Pacific Northwest
NATIONAL LABORATORY

PNNL- 31543

NRAP-Open-IAM: Open Wellbore Component v2.0

July 2021

Diana H Bacon
Lehua Pan*
Curtis M Oldenburg*

*Lawrence Berkeley National Laboratory

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- 31543

NRAP-Open-IAM: Open Wellbore Component v2.0

July 2021

Diana H Bacon
Lehua Pan*
Curtis M Oldenburg*

*Lawrence Berkeley National Laboratory

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

Pacific Northwest National Laboratory
Richland, Washington 99354

Abstract

The Open Wellbore component of NRAP-Open-IAM is applicable to the calculation of Area of Review at a carbon storage site and fast estimation of dense gas dispersion from multiple continuous CO_2 surface leakage sources for risk assessment. The Open Wellbore component of NRAP-Open-IAM is a look-up table model, generated by performing multiple runs of the drift-flux simulator T2Well. The National Risk Assessment Partnership's second-generation integrated assessment model, NRAP-Open-IAM, is open-source software written in Python for use in systems-level performance and quantitative risk assessment of geologic carbon storage (GCS). The look-up table for open well leakage is an updated version of the previous model which provides finer resolution of reservoir depth and more accurate fluid properties than the previous open well look-up table, especially in calculating the gas (CO_2-rich) phase properties. The component model input parameters include the reservoir transmissivity, aquifer transmissivity, brine salinity, well radius, and well top and bottom depths. The reservoir transmissivity and brine salinity were set to values consistent with the reservoir component, and the well top and bottom depths varied with location. The possible outputs from the Open Wellbore component are leakage rates of CO_2 and brine to either an aquifer or the atmosphere, depending on the depth of the well top. Several python test scripts are presented that demonstrate that the Open Wellbore component reproduces the results of T2Well. Use of the open wellbore model in NRAP-Open-IAM may result in a large CO_2 leakage rates, comparable to the leakage rates of a CO_2 blow out.

Acknowledgments

This work was completed as part of the U.S. Department of Energy's (DOE's) National Risk Assessment Partnership (NRAP) project, supported by the DOE Office of Fossil Energy's Cross-Cutting Research Program. The work was performed by Pacific Northwest National Laboratory under DOE contract number DE-AC05-76RL01830. A portion of the modeling research was performed using PNNL's Research Computing cluster. Many thanks to Nicholas Huerta for a helpful review.

Acronyms and Abbreviations

AoR	Area of Review
CCUS	Carbon, Capture, Utilizations, and Storage
EDX	Energy Data Exchange
GCS	Geologic Carbon Sequestration
MATK	Model Analysis Tool Kit
NETL	National Energy Technology Laboratory
NRAP	National Risk Assessment Partnership
PISC	Post Injection Site Care
ROM	Reduced Order Model
SDWA	Safe Drinking Water Act
US	United States
US DOE	United States Department of Energy
USDW	Underground Sources of Drinking Water
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, 2018a).

Within the US, CO_2 injection activities are overseen by the US Environmental Protection Agency (USEPA) following regulations (the Class VI Rule) promulgated under the Safe Drinking Water Act (SDWA) (USEPA, 2010b). 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 (1) 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 commercial-scale CO_2 injection; (2) a lack of site-specific data on the geometry of the porous media and resulting transport of the injected CO_2 in the reservoir; and (3) limited understanding of the changes in the transport behavior of CO_2 caused by changes in pressure and/or temperature and the buoyant nature of CO_2 over the long time scale required for geologic sequestration to have long-term benefit to atmospheric CO_2 concentration. 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, DOE established the National Risk Assessment Partnership (NRAP), an initiative across five 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 geological carbon sequestration (GCS) (NETL 2021a), including the open-source integrated assessment model NRAP-Open-IAM (NETL 2021b). The NRAP-Open-IAM tool adopts a system-level stochastic approach that includes uncertainties in components (e.g., storage reservoirs, leakage scenarios, and shallow groundwater impacts). The tool 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 possible impacts to water quality caused by CO_2 and brine leakage from a storage reservoir into drinking water aquifers to make informed decisions on monitoring and mitigation to reduce risks.

1.1 Overview

The Open Wellbore Component model is a lookup table ROM based on the drift-flux approach as described by Pan et al. (2011). This model treats the leakage of CO_2 up an open wellbore or up an open (i.e., uncemented) casing/tubing. The lookup table is populated using T2Well (Pan and Oldenburg, 2014), which treats the non-isothermal flow of CO_2 and brine up an open wellbore, allows for the phase transition of CO_2 from supercritical fluid to gas, with Joule-Thompson cooling, and considers exsolution of CO_2 from the liquid (i.e., brine) phase.

This look-up table for open well leakage is an updated version of the previous open well leakage ROM. The updated look-up table provides finer resolution of reservoir depth and it is generated using the most recent version of T2Well with the property module ECO2N V2.0. The property module ECO2N V2.0 is more accurate than the module ECO2H used in development of the previous open well look-up table, especially in calculating the gas (CO_2-rich) phase properties (Pan et. al., 2017).

The component model input parameters include the reservoir transmissivity, aquifer transmissivity, brine salinity, well radius, and well top and bottom depths. The reservoir transmissivity and brine salinity were set to values consistent with the reservoir component, and the well top and bottom depths varied with location.

The possible outputs from the Open Wellbore component are leakage rates of CO_2 and brine to either an aquifer or atmosphere, depending on the depth of the well top.

Several tests are presented that demonstrate that the Open Wellbore reproduces the original T2Well simulations.

2.0 Conceptual Model and Simulations

The simulations used to develop the Open Wellbore lookup table include a wellbore fully perforated through a 20 m thick reservoir that is used for CO_2 storage. The radially symmetric reservoir is assumed to have initially uniform pressure, gas saturation, and brine salinity, and it is assumed these conditions are held constant at the far-field boundary (100 m away from the wellbore). The temperature distribution within the cap rock and reservoir is assumed to vary linearly with depth (25°C/km and 15°C at surface). Note that because of this simplified reservoir model (a single model layer), the effect of stratification of CO_2 in the reservoir due to buoyancy, as well as effects due to other heterogeneous features or geometric effects (e.g., different perforation lengths of the well), are ignored. The model considers variation in pressure from the far-field boundary to the well, but not possible overall far-field pressure-depletion effects that could arise from long-term leakage.

T2Well/ECO2N V2.0 is a numerical simulator that can simulate nonisothermal, multiphase, and multicomponent (H_2O, NaCl, CO_2) fluid and energy flow in the integrated wellbore-reservoir system. The code has been verified against analytical and numerical solutions and field CO_2 production testing data and has been applied to solve various problems involving coupled wellbore-reservoir flow processes (e.g., Pan and Oldenburg, 2014). The new code also includes the gravity potential energy in the energy balance equation. Simulations were terminated after 1,000 time steps or 100 hours, whichever came first, for each case. The maximum as well as the average leakage rates of gas (CO_2-rich) phase and brine (H_2O-rich) phase during the simulated leakage time were recorded as the output. The simulated leakage time was also recorded as the last output for reference information for the users.

A one-dimensional grid of 20 m long grid cells was used to represent the wellbore. The deepest grid cell of the well is connected to another one-dimensional (radial) grid that represents the reservoir; whose radial cell width varies from 0.1 m near well to 29 m at the boundary of the model. The reservoir thickness is 20 m with a permeability that matches the given reservoir transmissivity. Examples of T2Well grids can be found in Pan and Oldenburg (2014).

3.0 Model Input

There are many parameters (or factors) that could affect the leakage rates to various degrees. The parameters selected for variation have been observed to be the most important for estimating the leakage rates from an open wellbore (Pan and Oldenburg, 2014). The seven model input parameters are described in the following sections.

3.1 Wellbore Top Depth

A leakage-to-surface scenario was considered (wellhead is at 0 m, i.e., open to atmosphere), as well as two subsurface depths of the well top (100 m and 500 m below ground surface) for leakage to USDWs (i.e., aquifer impacts). Constant-pressure and temperature boundary conditions are used at the top of the model wellbore for all cases.

3.2 Reservoir Depth

GCS reservoirs are usually restricted to certain depth ranges for economic and storage efficiency reasons. In this model, the depth of the reservoir will determine the initial pressure (i.e., the hydrostatic pressure) and temperature in the reservoir, which will greatly affect the behavior of the injected CO_2 and the length of the hypothetical wellbore. Thirty-one depths are selected in this study, ranging from 1,000 to 4,000 m below land surface (every 100 m).

3.3 Reservoir Transmissivity

The CO_2 storage reservoir may have spatially varying permeability and thickness. These two parameters can be multiplied together to give a single parameter called the transmissivity of the reservoir. In contrast to the previous ROM in which reservoir transmissivity was a function of depth, the new look-up table uses transmissivity as an independent parameter spanning three orders of magnitude. The reservoir thickness in this model is fixed, so the variation in reservoir transmissivity is realized by varying permeability.

3.4 Aquifer Transmissivity

The overlying aquifer may have spatially variable permeability and thickness, which affect the overall resistance to leakage from the wellbore. These two parameters can be multiplied together to give a single parameter called the transmissivity of the aquifer. This parameter is not relevant for the surface leakage simulations where the depth of the wellbore top = 0 m.

3.5 Injection-Induced Pressure Perturbation

CO_2 injection causes reservoir pressure to increase from its initial pressure. This pressure perturbation is the main driving force for brine and CO_2 leakage through an open wellbore. This study assumes that the initial reservoir pressure is at hydrostatic pressure at the given depth with fixed geothermal gradient. Five levels of pressure perturbation from 0 to 20 MPa are simulated.

3.6 Reservoir Gas Phase Saturation

Supercritical, herein referred to as gas (CO_2-rich) phase saturation is one of the most important factors controlling the leakage rates. Gas saturation decreases from the vicinity of the injection well to the boundary of the plume. A leaking well could intersect quite different gas saturations, even in the same reservoir, depending on the leaky well's location. Five levels of gas-phase saturation from 0.01 to 0.99 were simulated.

3.7 Brine Salinity

Brine salinity could affect the density of leaking brine and the solubility of CO_2 in the aqueous phase. Three levels of brine salinity were simulated: 0, 0.1, and 0.2 salt mass fraction in the aqueous (liquid) phase.

4.0 Model Output

4.1 Average leakage rate of gas (CO_2-rich) phase to atmosphere

The average leakage rate of gas (CO_2-rich) phase to the atmosphere is provided when the input parameter Wellbore Top Depth is set equal to zero.

4.2 Average leakage rate of brine (H_2O-rich) phase to ground surface

The average leakage rate of brine (H_2O-rich) phase to the ground surface is provided when the input parameter Wellbore Top Depth is set equal to zero.

4.3 Average leakage rate of gas (CO_2-rich) phase to aquifer

The average leakage rate of gas (CO_2-rich) phase to an aquifer is provided when the input parameter Wellbore Top Depth is set to a value greater than zero.

4.4 Average leakage rate of brine (H_2O-rich) phase to aquifer

The average leakage rate of brine (H_2O-rich) phase to an aquifer is provided when the input parameter Wellbore Top Depth is set to a value greater than zero.

5.0 Lookup Table Format

The lookup table is an ASCII text file containing model input parameter and output variable values. The model input parameters described in Section 3.0 are listed in the first seven header rows of the lookup table (Table 1).

Table 1. Model input parameters in header rows of lookup table

Line Number	Parameter	Values
1	Wellbore Top Depth (m)	100, 500
2	Reservoir Depth (m)	1000.0, 1100.0, 1200.0, 1300.0, 1400.0, 1500.0, 1600.0, 1700.0, 1800.0, 1900.0, 2000.0, 2100.0, 2200.0, 2300.0, 2400.0, 2500.0, 2600.0, 2700.0, 2800.0, 2900.0, 3000.0, 3100.0, 3200.0, 3300.0, 3400.0, 3500.0, 3600.0, 3700.0, 3800.0, 3900.0, 4000.0
3	Reservoir Transmissivity (log m^3)	-11.27, -10.55, -9.83, -9.11, -8.39
4	Aquifer Transmissivity (log m^3)	-11.27, -10.55, -9.83, -9.11, -8.39
5	Pressure perturbation (Pa)	0.0, 34473.8, 68947.6, 100000.0, 376060.3, 1414213.6, 5318295.9, 20000000.0
6	Gas saturation (-)	0.01, 0.255, 0.5, 0.745, 0.99
7	Mass fraction of salt (-)	0.0, 0.1, 0.2

The model output variables described in Section 4.0 are listed in the remaining lines of the lookup table. Each line in the lookup table contains two variable values as listed in Table 1.

Table 2. Location of model output values in lookup table

Line Numbers	Variable 1	Variable 2	Depth of the Wellbore Top (m)	Sorted by Parameters (Table 1)
8 to 18,607	Average leakage rate of gas (CO_2-rich) phase to surface	Average leakage rate of brine (H_2O-rich) phase to surface	0	2, 3, 5, 6, 7
18,608 to 111,607	Average leakage rate of gas (CO_2-rich) phase to aquifer	Average leakage rate of brine (H_2O-rich) phase to aquifer	100	2, 3, 4, 5, 6, 7
111,608 to 204,607	Average leakage rate of gas (CO_2-rich) phase to aquifer	Average leakage rate of brine (H_2O-rich) phase to aquifer	500	2, 3, 4, 5, 6, 7

Lines 8 to 18,607 of the file contain 18,600 pairs of leakage rates. This first set of values are for leakage to the atmosphere or ground surface. This set of values is sorted in ascending order of the parameters from lines 2, 3, 5, 6 and 7 of the lookup table. The next two sets of output variables each contain 93,000 pairs of leakage rates to aquifers at depths of 100 and 500 m, respectively. Each set of values is sorted in ascending order of the parameters from lines 2, 3, 4, 5, 6 and 7 of the lookup table.

6.0 Open Wellbore Component

The Open Wellbore model used in NRAP-Open-IAM reads the lookup table described in Section 5.0 and interpolates or extrapolates where appropriate. For parameter values that fall within the minimum and maximum values in the table, multilinear table interpolation is used to estimate the flux values (Weiser et al, 1988).

For parameter values that fall outside the minimum and maximum values in the table, the closest parameter value is used. One exception to this is when the input gas saturation is less than the minimum value of 0.01. In this case the gas leakage rate is assumed to be zero when the gas saturation is zero, and linear interpolation is used. The other exception is for well top depths less than zero but not equal to the table depths of 100 or 500 m. In that case linear interpolation is used to estimate the leakage rates at other depths.

The Open Wellbore component calls the Open Wellbore model and provides the interface with other NRAP-Open-IAM components. The NRAP-Open-IAM tool is written in the Python 3 programming language, which provides the NRAP-Open-IAM cross-platform capabilities; an extensive number of libraries for data handling, analysis, and visualization; and the flexibility to include libraries and code written in other programming languages that may be used for development of imbedded component models (e.g., FORTRAN, C++, etc.). NRAP-Open-IAM has both serial and parallel (concurrent) execution capabilities. NRAP-Open-IAM is built upon the functionality within the Model Analysis ToolKit (MATK) Python package (Harp, 2021). MATK provides the NRAP-Open-IAM with core functionality to create and couple various component models into a system model, sample distributions for stochastic model parameters, run the system model to calculate geologic system performance over time, and analyze the results. The component structure of the NRAP-Open-IAM framework provides flexibility to represent site-specific GCS scenarios (Figure 1).

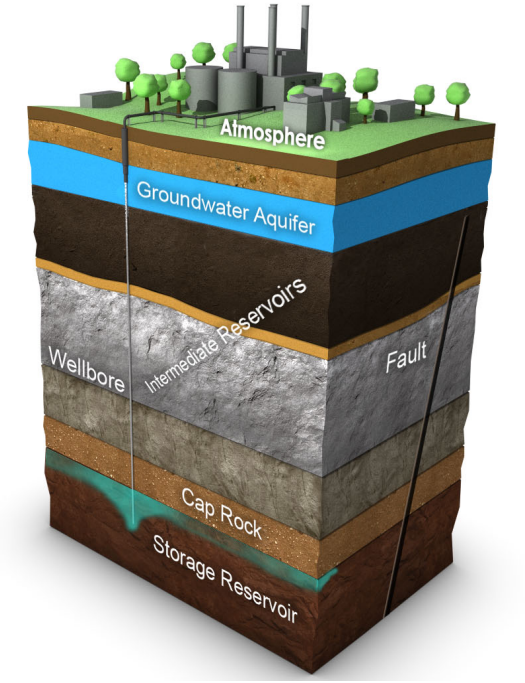

Figure 1. Components of a geologic carbon storage site (NETL 2021c).

6.1 Example applications

The Open Wellbore Component has been used in several peer-reviewed applications, from the calculation of Area of Review at a carbon storage sites (White et al. 2020; Bacon et al. 2020) to the fast estimation of dense gas dispersion from multiple continuous CO_2 surface leakage sources for risk assessment (Zhang et al. 2016). Example scripts related to these publications are distributed with NRAP-Open-IAM in the examples/scripts subfolder (NETL 2021b).

The example script *iam_sys_reservoir_openwell_aquifer_lhs.py* couples several components of NRAP-Open-IAM for uncertainty quantification using Latin Hypercube sampling. The flow of information through the system model components is as follows:

simple reservoir > open wellbore > rate-to-mass adapter > carbonate aquifer

CO_2 is injected into the reservoir at a constant rate for 50 years, and due to increasing pressure and CO_2 saturation in the reservoir, leakage from the open wellbore 100 m away begins immediately and leaks continuously. The resulting impacts on the carbonate aquifer in this extreme scenario are quite significant (Figure 2). Use of the open wellbore model in NRAP-Open-IAM may result in a large CO_2 leakage rates, comparable to the leakage rates listed as examples of CO_2 blow out by Aines et al. (2009), which ranges from 0.2 kg/s (Leroy Gas Storage Facility, WY) ~120 kg/s (Sheep Mt., CO).

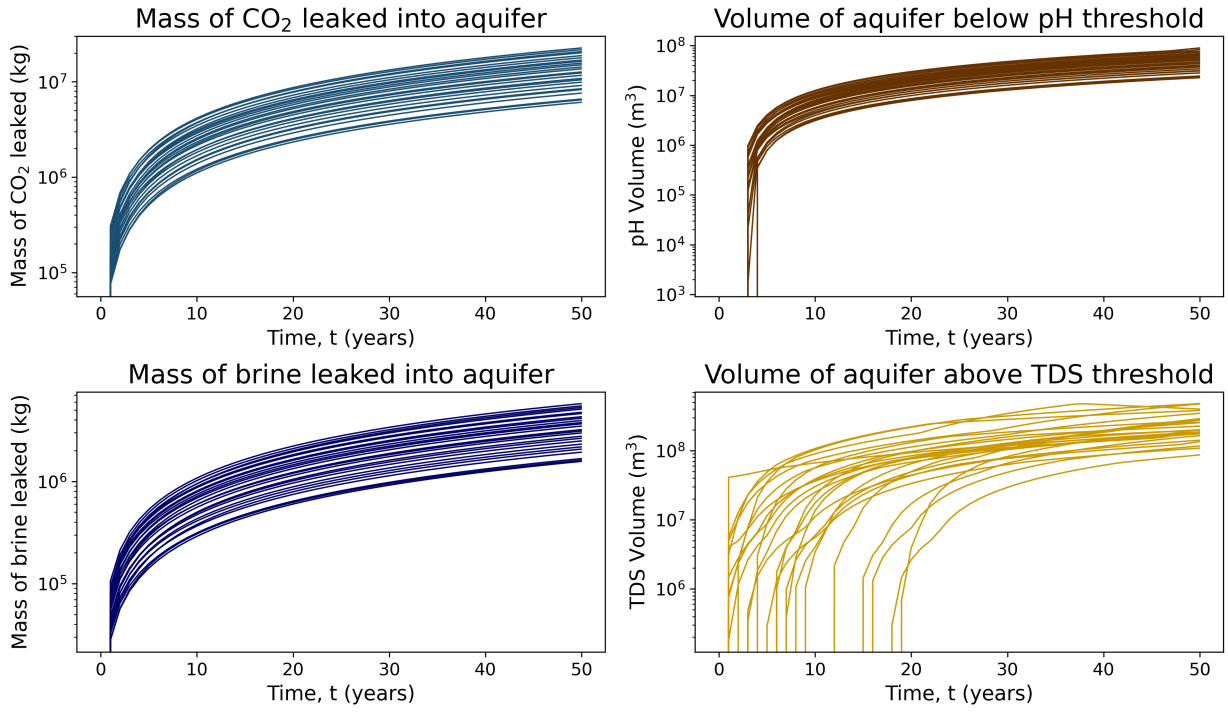

Figure 2. Aquifer impacts resulting from leakage from an open wellbore. The aquifer impact thresholds assumed were 6.9 for pH and 330 mg/L for TDS.

7.0 Testing

The following sections document three levels of testing: unit, integration, and system. The unit tests show that the Open Wellbore model produces expected results when called directly. The integration test shows that the Open Wellbore model produces expected results when invoked by the Open Wellbore component of NRAP-Open-IAM. The system tests show that the Open Wellbore model receives input correctly and produces expected results while running as part of a system model, connected to other components.

7.1 Unit Tests

These tests demonstrate that the Open Wellbore model produces expected results when called directly.

7.1.1 Reproduce original table values

This test confirms that the Open Wellbore Model produces the same brine and CO_2 leakage rates as contained in the original table of T2Well results. A python script (Appendix A.1) reads each row from the original files containing T2Well input parameters and output variables, uses those parameters to call *open_wellbore_ROM.py*, and plots the table values against the Open Wellbore Model output. These comparisons are shown for well top depths of 0, 100, and 500 m in Figure 3. The values match exactly and plot on at 1:1 line.

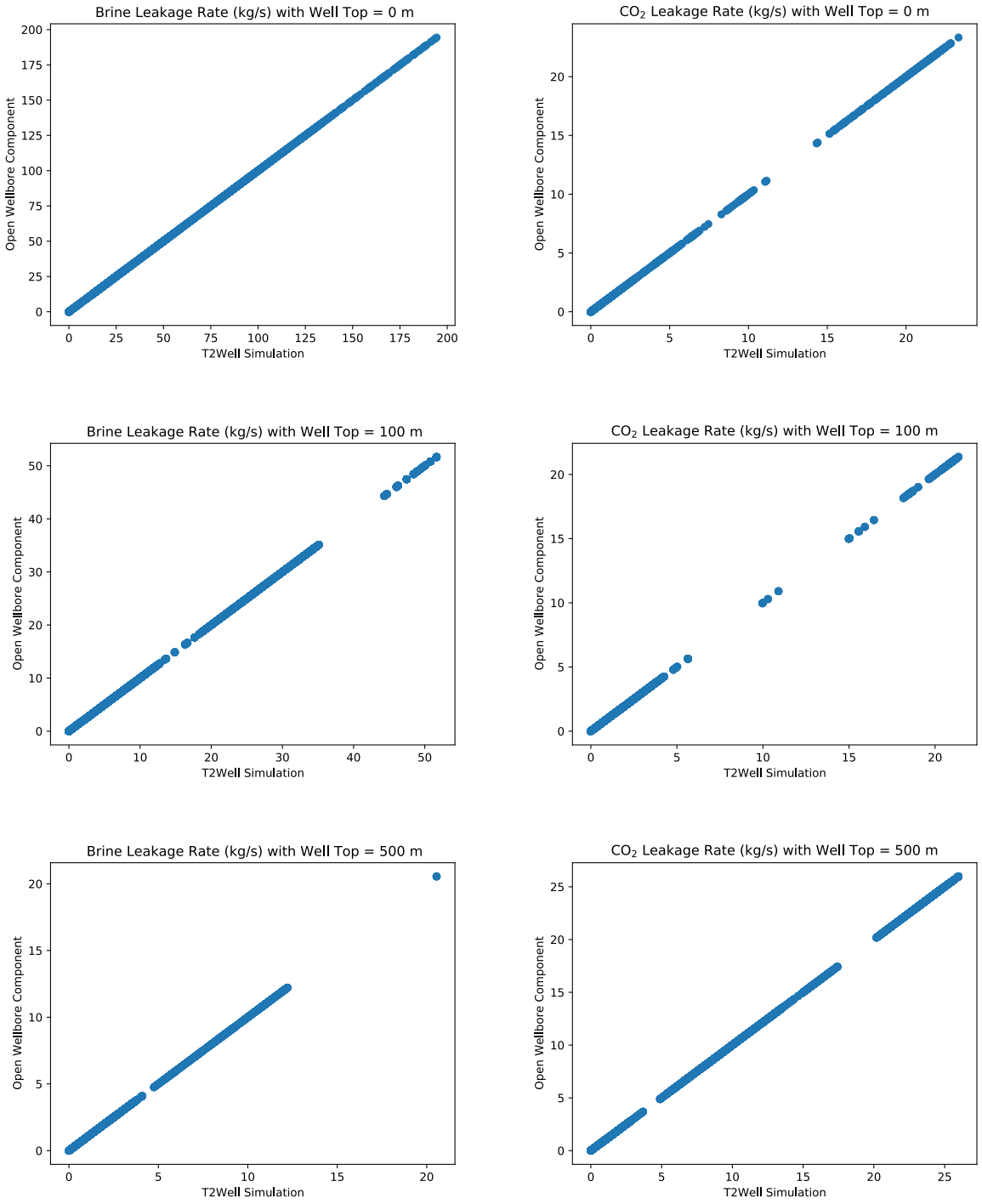

Figure 3. Comparison of brine and CO_2 leakage rates from the Open Wellbore Model and T2Well

7.1.2 Interpolate between table values

This test demonstrates that interpolation between values in the table of T2Well results works correctly. The python script to perform this test is listed in Appendix 0. Table 3 shows the input parameters for two points in the lookup table, which differ by the mass fraction of salt. Figure 4 shows the predicted values of brine and CO_2 leakage rate (shown in the center of the figures) interpolated between the two points in the data table (shown in the upper left and lower right corners of the figures). The interpolated value falls on a straight line between the two values from the table. The open wellbore model correctly interpolates between these two values.

Table 3. Two data points from the open wellbore lookup table

Wellbore Top Depth (m)	Reservoir Depth (m)	Reservoir Transmissivity (log m^3)	Aquifer Transmissivity (log m^3)	Pressure perturbation (Pa)	Gas saturation (-)	Mass fraction of salt (-)
100	1000	5.40E-12	2.82E-11	68947.6	0.745	0
100	1000	5.40E-12	2.82E-11	68947.6	0.745	0.1

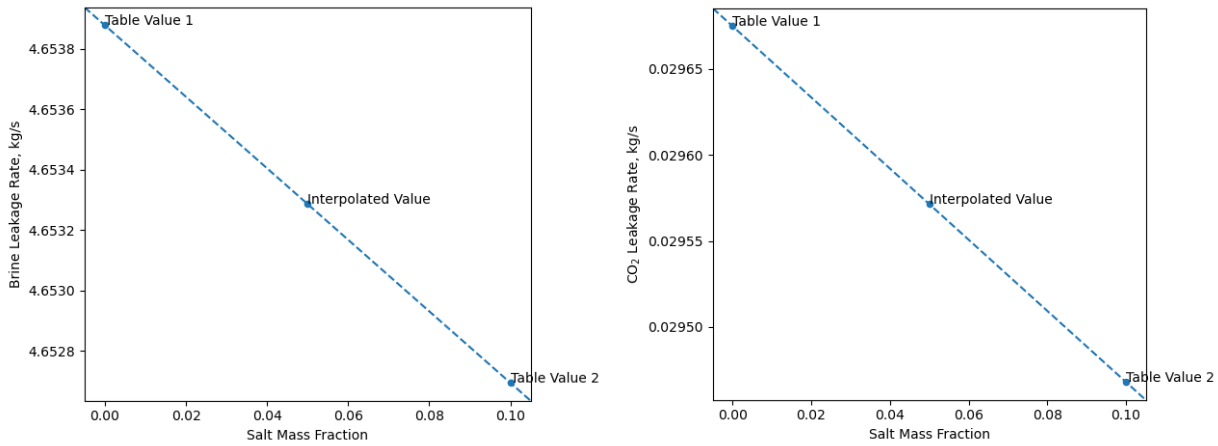

Figure 4. Interpolated brine and CO_2 leakage rates between two points in the open wellbore lookup table.

7.2 Integration Test

This test demonstrates that the Open Wellbore Model produces the same results either when called directly or when called by the Open Wellbore Component of NRAP-Open-IAM. Input parameters are randomly selected within the ranges listed in Table 4.

Table 4. Input parameter ranges for integration test

Parameter	Range
Wellbore Top Depth (m)	0 to 500
Reservoir Depth (m)	1000 to 4000

Reservoir Transmissivity (log m^3)	-11.27 to -8.40
Aquifer Transmissivity (log m^3)	-11.27 to -8.40
Pressure perturbation (Pa)	0.0 to 20000000.0
Gas saturation (-)	0.0 to 1.0
Mass fraction of salt (-)	0 to 0.2

The python script to perform this test is listed in Appendix A.3. The CO_2 and brine leakage rates calculated by the Open Wellbore model and the Open Wellbore component of NRAP-Open-IAM are shown in Figure 5. The values match exactly and plot on at 1:1 line.

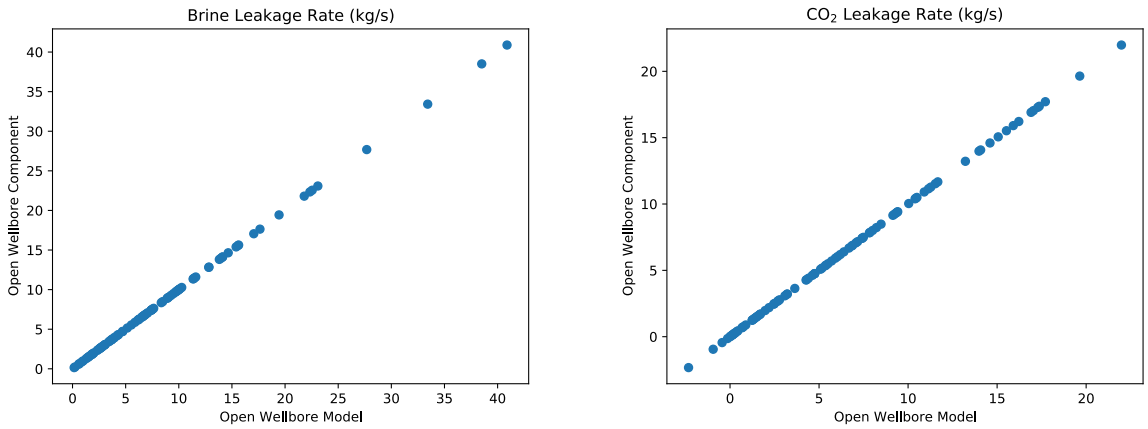

Figure 5. Comparison between leakage rates predicted by Open Wellbore model and Open Wellbore component of NRAP-Open-IAM

7.3 System Tests

These tests demonstrate that the Open Wellbore component of NRAP-Open-IAM operates as expected when part of a system model. NRAP-Open-IAM components have specific interface requirements, depending on what type of component they are. The Open Wellbore component is a type of wellbore component. A wellbore component must accept pressure and saturation as a fraction of pore volume vs. time in the reservoir as inputs and produce CO_2 and brine leakage rates vs. time as outputs.

The goal of the system tests is to demonstrate that the Open Wellbore component is in compliance with the interface requirements for an NRAP-Open-IAM wellbore component by demonstrating that it communicates correctly with a reservoir component and a rate-to-mass adapter component. The python script that runs all three of the following tests is listed in Appendix A.4.

7.3.1 Run as system model

The stratigraphic layering assumed for the system model is shown in Table 5, with three shale layers separating the reservoir and two aquifers.

Table 5. Stratigraphic layering for the system model

	Thickness (m)
Atmosphere	
Shale 3	11.2
Aquifer 2	400.0
Shale 2	1000.0
Aquifer 1	22.4
Shale 1	1000.0
Reservoir	51.2

CO_2 is injected into the reservoir, which has a radius of 10 km, at the default rate of 1.85E-02 kg/s for 2 years. All other input parameters for the Analytical Reservoir component assumed default values.

The open wellbore was assumed to have a radius of 0.0015 m and to be located 100 m away from the injection well location. The bottom of the open wellbore is in the reservoir at a depth of 2433.6 m and the top of the open wellbore is in aquifer 2 at a depth of 411.2 m. Both the reservoir and aquifer 2 are assumed to have transmissivities of 10^{-10} m^2. The reservoir brine was assumed to have a salt mass fraction equal to 0.1.

The pressure and saturation predicted by the analytical reservoir component are given in Table 6.

Table 6. Pressure and saturation in the reservoir at the bottom of the open wellbore

Time (year)	Pressure, (MPa)	Saturation (-)
0	25.29	0.00
1	27.92	0.54
2	27.60	0.84

The CO_2 and brine leakage rates calculated by the open wellbore component are given in Table 7.

Table 7. Leakage rates from the open wellbore

	Leakage Rates (kg/s)	
Time (year)	CO_2	Brine

0	0.00E+00	0.00E+00
1	6.34E-03	1.32E-03
2	6.15E-03	1.24E-03

The cumulative CO_2 and brine mass leaked, as calculated by the rate-to-mass adapter component in the system model, are given in Table 8.

Table 8. Cumulative mass leaked from the open wellbore

	Cumulative Mass Leaked (kg)	
Time (year)	CO_2	Brine
0	0.00E+00	0.00E+00
1	2.00E+05	4.17E+04
2	3.94E+05	8.08E+04

7.3.2 Receives input correctly

To verify that the open wellbore component receives input pressures and saturations correctly as part of a system model, the pressures and saturations listed in Table 6 were used as dynamic keyword arguments to run the open wellbore independently. As expected, the resulting CO_2 and brine leakage rates calculated by the open wellbore when run independently are identical to those listed in Table 7.

7.3.3 Passes output correctly

To verify that the open wellbore component outputs CO_2 and brine leakage rates correctly to a rate-to-mass adapter as part of an NRAP-Open-IAM system model, the CO_2 and brine leakage rates listed in Table 7 were used to independently calculate the cumulative mass leaked, according to the following formula

$$mass_t = mass_{t-1} + rate_t * 86400\ seconds/year$$

where
- t = time in years
- $rate_t$ = leakage rate in kg/s at the current time
- $mass_{t-1}$ = cumulative mass leaked at the previous time step
- $mass_t$ = cumulative mass leaked at the current time step

As expected, the calculated cumulative masses leaked of CO_2 and brine are identical to those listed in Table 8.

8.0 Conclusions

The Open Wellbore component of NRAP-Open-IAM has been applied to the calculation of Area of Review at a carbon storage site, and fast estimation of dense gas dispersion from multiple continuous CO_2 surface leakage sources for risk assessment. The tests documented in this report confirm that the updated Open Wellbore component reproduces the results of T2Well within given ranges of its input parameters and works correctly as a wellbore component of NRAP-Open-IAM.

However, the assumption of a completely open borehole that penetrates the storage reservoir and connects it to the shallow drinking water aquifer can lead to unrealistically high leakage rates (flux of brine and CO_2) and aquifer impacts (resulting from chemical constituent concentrations in the shallow drinking water aquifer) depending on pre-injection overpressure. Unrealistic because, if an open wellbore were actually present the reservoir would likely not remain overpressured (Oldenburg et al., 2016). In the NRAP-Open-IAM system model, there is no feedback from the wellbore component to the reservoir component which would relieve the pressure driving wellbore leakage.

9.0 References

Aines RD, MJ Leach, TH Weisgraber, MD Simpson, S Julio Friedmann, and CJ Bruton. 2009. "Quantifying the potential exposure hazard due to energetic releases of CO_2 from a failed sequestration well." *Energy Procedia* 1(1):2421-2429. 10.1016/j.egypro.2009.02.003

Bacon DH, DI Demirkanli, and SK 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. 10.1016/j.ijggc.2020.103153.

Harp, 2021. *Model Analysis ToolKit (MATK) - Python toolkit for model analysis*. Accessed June 15, 2021. http://dharp.github.io/matk/

NETL. 2015. *Carbon Storage Atlas*, 5th edition. Pittsburgh, PA.

NETL. 2018. *Carbon Storage Research*. Accessed May 7, 2021. https://netl.doe.gov/coal/carbon-storage/publications

NETL. 2021a. *National Risk Assessment Partnership*. Accessed May 26, 2021. https://edx.netl.doe.gov/nrap

NETL. 2021b. *NRAP-Open-IAM*. Accessed May 26, 2021. https://gitlab.com/NRAP/OpenIAM

NETL. 2021c. *NRAP Approach*. Accessed June 15, 2021. https://edx.netl.doe.gov/nrap/nrap-approach

Pan LH, SW Webb, and CM Oldenburg. 2011. "Analytical solution for two-phase flow in a wellbore using the drift-flux model." *Advances in Water Resources* 34(12):1656-1665. 10.1016/j.advwatres.2011.08.009.

Pan LH and CM Oldenburg. 2014. "T2Well-An integrated wellbore-reservoir simulator." *Computers & Geosciences* 65:46-55. 10.1016/j.cageo.2013.06.005.

Pan L, N Spycher, C Doughty, and K Pruess. 2017. "ECO2N V2.0: A TOUGH2 fluid property module for modeling CO2-H2O-NACL systems to elevated temperatures of up to 300 degrees C." *Greenhouse Gases-Science and Technology* 7(2):313-327. 10.1002/ghg.1617.

USEPA. 2010. "Federal Requirements Under the Underground Injection Control (UIC) Program for Carbon Dioxide (CO_2) Geologic Sequestration (GS) Wells."

Weiser A and SE Zarantonello. 1988. "A Note on Piecewise Linear and Multilinear Table Interpolation in Many Dimensions." *Mathematics of Computation* 50(181):189-196. DOI 10.2307/2007922.

Zhang Y, CM Oldenburg, and L Pan. 2016. "Fast estimation of dense gas dispersion from multiple continuous CO_2 surface leakage sources for risk assessment." *International Journal of Greenhouse Gas Control* 49:323-329. 10.1016/j.ijggc.2016.03.002.

White SK, S Carroll, S Chu, D Bacon, R Pawar, L Cumming, J Hawkins, M Kelley, I Demirkanli, RS Middleton, J Sminchak, and A Pasumarti. 2020. "A Risk-Based Approach to Evaluating the

Area of Review and Leakage Risks at CO_2 Storage Sites." *International Journal of Greenhouse Gas Control* 93:102884. 10.1016/j.ijggc.2019.102884.

Appendix A – Python Scripts

A.1 Reproduce table values

```
# confirm that model produces original values
import pandas as pd
import numpy as np
import sys, os
import matplotlib.pyplot as plt

# import the model
home_dir = 'YourHomeDirectoryHere'
source_dir = os.path.join(home_dir,'UQ_example_setup','source')
header_file_dir = os.path.join(source_dir,'components','wellbore','open')
sys.path.append(header_file_dir)
import open_wellbore_ROM as owrom
sol = owrom.Solution(header_file_dir)

def build_dataframe(data1, data2):
    df1 = pd.read_csv(data1, delim_whitespace=True)
    df2 = pd.read_csv(data2, delim_whitespace=True)
    df = pd.concat([df1,df2]).reset_index(drop=True)
    return df

def add_output(df, top):
    df['CO2LeakageRate'] = np.nan
    df['brineLeakageRate'] = np.nan
    for index, row in df.iterrows():
        # print(top, index)
        depth = row['Depth(m)']
        trans = row['PermXthickness(m^3)']
        if top == 0:
            aquif = 10**-11.26 # not used for surface leakage
        else:
            aquif = row['AquTransm(m^3)']
        dpres = row['DP(Pa)']
        sat = row['Sg']
        brine = row['Xsalt']
        fgas = row['Fgas(kg/s)']
        fliq = row['Fliq(kg/s)']
        input_array = np.array([top, depth, np.log10(trans), np.log10(aquif), dpres,
sat, brine])
        sol.find(input_array)
        if top == 0:
            row['CO2LeakageRate'] = sol.CO2LeakageRates[0]
            row['brineLeakageRate'] = sol.brineLeakageRates[0]
        else:
            row['CO2LeakageRate'] = sol.CO2LeakageRates[1]
            row['brineLeakageRate'] = sol.brineLeakageRates[1]
    return df

def plot_rate_comparison(df, top):
    fig, ax = plt.subplots()
    ax.scatter(df['Fgas(kg/s)'],df['CO2LeakageRate'])
    ax.set_ylabel('Open Wellbore Component')
    ax.set_xlabel('T2Well Simulation')
    ax.set_title('CO' + r'$\mathregular{_2}$' + ' Leakage Rate (kg/s)')
    plt.savefig('aqu' + str(top) + '_co2.pdf', dpi=600)
    plt.clf()
    fig, ax = plt.subplots()
    ax.scatter(df['Fliq(kg/s)'],df['brineLeakageRate'])
```

```
    ax.set_ylabel('Open Wellbore Component')
    ax.set_xlabel('T2Well Simulation')
    ax.set_title('Brine Leakage Rate (kg/s)')
    plt.savefig('aqu' + str(top) + '_brine.pdf', dpi=600)

# read original simulation results
# combine into dataframes for different well top depths: 0, 100 and 500 m

aqu0 = build_dataframe(
    '../table_data/NRAP_LOOKUP_lowDP_Sur.dat',
    '../table_data/NRAP_lookup_sur.dat')
aqu100 = build_dataframe(
    '../table_data/NRAP_lookup_lowDP_aqu100.dat',
    '../table_data/NRAP_LOOKUP_aqu100.dat')
aqu500 = build_dataframe(
    '../table_data/NRAP_lookup_lowDP_aqu500.dat',
    '../table_data/NRAP_lookup_aqu500.dat')

# run model and add results to each row
aqu0 = add_output(aqu0, 0)
aqu100 = add_output(aqu100, 100)
aqu500 = add_output(aqu500, 500)

# should plot 1:1 line
plot_rate_comparison(aqu0, 0)
plot_rate_comparison(aqu100, 100)
plot_rate_comparison(aqu500, 500)
```

A.2 Interpolation between table values

```
# confirm that model interpolates between table values
import pandas as pd
import numpy as np
import sys
import matplotlib.pyplot as plt

# import the model
home_dir = 'YourHomeDirectoryHere'
source_dir = os.path.join(home_dir,'UQ_example_setup','source')
header_file_dir = os.path.join(source_dir,'components','wellbore','open')
sys.path.append(header_file_dir)
import open_wellbore_ROM as owrom
sol = owrom.Solution(header_file_dir)

def build_dataframe(data1, data2):
    df1 = pd.read_csv(data1, delim_whitespace=True)
    df2 = pd.read_csv(data2, delim_whitespace=True)
    df = pd.concat([df1,df2]).reset_index(drop=True)
    return df

def add_output(row, top):
    depth = row['Depth(m)']
    trans = row['PermXthickness(m^3)']
    if top == 0:
        aquif = -11.26 # not used for surface leakage
    else:
        aquif = row['AquTransm(m^3)']
    dpres = row['DP(Pa)']
    sat = row['Sg']
    brine = row['Xsalt']
    fgas = row['Fgas(kg/s)']
    fliq = row['Fliq(kg/s)']
```

```
    input_array = np.array([top, depth, np.log10(trans), np.log10(aquif), dpres, sat,
brine])
    sol.find(input_array)
    if top == 0:
        row['CO2LeakageRate'] = sol.CO2LeakageRates[0]
        row['brineLeakageRate'] = sol.brineLeakageRates[0]
    else:
        row['CO2LeakageRate'] = sol.CO2LeakageRates[1]
        row['brineLeakageRate'] = sol.brineLeakageRates[1]
    return row

# read original simulation results
# combine into dataframes for different well top depths: 0, 100 and 500 m

aqu0 = build_dataframe(
    '../table_data/NRAP_LOOKUP_lowDP_Sur.dat',
    '../table_data/NRAP_lookup_sur.dat')
aqu100 = build_dataframe(
    '../table_data/NRAP_lookup_lowDP_aqu100.dat',
    '../table_data/NRAP_LOOKUP_aqu100.dat')
aqu500 = build_dataframe(
    '../table_data/NRAP_lookup_lowDP_aqu500.dat',
    '../table_data/NRAP_lookup_aqu500.dat')


row1 = aqu100.iloc[84]
row2 = aqu100.iloc[85]
row3 = (row1 + row2) / 2.
row1 = add_output(row1, 100)
row1['label'] = 'Table Value 1'
row2 = add_output(row2, 100)
row2['label'] = 'Table Value 2'
row3 = add_output(row3, 100)
row3['label'] = 'Interpolated Value'
df = pd.concat([row1,row2,row3], axis=1).transpose().reset_index(drop=True)
df.to_csv('data_points.csv')

ax1 = df.plot(x='Xsalt', y='CO2LeakageRate', kind='scatter')
df[['Xsalt','CO2LeakageRate','label']].apply(lambda row: ax1.text(*row),axis=1)
ax1.axline((row1['Xsalt'], row1['CO2LeakageRate']), (row2['Xsalt'],
row2['CO2LeakageRate']), linestyle="--")
plt.xlabel('Salt Mass Fraction')
plt.ylabel('CO$_2$ Leakage Rate, kg/s')
plt.tight_layout()
plt.savefig('test_interpolate_co2.pdf', dpi=600)

plt.clf()
ax2 = df.plot(x='Xsalt', y='brineLeakageRate', kind='scatter')
df[['Xsalt','brineLeakageRate','label']].apply(lambda row: ax2.text(*row),axis=1)
ax2.axline((row1['Xsalt'], row1['brineLeakageRate']), (row2['Xsalt'],
row2['brineLeakageRate']), linestyle="--")
plt.xlabel('Salt Mass Fraction')
plt.ylabel('Brine Leakage Rate, kg/s')
plt.tight_layout()
plt.savefig('test_interpolate_brine.pdf', dpi=600)
```

A.3 Compare component and model results

```
import sys
import os
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

home_dir = 'YourHomeDirectoryHere'
source_dir = os.path.join(home_dir,'UQ_example_setup','source')
sys.path.insert(0, source_dir)

def run_model(top, depth, trans, aquif, dpres, sat, brine):
    header_file_dir = os.path.join(source_dir,'components','wellbore','open')
    sys.path.append(header_file_dir)
    import open_wellbore_ROM as owrom
    sol = owrom.Solution(header_file_dir)
    input_array = np.array([top, depth, trans, aquif, dpres, sat, brine])
    sol.find(input_array)
    if top == 0:
        return sol.CO2LeakageRates[0], sol.brineLeakageRates[0]
    else:
        return sol.CO2LeakageRates[1], sol.brineLeakageRates[1]

def run_component(top, depth, trans, aquif, dpres, sat, brine):
    from openiam import (SystemModel, OpenWellbore)

    # Define keyword arguments of the system model
    time_array = 365.*np.arange(0, 2)
    sm_model_kwargs = {'time_array': time_array}  # time is given in days

    # Create system model
    sm = SystemModel(model_kwargs=sm_model_kwargs)

    # Add open wellbore component
    ow = sm.add_component_model_object(OpenWellbore(name='ow', parent=sm))

    # Add parameters of open wellbore component
    ow.add_par('wellRadius', value=0.05)
    ow.add_par('logReservoirTransmissivity', value=trans)
    ow.add_par('logAquiferTransmissivity', value=aquif)
    ow.add_par('brineSalinity', value=brine, vary=False)

    # Initial conditions
    g = 9.8
    brineDensity = 1000.0
    datumPressure = 101325.0
    initialPressure = datumPressure + (brineDensity*g*depth)
    initialSaturation = 0.

    # Add keyword arguments of the open wellbore component model
    dynamicPres = np.array([initialPressure, initialPressure+dpres])
    dynamicSat = np.array([initialSaturation, sat])
    ow.add_dynamic_kwarg('pressure', dynamicPres)
    ow.add_dynamic_kwarg('CO2saturation', dynamicSat)

    # Add composite parameters of open wellbore component
    ow.add_par('reservoirDepth', depth)
    ow.add_par('wellTop', top)

    # Add observations of open wellbore component model
    ow.add_obs('CO2_aquifer')
    ow.add_obs('brine_aquifer')
    ow.add_obs('CO2_atm')
```

```
    ow.add_obs('brine_atm')

    sm.forward()

    CO2leakrates_aq1 = sm.collect_observations_as_time_series(ow, 'CO2_aquifer')
    CO2leakrates_atm = sm.collect_observations_as_time_series(ow, 'CO2_atm')

    brine_leakrates_aq1 = sm.collect_observations_as_time_series(ow, 'brine_aquifer')
    brine_leakrates_atm = sm.collect_observations_as_time_series(ow, 'brine_atm')

    if top == 0:
        return CO2leakrates_atm[1], brine_leakrates_atm[1]
    else:
        return CO2leakrates_aq1[1], brine_leakrates_aq1[1]

def add_output(df):
    df['co2_rate_model'] = np.nan
    df['brine_rate_model'] = np.nan
    df['co2_rate_component'] = np.nan
    df['brine_rate_component'] = np.nan
    for index, row in df.iterrows():
        print('sample',index)
        top = row['top']
        depth = row['depth']
        trans = row['trans']
        aquif = row['aquif']
        dpres = row['dpres']
        sat = row['sat']
        brine = row['brine']

        row['co2_rate_model'], row['brine_rate_model'] = run_model(top, depth, trans,
aquif, dpres, sat, brine)
        row['co2_rate_component'], row['brine_rate_component'] = run_component(top,
depth, trans, aquif, dpres, sat, brine)
    return df

def plot_rate_comparison(df):
    fig, ax = plt.subplots()
    ax.scatter(df['co2_rate_model'],df['co2_rate_component'])
    ax.set_ylabel('Open Wellbore Component')
    ax.set_xlabel('Open Wellbore Model')
    ax.set_title('CO' + r'$\mathregular{_2}$' + ' Leakage Rate (kg/s)')
    plt.savefig('aqu_co2.pdf', dpi=600)
    plt.clf()
    fig, ax = plt.subplots()
    ax.scatter(df['brine_rate_model'],df['brine_rate_component'])
    ax.set_ylabel('Open Wellbore Component')
    ax.set_xlabel('Open Wellbore Model')
    ax.set_title('Brine Leakage Rate (kg/s)')
    plt.savefig('aqu_brine.pdf', dpi=600)

def generate_samples(num_samples):
    from matk import matk,pyDOE
    p = matk()
    p.add_par('top',min=0.0,max=500.0)
    p.add_par('depth',min=1000.0,max=4000.0)
    p.add_par('trans',min=-11.27,max=-8.40)
    p.add_par('aquif',min=-11.27,max=-8.40)
    p.add_par('dpres',min=0.0,max=20000000.0)
    p.add_par('sat', min=0.0, max=1.0)
    p.add_par('brine', min=0.0, max=0.2)
```

```
    # Create sampleset using pyDOE and parameter mins and maxs
    s = p.parmins + pyDOE.lhs(len(p.pars),samples=num_samples) * (p.parmaxs-p.parmins)
    df = pd.DataFrame(s, columns=p.parnames)
    return df

if __name__ == "__main__":

    df = generate_samples(100)
    add_output(df)
    plot_rate_comparison(df)
```

A.4 System Test

```
'''
This test verifies that the open wellbore component runs correctly
as part of a system model coupled with the analytical reservoir and
a rate-to-mass adapter. The results of the system model are compared
to running the open wellbore component independently and calculating
the cumulative mass directly.

Example of run:
$ python iam_sys_analytres_openwell_adapter.py
'''

import sys
import os

import numpy as np
import matplotlib.pyplot as plt

home_dir = 'YourHomeDirectoryHere'
source_dir = os.path.join(home_dir,'UQ_example_setup','source')
sys.path.insert(0, source_dir)


def run_system(time_array, rad, trans, aquif, brine):

    from openiam import (SystemModel, AnalyticalReservoir, OpenWellbore,
      RateToMassAdapter)

    # Define keyword arguments of the system model
    sm_model_kwargs = {'time_array': time_array}  # time is given in days

    # Create system model
    sm = SystemModel(model_kwargs=sm_model_kwargs)

    # Add reservoir component
    ares = sm.add_component_model_object(AnalyticalReservoir(name='ares', parent=sm))

    # Add parameters of reservoir component model
    ares.add_par('numberOfShaleLayers', value=3)
    ares.add_par('shale1Thickness', value=1000.0)
    ares.add_par('shale2Thickness', value=1000.0)
    ares.add_par('shale3Thickness', value=11.2)
    ares.add_par('aquifer1Thickness', value=22.4)
    ares.add_par('aquifer2Thickness', value=400)
    ares.add_par('reservoirThickness', value=51.2)
    ares.add_par('reservoirRadius', value=10000)

    # Add observations of reservoir component model
    ares.add_obs('pressure')
```

```
ares.add_obs('CO2saturation')
ares.add_obs_to_be_linked('pressure')
ares.add_obs_to_be_linked('CO2saturation')

# Add open wellbore component
ow = sm.add_component_model_object(OpenWellbore(name='ow', parent=sm))

# Add parameters of open wellbore component
ow.add_par('wellRadius', value=rad)
ow.add_par('logReservoirTransmissivity', value=trans)
ow.add_par('logAquiferTransmissivity', value=aquif)
ow.add_par('brineSalinity', value=brine)

# Add keyword arguments of the open wellbore component model
ow.add_kwarg_linked_to_obs('pressure', ares.linkobs['pressure'])
ow.add_kwarg_linked_to_obs('CO2saturation', ares.linkobs['CO2saturation'])

# Add composite parameters of open wellbore component
ow.add_composite_par('reservoirDepth',
    expr='+'.join(['ares.shale1Thickness',
          'ares.shale2Thickness',
          'ares.shale3Thickness',
          'ares.aquifer1Thickness',
          'ares.aquifer2Thickness']))
ow.add_composite_par(
    'wellTop', expr='ares.shale3Thickness + ares.aquifer2Thickness')

reservoirDepth = ares.pars['shale1Thickness'] + ares.pars['shale2Thickness'] +
    ares.pars['shale3Thickness'] + ares.pars['aquifer1Thickness'] +
    ares.pars['aquifer2Thickness']
wellTop = ares.pars['shale3Thickness'] + ares.pars['aquifer2Thickness']

# Add observations of open wellbore component model
ow.add_obs_to_be_linked('CO2_aquifer')
ow.add_obs_to_be_linked('brine_aquifer')
ow.add_obs_to_be_linked('brine_atm')
ow.add_obs_to_be_linked('CO2_atm')
ow.add_obs('CO2_aquifer')
ow.add_obs('brine_aquifer')
ow.add_obs('CO2_atm')  # zero since well top is in aquifer
ow.add_obs('brine_atm')  # zero since well top is in aquifer

# Add adapter that transforms leakage rates to accumulated mass
adapt = sm.add_component_model_object(RateToMassAdapter(name='adapt', parent=sm))
adapt.add_kwarg_linked_to_collection('CO2_aquifer',
    [ow.linkobs['CO2_aquifer'], ow.linkobs['CO2_atm']])
adapt.add_kwarg_linked_to_collection('brine_aquifer',
    [ow.linkobs['brine_aquifer'], ow.linkobs['brine_atm']])
adapt.add_obs_to_be_linked('mass_CO2_aquifer')
adapt.add_obs_to_be_linked('mass_brine_aquifer')
adapt.add_obs('mass_CO2_aquifer')
adapt.add_obs('mass_brine_aquifer')

CO2_rate_obs_list = []
brine_rate_obs_list = []
CO2_mass_obs_list = []
brine_mass_obs_list = []
CO2_rate_obs_list.append(ow.linkobs['CO2_aquifer'])
brine_rate_obs_list.append(ow.linkobs['brine_aquifer'])
CO2_mass_obs_list.append(adapt.linkobs['mass_CO2_aquifer'])
brine_mass_obs_list.append(adapt.linkobs['mass_brine_aquifer'])
```

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

    pressure = sm.collect_observations_as_time_series(ares, 'pressure')
    CO2saturation = sm.collect_observations_as_time_series(ares, 'CO2saturation')
    CO2_aquifer = sm.collect_observations_as_time_series(ow, 'CO2_aquifer')
    CO2_atm = sm.collect_observations_as_time_series(ow, 'CO2_atm')
    brine_aquifer = sm.collect_observations_as_time_series(ow, 'brine_aquifer')
    brine_atm = sm.collect_observations_as_time_series(ow, 'brine_atm')
    mass_CO2_aquifer = sm.collect_observations_as_time_series(adapt,
'mass_CO2_aquifer')
    mass_brine_aquifer = sm.collect_observations_as_time_series(adapt,
'mass_brine_aquifer')

    return wellTop, reservoirDepth, pressure, CO2saturation, CO2_aquifer,
brine_aquifer, mass_CO2_aquifer, mass_brine_aquifer

def run_open_wellbore(time_array, rad, top, depth, trans, aquif, pres, sat, brine):

    from openiam import (SystemModel, OpenWellbore)

    # Define keyword arguments of the system model
    sm_model_kwargs = {'time_array': time_array}  # time is given in days

    # Create system model
    sm = SystemModel(model_kwargs=sm_model_kwargs)

    # Add open wellbore component
    ow = sm.add_component_model_object(OpenWellbore(name='ow', parent=sm))

    # Add parameters of open wellbore component
    ow.add_par('wellRadius', value=rad)
    ow.add_par('logReservoirTransmissivity', value=trans)
    ow.add_par('logAquiferTransmissivity', value=aquif)
    ow.add_par('brineSalinity', value=brine)


    # Add keyword arguments of the open wellbore component model
    dpres = pres - pres[0]
    ow.add_dynamic_kwarg('pressure', dpres)
    ow.add_dynamic_kwarg('CO2saturation', sat)

    # Add composite parameters of open wellbore component
    ow.add_par('reservoirDepth', depth)
    ow.add_par('wellTop', top)

    # Add observations of open wellbore component model
    ow.add_obs('CO2_aquifer')
    ow.add_obs('brine_aquifer')
    ow.add_obs('CO2_atm')
    ow.add_obs('brine_atm')

    sm.forward()

    CO2leakrates_aq1 = sm.collect_observations_as_time_series(ow, 'CO2_aquifer')
    CO2leakrates_atm = sm.collect_observations_as_time_series(ow, 'CO2_atm')

    brine_leakrates_aq1 = sm.collect_observations_as_time_series(ow, 'brine_aquifer')
    brine_leakrates_atm = sm.collect_observations_as_time_series(ow, 'brine_atm')
```

```
    if top == 0:
        return CO2leakrates_atm, brine_leakrates_atm
    else:
        return CO2leakrates_aq1, brine_leakrates_aq1

def calculate_mass(time, co2_rate, brine_rate):
    co2_mass = np.zeros_like(time)
    brine_mass = np.zeros_like(time)
    for i in range(len(time)):
        if i == 0:
            continue
        else:
            co2_mass[i] = co2_mass[i-1] + (time[i] - time[i-1])*86400 * co2_rate[i]
            brine_mass[i] = brine_mass[i-1] + (time[i] - time[i-1])*86400 *
brine_rate[i]
    return co2_mass, brine_mass

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

    rad = 0.0015
    trans = -10.0
    aquif = -10.0
    brine = 0.1

    num_years = 2
    time_array = 365*np.arange(0.0, num_years+1)

    top, depth, pres, sat, co2_rate, brine_rate, co2_mass, brine_mass =
run_system(time_array, rad, trans, aquif, brine)
    print('pres', pres)
    print('sat', sat)
    print('top', top)
    print('depth', depth)
    co2_rate_2, brine_rate_2 = run_open_wellbore(time_array, rad, top, depth, trans,
aquif, pres, sat, brine)
    print('co2_rate', co2_rate, co2_rate_2)
    print('brine_rate', brine_rate, brine_rate_2)
    co2_mass_2, brine_mass_2 = calculate_mass(time_array, co2_rate, brine_rate)
    print('co2_mass', co2_mass, co2_mass_2)
    print('brine_mass', brine_mass, brine_mass_2)
```

PNNL- 31543

Pacific Northwest National Laboratory

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

www.pnnl.gov