercube Sampling (Iman, Helton, and Campbell 1981). The parameters listed in Table 2 were assumed to be uniformly distributed. For both the aquifer (Figure 10) and AZMI (Figure 11) models, 480 different combinations of the input parameters were generated.

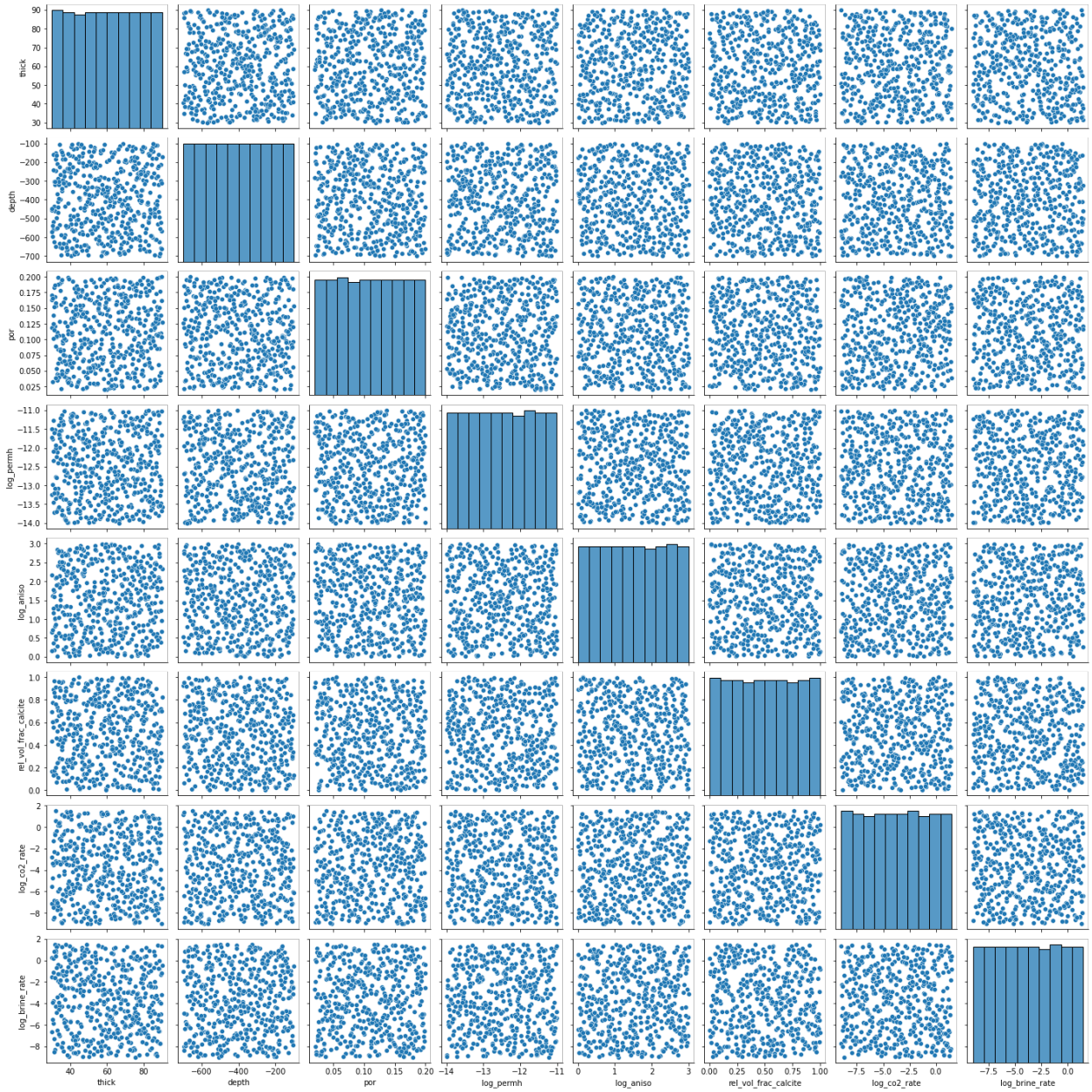

Figure 10. Pair plot for input parameters of simulations used to train the aquifer model, showing the uniform distribution of values in 480 samples.

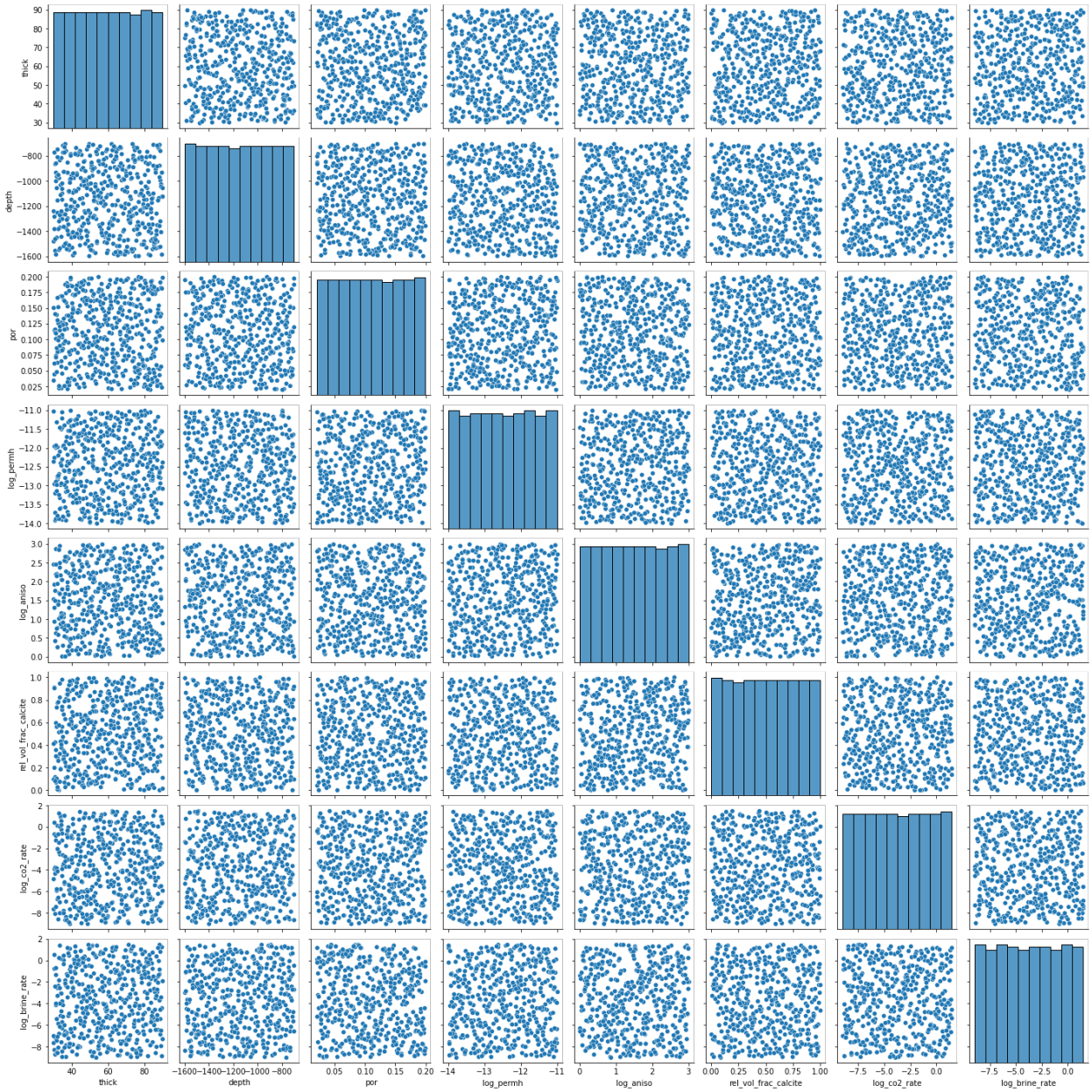

Figure 11. Pair plot for input parameters of simulations used to train the AZMI model, showing the uniform distribution of values in 480 samples

3.4 STOMP Simulations

A series of STOMP simulations were conducted using the parameter samples from section 3.3. Simulations were performed using STOMP-CO2E-R (multiphase flow of CO_2, brine and heat with geochemical reactions). Nonisothermal simulations were performed for training the AZMI component ROM and isothermal simulations were performed for training the aquifer component ROM.

In order to encompass impact plumes for both very small and very large CO_2 or brine leaks, a radial grid of 50 miles (80,467 meters) in radius was used. Grid radii ranged in size from 3.24 m to 5,877 m in the horizontal direction. Ten vertical grids were used, varying in height from 3 to 9 m, for a total of 1000 nodes. For each of the leakage scenarios, distributions of pressure, temperature, dissolved CO_2, pH and TDS were calculated.

The initial temperature and salinity in the aquifer/AZMI simulations were assumed to be a function of depth, based on site characterization data in section 3.1. Hydrostatic initial pressures were assumed.

Figure 12 shows supercritical CO_2, pH and TDS distributions in the St. Peter Sandstone after 20 years of a single, illustrative leakage simulation with equal CO_2 and brine leakage rates of $2.5x10^{-4}$ kg/s. Free-phase CO_2 accumulates within the top 60 m of the aquifer, extending about 260 m from the leakage location. The pH is inversely related to dissolved CO_2, and it decreases from the background value of 7.91 to 4.8. This effect is seen where supercritical CO_2 migrates and where dissolved CO_2 spreads beyond the supercritical CO_2 plume due to density effects and mixing with brine.

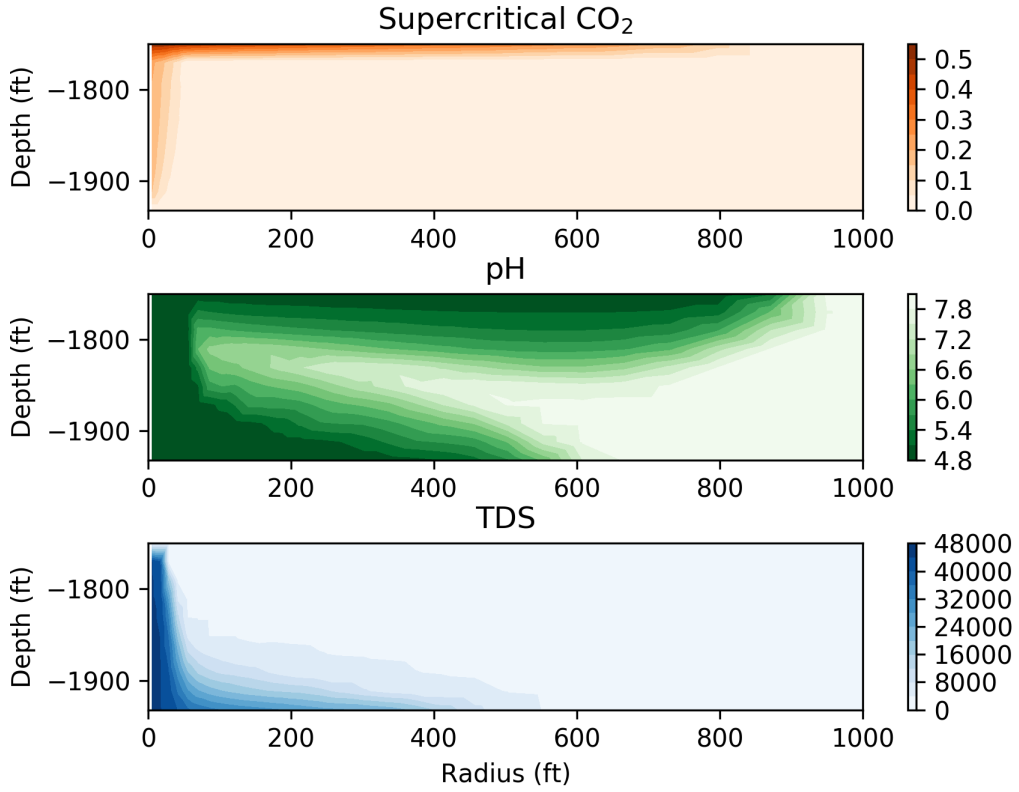

Figure 12. Supercritical CO_2 saturation, pH and TDS in a single aquifer leakage simulation after 20 years.

3.5 Impact Thresholds

In order to delineate the impact in the AZMI or aquifer, a threshold that represents a monitoring detection must be set. The precision thresholds of various sensors were used when developing the FutureGen2 aquifer and AZMI ROMs (Table 4).

Table 4. Monitoring detection thresholds.

	Table A.5 & A.7 (FutureGen Industrial Alliance 2013a)				ROM	
Variable	**Min**	**Max**	**Unit**	**Precision +/-**	**Indicator**	**Threshold**
Pressure	0	2500	psi	0.065%	relative	0.00065
Temperature	0	150	F	0.03%	relative	0.0003
DIC	0.2	--	mg/L	20%	relative	0.2
pH	2	12	pH	0.2	absolute	0.2
TDS	10	--	mg/L	10%	relative	0.1

The original monitoring plan states that “Central to this monitoring strategy is the measurement of CO_2 saturation … using pulsed-neutron capture logging” (FutureGen Industrial Alliance 2013a). However, dissolved inorganic carbon (DIC) was used instead as it would indicate the presence of small leaks where free-phase CO_2 would not be present. Romanak et al. (2012), found that dissolved inorganic carbon (DIC) was a useful monitoring metric because changes in DIC with CO_2 leakage were consistent across geochemical environments, indicating that prior characterization of aquifer minerals may not be necessary if DIC is used as the primary monitoring parameter.

3.6 Plume Delineation

For each original STOMP-CO2E-R leakage simulation, the total volume and dimensions of impacted aquifer/AZMI are calculated for each of the five monitoring variables. For example, the volume of impacted aquifer for pH is calculated by summing the volume of each grid cell in the model where pH changes by more than 2 pH points. The dataset includes the width (dx, dy), height (dz) and volume of the impacted aquifer recorded at 17 times (0, 1, 2, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, and 70 years). Plots of these plume dimensions vs. time are shown in Figure 13 (aquifer) and Figure 14 (AZMI).

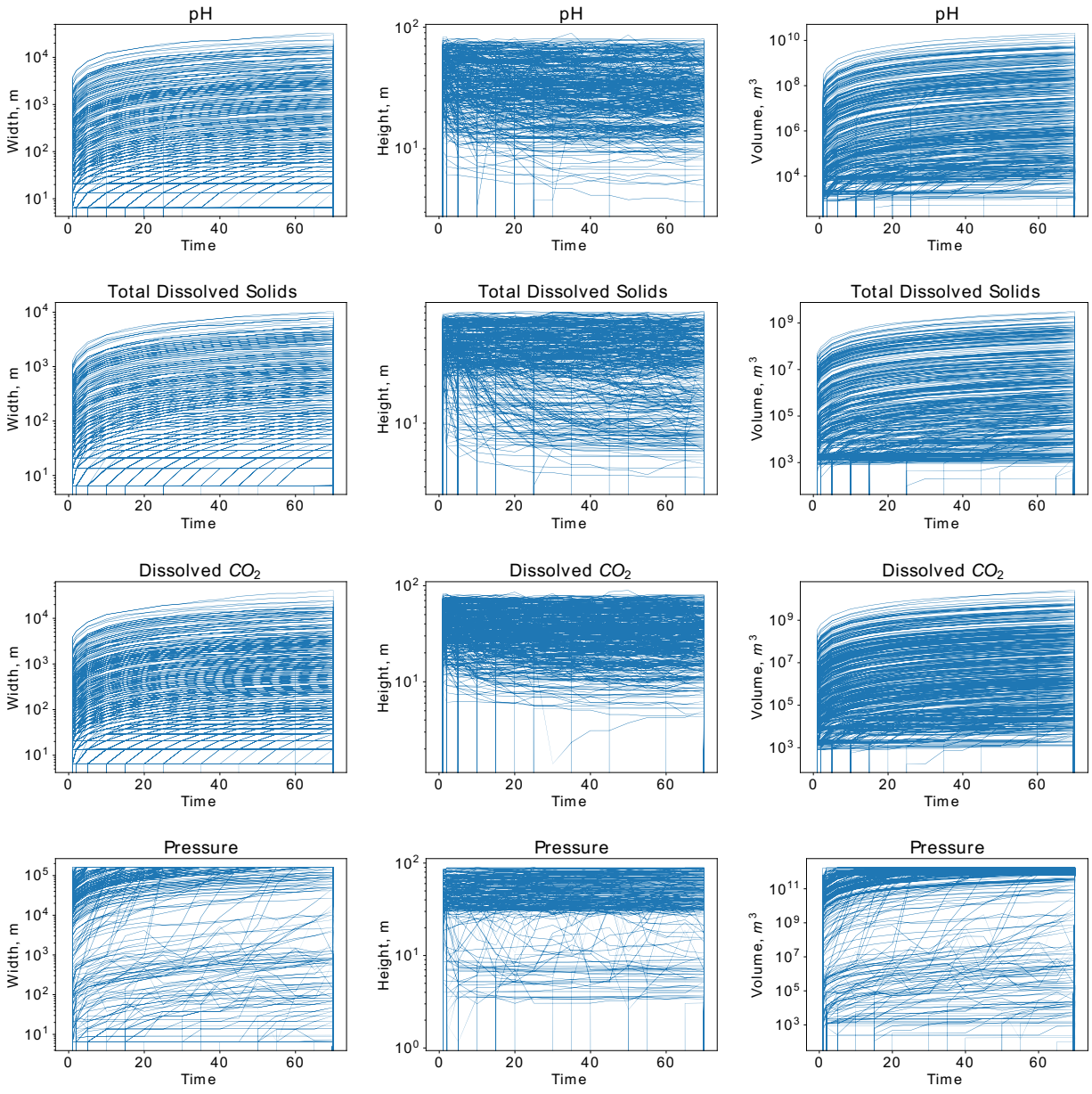

Figure 13. Plume dimensions vs. time for aquifer simulations

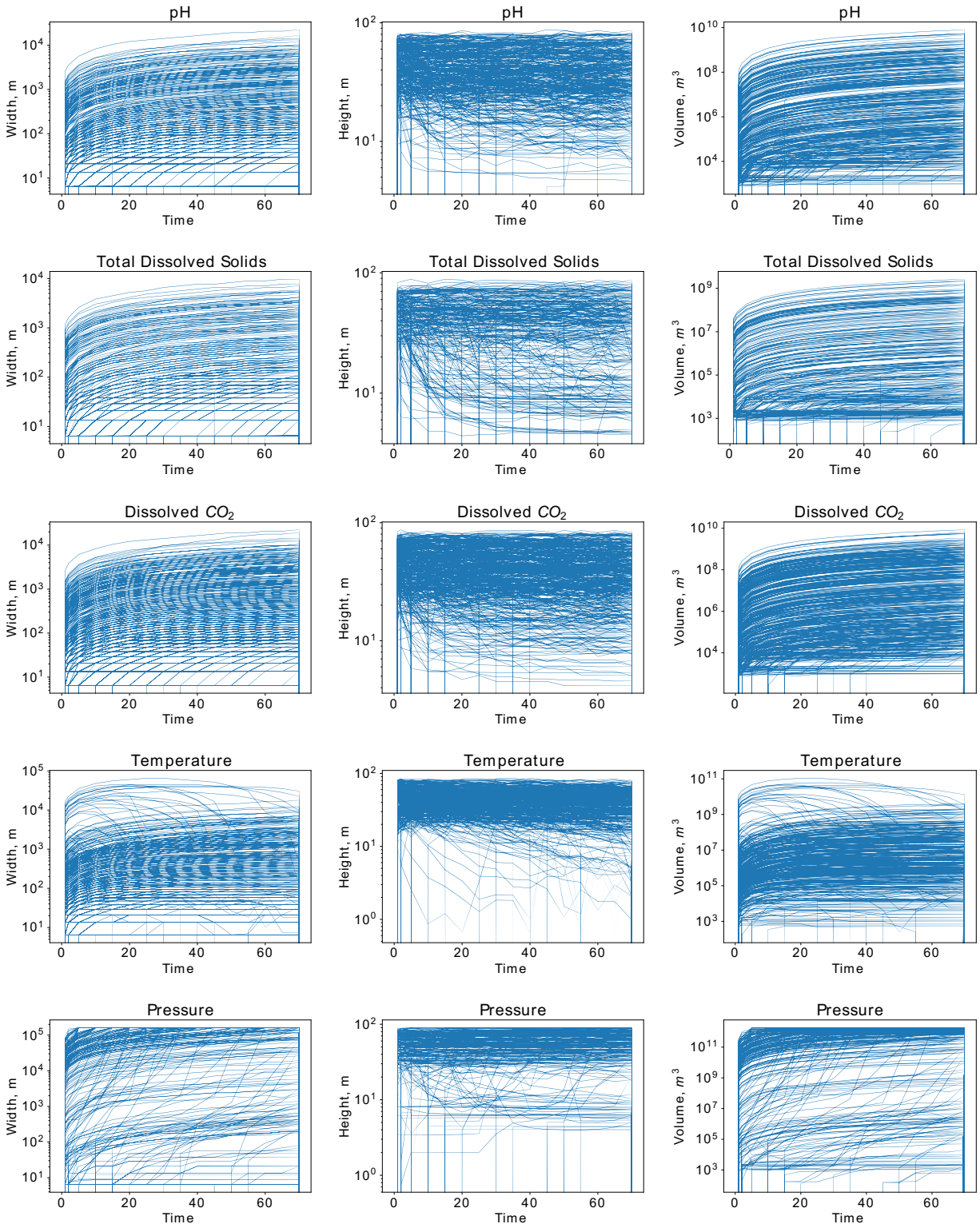

Figure 14. Plume dimensions vs. time for AZMI simulations.

3.7 Training

The FutureGen2 Aquifer component model is a regression model, fit to the simulations of CO_2 and brine leakage into an aquifer, described in the previous chapter. Each of the 480-simulation datasets, one for the aquifer model and one for the AZMI model, were randomly split 70%/30% into training and testing sets.

The regression model was trained using using py-earth (https://contrib.scikit-learn.org/py-earth/index.html). The py-earth package is a Python implementation of the Multivariate Adaptive Regression Splines algorithm (Friedman 1991) in the style of scikit-learn (http://scikit-learn.org), a library of machine-learning methods written in Python.

Multivariate adaptive regression splines is a flexible regression method that automatically searches for interactions and non-linear relationships. Earth models can be thought of as linear models in a higher dimensional basis space (specifically, a multivariate truncated power spline basis). Each term in an Earth model is a product of so called “hinge functions”. A hinge function is a function that’s equal to its argument where that argument is greater than zero and is zero everywhere else.

The multivariate adaptive regression splines algorithm has two stages. First, the forward pass searches for terms in the truncated power spline basis that locally minimize the squared error loss of the training set. Next, a pruning pass selects a subset of those terms that produces a locally minimal generalized cross-validation (GCV) score. The GCV score is not actually based on cross-validation, but rather is meant to approximate a true cross-validation score by penalizing model complexity. The final result is a set of terms that is nonlinear in the original feature space, may include interactions, and is likely to generalize well.

The training statistics for the aquifer (Table 5) and AZMI (Table 6) model are shown below. The volume (vol) and dimensions in the x-, y- and z- dimensions are given. The statistics for dx and dy are always the same, because these are radially symmetric models. By convention, NRAP-Open-IAM aquifer models predict the dimensions of impact plumes in three dimensions. Ideally, the mean square error (MSE) and generalized cross-validation (GCV) scores should be similar and relatively low, indicating that the model is not overfitting. Similarly, the generalized R^2 error (RSQ) and the R^2-like score based on the GCV should be similar and close to one.

Table 5. Train/test statistics for the aquifer model.

Variable		Metric	MSE	GCV	RSQ	GRSQ	Prediction Score
Dissolved CO_2	log	dx	0.3662	0.3746	0.9418	0.9405	0.9437
Dissolved CO_2	log	dy	0.3662	0.3746	0.9418	0.9405	0.9437
Dissolved CO_2	log	dz	0.2651	0.2759	0.7859	0.7773	0.7674
Dissolved CO_2	log	vol	2.2255	2.2602	0.9283	0.9272	0.9307
pH	log	dx	0.7156	0.7446	0.9201	0.9169	0.9130
pH	log	dy	0.7156	0.7446	0.9201	0.9169	0.9130
pH	log	dz	0.545	0.5579	0.785	0.78	0.7768
pH	log	vol	6.0776	6.1848	0.8785	0.8764	0.8598
Pressure	log	dx	2.0994	2.1582	0.9252	0.9231	0.9213
Pressure	log	dy	2.0994	2.1582	0.9252	0.9231	0.9213
Pressure	log	dz	0.6325	0.6535	0.8275	0.8218	0.8103
Pressure	log	vol	11.7228	12.0508	0.9221	0.92	0.9183
TDS	log	dx	0.2298	0.2339	0.968	0.9675	0.9671
TDS	log	dy	0.2298	0.2339	0.968	0.9675	0.9671
TDS	log	dz	0.4261	0.4398	0.8423	0.8373	0.8310
TDS	log	vol	1.5188	1.5503	0.964	0.9633	0.9612

Table 6. Train/test statistics for the AZMI model.

Parameter		Metric	MSE	GCV	RSQ	GRSQ	Prediction Score
Dissolved CO2	log	dx	0.3403	0.3468	0.9402	0.9391	0.9475
Dissolved CO2	log	dy	0.3403	0.3468	0.9402	0.9391	0.9475
Dissolved CO2	log	dz	0.2752	0.2823	0.7846	0.7791	0.7883
Dissolved CO2	log	vol	2.0688	2.1044	0.9295	0.9283	0.9424
pH	log	dx	0.7289	0.747	0.9084	0.9061	0.9061
pH	log	dy	0.7289	0.747	0.9084	0.9061	0.9061
pH	log	dz	0.4022	0.413	0.8455	0.8415	0.8505
pH	log	vol	5.8461	5.9801	0.8743	0.8714	0.8586
Pressure	log	dx	2.3855	2.4425	0.9078	0.9057	0.9009
Pressure	log	dy	2.3855	2.4425	0.9078	0.9057	0.9009
Pressure	log	dz	0.6414	0.6591	0.829	0.8243	0.8239
Pressure	log	vol	13.4567	13.7523	0.9045	0.9024	0.8976
TDS	log	dx	1.0008	1.0161	0.8601	0.858	0.8612
TDS	log	dy	1.0008	1.0161	0.8601	0.858	0.8612
TDS	log	dz	0.6665	0.6927	0.7954	0.7875	0.8004
TDS	log	vol	6.8869	6.9729	0.8496	0.8477	0.8524
Temperature	log	dx	0.4575	0.4662	0.9437	0.9427	0.9370
Temperature	log	dy	0.4575	0.4662	0.9437	0.9427	0.9370
Temperature	log	dz	0.372	0.3816	0.8585	0.8549	0.8569
Temperature	log	vol	2.9594	3.0076	0.9385	0.9376	0.9336

3.8 Testing

The prediction scores for the aquifer (Table 5) and AZMI (Table 6) models on the test set are generally acceptable. They are similar in magnitude to the RSQ and GRSQ obtained on the training set, indicating that the model is not overfitting to the training set. The plume volume (vol) and width (dx and dy) scores are higher, between 0.85 and 0.96, while the plume height (dz) scores are lower, between 0.78 and 0.85.

Plots of the predicted vs actual plume dimensions are shown in Figures 7 and 8. Each of the plume dimensions, volume, width (dx and dy), and height (dz), were scaled using a ln(x + 1) transform, referred to in the numpy library (Harris et al. 2020) as log1p. This transform resulted in a better fit across the entire range of simulations.

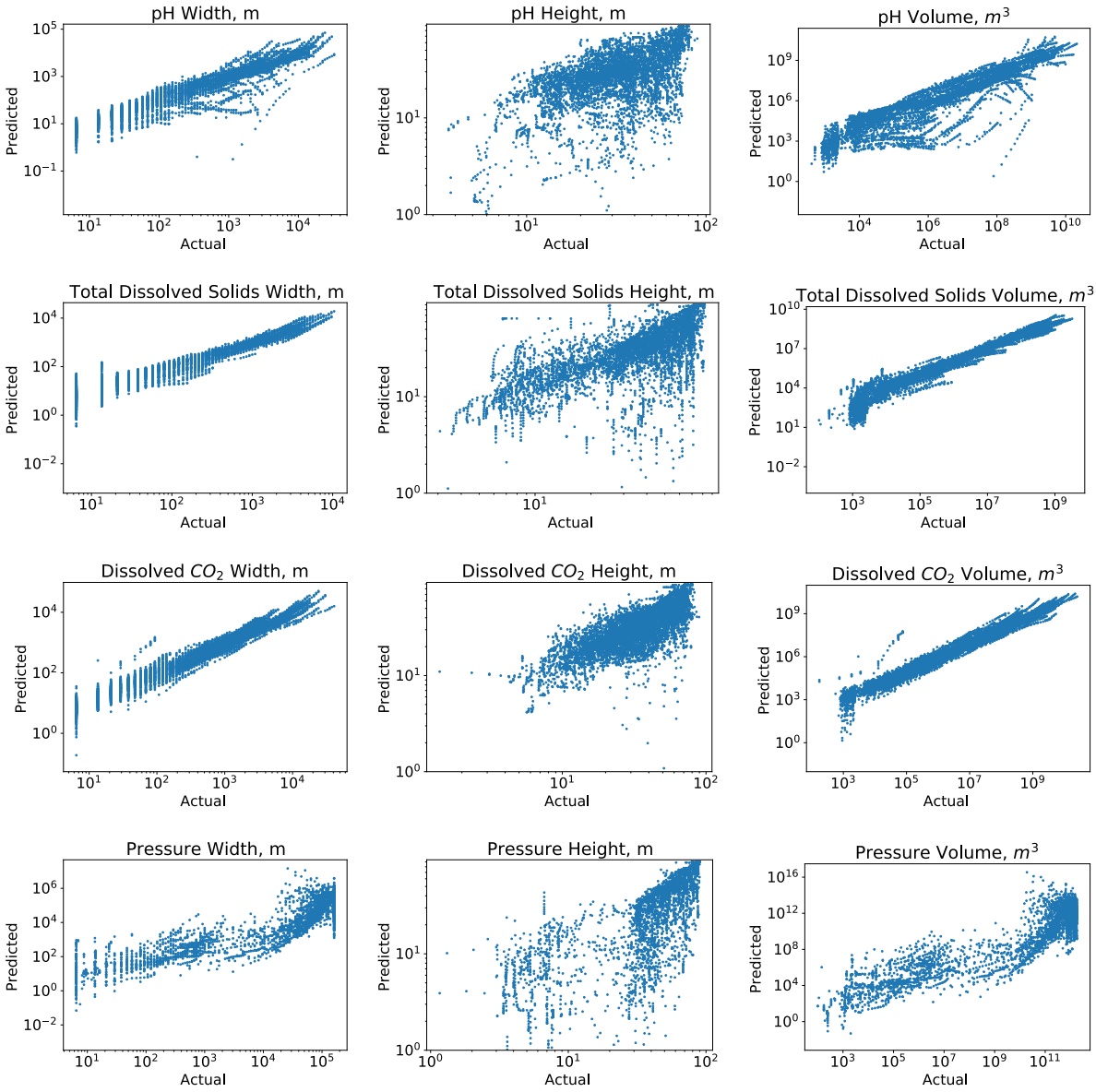

Figure 15. Comparison between aquifer model predicted and actual plume dimensions.

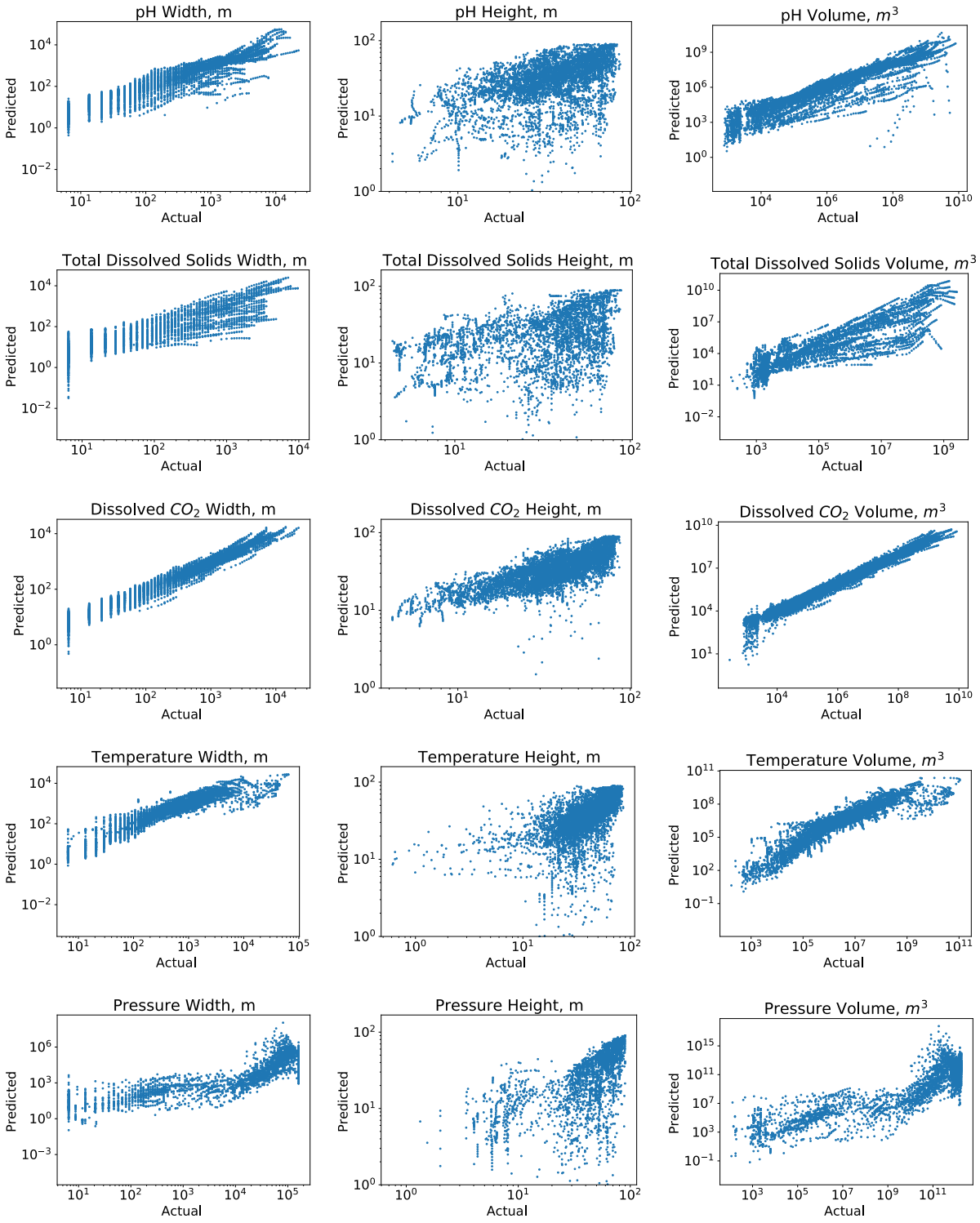

Figure 16. Comparison between AZMI model predicted and actual plume dimensions.

4.0 Applications

The FutureGen2 components have been used in two recent applications.

NRAP-Open-IAM and DREAM (Figure 17) were used to determine a risk-based PISC period and optimized monitoring network for a commercial-scale CO2 storage project (Bacon et al. 2019). Realizations from NRAP-Open-IAM revealed that maximum simulated leakage rates of brine were small, on the order of 10−5 kg/s, and maximum simulated leakage rates of CO_2 were on the order of 10-3 kg/s and could be detected earliest during the injection phase in the Ironton-Galesville, the thief zone immediately overlying the injection reservoir. Using this information to design an optimized monitoring well network eliminated one of the three originally planned monitoring wells, resulting in a cost reduction for the project. Perhaps the most significant finding from this effort is that NRAP-Open-IAM can be used to define a risk-based, and substantially shorter, PISC period for the site. NRAP-Open-IAM realizations indicate that most of the risk of endangerment to USDWs decreases within the first 5 years after CO_2 injection ends. Doubling this timeframe would still lead to a net PISC period reduction of 40-years and an operational cost reduction of more than $50 M for the project. An example python script for this application, named *iam_sys_lutreservoir_mswell_futuregen_dream.py*, is included in the NRAP-Open-IAM source code.

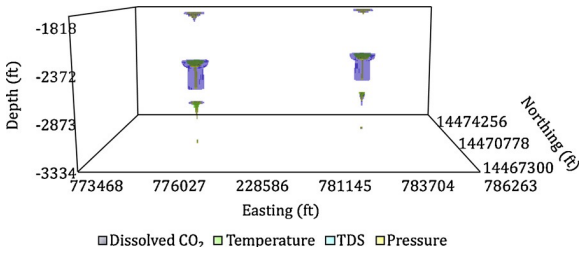

Figure 17. DREAM generated solution space which was used for optimization of a leakage monitoring network. Colored zones represent all locations where the leakage threshold of the respective monitoring parameter was exceeded.

NRAP-Open-IAM was used to develop a probabilistic estimate of impact risk to USDW quality (Bacon, Demirkanli, and White 2020). CO_2 and pressure predictions from the reservoir modeling conducted using the STOMP-CO2 simulator for the FutureGen 2.0 site are used in a NRAP-Open-IAM model with reservoir, wellbore, and aquifer components to: (1) assess the extent of potential leakage into the USDW for the predicted reservoir pressure conditions; (2) evaluate the extent of potential impact using “no-net-degradation” thresholds; and (3) account for uncertainty in reservoir permeabilities. Regulatory oversight of a geologic carbon sequestration (GCS) project relies on iterative estimations, throughout the project lifetime, of the area where increased risks to underground sources of drinking water (USDWs) may occur due to injection of CO_2. This area, referred to as Area of Review (AoR), is typically delineated by predicting the migration of fluid between the reservoir and the lowermost USDW via an open wellbore using predictions from physics-based reservoir simulators. Accounting for the probability of aquifer impact using NRAP-Open-IAM results in a smaller “risk-based Area of Review” (Figure 18). An example python script for this application (*iam_sys_lutreservoir_openwell_futuregen_aor.py*) is included in the NRAP-Open-IAM source code.

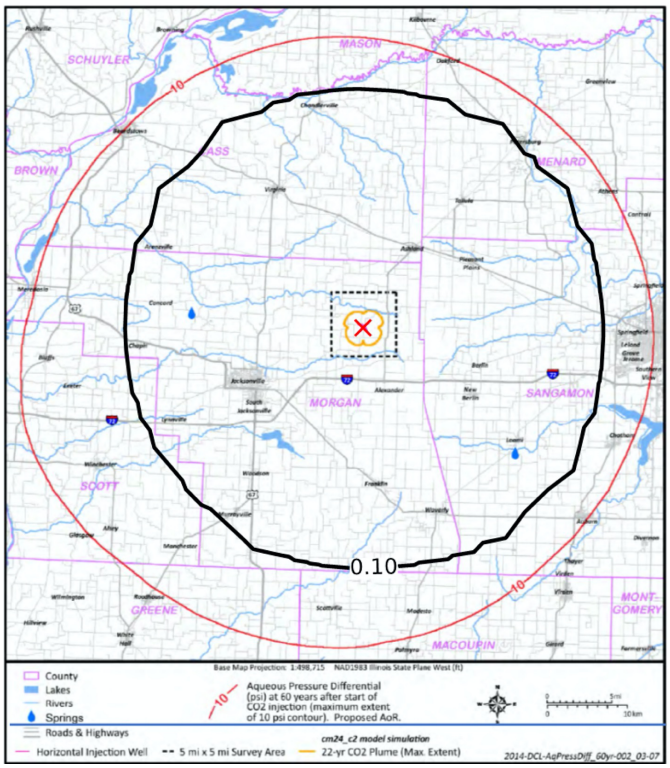

Figure 18. Risk-based AoR defined by the pressure differential corresponding to the pH and TDS nonzero impact probability distributions (black line indicating 0.1 MPa) compared to the permitted AoR for the FutureGen 2.0 site (pressure differential of 0.69 MPa/10 psi at 60 years after the start of injection) (Bacon et al. 2020).

5.0 Summary and Conclusions

The FutureGen2 reservoir lookup table component provides a fast surrogate model for exploring the effect of variations in reservoir permeability. The reservoir component exactly reproduces the results of the multiphase flow simulator STOMP for a model of the FutureGen 2.0 site. The FutureGen2 Aquifer and AZMI components for NRAP-Open-IAM quickly calculate the potential impact of brine and CO_2 leakage into high permeability formations overlying the Mt. Simon sandstone, including the Ironton-Galesville, Potosi Dolomite, New Richmond, and St. Peter sandstone. Model predictions of impact volume and width are more accurate, whereas the prediction of aquifer height is more challenging. The FutureGen2 components have been useful in two recent applications of NRAP-Open-IAM: calculating a risk-based post-injection site care (PISC) period and a risk-based area of review (AoR).

6.0 References

Bacon, D. H., D. I. Demirkanli, and S. K. White. 2020. "Probabilistic risk-based Area of Review (AoR) determination for a deep-saline carbon storage site." *International Journal of Greenhouse Gas Control* 102: 103153. https://doi.org/10.1016/j.ijggc.2020.103153.

Bacon, D. H., C. M. R. Yonkofski, C. F. Brown, D. I. Demirkanli, and J. M. Whiting. 2019. "Risk-based post injection site care and monitoring for commercial-scale carbon storage: Reevaluation of the FutureGen 2.0 site using NRAP-Open-IAM and DREAM." *International Journal of Greenhouse Gas Control* 90: 102784. https://doi.org/10.1016/j.ijggc.2019.102784.

Friedman, J. H. 1991. "Multivariate adaptive regression splines." *Annals of Statistics* 19 (1): 1-67.

FutureGen Industrial Alliance, I. 2013a. *Underground Injection Control Permit Applications for FutureGen 2.0 Morgan County Class VI UIC Wells 1, 2, 3, and 4 – ATTACHMENT C: TESTING AND MONITORING PLAN.* FG-RPT-017-Revision 1. Jacksonville, Illinois. https://archive.epa.gov/region5/water/uic/futuregen/web/pdf/attachment-c.pdf.

□. 2013b. *Underground Injection Control Permit Applications for FutureGen 2.0 Morgan County Class VI UIC Wells 1, 2, 3, and 4 – Chapter 2.0 Geology and Hydrology.* FG-RPT-017-Revision 1. Jacksonville, Illinois. https://archive.epa.gov/region5/water/uic/futuregen/web/pdf/futuregen-permitapp-201303.pdf.

□. 2013c. *Underground Injection Control Permit Applications for FutureGen 2.0 Morgan County Class VI UIC Wells 1, 2, 3, and 4 – Supporting Documentation.* FG-RPT-017-Revision 1. Jacksonville, Illinois.

Groschen, G. E., M. A. Harris, R. B. King, P. J. Terrio, and K. L. Warner. 2000. *Water Quality in the Lower Illinois River Basin, Illinois, 1995–98: U.S. Geological Survey Circular 1209.* https://pubs.water.usgs.gov/circ1209/.

Harris, C. R., K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Rio, M. Wiebe, P. Peterson, P. Gerard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. 2020. "Array programming with NumPy." *Nature* 585 (7825): 357-362. https://doi.org/10.1038/s41586-020-2649-2.

Iman, R. L., J. C. Helton, and J. E. Campbell. 1981. "An Approach to Sensitivity Analysis of Computer Models: Part I—Introduction, Input Variable Selection and Preliminary Variable Assessment." *Journal of Quality Technology* 13 (3): 174-183. https://doi.org/10.1080/00224065.1981.11978748.

NETL. 2015. *Carbon Storage Atlas, 5th Edition.* National Energy Technology Laboratory Pittsburgh, PA.

□. 2018. "Carbon Storage Research." Accessed 11/1. https://www.energy.gov/fe/science-innovation/carbon-capture-and-storage-research.

Romanak, K. D., R. C. Smyth, C. Yang, S. D. Hovorka, M. Rearick, and J. Lu. 2012. “Sensitivity of groundwater systems to CO2: Application of a site-specific analysis of carbonate monitoring parameters at the SACROC CO2-enhanced oil field.” *International Journal of Greenhouse Gas Control* 6: 142-152. https://doi.org/http://dx.doi.org/10.1016/j.ijggc.2011.10.011.

USEPA. 2010. Federal Requirements Under the Underground Injection Control (UIC) Program for Carbon Dioxide (CO2) Geologic Sequestration (GS) Wells. In *40 CFR Parts 124, 144, 145, 146, and 147*, edited by Environmental Protection Agency. Federal Register.

Vermeul, V. R., J. E. Amonette, C. E. Strickland, M. D. Williams, and A. Bonneville. 2016. “An overview of the monitoring program design for the FutureGen 2.0 CO 2 storage site.” *International Journal of Greenhouse Gas Control* 51: 193-206. https://doi.org/10.1016/j.ijggc.2016.05.023.

Zhang, Z. F., S. K. White, A. Bonneville, and T. J. Gilmore. 2014. “Local Sensitivity of Predicted CO2 Injectivity and Plume Extent to Model Inputs for the FutureGen 2.0 site.” *12th International Conference on Greenhouse Gas Control Technologies, Ghgt-12* 63: 3805-3814. https://doi.org/10.1016/j.egypro.2014.11.409.

Appendix A Python Scripts

A.1 plot_to_openiam.py

```
import os
import sys
import argparse
import string
from math import ceil
import numpy as np
from os.path import join, getsize
import pandas as pd
import glob

# STOMP plot file
class Plot_File:

    convert_length = {'m':1.0, 'ft':0.3048, 'km':1000., 'mi':1609.34,
'mile':1609.34,
        'nm':1e-9, 'cm':0.01, 'mm':0.001, 'yd':0.9144, 'in':0.0254}
    convert_pressure = {'pa':1.0, 'psi':6894.76, 'mpa':1000000.,
'atm':101325., 'bar':100000.}

    def __init__(self, file_name):

        self.file_name = file_name
        self.input_list = self.read_file(self.file_name)
        self.nx = self.get_parameter('Number of X or R-Direction Nodes =')
        self.ny = self.get_parameter('Number of Y or Theta-Direction Nodes
=')
        self.nz = self.get_parameter('Number of Z-Direction Nodes =')
        self.nfield = self.get_parameter('Number of Field Nodes =')
        self.nactive = self.get_parameter('Number of Active Nodes =')
        self.nvert = self.get_parameter('Number of Vertices =')

    # read file into list
    def read_file(self, file_name):
        input_list = []
        with open(file_name) as f:
            input_list = f.readlines()
            # remove whitespace characters
            input_list = [line.strip() for line in input_list]
        return input_list

    # read x, y, or z dimension
    def get_parameter(self, search_string):
        for m,line in enumerate(self.input_list):
            if search_string in line:
                # remove whitespace characters
                line_list = [item.strip() for item in line.split('=')]
        return int(line_list[-1])

    # get line number with variable name
```

```
    def find_variable(self, search_string):
        for m,line in enumerate(self.input_list):
            if search_string in line:
                break
        return m

    # read or calculate node centroids
    def get_centroids(self, direction, layer):
        zlist = []
        if direction == 'Z':
            m = self.find_variable(direction + ' Node-Centroid Position')
            if m < len(self.input_list) - 1:
                start = m+1
                end = start+int(ceil(self.nfield/10.))
                unit = self.input_list[m].split()[-1]
                for line in self.input_list[start:end]:
                    line_list =
[float(item.strip())*self.convert_length[unit] for item in line.split()]
                    zlist.extend(line_list)
        elif direction == 'X':
            m = self.find_variable('X-Direction Nodal Vertices')
            start = m+1
            end = start+self.nfield
            unit = self.input_list[m].split()[-1]
            for line in self.input_list[start:end]:
                line_list = [float(item.strip()) for item in line.split()]
                centroid = self.convert_length[unit] * (line_list[0] +
line_list[1]) / 2.
                zlist.append(centroid)
        elif direction == 'Y':
            m = self.find_variable('Y-Direction Nodal Vertices')
            start = m+1
            end = start+self.nfield
            unit = self.input_list[m].split()[-1]
            for line in self.input_list[start:end]:
                line_list = [float(item.strip()) for item in line.split()]
                centroid = self.convert_length[unit] * (line_list[0] +
line_list[2]) / 2.
                zlist.append(centroid)
        else:
            sys.exit('Plot file must contain Z Node-Centroid Position')
        zarray = np.array(zlist).reshape((self.nx, self.ny, self.nz),
order='F')
        zlayer = zarray[:,:,layer-1]
        return zlayer.flatten(order='F')

    def get_variable(self, name, layer, unit_conversion=None):
        vlist = []
        m = self.find_variable(name)
        start = m+1
        end = start+int(ceil(self.nfield/10.))
        unit = self.input_list[m].split()[-1]
        if unit_conversion is not None:
            conversion_factor = unit_conversion[unit]
        else:
```

```
            conversion_factor = 1.0
        for line in self.input_list[start:end]:
            line_list = [float(item.strip())*conversion_factor for item in
line.split()]
            vlist.extend(line_list)
        varray = np.array(vlist).reshape((self.nx, self.ny, self.nz),
order='F')
        vlayer = varray[:,:,layer-1]
        return vlayer.flatten(order='F')

    def get_time(self,unit):
        m = self.find_variable('Time = ')
        line_list = [item.strip() for item in self.input_list[m].split()]
        time_list = [i.split(',') for i in line_list[2:]]
        time_dict = {x[1]: x[0] for x in time_list}
        return time_dict[unit]

# input arguments
parser = argparse.ArgumentParser(description='Convert STOMP plot files into
OpenIAM reservoir lookup table file')
parser.add_argument('--dir', default='.', help='directory with results of
STOMP simulation')
# parser.add_argument('--inact', help='inactive nodes file name')
parser.add_argument('--layer', default=9, type=int, help='model layer to
extract results from')
parser.add_argument('--out', default='output.csv', help='output file name')
args = parser.parse_args()

layer = args.layer
times = []

# get plot file names
files = glob.glob(os.path.join(args.dir,'plot.*'))
files.sort()

# get grid info from first file
pf = Plot_File(files[0])

# validate layer number
if args.layer < 1 or args.layer > pf.nz:
    sys.exit('ERROR: User-specified layer '+str(args.layer)+' is out of
range: '+str(1)+' to '+str(pf.nz))

x = pf.get_centroids('X',layer)
y = pf.get_centroids('Y',layer)
z = pf.get_centroids('Z',layer)
z = z - 633 * 0.3048 # Kelly bushing
df = pd.DataFrame({'x': x, 'y': y, 'z': z})

df['area'] = pf.get_variable('Z-Dir. Surface Area',layer)
df['rock'] = pf.get_variable('Rock/Soil Type',layer)

# read variables and times from each plot file
for i,file_path in enumerate(files):
```

```
        # create plot file object
        pf = Plot_File(file_path)

        t = pf.get_time('yr')
        times.append(t)
        print(t)

        df['pressure_' + str(i+1)] = pf.get_variable('Gas
Pressure',layer,pf.convert_pressure)
        df['CO2saturation_' + str(i+1)] = pf.get_variable('Gas
Saturation',layer)
        df['salinity_' + str(i+1)] = pf.get_variable('Aqueous Salt Mass
Fraction',layer)

# use rock type to filter out inactive nodes, then remove
df.drop(df[df.rock == 0].index, inplace=True)

# write OpenIAM reservoir lookup table
df.drop(['rock'], axis=1).to_csv(args.out,index=False)

# write times to file
filename = 'time_points.csv'
f = open(filename,"w+")
f.write(','.join(times) + '\n')
f.close()
```

A.2 Test script for FutureGen2 reservoir lookup table

```
'''
Test FutureGen2 Reservoir Lookup Table

Compares STOMP output to FutureGen2 reservoir lookup table predictions.

This example requires the additional FutureGen 2.0 data set.
FutureGen 2.0 data set can be downloaded from the following source:
https://edx.netl.doe.gov/dataset/futuregen-2-0-1008-simulation-reservoir-
lookup-table

The downloaded data set should be placed here:
    source/components/reservoir/lookuptables/FutureGen2/1008_sims

Usage examples:
$ python test_iam_sys_lutreservoir.py --run 1
'''
# @author: Diana Bacon
# diana.bacon@pnnl.gov

import os
import sys
import argparse
import datetime
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from plot_to_openiam import Plot_File
```

```
import pandas as pd
import glob


base_path =
os.path.join os.sep,'Users','d3a926','Desktop','OpenIAM','UQ_example_setup','
source'
sys.path.insert 0, base_path
from openiam import  SystemModel, ReservoirDataInterpolator,
LookupTableReservoir

# read STOMP plot file into dataframe
def plot_to_df file, layer

    pf = Plot_File file

    # validate layer number
    if layer < 1 or layer > pf.nz
        sys.exit 'ERROR: User-specified layer '+str layer +' is out of range:
'+str 1 +' to '+str pf.nz

    x = pf.get_centroids 'X',layer
    y = pf.get_centroids 'Y',layer
    z = pf.get_centroids 'Z',layer
    z = z - 633 * 0.3048 # Kelly bushing
    df = pd.DataFrame  'x': x, 'y': y, 'z': z

    df 'area'  = pf.get_variable 'Z-Dir. Surface Area',layer
    df 'rock'  = pf.get_variable 'Rock/Soil Type',layer

    t = pf.get_time 'yr'

    df 'pressure'  = pf.get_variable 'Gas
Pressure',layer,pf.convert_pressure
    df 'CO2saturation'  = pf.get_variable 'Gas Saturation',layer
    df 'salinity'  = pf.get_variable 'Aqueous Salt Mass Fraction',layer

    # use rock type to filter out inactive nodes, then remove
    df.drop df df.rock == 0 .index, inplace=True

    return  df, float t

if __name__ == "__main__"
    # For multiprocessing in Spyder
    __spec__ = None

    file_directory = os.sep.join  base_path, 'components', 'reservoir',
                                  'lookuptables', 'FutureGen2', '1008_sims'

    if not os.path.exists os.sep.join  file_directory, 'fg1.csv'
        url = ''.join
            'https://edx.netl.doe.gov/dataset/',
            'futuregen-2-0-1008-simulation-reservoir-lookup-table \n'
        msg = ''.join
            '\nFutureGen 2.0 data set can be downloaded ',
```

```
            'from the following source:\n',
            url,
            'Check this example description for more information.'
        print msg

    # Input arguments
    parser = argparse.ArgumentParser description='Test FutureGen2 reservoir
lookup table'
    parser.add_argument '--run', default='1', help='run number to process'
    parser.add_argument '--file', default='.', help='plot file with results
of STOMP simulation'
    parser.add_argument '--layer', default=9, type=int, help='model layer to
extract results from'
    args = parser.parse_args
    run = int args.run

    # read STOMP results
    df,time = plot_to_df args.file, args.layer
    print df.head

    # Define keyword arguments of the system model
    num_years = int time
    time_array = 365*np.arange 0.0, num_years+1
    sm_model_kwargs =  'time_array': time_array  # time is given in days

    # Read file with signatures of interpolators and names of files with the
corresponding data
    signature_data = np.genfromtxt
        os.sep.join  file_directory, 'parameters_and_filenames_trunc.csv'  ,
        delimiter=",", dtype='str'

    # The first row (except the last element) of the file contains names of
the parameters
    par_names = signature_data 0, 1:-1

    num_pars = len par_names

    num_interpolators = signature_data.shape 0 -1  # -1 since the first line
is a header

    def last_results coords

        x,y,z = coords

        # Create system model
        sm = SystemModel model_kwargs=sm_model_kwargs

        # Create and add interpolator to the system model
        ind = run-1
        signature =  par_names j : float signature_data ind+1, j+1   for j in
range num_pars

        sm.add_interpolator ReservoirDataInterpolator
            name='int{}'.format ind+1 , parent=sm,
            header_file_dir=file_directory,
```

```
            time_file='time_points.csv',
            data_file='fg{}.csv'.format ind+1 ,
            index=int signature_data ind+1, 0  ,
            signature=signature , intr_family='reservoir'

        # Add reservoir component
        ltres = sm.add_component_model_object LookupTableReservoir
            name='ltres', parent=sm, intr_family='reservoir', locX=x,
locY=y

        # Add parameters of reservoir component model
        for j in range num_pars
            # add arbitrary line of values from signature_file
            ltres.add_par par_names j , value=float signature_data run,
j+1  , vary=False

        # Add observations of reservoir component model
        ltres.add_obs 'pressure'
        ltres.add_obs 'CO2saturation'
        ltres.add_obs_to_be_linked 'pressure'
        ltres.add_obs_to_be_linked 'CO2saturation'

        # Run system model using current values of its parameters
        sm.forward    # system model is run deterministically

        # Get results
        pressure = sm.collect_observations_as_time_series ltres, 'pressure'
        sat = sm.collect_observations_as_time_series ltres, 'CO2saturation'

        return  pressure -1 , sat -1

    from sys import platform
    if platform == "win32"
        # Loop replaces parallel execution of the simulations on Windows
        results =  last_results  x,y,z   for x,y,z in
zip df 'x' ,df 'y' ,df 'z'
    else
        # The following code should work on Mac but not on Windows.
        from multiprocessing import Pool
        with Pool processes=8  as pool
            results = pool.map last_results,   x,y,z  for x,y,z in
zip df 'x' ,df 'y' ,df 'z'
    results = np.array results

    def scatterplot y_true, y_pred, title
        plt.figure
        plt.plot y_true, y_pred, '.', markersize=4
        plt.xlabel('STOMP')
        plt.ylabel('NRAP-Open-IAM')
        plt.title title.title
        plt.savefig title+'.png'
        plt.close

    scatterplot df 'CO2saturation' , results :, 1 , 'saturation'
    scatterplot df 'pressure' , results :, 0 , 'pressure'
```

A.3 Test script for reservoir lookup table interpolation

```
'''
Test FutureGen2 Reservoir Lookup Table

Compares STOMP output to FutureGen2 reservoir lookup table
predictions.

This example requires the additional FutureGen 2.0 data set.
FutureGen 2.0 data set can be downloaded from the following
source:
https://edx.netl.doe.gov/dataset/futuregen-2-0-1008-simulation-
reservoir-lookup-table

The downloaded data set should be placed here:

source/components/reservoir/lookuptables/FutureGen2/1008_sims

Usage examples:
$ python test_iam_sys_lutreservoir.py --run 1
'''
# @author: Diana Bacon
# diana.bacon@pnnl.gov

import os
import sys
import argparse
import datetime
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
# from plot_to_openiam import Plot_File
# import pandas as pd
import glob

base_path =
os.path.join(os.sep,'Users','d3a926','Desktop','OpenIAM','UQ_exa
mple_setup','source')
sys.path.insert(0, base_path)
from openiam import (SystemModel, ReservoirDataInterpolator,
LookupTableReservoir)

if __name__ == "__main__":
```

```
    # For multiprocessing in Spyder
    __spec__ = None

    file_directory = os.sep.join([base_path, 'components',
'reservoir',
                                  'lookuptables', 'FutureGen2',
'1008_sims'])

    if not os.path.exists(os.sep.join([file_directory,
'fg1.csv'])):
        url = ''.join([
            'https://edx.netl.doe.gov/dataset/',
            'futuregen-2-0-1008-simulation-reservoir-lookup-
table \n'])
        msg = ''.join([
            '\nFutureGen 2.0 data set can be downloaded ',
            'from the following source:\n',
            url,
            'Check this example description for more
information.'])
        print(msg)

    # Input arguments
    parser = argparse.ArgumentParser(description='Test
FutureGen2 reservoir lookup table')
    parser.add_argument('--run', default='1', help='run number
to process')
    parser.add_argument('--file', default='.', help='plot file
with results of STOMP simulation')
    args = parser.parse_args()
    run = int(args.run)


    def get_results(x, y, time_array):

        # Define keyword arguments of the system model

        sm_model_kwargs = {'time_array': time_array} # time is
given in days

        # Read file with signatures of interpolators and names
of files with the corresponding data
        signature_data = np.genfromtxt(
            os.sep.join([file_directory,
'parameters_and_filenames_trunc.csv']),
            delimiter=",", dtype='str')
```

```
        # The first row (except the last element) of the file
contains names of the parameters
        par_names = signature_data[0, 1:-1]

        num_pars = len(par_names)

        num_interpolators = signature_data.shape[0]-1  # -1
since the first line is a header
        # Create system model
        sm = SystemModel(model_kwargs=sm_model_kwargs)

        # Create and add interpolator to the system model
        ind = run-1
        signature = {par_names[j]: float(signature_data[ind+1,
j+1]) for j in range(num_pars)}

        sm.add_interpolator(ReservoirDataInterpolator(
            name='int{}'.format(ind+1), parent=sm,
            header_file_dir=file_directory,
            time_file='time_points.csv',
            data_file='fg{}.csv'.format(ind+1),
            index=int(signature_data[ind+1, 0]),
            signature=signature), intr_family='reservoir')

        # Add reservoir component
        ltres =
sm.add_component_model_object(LookupTableReservoir(
            name='ltres', parent=sm, intr_family='reservoir',
locX=x, locY=y))

        # Add parameters of reservoir component model
        for j in range(num_pars):
            # add arbitrary line of values from signature_file
            ltres.add_par(par_names[j],
value=float(signature_data[run, j+1]), vary=False)

        # Add observations of reservoir component model
        ltres.add_obs('pressure')
        ltres.add_obs('CO2saturation')
        ltres.add_obs_to_be_linked('pressure')
        ltres.add_obs_to_be_linked('CO2saturation')

        # Run system model using current values of its
parameters
        sm.forward()  # system model is run deterministically
```

```
        # Get results
        pres = sm.collect_observations_as_time_series(ltres,
'pressure')
        sat = sm.collect_observations_as_time_series(ltres,
'CO2saturation')

        return pres, sat

    num_years = 20
    time1 = 365*np.arange(0.0, num_years+1)
    time2 = 365*np.arange(0.0, num_years+1, 0.5)

    pres_x1, sat_x1 = get_results(236681.772, 4409421.814,
time1)
    pres_x2, sat_x2 = get_results(236834.172, 4409421.814,
time1)
    pres_interp, sat_interp = get_results(236757.972,
4409421.814,time2)

    plt.figure()
    plt.plot(time1/365.25, sat_x1, '.', markersize=4)
    plt.plot(time2/365.25, sat_interp, '.', markersize=4)
    plt.plot(time1/365.25, sat_x2, '.', markersize=4)
    plt.xlabel('Time')
    plt.ylabel('Saturation')
    plt.savefig('saturation_vs_time.png')
    plt.close()

    # pres_x1, sat_x1 = get_results(236681.772, 4409421.814,
time1)
    # pres_x2, sat_x2 = get_results(236834.172, 4409421.814,
time1)
    # pres_interp, sat_interp = get_results(236757.972,
4409421.814,time2)

    plt.figure()
    plt.plot(time1/365.25, pres_x1, '.', markersize=4)
    plt.plot(time2/365.25, pres_interp, '.', markersize=4)
    plt.plot(time1/365.25, pres_x2, '.', markersize=4)
    plt.xlabel('Time')
    plt.ylabel('Pressure')
    plt.savefig('pressure_vs_time.png')
    plt.close()
```

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99354
1-888-375-PNNL (7665)

www.pnnl.gov