ORNL/TM-2015/543

High Performance Computing Simulations for Shale Gas Formation: Flow Transport and Uncertainty Quantification Analysis

Charlotte Barbier
Dan Lu
Nathan Collier
Franklin Curtis
Clayton Webster
Yarom Polsky

Paolo Orsini

November 2015

DISTRIBUTION: Approved for public release; distribution is unlimited

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

DOCUMENT AVAILABILITY

Reports produced after January 1, 1996, are generally available free via US Department of Energy (DOE) SciTech Connect.

Websitehttp://www.osti.gov/scitech/

Reports produced before January 1, 1996, may be purchased by members of the public from the following source:

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
Telephone 703-605-6000 (1-800-553-6847)
TDD 703-487-4639
Fax 703-605-6900
E-mail info@ntis.gov
Websitehttp://www.ntis.gov/help/ordermethods.aspx

Reports are available to DOE employees, DOE contractors, Energy Technology Data Exchange representatives, and International Nuclear Information System representatives from the following source:

Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831
Telephone 865-576-8401
Fax 865-576-5728
E-mail reports@osti.gov
Websitehttp://www.osti.gov/contact.html

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-2015/543

Computational Sciences and Engineering Division

High Performance ComputingSimulations for Shale Gas Formation: Flow Transport and Uncertainty Quantification Analysis

Charlotte Barbier
Dan Lu
Nathan Collier
Franklin Curtis
Clayton Webster
Yarom Polsky

Paolo Orsini*

Date Published: September 2015

*Opengosim Ltd

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6283
managed by
UT-BATTELLE, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

Page

LIST OF FIGURES vii
LIST OF TABLES ix
EXECUTIVE SUMMARY xi
1. INTRODUCTION 1
2. TRANSPORT MODELING 2
2.1 Pflotran Introduction 2
2.2 Methane Thermodynamic properties 2
2.3 H_2O-CH_4-NACL dual miscibility and density 3
2.4 Modeling hydraulic fracturing 3
2.4.1 Approach 3
2.4.2 Mesh generation 4
2.4.3 Demonstration 4
2.5 Geomechanical modeling 5
2.6 Well boundary condition 8
2.6.1 Well modeling: IPR 9
2.6.2 Explicit well treatment 10
2.6.3 Well definition in the input file 11
2.6.4 PFLOTRAN code implementation 12
2.6.5 Well boundary condition demonstrations 13
3. UNCERTAINTY QUANTIFICATION 16
3.1 Uncertainty quantification problems 16
3.2 Sparse grid method and application 17
3.2.1 Sparse grid interpolation used in MCMC simulation 17
3.2.2 Application to an oil reservoir model 18
3.3 Multilevel Monte Carlo method and application 20
3.3.1 MLMC method in estimation of expectation 20
3.3.2 MLMC method in estimation of CDF 22
3.3.3 Application to an oil reservoir model 22
4. PARALLEL PERFORMANCE 26
4.1 TH “small” scaling and graphic visualization 26
4.2 CO_2/water test case 27
4.2.1 Problem setup 27
4.2.2 Sequential run results – computational grid 1 28
4.2.3 Parallel run results – computational grid 2 31
4.2.4 PFLOTRAN/TOUGH-ECO2N – detailed parallel performance analysis for computational grid 2 case 35
4.2.5 PFLOTRAN scaling test on a real reservoir problem – Sleipner CO_2 storage site 36
4.3 CH_4 examples 37
4.3.1 PFLOTRAN scaling test on a reservoir problem 37
4.3.2 TOUGH-EWASG and PFLOTRAN model comparison on Kissinger example 39
4.3.3 Results 42
5. PROPOSED FUTURE WORK 48
5.1 Moving from mphase to general phase in pflotran 48
5.2 Develop new code that leverage more petsc libraries 48
5.3 Expand geomechanics module 49
5.4 Discrete Fracture Network DFN 49

6. CONCLUSIONS 49
7. BIBLIOGRAPHY 49

LIST OF FIGURES

Page

Figure 1 Compression Factor of CH_4 3
Figure 2 Schematic of the different approaches to model flow in a fracture. 4
Figure 3 Illustration of the EPA problem 5
Figure 4 Contaminant concentration contours obtained with PFLOTRAN. 5
Figure 5 Injection flow rate versus time 7
Figure 6 Permeability evolution with time at 2.5m and 7.5m from the injection site. 8
Figure 7 Pressure and Temperature time evolutions at several locations away from the injection site. 8
Figure 8 Illustration of the well perforating several blocks 9
Figure 9 Well hydrostatic head correction illustration 10
Figure 10 Liquid pressure contours for a well injecting water with a BHP of 12.91 MPa at t=27 days with (right) and without (left) hydraulic pressure correction. 13
Figure 11 Volumetric flow rate at the well with and without hydraulic correction. 14
Figure 12 Pre-run to generate the initial condition for gas production at t=0 (left) and t=2000 days (right) 14
Figure 13 Gas saturation at the beginning of production (left), and after 1000 days (right). 15
Figure 14 Gas production rate knees 15
Figure 15 Gas saturation (left) and liquid pressure (right) at t=825 days 15
Figure 16 Gas saturation (left) and liquid pressure (right) at t=830 days 16
Figure 17 Flowchart of the MCMC run. The difference between the C-MCMC and the SG-MCMC simulations is that C-MCMC executes the actual physical model and SG-MCMC executes the SG-based surrogate model in each sample generation. 17
Figure 18 A seven-level adaptive sparse grid for interpolating a 1-D function $f(\vartheta) = 50000\,\vartheta^4(1-\vartheta)^{20}$ on bound [0, 1] with the error tolerance of 0.04. The resulting adaptive sparse grid has only 21 points as shown in black dots, whereas the full grid needs 65 points as shown in both black and gray dots. 18
Figure 19 Pair wise comparison of the 28 simulated temperature and 22 tracer con- centration of the original physical and surrogate model for 1000 parameter combinations randomly drawn from the prior distribution. 19
Figure 20 Marginal PPDF of the four parameters estimated using C-MCMC with 80,000 model executions and the SG-MCMC with 295 model executions, respectively. The black vertical lines represent true parameter values. Take the C-MCMC results as reference, the estimations by the SG-MCMC are accurate enough but with significantly computational cost reduction. 20
Figure 21 (a) True log permeability field log(k); I represents the injection well and P is the production well. (b) Locations of 36 sample data drawn from (a); conditioning on these samples, the realizations of random log(k) field are generated. (c) An example of one realization of random log(k) field. 23
Figure 22 Simulation meshes on different levels used in MLMC. The highest level 5 has the same mesh as the synthetic model. 23
Figure 23 Computational time of the standard MC and MLMC needed to achieve the desired accuracy. 24
Figure 24 (a) Time for one sample run (red solid line corresponds to left y-axis) and the required number of samples (blue dashed line corresponds to right y-axis) at the six levels of MLMC; (b) MLMC estimates of E[Q] considering different numbers of levels, and the standard MC estimate of E[Q]. 24
Figure 25 The reference CDF and the estimated CDF based on the standard MC and the MLMC methods. 25
Figure 26 (a) The variance decay of the indicator function, the smoothing function calculated based on the δ determined in this study (δ) and that calculated in Giles et al. (2015) (δ_G); (b) computational time of the standard MC and the MLMC with δ and δ_G. 26
Figure 27 Parallel efficiency of a strong scaling study of a simple TH problem 27
Figure 28 Isotropic case, 3D sg controur: Left, t=10 years; right, t=50 years 28
Figure 29 Anisotropic case, 3D sg controur: Left, t=10 years; right, t=50 years 31
Figure 30 Anisotropic case, t=10 years, 3D sg contour: Left, grid 1 (50 k cells), right grid 2 (1M cells). 32
Figure 31 PFLOTRAN parallel performance for the Sleipner CO_2 storage case 37

Figure 32 CH_4 gas plume for t=10 years and a coarse grid. The PFLOTRAN results are on the left and the TOUGH2-MP results are on the right. 38
Figure 33 Parallel performance for the CH_4 injection for 2.25 years. 39
Figure 34 Contour plots for the static pressure and brine density obtained from the TOUGH2-EWASG equilibrium run. 42
Figure 35 Gas saturation at the end of leakage – t=7.76 years: left TOUGH2-EWASG, right PFLOTRAN 43
Figure 36 CH_4 liquid molar fraction at the end of leakage – t=7.76 years: left TOUGH2-EWASG, right PFLOTRAN 43
Figure 37 Gas saturation at z=-1135 (first high permeability layers from the bottom) at the end of leakage – t=7.76 years: left TOUGH2-EWASG, right PFLOTRAN 44
Figure 38 CH_4 liquid molar fraction at z=-1135 (first high permeability layers from the bottom) at the end of leakage – t=7.76 years: left TOUGH2-EWASG, right PFLOTRAN 44
Figure 39 Gas saturation at the end of the simulation – t=100 years: left TOUGH2-EWASG, right PFLOTRAN 45
Figure 40 CH_4 liquid molar fraction at the end of the simulation – t=100 years: left TOUGH2-EWASG, right PFLOTRAN 45
Figure 41 Pressure contours for equilibration run with an initial salt mass fraction equal to 0.1. Left TOUGH2-EWASG, right PFLOTRAN 46
Figure 42 Liquid phase density contour for equilibration run with an initial salt mass fraction equal to 0.1. Left TOUGH2-EWASG, right PFLOTRAN 46
Figure 43 Gas saturation at the end of the simulation, XS=0.1 – t=100 years: left TOUGH2-EWASG, right PFLOTRAN. 47
Figure 44 CH_4 liquid molar fraction at the end of the simulation, XS=0.1 – t=100 years: left TOUGH2-EWASG, right PFLOTRAN. 47
Figure 45 Results of the convergence study proposed by Tracy against a variably saturated analytic solution. On the left panel, a picture of the steady solution on the well-shaped but not orthogonal triangular mesh corresponding to the tracy_tri.e runs in the right panel. 48

LIST OF TABLES

Page

Table 1True parameter values and their ranges of the four parameters 18
Table 2 Execution time TOUGH-ECO2N / PFLOTRAN-MPHASE comparison time in minutes 29
Table 3 Execution time comparison PFLOTRAN-MPHASE in sequential and using 4 cores, time in minutes. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, np number of mpi-core processes.....29
Table 4 Number of time steps comparison. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, np number of mpi-core processes.....30
Table 5 Number of iteration comparison. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, np number of mpi-core processes......30
Table 6 Simulation up to 1 year - execution time in minutes. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 32
Table 7 Simulation up to 1 year – number of time steps. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 33
Table 8 Simulation up to 1 year – number of Newton iterations. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 33
Table 9 Simulation up to 1 year – number of linear iterations. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 33
Table 10 Simulation up to 1 year – number of time step cuts. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 33
Table 11 Simulation up to 10 years - execution time in minutes. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 33
Table 12 Simulation up to 10 years – number of time steps. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 34
Table 13 Simulation up to 10 years – number of Newton iterations. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 34
Table 14 Simulation up to 10 years – number of linear iterations. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 34
Table 15 Simulation up to 10 years – number of time step cuts. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 34
Table 16 Simulation up to 1 year - execution time in minutes. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 35
Table 17 Simulation up to 1 year – number of time steps. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 35
Table 18 Simulation up to 1 year – number of Newton iterations. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 35
Table 19 Simulation up to 1 year – number of linear iterations. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 35
Table 20Simulation up to 1 year – number of time step cuts. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes. 36
Table 21PFLOTRAN performance on the Sleipner CO_2 storage site case 36
Table 22 Simulation time in seconds of the finer grid for a CH_4 injection of 2.25 years. 38

EXECUTIVE SUMMARY

Fracking is the controversial technique used to economically release natural gas and oil from shale formations. The process has been used for more than 60 years, but recently has employed horizontal drilling, which facilitates the extraction of significantly more natural gas and oil from each well than previously. Fracking is not only critical for energy production in the US, but for jobs and economic growth as well. However, this new practice carries possible environmental and public health risks, most notably contamination of aquifers. In this project, we developed a powerful simulation tool that would help maximize the benefits of fracking while curtailing possible negative impacts on ecosystems and surrounding communities. We added new capabilities within PFLOTRAN, a massively parallel subsurface reservoir modeling code, to enable the simulation of fracking processes on leadership-class systems like Titan. Due to the presence of inherent uncertainties involved in describing the subsurface structure and related physical parameters, new uncertainty quantification (UQ) methods were developed through ORNL Toolkit for Adaptive Stochastic Modeling and Non-Intrusive ApproximatioN (TASMANIAN) and are presented here.

In this report, details on the new capabilities implemented in PFLOTRAN and the UQ methods developed are presented. In addition, we demonstrate how PFLOTRAN and our UQ methods are much faster than the current state-of-the-art.

1. INTRODUCTION

Fracking (or hydraulic fracturing) technology enables the extraction of fluid (petroleum, natural gas, or water) from rock formations deep below the earth's surface (1.5 to 6.1 km deep). At such depth, the rock permeability or the reservoir pressure may be too low to allow the fluid to flow from the rock into the wellbore at economic rates. To overcome these engineering obstacles, a fracturing fluid is pumped into the wellbore at high pressure, causing the rock tensile strength to be exceeded and hydraulic fractures to be formed. Eventually the fluid is withdrawn, but proppant materials in the fracking fluid keep the fractures open, allowing oil or gas to be extracted more easily and economically.

Amid the U.S. focus on energy independence, fracking has become a key technique for releasing previously inaccessible hydrocarbons. Up to 95% of wells drilled today in the U.S. are hydraulically fractured, accounting for 43% and 67% of total U.S. oil and natural gas production, respectively. Although hydraulic fracturing was first used in 1947, it was only in 1998 that modern fracturing technology, referred to as horizontal slickwater fracturing, made possible the economical extraction of shale gas. However, fracking can have serious environmental impacts such as groundwater contamination, air quality deterioration, migration of gases and hydraulic fracturing chemicals to the surface, and surface contamination from spills and flowback. More than 750 distinct chemicals, ranging from benign to toxic, have been used in fracking solutions. Although these additives are generally less than 2% by volume of the total fracturing fluid, fracking is a water-intensive process, using between 2.3 million and 3.8 million gallons of water and at least 12,000 gallons of chemicals per well [1]. Even if benign fracking fluid is used, the recovered fluids may be contaminated once they come in contact with the shale, carrying an excess of sodium, chloride, bromide, arsenic, barium, or naturally occurring radioactive materials such as uranium, radium, and radon. Despite the possible water contamination, fracking is currently exempt from regulation under the Safe Drinking Water Act due to an exemption written in the Energy Policy act of 2005 ("Halliburton loophole"). For these reasons, hydraulic fracturing is the topic of much debate and has even been suspended or banned in some countries. Despite these concerns, experts believe that the exploration, extraction, and transportation of natural gas can be performed in a sustainable and environmentally responsible manner through the application of advanced technologies. Simulation will play a key role not only in understanding and limiting the potential negative impacts of fracking on the environment, but also in optimizing the technique for economic growth.

Due to modeling complexities and the presence of uncertainties in the modeling parameters (fracture size and distribution, porosity, etc.), the reliability of hydraulic fracture simulations can be debated. Even though the state-of-the-art in hydraulic fracture modeling allows the gas and oil industry to routinely design and evaluate stimulation treatments, there is no doubt that simulation has not reached a fully predictive capability that would enable large scale optimization strategies or serious risk assessment of possible water contamination. Despite a few analytical models as well as a small number of published numerical studies, there is little consensus regarding the large-scale flow behavior over time in tight/shale gas systems featuring a horizontal well with multiple hydraulic fractures, particularly regarding the dominant flow regimes and whether or not reservoir properties or flow characteristics can be estimated reliably from well performance data.

Parallelized variants of some state-of-the-art modeling tools such as TOUGH [2] exist, but cannot fully utilize today's High Performance Computing(HPC) capability, resulting in very long execution times that can range from several days to weeks for one scenario [1]. The lengthy execution time makes some state-of-the-art UQ investigation such as Monte Carlo (MC) and Markov chain Monte Carlo (MCMC) unaffordable or even prohibitive. No fracking simulation codes have yet been coupled with a UQ toolset that can provide a full suite of parameter sampling algorithms, global sensitivity analysis algorithms, predictive analysis tools, and constrained optimization tools to quantify uncertainty and guide uncertainty

reduction. Moreover, no efforts have been devoted to tackling the important inverse problem, where field data measurements and simulation output are used to optimally compute input parameters associated with the specification of the reservoir, such as porosity, permeability, fracture spacing, rock compressibility, etc.

To address these deficiencies, we developed new capabilities in PFLOTRAN [3] for fracking applications. PFLOTRAN can simulate coupled, thermal-hydrologic-chemical (T-H-C) processes in variably saturated, non-isothermal, porous media in three spatial dimensions. The code already contained a number of physics models relevant for fracking applications and had demonstrated petascale performance [4], performing simulations on large domains (km scale) with very high resolution. Additionally, PFLOTRAN has the ability to simultaneously launch multiple simulations of different problem realizations, which make it very suitable to quantify uncertainties associated with the subsurface structure, model input parameters and model output quantities.

2. TRANSPORT MODELING

2.1 PFLOTRAN INTRODUCTION

PFLOTRAN [5] is an open source, massively parallel subsurface flow and reactive transport code. It can solve multiphase, multicomponent and multiscale reactive flow and transport in porous materials. PFLOTRAN uses PETSc [6] (Portable Extensible Toolkit for Scientific Computation) libraries, a suite of data structures and routines allowing scalable solutions. At this time, PFLOTRAN can solve: Richard's equation, Thermo-Hydro-Chemical, Multiphase Water-Supercritical CO_2, Surface Flow, Aqueous Complexation, Sorption, Mineral Precipitation and Dissolution, Multiple Continuum for Heat, Subsurface Flow-Reactive Transport Coupling, Multiphase Ice-Water-Vapor Flow, Structured and Unstructured Grids, Multiple Realizations, multiple inputs, Parallel I/O.

In order to simulate the flow in a shale gas formation, the multiphase mode "MPHASE" of PFLOTRAN which solves the subsurface multiphase flow of Water-CO_2-brine was adapted. More specifically, an equation of state for CH_4 and models to compute the dual-miscibility and density for H_2O-CH_4-NaCl systems were implemented. The details of the implementations are presented in the following sections.

2.2 METHANE THERMODYNAMIC PROPERTIES

The thermodynamic properties of CH_4 are computed using the ideal gas equation of state. Over a wide range of temperature and pressure, CH_4 behavior is very close to an ideal gas. This is illustrated in Figure 1 in which the compression factor remains close to 1 for a wide range of temperature and pressure. The viscosity of CH_4 is computed using the Irvine & Liley equation [7] . The possibility of using a table was also implemented for getting more accurate thermodynamic properties if needed.

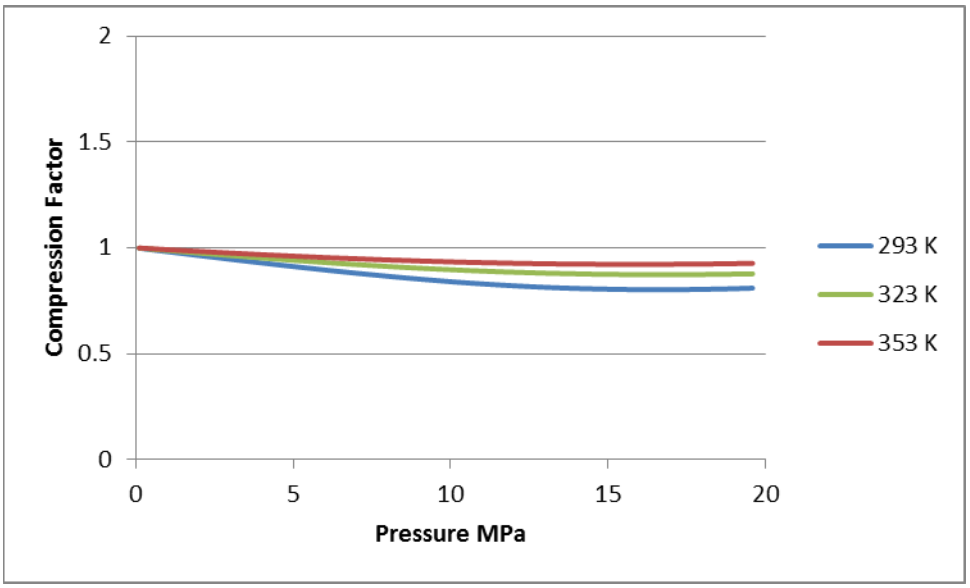

Figure 1 Compression Factor of CH_4

2.3 H_2O-CH_4-NACL DUAL MISCIBILITY AND DENSITY

When both aqueous and gas phases are present in a cell, the amount of dissolved CH_4 in brine and the amount of evaporated water needs to be evaluated. This is done by assuming thermodynamic equilibrium between the two phases. The concentration of CH_4 in brine is computed using a model by Duan and Mao [8]

$$ln\frac{y_{CH4}P}{m_{CH4}} = \frac{\mu_{CH4}^{l(0)}}{RT} - ln\varphi_{CH4} + ln\gamma_{CH4}$$

where P is the total pressure, y_{CH4} the mole fraction of CH_4 in the gas phase, $m_{CH4}$the molality (mol/kg) of CH_4 in the liquid phase, $\mu_{CH4}^{l(0)}$ the chemical potential in hypothetically ideal solution of unity molality, R the ideal gas constant, T the temperature, φ_{CH4} the CH_4 fugacity, and $\gamma_{CH4}$the CH_4 activity coefficient. This was implemented in ch4eos.F90 with the subroutine Henry_duan_mao.

The CH_4 enriched brine density is computed using [8]. This is implemented in eos_water.F90 in the subroutine EOSWaterDuanMixture, which reads pressure, temperature, NaCl molar fraction, liquid phase formula weight, and brine density and then computes the density of the liquid mixture.

2.4 MODELING HYDRAULIC FRACTURING

2.4.1 Approach

Two main methods are used in the literature to simulate the fractures. A discrete approach where each fracture is meshed; although it is the most accurate method, it is very mesh intensive, unsuited for operational tasks such as evaluating uncertainties, and requires small time steps to get good convergence. Another approach is a continuum approach where a single equation is used for both fracture and matrix flow. The mesh is then much simpler (structured mesh most of the time) and it is very suitable for the inverse problem. However, with this approach, poorly connected fracture networks or large faults cannot be correctly simulated and it focuses on macroscopic behavior. Our approach is a combination of these methods where the fracture network is mapped onto a regular mesh and equivalent properties are mapped onto the mesh. The main advantage of this method is that the dual-continuum method implemented in PFLOTRAN can still be used.

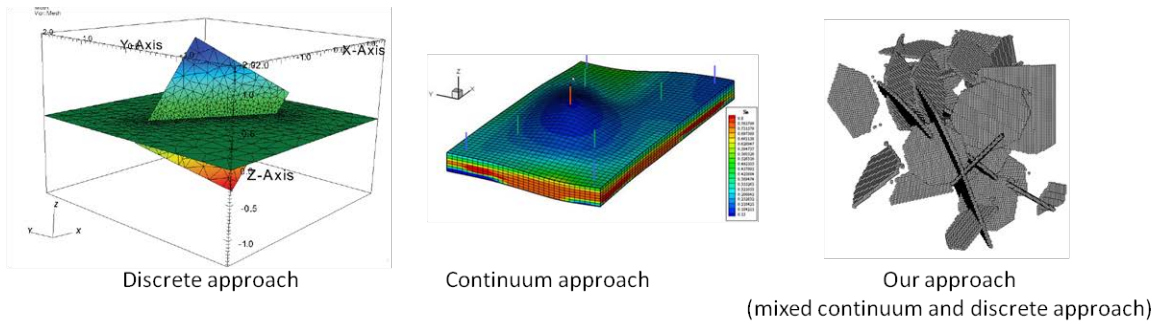

Figure 2 Schematic of the different approaches to model flow in a fracture.

Since the fractures are discretized much larger than reality, an equivalent permeability is used to keep the flow between the matrix and the fracture realistic. To do this, we keep the fracture conductivity C_f constant:

$$C_f = \frac{k_f w}{k_m x_f}$$

Where $w$is the fracture width, k_m, k_f the matrix and fracture permeability, and x_f the half length of the fracture.

2.4.2 Mesh generation

While PFLOTRAN supports both structured and unstructured grid types, we found that unstructured meshes heavily graded to better capture small crack features caused performance problems in the solver. For this reason we focused on developing meshes which were uniformly fine in all dimensions. However, this greatly multiplied the number of cells, many of which fall into the overburden region which forms a nearly impermeable boundary. In these regions, we simply removed cells as can be seen in Figure 4. We found that this balanced the cost of simulation and robustness of the solver. So while the mesh itself originates from a structured grid (a tensor product of 1D grids in each dimension), the removal of cells requires the mesh connectivity to be coded as if the mesh were unstructured.

2.4.3 Demonstration

The method was demonstrated on an example extracted from the 2012 EPA report on hydraulic fracturing and is illustrated in Figure 3. The stimulation of the fracture is modeled by injecting contaminated water at high pressure in the horizontal well for 30 minutes and checking if the contaminant can reach the aquifer. The simulations suggested that contamination may happen years after the injection but that if the pathway is large enough, the contamination is negligible. However, in our simulations, several physics such as flow back, fluid compressibility effects and continuum damage are not taken into account and thus, no definitive conclusions can be done on the possible contamination.

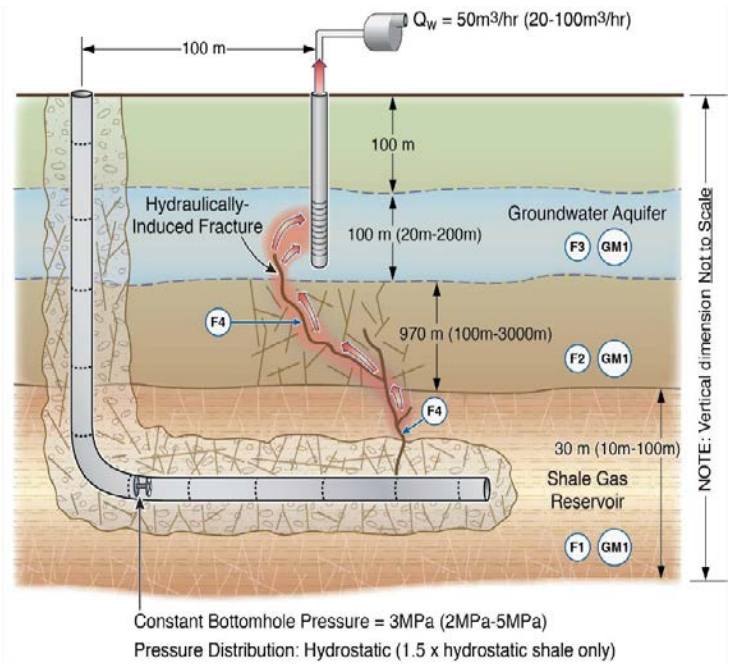

Figure 3 Illustration of the EPA problem

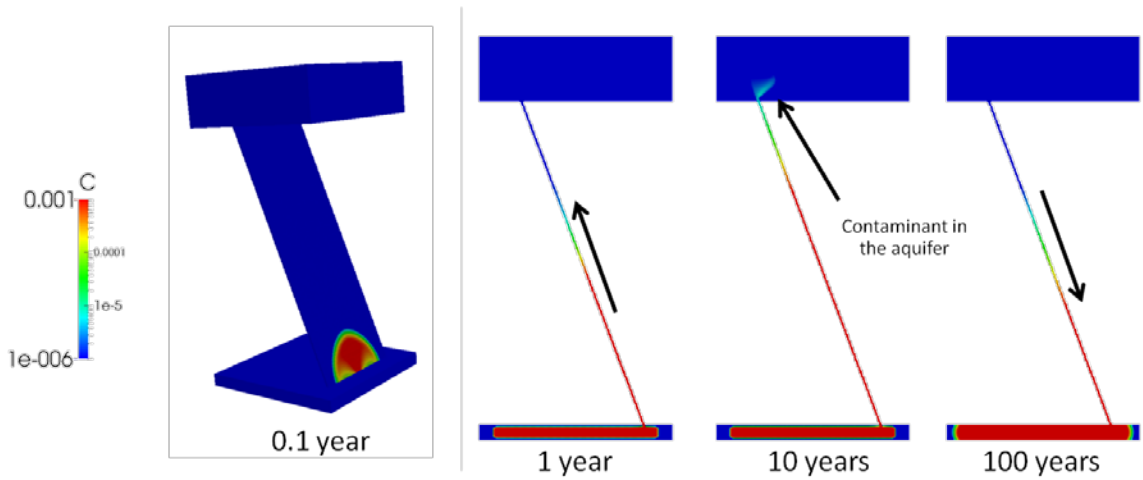

Figure 4 Contaminant concentration contours obtained with PFLOTRAN.

2.5 GEOMECHANICAL MODELING

In PFLOTRAN, a linear elasticity model is assumed as the constitutive model for deformation of the rock. The strong form is to find the displacements u_i such that

$$\sigma_{ij} + f_i = 0 \text{ on} \Omega$$
$$u_i = g_i \text{on} \Gamma_D$$
$$\sigma_{ij} n_i = h_i \text{on} \Gamma_N$$

where the stress is related to the symmetric part of the displacement gradient by a linear constitutive law known as Hooke's law as well as to the temperature and pressure of the fluid flow model.

$$\sigma_{ij} = c_{ijkl}\varepsilon_{kl} - \beta(P - P_0) - \alpha(T - T_0)$$
$$\varepsilon_{kl} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right)$$
$$c_{ijkl} = \lambda\delta_{ij}\delta_{kl} + \mu\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right)$$

whereλ and μ are the so-called Lamé constants, δ_{ij} is the Kronecker delta, β is the Biot coefficient, and α is the coefficient of thermal expansion. This system is solved using a Galerkin finite element discretization of the weak form. The weak form is to find $u_i \in \mathcal{S}_i$ which satisfies the Dirichlet conditions such that for all $w_i \in \mathcal{V}_i$ which satisfy $w_i = 0$ on Γ_D and

$$\int_\Omega w_{(i,j)}\, \sigma_{ij} d\Omega = \int_\Omega w_i\, f_i d\Omega + \int_{\Gamma_N} w_i\, h\, d\Gamma_N$$

In PFLOTRAN, the initial implementation of this weak form was by hard coding the tensors as they appear in the equations. These contractions are up to four-dimensional and involve the computing of many terms that are zero. A more computationally efficient approach is to perform the contractions analytically which culminates in solving the following linear system

$$Kd = F$$

where the improvements to the stiffness matrix are given by

$$K_{AB}^{ij} = \int_\Omega \lambda N_{A,i}N_{B,j} + \mu\left(N_{A,j}N_{B,i} + N_{A,l}N_{B,l}\delta_{ij}\right)\, d\Omega$$

Shifting to this implementation of the stiffness matrix allowed us to avoid unnecessary computation of zeroes as well as memory allocation and yielded a speedup of 3-4 times.

The linear elasticity system is loosely coupled to the flow system. That is, first the thermal hydrology equations are solved for temperature and pressure in each cell. Then the linear elasticity equations are solved given these values of temperature and pressure. The state of strain is then used to modify the porosity in the subsequent thermal hydrology step. This modification is represented by:

$$\varphi = \frac{\varphi_0}{1 + (1 - \varphi_0)\varepsilon_{ii}}$$

whereφ_0 is the unstressed porosity and ε_{ii} is the trace of the strain tensor. To include some continuum damage (Mohr Coulomb excess shear stress model), an option in the input files was added such that the intrinsic permeability becomes a function of pore pressure and change in temperature of the form:

$$MC_{stress} = \left(\frac{\sigma_1' - \sigma_3'}{2}\right)(\mu^2 + 1)^{0.5} - \mu\left(\frac{\sigma_1' + \sigma_3'}{2}\right) - S_0$$
$$k = k_0\left[\left(\frac{MC_{stress}}{MC_{stress}^{ramp}}\right)(p_F - 1) + 1\right]$$

$$\sigma_{ij}' = \sigma_{ij} + \beta(P - P_0) + \alpha(T - T_0)$$

whereσ_1' and $\sigma_3'$are the effective maximum and minimum stress,σ_1 and $\sigma_3$are their corresponding total stresses, μ the static friction coefficient, and S_0 the cohesion. The mechanical sign convention is used, i.e. tensile stresses are positive. This formulation was originally tested on simple test cases to assess the

efficiency improvement obtained by introducing the tensor analytical contraction. Since the traction boundary is not yet implemented in the geomechanics module of PFLOTRAN, these tests were performed imposing only boundary displacement conditions.

PFLOTRAN was also tested on problem 2 from [9]: Cold water is injected at high pressure in the center of a 3D domain with outside boundaries held fixed at initial conditions pressure, temperature, and far-field stress. Only 1/8th of the domain is modeled to exploit the problem symmetry. Since traction boundary conditions to model the far-field stresses are currently not available in PFLOTRAN, nodal forces were used to mimic the load. For a uniform mesh and a constant load, the nodal force can be estimated with:

$$F_{node} = \frac{\sigma_v A_{tot}}{n_{nodes}}$$

But the stress field computation is affected by larger errors in the edges and corners of the computational domain, included those in the symmetry planes. Being the injection point located in proximity of the symmetry planes, the result of such model would not be realistic. To obviate these issues, the model was solved with a local approximation for the effective stresses, without invoking the geomechanics module:

$$\sigma_1' = \sigma_1 + \left[\frac{E}{3(1-2\nu)}\right]\alpha_T \Delta T - \alpha_P P$$
$$\sigma_3' = \sigma_3 + \left[\frac{E}{3(1-2\nu)}\right]\alpha_T \Delta T - \alpha_P P$$

In the equation above compression stresses are taken positive, E is the Young's modulus, ν the Poisson's ratio, α_T the coefficient of thermal expansion, ΔT the change in temperature, α_P the Biot's coefficient, and P the pore pressure.

Figure 5 shows the mass flow rate versus time: the injection rate drops quickly early in time, then the cold injection causes lower temperatures causing fracture failing and yielding to higher permeability (Figure 6). Figure 7 shows the pressure and temperature evolution in time at several locations away from the injection site.

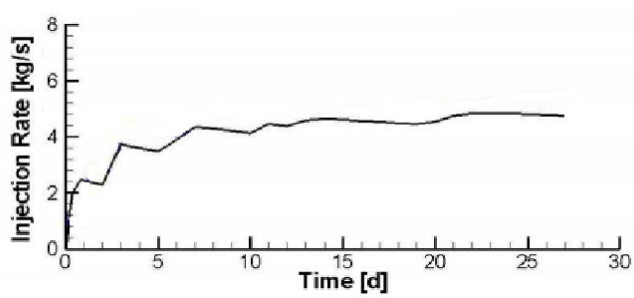

Figure 5 Injection flow rate versus time

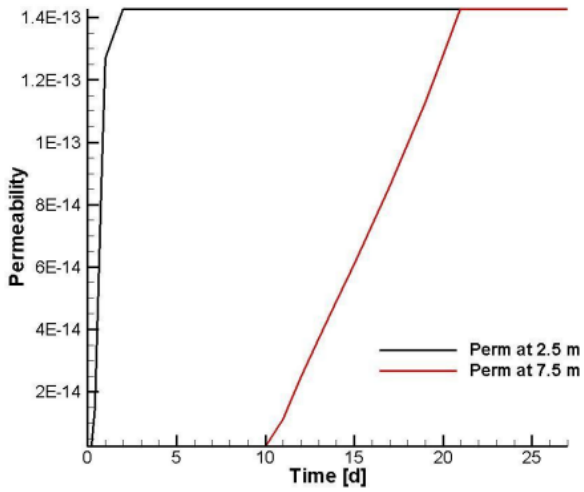

Figure 6 Permeability evolution with time at 2.5m and 7.5m from the injection site.

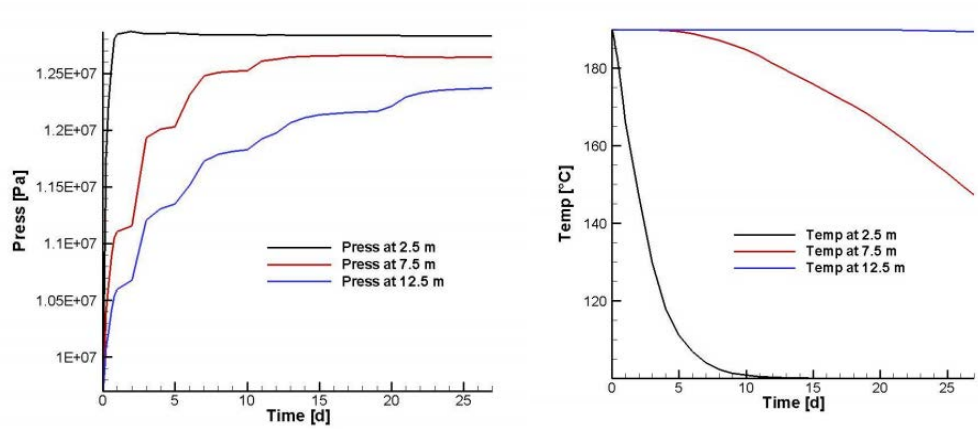

Figure7 Pressure and Temperature time evolutions at several locations away from the injection site.

2.6 WELL BOUNDARY CONDITION

Prior to this project, PFLOTRAN could model multi-block well only by imposing the same bottom hole pressure (BHP) or flow rate to all grid blocks. In addition BHP was available only for gas injector. During this project, additional well modeling capabilities were added in PFLOTRAN. More specifically, an explicit well treatment for gas/water producers and injectors over several well blocks was implemented with the following control options:

- mass rate
- volumetric rate
- BHP

In other words, the capability of controlling multi-block wells, prescribing a single control parameter for each well, was added. In addition, well hydrostatic pressure corrections, well factors (drilling direction, well diameter and skin factor), well operational limits (max BHP for injectors, min BHP for producers, max mass flow rate, max volumetric flow rates, max water cut for gas producer), and variable temperature injection were implemented. All these features can run in parallel with automatic domain decomposition following the PFLOTRAN philosophy.

2.6.1 Well modeling: IPR

The Inflow Performance Relation (IPR) is used to model the wells. The limitation of this formulation is that the near-well Stokes flow and the large pressure gradient near the casing are not captured. The IPR model is a synthetic way to model the pressure drop between the well and a perforated grid block, accounting the near-well effects:

$$Q_{\alpha,j} = T_{w,j}\lambda_{\alpha,j}\left(p_{\alpha,j} - p_w\right)$$

Where $T_{w,j}$ is the well factor, $p_{\alpha,j}$ the grid block pressure, and p_w the well flowing BHP.

The well factor is computed with the Peaceman [10] formulation:

$$T_{w,j} = \frac{2\pi kh}{\ln\left(r_0/r_w\right) + S}$$

Where r_w is the well radius, r_o the grid block pressure equivalent radius (radius where the pressure is equal to the grid block average pressure), S the skin factor (accounting for damage around the casing, etc.), k the effective permeability (in the plane perpendicular to the drilling direction), and h the thickness in the drilling direction.

$$r_0 = 0.28\frac{\left[\Delta x_i^2\left(\frac{K_j}{K_i}\right)^{1/2} + \Delta x_j^2\left(\frac{k_i}{k_j}\right)^{1/2}\right]^{1/2}}{\left(\frac{k_j}{k_i}\right)^{1/4} + \left(\frac{k_i}{k_j}\right)^{1/4}} \qquad k = \left(K_i K_j\right)^{1/2}$$

Figure 8 illustrates how a well is perforating multiple grid blocks and how the multi-block well is controlled by a single reference pressure.

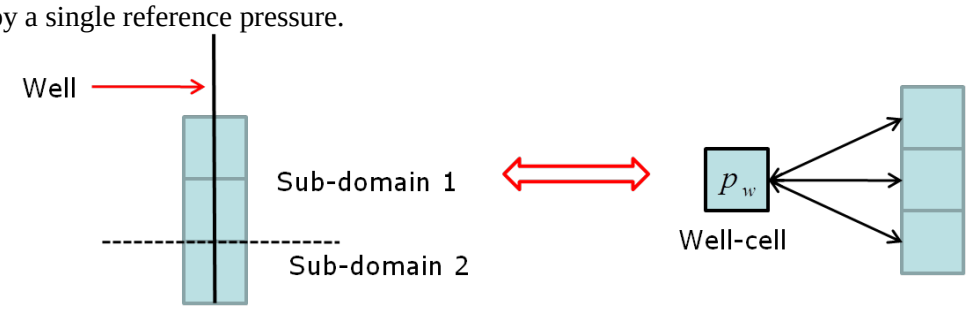

Figure 8 Illustration of the well perforating several blocks

Multi-block wells controlled by a single reference pressure require pressure corrections (see Figure 9) to account for hydrostatic pressure effects (was not available in PFLOTRAN). The hydrostatic corrections depend on the wellbore fluid density and pressure. The literature indicates they are usually treated explicitly for stability reasons.

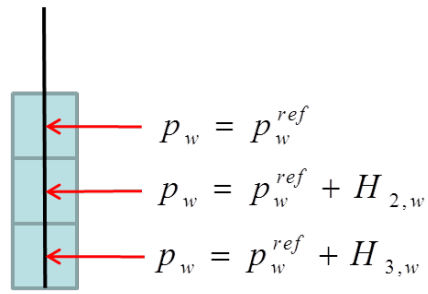

Figure 9 Well hydrostatic head correction illustration

For each multi-block well a new unknown is introduced (flow rate or the reference BHP) which requires an additional equation. Phase conservation equations for the volumetric or mass rate are used to close the system [11]:

- Well phase volumetric equation:

$$Q_{w,\alpha} = \sum_{i=1}^{n_{perf}} T_{i,w}\lambda_{i,\alpha}\left(p_{i,\alpha} - p_w^{ref} - H_{i,w}\right) \qquad \alpha = l, g$$

- Well phase mass conservation equation:

$$M_{w,\alpha} = \sum_{i=1}^{n_{perf}} T_{i,w}\lambda_{i,\alpha}\rho_\alpha\left(p_{i,\alpha} - p_w^{ref} - H_{i,w}\right) \qquad \alpha = l, g$$

Each well is then controlled by imposing either a reference BHP or a flow rate (Q or M) for the water (l) or the gas (g).

In MPHASE, the well reference BHP obtained from the well conservation equation is used to compute the chemical component and energy fluxes in each well connection:

- Fluid volumetric rate for each phase:

$$q_{i,\alpha} = T_{i,w}\lambda_{i,\alpha}\left(p_{i,\alpha} - p_w^{ref} - H_{i,w}\right) \qquad \alpha = l, g$$

- Molar flow rate for each component (e.g. H_2O,CH_4):

$$q_{i,x^j} = \sum_{\alpha=1}^{np} q_{i,\alpha}\eta_{i,\alpha}x_{i,\alpha}^j \qquad \alpha = l, g$$

- Energy flow rate:

$$q_{i,E} = \sum_{\alpha=1}^{np} q_{i,\alpha}\eta_{i,\alpha}h_{i,\alpha} \qquad \alpha = l, g$$

Where i refers to the ith well connection, α the phase (gas or liquid), $X^j$the molar fraction for the chemical component$j$, and h the enthalpy.

2.6.2 Explicit well treatment

Although explicit well equations require smaller time steps and cannot model cross-flow between perforated grid blocks, they are simpler to implement than implicit ones, for which an extension of the Jacobian is required. With the explicit approach the hydrostatic corrections, the well fluid properties and the well factors are computed at the previous time step, while the pressure, the mobilities and the densities of the perforated grid blocks are evaluated at the previous Newton iteration.

If the well is controlled by a flow rate (given for each time step), the reference BHP is computed at the latest Newton iteration with:

$$p_w^{ref} = \frac{\sum_{i=1}^{n_{perf}} T_{i,w}\lambda_{i,\alpha} p_{i,\alpha} - \sum_{i=1}^{n_{perf}} T_{i,w}\lambda_{i,\alpha} H_{i,w} - Q_{w,\alpha}}{\sum_{i=1}^{n_{perf}} T_{i,w}\lambda_{i,\alpha}}$$

If instead, the well is controlled by reference BHP, the well volumetric (or mass) flow rates of the fluid phases being produced or injected are computed at each Newton iteration with:

$$Q_{w,\alpha} = \sum_{i=1}^{n_{perf}} T_{i,w}\lambda_{i,\alpha}\left(p_{i,\alpha} - p_w^{ref} - H_{i,w}\right) \qquad \alpha = l, g$$

Next, the well reference pressure is used to compute the well connection volumetric fluxes:

$$q_{i,\alpha} = T_{i,w}\lambda_{i,\alpha}\left(p_{i,\alpha} - p_w^{ref} - H_{i,w}\right)$$

Finally, the component and energy fluxes at each well connection can be computed and added to the residual equations of the perforated grid blocks:

$$q_{i,x^j} = \sum_{\alpha=1}^{np} q_{i,\alpha}\eta_{i,\alpha} x_{i,\alpha}^j \qquad \alpha = l, g$$

$$q_{i,E} = \sum_{\alpha=1}^{np} q_{i,\alpha}\eta_{i,\alpha} h_{i,\alpha} \qquad \alpha = l, g$$

2.6.3 Well definition in the input file

A well is defined as a group of grid blocks containing sources/sinks, whose values depend on the well control variables. Therefore, it is defined as a SOURCE_SINK in the PFLOTRAN input file. A WELL_SPEC card, inserted as a subcard of a SOURCE_SINK coupler defines the well. In the example below, the minimum number of input parameters required to define a well are reported.

```
SOURCE_SINK wat_inj
  FLOW_CONDITION injection
  REGION well1
  WELL_SPEC
    WELL_TYPE GAS_INJECTOR
    VAR_CONTROL MASS_RATE
    CONSTANT_WELL_FACTOR 2.6e-15
END
END
```

The user can enter parameters to overwrite the default values, see below:

```
SOURCE_SINK wat_inj
  FLOW_CONDITION injection
  REGION well1
  WELL_SPEC
```

```
    WELL_TYPE GAS_INJECTOR
    VAR_CONTROL MASS_RATE
    RADIUS 0.1 m
    SKIN_FACTOR 0.001 !default is 0
    CONST_DRILL_DIR DIR_X  !drilling direction in the well factor formula, default value is DIR_Z
    THETA_FRACTION 0.25d0 !well angle exposure – e.g. For a corner well 0.25 – default is 1
!CONSTANT_WELL_FACTOR 2.6e-15 ! If specified the well factor computation is skipped
  END
END
```

The values of the well control variables and operational limits can be time dependent, thus specified in the flow condition using the sub flow condition WELL. See below for an example:

```
FLOW_CONDITION injection
  UNITS Pa,C,M,yr
  TYPE
    WELL well
    TEMPERATURE dirichlet
    CONCENTRATION zero_gradient
    ENTHALPY dirichlet
  /
  WELL file wfl_inj.txt        ! See example for format
  TEMPERATURE 100.d0     !this value is ignored – specified in wfl_inj.txt
  CONCENTRATION 1.D-12 !this is ignored – specified in wfl_inj.txt
  ENTHALPY 0.d0 0.d0        ! Not used – computed internally for injecting and producing wells
/
```

2.6.4 PFLOTRAN code implementation

Two classes have been added:

- Well_AuxVar_Base (well_auxvar_base.F90)
- Well_AuxVar_MPhase (well_auxvar_mphase.F90)

These classes are used by the coupler to define a well. A coupler allocates a Well_AuxVar object only if WELL_SPEC is found in its definition. Well_AuxVar_Base contains data and function members usable by all PFLOTRAN flow modes, while the Well_AuxVar_modeXXX implements mode specific members. Only the MPHASE extension has been implemented so far. A Well_module (well.F90) has been also implemented, which works as an interface between the well classes and other classes.

The functions implemented in Well_AuxVar_Base are the following:

- Create a well communicator and group of processors. A well can be distributed between more processors, its partitioning follows the “localization” of the region the well is associated to. For each well a communicator and a group of processors is created, in which only the processors containing a portion of the well are included.
- Determine the well “controlling” connection: i.e. the well connection where the well reference pressure and temperature are defined (the elevation of this connection is named reference elevation: z_ref). Only one method is currently available, which identifies the controlling connection in the shallower grid block.
- Detect the processor that contains the “controlling” connection.

- Sort the well connections in ascending elevation order to facilitate the computation of the hydrostatic pressure corrections.

The functions implemented in Well_AuxVar_MPhase are:
- Solve explicitly the well conservation equations for which MPhase auxvars are required
- Perform checks on operational limits specific to MPhase
- Compute the well pressure hydrostatic corrections for which MPhase auxvars are required

For each well, a text file is outputted in Tecplot format such that at each flow time step, the following variables are printed:
- reference pressure
- reference temperature
- reference water density
- reference gas density
- water volumetric rate being produced/injected
- gas volumetric rate being produced/injected
- water mass rate being produced/injected
- gas mass rate being produced/injected

2.6.5 Well boundary condition demonstrations

2.6.5.1 Water injector

Water is injected at the corner of the domain with a BHP of 12.91 MPa with and without hydraulic pressure correction. As expected, a higher maximum pressure and a larger injection flow rate is observed with the hydraulic pressure correction (Figure 10 and Figure 11)

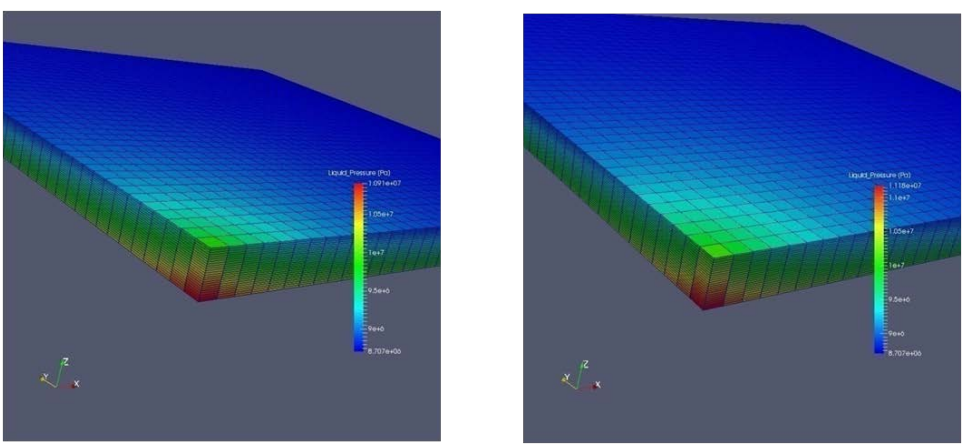

Figure 10 Liquid pressure contours for a well injecting water with a BHP of 12.91 MPa at t=27 days with (right) and without (left) hydraulic pressure correction.

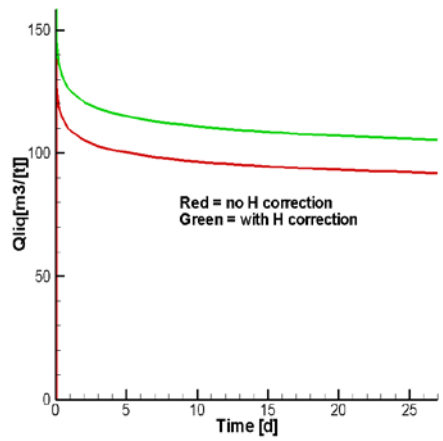

Figure 11 Volumetric flow rate at the well with and without hydraulic correction.

2.6.5.2 Gas producer demonstration

First, the initial condition was generated from an initial mass of gas located in the top left side of the domain, which is left to spread gradually for 2000 days (see Figure 12). Then two wells are activated in the west side corners, both with a BHP of 0.3 MPa, and produce about 5% of the gas after 1000 days. As illustrated in Figure 13, the gas spreading due to buoyancy is still occurring causing large changes in saturation. A gas production rate "knee", occurring during the extraction, is shown in Figure 14 with and without hydraulic pressure correction. Figure 15 and Figure 16 show how one of the perforated grid blocks produces a pressure discontinuity due to the switch from fully to partially gas saturated (the capillary pressure jumps from zero to a finite value). The phenomenon is magnified here due to the low permeability. With one block partially saturated, the well reduces its production ("knee" observed in Figure 14).

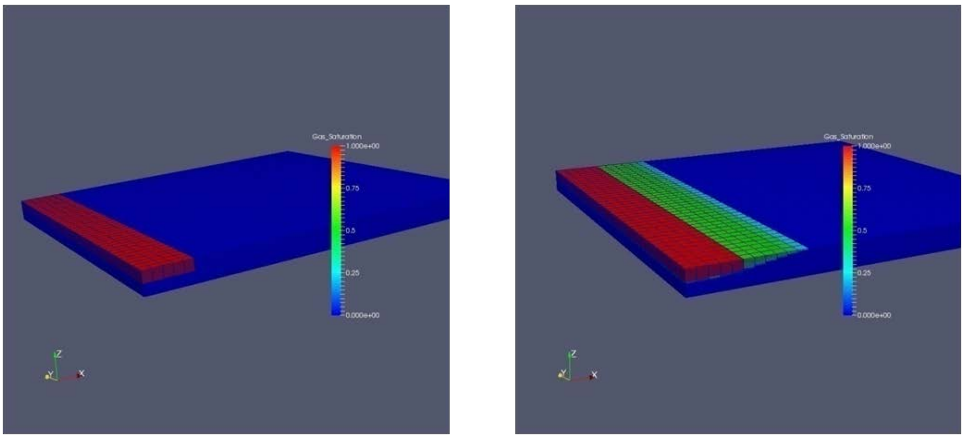

Figure 12 Pre-run to generate the initial condition for gas production at t=0 (left) and t=2000 days (right)

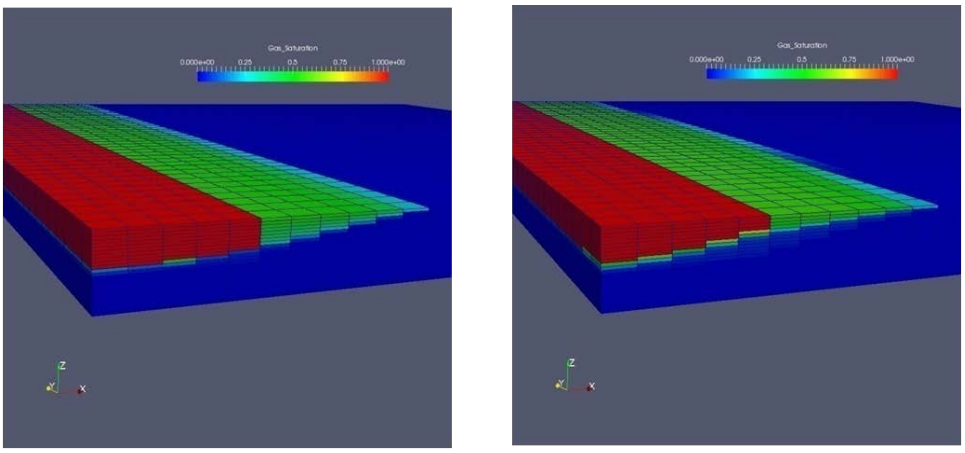

Figure 13 Gas saturation at the beginning of production (left), and after 1000 days (right).

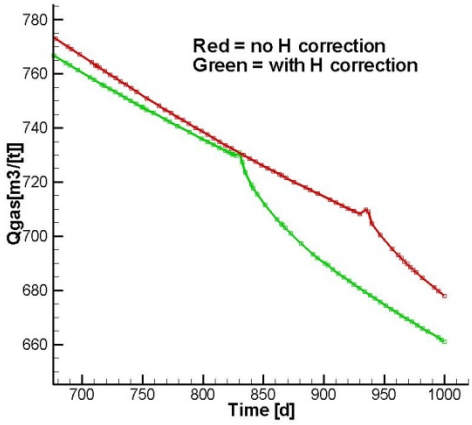

Figure 14 Gas production rate knees

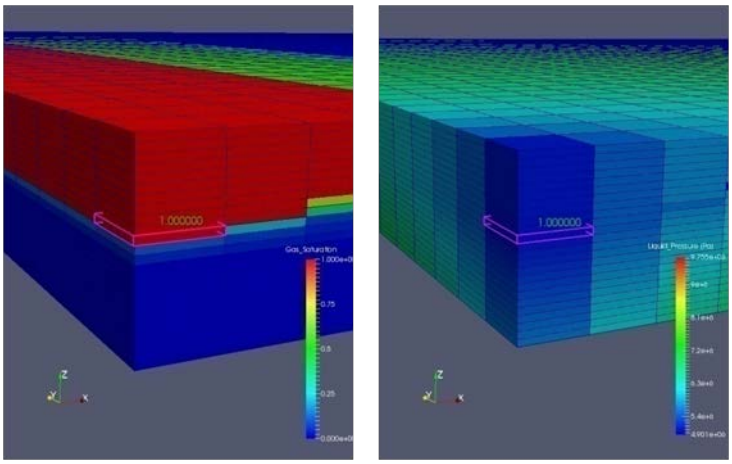

Figure 15 Gas saturation (left) and liquid pressure (right) at t=825 days

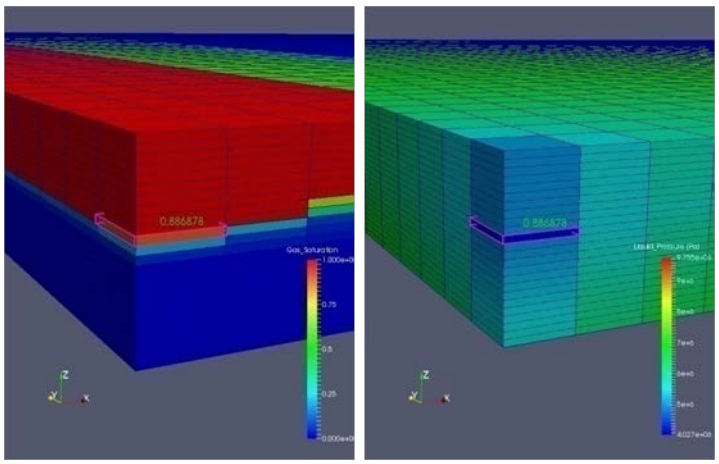

Figure 16 Gas saturation (left) and liquid pressure (right) at t=830 days

3. UNCERTAINTY QUANTIFICATION

3.1 UNCERTAINTY QUANTIFICATION PROBLEMS

Subsurface flow and transport models are routinely employed in the prediction of fluid behavior, and in making economic and management decisions. These models describe fluid flow (e.g., oil, gas, and water) through a porous medium, so the porous media properties (e.g., permeability, porosity and capillary) have significant impact on the model outputs such as mass flow rate and pressure. Almost all the porous media properties used in subsurface flow simulations are subject to uncertainty. This uncertainty may be quite large, as direct measurements of these properties are available at only a limited number of boreholes, and indirect measurements inferred from logs, well tests or subsurface data involve averaged responses over a large scale. Analyzing the uncertainties of these properties and quantification of their influence on predictions of the fluid behavior has become increasingly important to facilitate science-informed decisionmaking in subsurface environment management.

Generally speaking, there are two types of uncertainty quantification (UQ) problems, inverse UQ and forward UQ. In inverse UQ, we evaluate the uncertainty of model parameters based on observation data; in forward UQ, we quantify the uncertainty of quantities of interest (QoI) propagated from the parameter uncertainty (Lu et al. 2015a).

The uncertainty parameter is usually evaluated by regression and Bayesian approaches.The regression methods quantify uncertainties using confidence intervals. They are computationally cheap but often need significant restrictive assumptions for effective evaluation. The Bayesian methods quantify uncertainties using the entireprobability density function and they can be applied to complex nonlinear models incorporating multiple types of data and prior information (Hill et al. 2015). However, the computational cost of the Bayesian methods is very expensive (Lu et al. 2014). In this project, we propose a sparse-grid (SG) stochastic collocation method to improve the computational efficiency in applying the Bayesian methods to quantify the parameter uncertainty.

A wildly used forward UQ methodology in subsurface flow is to treat the porous media properties as random fields first and then apply stochastic techniques to quantify the uncertainty of the QoI. Among the techniques, perturbation/moment equation methods, generalized polynomial chaos expansions, and stochastic collocation methods are not suitable for highly heterogeneous porous media; the standard Monte Carlo (MC) simulation is widely used for the highly stochastic problems but it is very computationally demanding. In this project, we propose a multilevel Monte Carlo (MLMC) method to

quantify theuncertainty of subsurface flow in a highly heterogeneous porous media. The MLMC method is shown to be more computationally efficientthan the standard MC.

3.2 SPARSE GRID METHOD AND APPLICATION

In this section, we are interested in quantifying the uncertainty of porous media properties based on the observation data using the Bayesian approach; specifically, we use conventional Markov chain Monte Carlo (C-MCMC) and SG-based MCMC (SG-MCMC) simulations to estimate the posterior probability density function (PPDF) of the model parameters. In the discussion, we concentrate on comparing the computational efficiency of the two methods.

3.2.1 Sparse grid interpolation used in MCMC simulation

The Bayesian method generally uses MCMC simulations to estimate the PPDF of model parametersbased on the procedure in Figure 17. MCMC simulation is typically computationally expensive especially for complex models with a large number of parameters. As shown in Figure 17,generation of each parameter sample requires one forward model run, and each application model runcan take several minutes, hours, or even days. In addition, to achieve the convergence, often tens of thousands, hundreds of thousands, or even millions of model executions are needed. Thus, one of the most effective ways to improve the computational efficiency of the MCMC processis to build a surrogate system of the actual physical model to reduce the time of each model run.

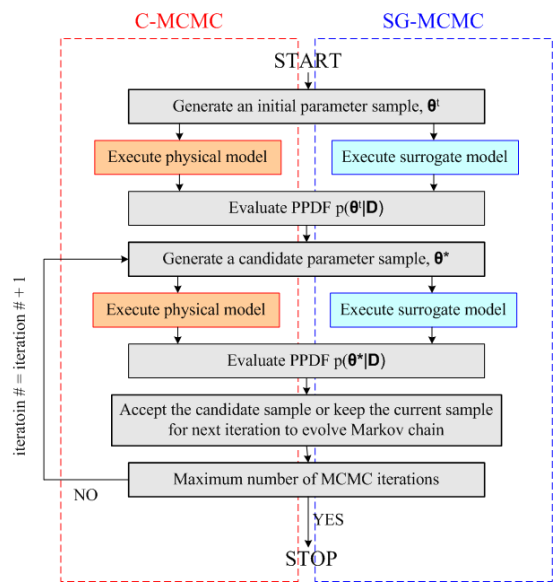

Figure 17 Flowchart of the MCMC run. The difference between the C-MCMC and the SG-MCMC simulations is that C-MCMC executes the actual physical model and SG-MCMC executes the SG-based surrogate model in each sample generation.

In this project, we use the adaptive SG interpolation to build the surrogate system. The advantage of this powerful interpolation method is that it requires significantly fewer support points on a sparse grid than conventional interpolation on a full grid. An interpolation point means one physical model execution and the difference in the number of required points between the full and sparse grids can be several orders of magnitude with increasing problem dimension. Thus, it can be significantly more efficientto use the SG interpolation to build the surrogate system. An additional benefit of the adaptive SG is its hierarchical structure, which can terminate the increase of interpolation points automatically when a desired accuracy is reached. Take a one-dimensional function $f(\theta)$ (in application $f(\theta)$ represents a physical model with

parameter θ) in Figure 18 for example. To achieve the same interpolation accuracy, the adaptive SG only needs 21 points (21 function evaluations), less than one third of the points required by the full grid. Practically, when the simulation time of one function evaluation is costly, the saved interpolation points can lead to outstanding computational cost savings. More importantly, in SG-MCMC simulation, after the surrogate model is built, no more physical model execution is needed as shown in Figure 17. So a longer MCMC process usually results in larger cost savings of the SG-MCMC compared to the C-MCMC, because the execution of the surrogate model is just polynomial evaluation. More information about the SG-MCMC is presented in Lu et al. 2015b.

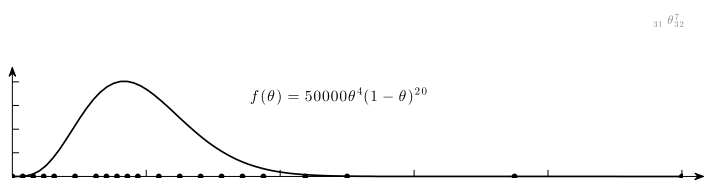

Figure 18 A seven-level adaptive sparse grid for interpolating a 1-D function $f(\theta) = 50000\,\theta^4(1-\theta)^{20}$ on bound [0, 1] with the error tolerance of 0.04. The resulting adaptive sparse grid has only 21 points as shown in black dots, whereas the full grid needs 65 points as shown in both black and gray dots.

3.2.2 Application to an oil reservoir model

We apply both the C-MCMC and SG-MCMC methods to a synthetic oil and gas reservoir model revised from Kissinger et al. (2013). The model has four parameters: porosity and permeability of upper Permian layer and the same two parameters of the fault zone. We assume that all parameters have uniform prior distribution with ranges listed in Table 1. The model is first simulated using PFLOTRAN with the true valuesin Table 1to generate 28 temperature and 22 tracer concentration data. The 50 simulated values are then perturbedby some Gaussian random noises and the noisy data are used in the MCMC simulation to evaluate the PPDF. The random noise for the temperature data has standard deviation of 0.01, and the 2% random noise is added to the tracer concentration. One model simulation takes about three minutes using eight processors. The computational cost is intensive for the C-MCMC simulation, so a surrogate model is necessary to improve the computational efficiency in the Bayesian uncertainty analysis.

Table 1True parameter values and their ranges of the four parameters

	Por_UP	$\log_{10}(Perm_UP)$	Por_F	$\log_{10}(Perm_F)$
True value	0.005	-13.398	0.01	-13.046
Range	[0.001, 0.01]	[-13.8, -12.8]	[0.008, 0.012]	[-13.2, -12.8]

The surrogate system is constructed on an adaptive grid with cubic basis functions. For this four-dimensional problem, only 295 SG points are needed to satisfy the error tolerance of 0.01. To verify whether the surrogate system captured the model features sufficiently well, we compare the simulated temperature and tracer concentration between the original physical and surrogate models for 1000 parameter combinations randomly drawn from their prior distributions. The comparison results are summarized in Figure 19. The figure indicates that the surrogate model can adequately capture the behavior of the original CPU-intensive model in recovering the simulation results. Evidence of this is given by the correlation coefficient (r^2) and the root mean square error (RMSE) of the two simulations. Figure 19shows that for both temperature and tracer concentration $r^2 = 1$, which indicates the extremely strong consistency between the two sets of simulations; the negligible RMSE of the two types of data suggests that essentially there is no difference between the original and surrogate models. These findings give confidence that the surrogate model provides a sufficiently accurate substitute for the original model in producing the simulations.

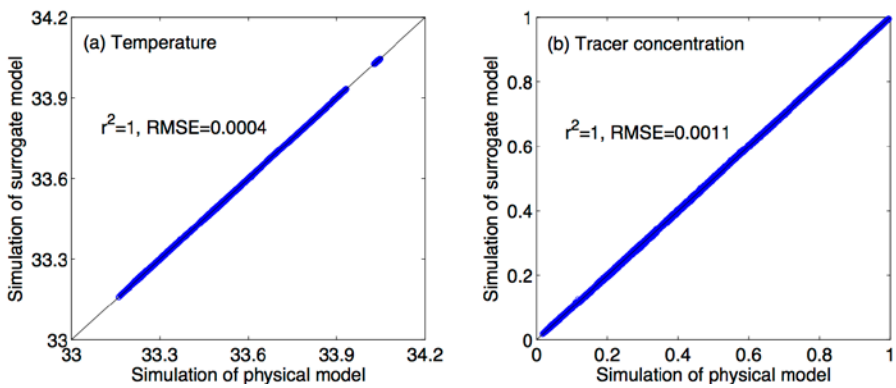

Figure 19 Pair wise comparison of the 28 simulated temperature and 22 tracer con- centration of the original physical and surrogate model for 1000 parameter combinations randomly drawn from the prior distribution.

We used both C-MCMC and SG-MCMC simulations to estimate the PPDF of the four parameters. The results of the C-MCMC are used as reference to evaluate the accuracy and efficiency of the SG-MCMC simulation. Both simulations are initialized from the prior distributions and each MCMC simulation draws 80,000 parameter samples using eight Markov chains; each of which evolves 10,000 generations. Convergence of the Markov chains is examined using the Gelman-Rubin R statistics (Gelman et al., 1995), which indicates that the chains converge after 1000 generations for both MCMC runs. So the first 1000 generations of each chain are discarded, and the remaining 80,000 - 8 × 1,000 = 72,000 samples are used to estimate the PPDF.

Figure 20 plots the marginal PPDFs of the four parameters. The figure indicates that the SG-MCMC results are effectively close to those of C-MCMC. However, the SG-MCMC needs significantly fewer model executions. In comparisonwith the 80,000 model executions used inthe C-MCMC, the number of model executions of the SG-MCMC is only 295 for the construction of the surrogate system, and this number can be expected to be smaller when a higher-order polynomial basis is used. After that, SG-MCMC does not require any model runs to draw the 80,000 parameter samples but negligible computational time for polynomial evaluation. The improvement of computational efficiency of the SG-MCMC is more evidentwhen more parameter samples are drawn.

To sum up, this section presents an adaptive SG method to improve the computational efficiency of Bayesian inference for parameter uncertainty quantification. The method is model independent and flexible to be used together with any MCMC algorithms.

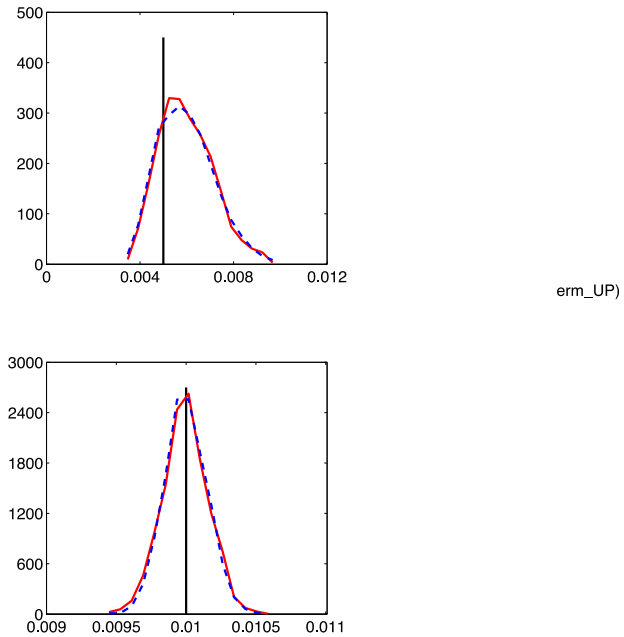

Figure 20 Marginal PPDF of the four parameters estimated using C-MCMC with 80,000 model executions and the SG-MCMC with 295 model executions, respectively. The black vertical lines represent true parameter values. Take the C-MCMC results as reference, the estimations by the SG-MCMC are accurate enough but with significantly computational cost reduction.

3.3 MULTILEVEL MONTE CARLO METHOD AND APPLICATION

In this section, we are interested in investigating the influence of uncertain permeability field on mass flow rate Q; specifically, we use the expectation E[Q] and cumulative distribution function (CDF) F(q) to quantify the uncertainty of Q. We use both the standard MC and MLMC methods and focus on comparing computational efficiency of the two methods.

3.3.1 MLMC method in estimation of expectation

In the MC simulation, the simulated domain is usually discretized into a grid mesh with M grid cells, and the discretized solution Q_M is used to approximate the actual solution Q. The standard MC estimator for E[Q_M] is,

$$\hat{Q}_M^{MC} = \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} Q_M^{(i)},$$

where $Q_M^{(i)}$ is the ith sample of Q_M and N_{MC} is the total number of independent samples. Note that $\hat{Q}_M^{MC}$ is an unbiased estimator, i.e., $E[\hat{Q}_M^{MC}] = E[Q_M]$. For the estimation of E[Q], $\hat{Q}_M^{MC}$ includes two sources of error, the MC sampling error in estimating E[Q_M] and the discretization error between Q_M and Q. Specifically, the mean square error (MSE) of $\hat{Q}_M^{MC}$ can be represented as

$$e\left(\hat{Q}_M^{MC}\right)^2 = E\left[\left(\hat{Q}_M^{MC} - E[Q]\right)^2\right]$$
$$= N_{MC}^{-1} V[Q_M] + \left(E[Q_M - Q]\right)^2$$

where the first term of sampling error can be reduced by increasing the number of samples, and the second term of discretization error can be reduced by refining the model grid M. Since both methods require an increase of computational cost, the MLMC method improves efficiency by working on the variance.

Using MLMC for E[Q] estimation, the terminology "multilevel" refers to the fact the E[Q] is not estimated by E[Q_M] using simulations of Q_M on a single level associated to the grid M, but by $E[Q_{M_L}]$ using simulations from an increasing sequence of levels $\left\{Q_{M_l}, l = 0, \cdots, L\right\}$ with $M_0 < M_1 < \ldots < M_L = M$. Usually $M_{l-1} = 2^{-d} M_l$ where d is the dimension of the spatial domain. Here l is the mesh resolution level; M_l is the number of grid cells of the discretized mesh for Q_{M_l} simulation. Mathematically, the expectation E[Q_M] can be expressed as

$$E[Q_M] = E[Q_{M_0}] + \sum_{l=1}^{L} E[Q_{M_l} - Q_{M_{l-1}}] = \sum_{l=0}^{L} E[Y_l],$$

and each E[Y_l] can be estimated by

$$E[Y_0] \approx \hat{Y}_0 = \frac{1}{N_0}\sum_{i=1}^{N_0} Q_0^{(i)} \text{ and } E[Y_l] \approx \hat{Y}_l = \frac{1}{N_l}\sum_{i=1}^{N_l} (Q_l^{(i)} - Q_{l-1}^{(i)}).$$

Then the MLMC estimator is simply

$$\hat{Q}_M^{ML} = \sum_{l=0}^{L} \hat{Y}_l.$$

The idea of MLMC indicates that the expectation on the finest level is equal to the expectation on the coarsest level plus a sum of difference in expectation between simulations on consecutive levels. In this way, the less accurate estimate on the coarsest level is sequentially corrected by the estimates on the following finer levels, thereby the MLMC estimator can achieve the same accuracy as the standard MC on the single finest level.

In the same manner, the MSE of MLMC can be decomposed into the sampling error and the discretization error as

$$e\left(\hat{Q}_M^{ML}\right)^2 = E\left[\left(\hat{Q}_M^{ML} - E[Q]\right)^2\right] = \sum_{l=0}^{L} N_l^{-1} V[Y_l] + \left(E[Q_M - Q]\right)^2$$

where the discretization error is exactly the same as the standard MC estimator. However, thecomputational cost to achieve the same RMSE accuracyεis dramatically different between the two MC methods. To achieve ε, a sufficient condition of the standard MC estimator is that $N_{MC} = 2\varepsilon^{-2} V[Q_M]$. Assuming the computational cost of simulating one sample of Q_M is C_M, the total cost of the standard MC is $C(\hat{Q}_M^{MC}) = N_{MC} C_M = 2\varepsilon^{-2} V[Q_M] C_M$. In the similar analysis, the total cost of the MLMC to achieve εis $C(\hat{Q}_M^{ML}) = \sum_{l=0}^{L} N_l C_l$, where $N_l = 2\varepsilon^{-2}\sqrt{V[Y_l]/C_l}\left(\sum_{k=0}^{L}\sqrt{V[Y_k]C_k}\right)$ is the number of samples on level l and $C_l = C_{M_l} + C_{M_{l-1}}$ is the cost of simulating one sample on level l.

It can be shown that to achieve the same accuracyε, the cost of MLMC is asymptotically lower than that of the standard MC as long as the variance $V[Y_l]$ decays reciprocally with l. The reason is that as$\varepsilon \rightarrow 0$, then$l \rightarrow \infty$; when $l \rightarrow \infty$, if $V[Y_l]$ decays reciprocally with l, then $V[Y_l] \rightarrow 0$ and $N_l \rightarrow 0$; since $V[Q_M]$ is approximately constant, independent of M, this results in $V[Y_l] << V[Q_M]$ and consequently $N_l << N_{MC}$, especially for large l. That is, MLMC simulates more samples at the coarser levels with inexpensive model runs and fewer samples at the finer levels with expensive model runs. In this way, MLMC saves computational cost compared to the standard MC that simulates a relatively large number of samples solely on the finest level with the most expensive model runs. The cost savings of MLMC can be significant when the variance $V[Y_l]$has a fast decay. More information about the application of the MLMC in expectation estimation is presented in Lu et al. 2015c.

3.3.2 MLMC method in estimation of CDF

By definition, the CDF F(q) on a bounded range of [a,b] can be represented by $F(q) = \int_{-\infty}^{q} f(Q)\,dQ$, where $f(Q)$ is the corresponding probability density function. The integral can be formatted as an expectation of an indicator function $I(-\infty,q](Q)$, i.e.,

$$F(q) = E[I_{(-\infty,q]}(Q)] = \int_{-\infty}^{+\infty} I_{(-\infty,q]}(Q) f(Q)\,dQ$$

where $I_{(-\infty,q]}(Q) = 1 \text{ for } Q \in (-\infty,q] \text{ and } I_{(-\infty,q]}(Q) = 0 \text{ for } Q \in (q,+\infty]$. In approximation of the continuous function F(q), we first discretize the range [a,b] into a sequence of equidistant points q_n and then estimate $F(q_n)$ by the two MC methods. Last, we extend the point approximations to the approximation of function F(q) using piecewise linear Lagrange interpolation.

As discussed before, the prerequisite of MLMC to be superior to the standard MC is the decay of variance, and a faster decay leads to a larger cost saving of the MLMC. However, the indicator function used in the CDF estimation is a step function with singularity at pointsq_n and this singularity hampers the variance decay and consequently the MLMC performance. To resolve this problem and improve the MLMC's efficiency, a smoothing step is necessary. Simply speaking, the smoothing step uses a smooth function $g(Q)$ to approximate the actual step function $I(Q)$, and we estimate the expectation of $E[g(Q)]$ instead of $E[I(Q)]$ in the CDF estimation.Since $g(Q)$ is a smooth function, a relatively large variance reduction is expected. The smoothness of the function is controlled by a smoothing factorδ. Carefull calculation of δ has an important influence of the MLMC performance. Giles et al. (2015) calculatedδ in an asymptotic sense with the assumption of $\varepsilon \rightarrow 0$. Because$\varepsilon$ can never practically approach zero due to the irreducible discretization error, their determined δ may deteriorate the computational efficiency of the MLMC. In this project, we propose a posteriori approach to estimate the optimal δ and thus maximize the MLMC efficiency.More information about the application of the MLMC in CDF estimation is presented in Lu et al. 2015d.

3.3.3 Application to an oil reservoir model

To illustrate the MLMC method and demonstrate its effectiveness and efficiency, we applied the algorithm to a synthetic study of an oil reservoir simulation. The synthetic case is designed based on the fine grid model from the tenth SPE comparative solution project. The synthetic model is two-dimensional with total 60 × 220 cells. The permeability (k) field is shown in Figure 21(a), which also shows the location of the injection and production wells. We are interested in investigating the influence of the permeability uncertainty on the mass flow rate Qat the production well P. Specifically, we want to estimate the expectation and CDF of Qusing the standard MC and the MLMC methods.

We assumed that the random permeability field log(k) is a Gaussian process whose mean and covariance structure can be inferred from the 36 synthetic data regularly situated in Figure 21(b). Conditioned on the 36 data, random realizations of the permeability field were generated, where one realization is shown in Figure 21(c). In the application of the MLMC method, the finest mesh has $M = 60 \times 220$ grid, based on which we define a total of six levels as shown in Figure 22. All six levels are used in the expectation estimation and the last five levels are used in the CDF estimation according to the variance decay.

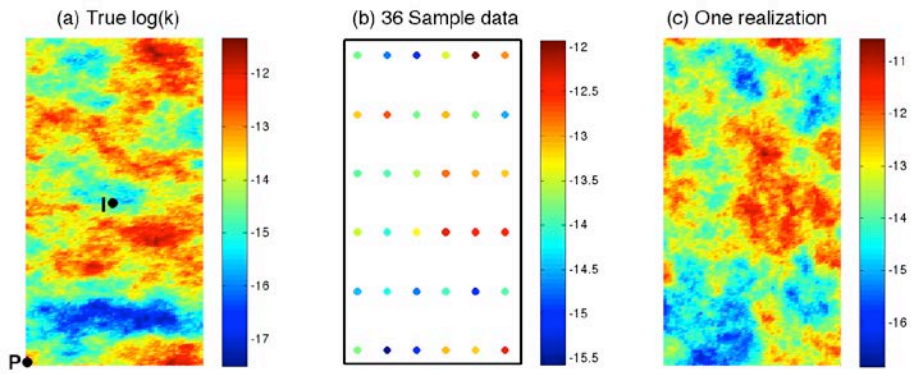

Figure 21(a) True log permeability field log(k); I represents the injection well and P is the production well. (b) Locations of 36 sample data drawn from (a); conditioning on these samples, the realizations of random log(k) field are generated. (c) An example of one realization of random log(k) field.

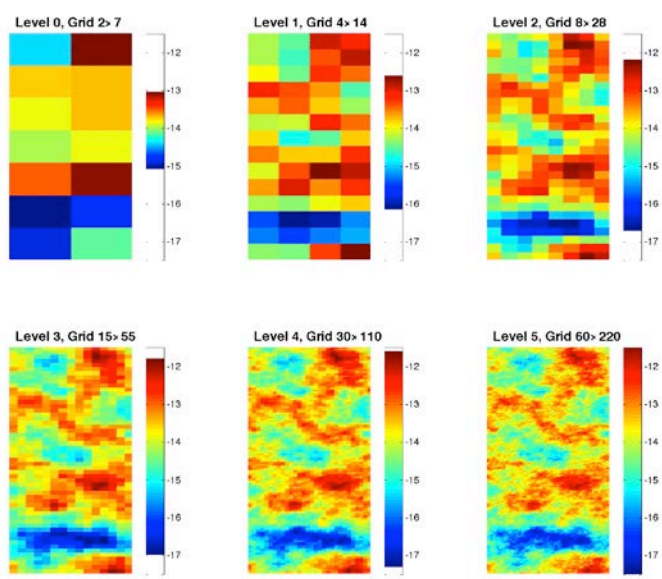

Figure 22 Simulation meshes on different levels used in MLMC. The highest level5 has the same mesh as the synthetic model.

Figure 23 plots the computational time of the standard MC and the MLMC to achieve the same accuracy in estimation of the expectation. The figure indicates that for the same desired ε, MLMC needs significantly less computational time; as the value of ε decreases, the increase ofthe computational time of the standard MC is much faster than that of the MLMC, suggesting that for a higher accuracy requirement, the MLMC will have superior performance.

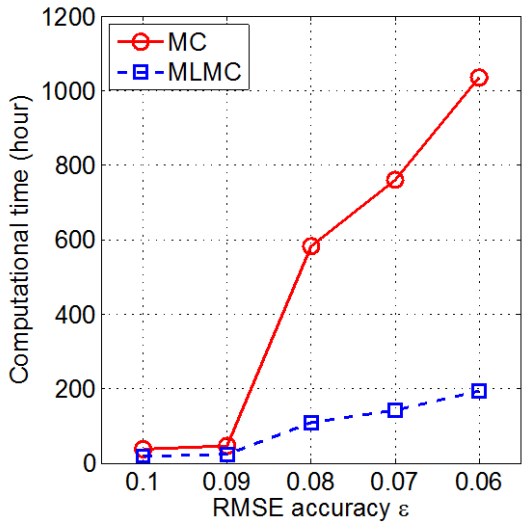

Figure 23 Computational time of the standard MC and MLMC needed to achieve the desired accuracy.

We will now explore the reasons of high computational efficiency in the MLMC estimation. Take the results ofε=0.06 as an example, Figure 24(a) plots the time for one sample run and the required number of samples at all six levels. The figure shows that as the level increases and the grid resolution gets finer, the required time for one sample run increases (the red solid line), but in the meantime, the required number of samples decreases (the dashed blue line). This suggests that MLMC puts the right effort where it is needed, i.e., most of the model executions are conducted on the computationally frugal coarse grids (lower levels) and just a few on the computationally expensive fine grids (higher levels). In this way, the total computational time is significantly less compared to the standard MC that spends all its effort on the computationally most expensive finest grid. With less computational time, MLMC can achieve asimilar estimation as the standard MC because it considers the estimates on a sequence of grids so that the less accurate estimates on the preceding coarser grids can be sequentially corrected by estimates on the following finer grids.Figure 24(b) illustrates how the MLMC estimate gets closer to the standard MC estimate as more levels are included.

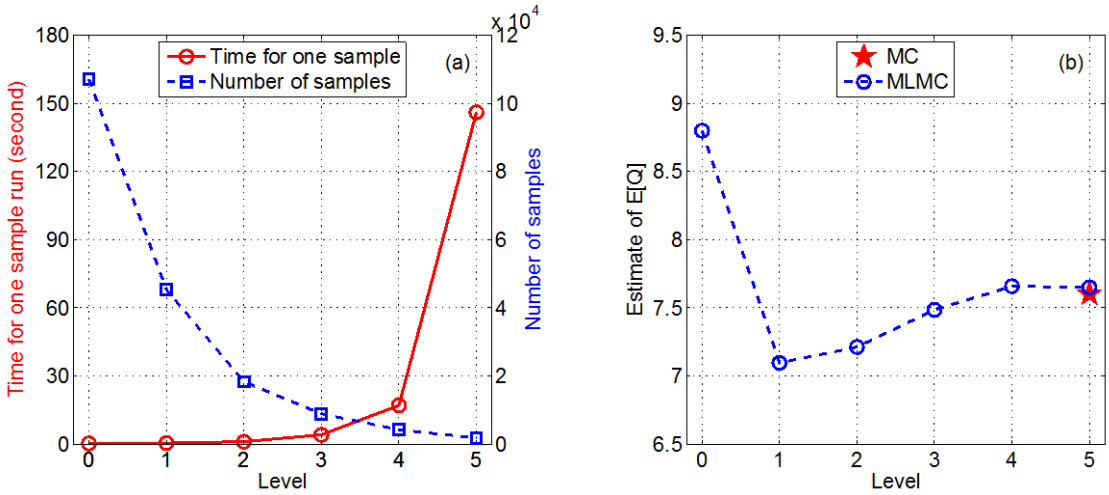

Figure 24 (a) Time for one sample run (red solid line corresponds to left y-axis) and the required number of samples (blue dashed line corresponds to right y-axis) at the six levels of MLMC; (b) MLMC estimates of E[Q] considering different numbers of levels, and the standard MC estimate of E[Q].

In the estimation of the CDF, we worked on the log(Q) because the samples of $Q$are distributed in a wide range and a relatively large portion is close to zero. To evaluate accuracy of the two MC estimators, we use the result of 100,000 samples simulated on the finest mesh as a reference. The reference CDF is shown in Figure 25. In the estimation, we first estimate the values of $F(q_n)$ at the 20 points uniformly distributed in the range [-3, 5], and then we approximate the CDF using piecewise linear Lagrange interpolation.

Figure 25 indicates that the estimation of both the standard MC and MLMC are very close to the reference CDF. However, to achieve the same accuracyε, the computational costs of the two MC methods are dramatically different as shown inFigure 26. Figure 26(b) shows that the computational time of the MLMC based on the δ evaluated in this study is much smaller than that of the standard MC,providing significant cost savings, but the MLMC with the δ calculated in Giles et al. (2015) is actually inferior to the standard MC. The reason is that our δ causes faster variance decay than that of Giles et al. (2015), as shown in Figure 26(a). This suggests that δ is vitally important in the application of MLMC and our δ can give great results.

To sum up, this section presents a MLMC method in estimation of expectation and CDF of QoI. We compare its computational cost with the standard MC mathematically and analyze the possible factors affecting the MLMC efficiency. To improve the efficiency of the MLMC in CDF estimation, we propose using a smoothing function with an optimally determined smoothing factor. The oil reservoir simulation demonstrates that the MLMC performs better than the standard MC with significant cost savings and our smoothing factor causes better results of MLMC than that determined in current literature. The MLMC method can be applied to high-dimensional UQ problems to improve the computational efficiency.

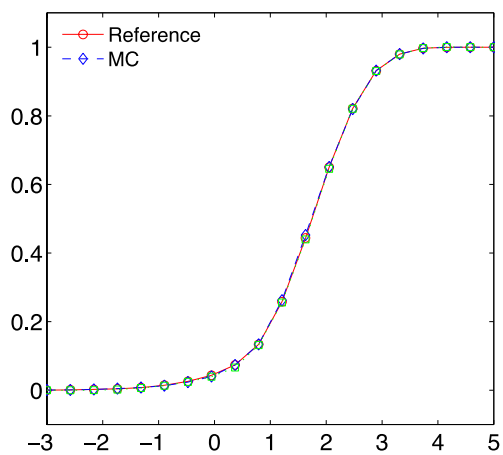

Figure 25The reference CDF and the estimated CDF based on the standard MC and the MLMC methods.

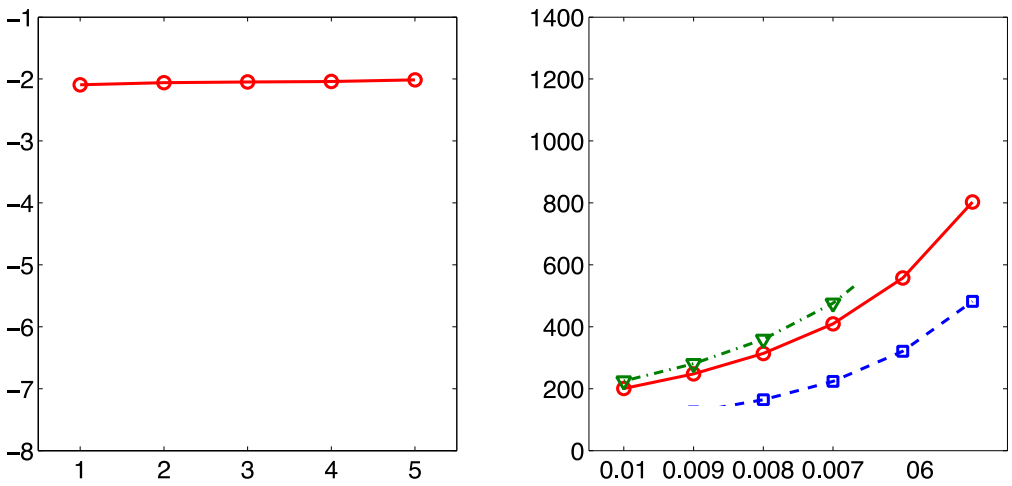

Figure 26 (a) The variance decay of the indicator function, the smoothing function calculated based on the δ determined in this study (δ) and that calculated in Giles et al. (2015) (δ_G); (b) computational time of the standard MC and the MLMC with δ and δ_G.

4. PARALLEL PERFORMANCE

4.1 TH "SMALL" SCALING AND GRAPHIC VISUALIZATION

In order to get an idea for how well PFLOTRAN uses a node of a super computer, we first analyzed the strong scaling of TH mode on a workstation with 12 cores. While different from multiphase flow, the test simulation helps us understand what performance we can expect. Also, the simulation is relevant as some fracking simulations are done using these equations.

We solve a simple problem on a range of discretizations in 3D where the number of cells varies from 20^3 to 50^3. In FigureFigure 27 Parallel efficiency of a strong scaling study of a simple TH problem we plot the parallel efficiency of each of the runs. In place of specific numbers that do not mean much, we opt to plot regions that reflect where the scaling is

- good (green, efficiency > 75%)
- acceptable (yellow, 50% < efficiency < 75%)
- poor (red, efficiency < 50%)

This means that on a 12 core node, PFLOTRAN only effectively makes use of less than half its cores. Unfortunately, computing centers will charge users for all cores on a node and thus there is no incentive to efficiently use a single node.

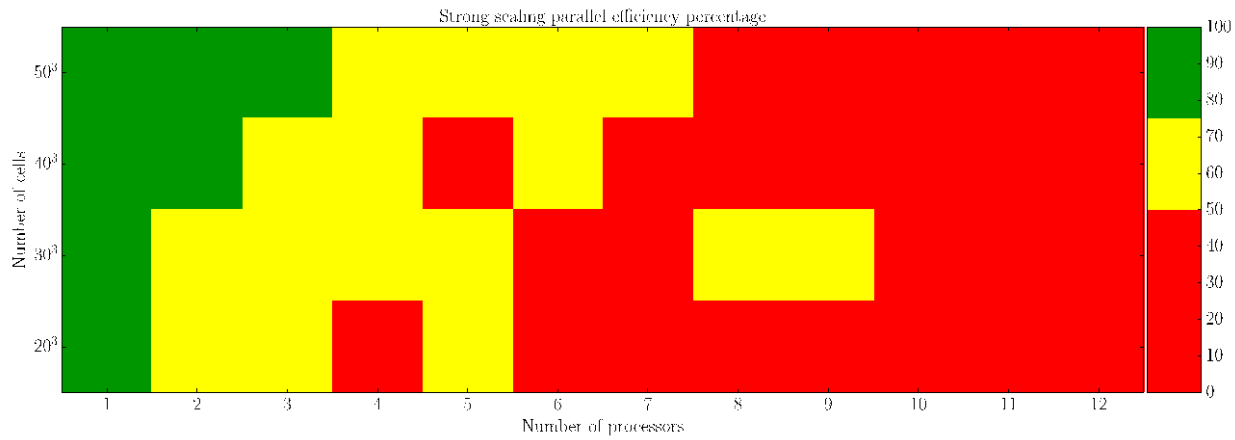

Figure 27 Parallel efficiency of a strong scaling study of a simple TH problem

4.2 CO_2/WATER TEST CASE

4.2.1 Problem setup

The computational domain chosen for the test case is a layer 200 m high, which extends ~ 10 km in both horizontal directions, at a depth of 2000 m (P~200 bar). An injection well is located in the centerof the domain and extends from a depth of 2100 m to 2140 m. A constant injection rate of 0.5 Mton/year of CO_2 is uniformly distributed between the injection cells falling within the well vertical extension, and the injection is carried out for 10 years. The simulation is then continued for additional 40 years following the injection stop to study the dissolution effects. The total simulation time is 50 years.

Due to the problem's symmetry, only a quarter of its domain is considered, including two symmetry boundaries that cross the vertical well. The other two vertical boundaries, parallel to the symmetry planes are treated as open imposing a pressure hydrostatic boundary condition, while the top and bottom surfaces are considered to be impermeable (perfect seals). Initial conditions: fully water saturated media, hydrostatic pressure, initial temperature assigned imposing 50 ^{0}C at a depth of 2000m and a thermal gradient of 0.025 ^{0}C/m, zero CO_2 concentration in water. Only pure water will be considered.

Constant porosity and saturated permeability in the layer, with the following data:

```
POROSITY 0.12d0
PERM_X 1.d-12  (m²)
PERM_Y 1.d-12  (m²)
PERM_Z 1.d-13  (m²)
```

Constant rock properties

```
ROCK_DENSITY 2650   (kg/m³)
```

Constant rock properties (assuming thermal equilibrium):

```
SPECIFIC_HEAT 1000          (J/Kg/K)
THERMAL_CONDUCTIVITY 0.5    (W/m/K)
```

Capillary pressure: function (Van Genutchen)
Relative permeability functions: Modified Brook and Corey relations

For the computational grid, two discretizations have been chosen,

1. dx= dy=100m, dz=10m, 150k DOF, to allow for a serial computing test
2. dx= dy=50m, dz=2m, 3M DOF, parallel test, min DOF per core ~ 3000 (assuming a max of 1024 cores)

The mesh sizes are chosen to push the code to work with a minimum number of DOF per core beyond the recommended value (10k DOF). This should allow to test in full the MPHASE parallel performance, up to the point where no benefits are found by increasing the number of cores, because the excessive communication between computing nodes hampers the computational time saving given by the domain decomposition.

In the parallel test, the number of computational cores are increased following a power-2 law (16, 32, 64, 128, 256, 512, 1024).

4.2.2 Sequential run results – computational grid 1

Physical Solution

In the first case adopted in this comparison, an isotropic saturated permeability is assigned to the only homogeneous material (kx=ky=kz=10^{-13} m^2). There are not notable differences between the PFLOTRAN and the TOUGH-ECO2N solutions, for this reason only one of the two solutions (TOUGH-ECO2N) is reported. Below the 3d contour of the gas saturations are shown, for two times: t=10 years (end of the injection), and t=50 years (end of the simulation).

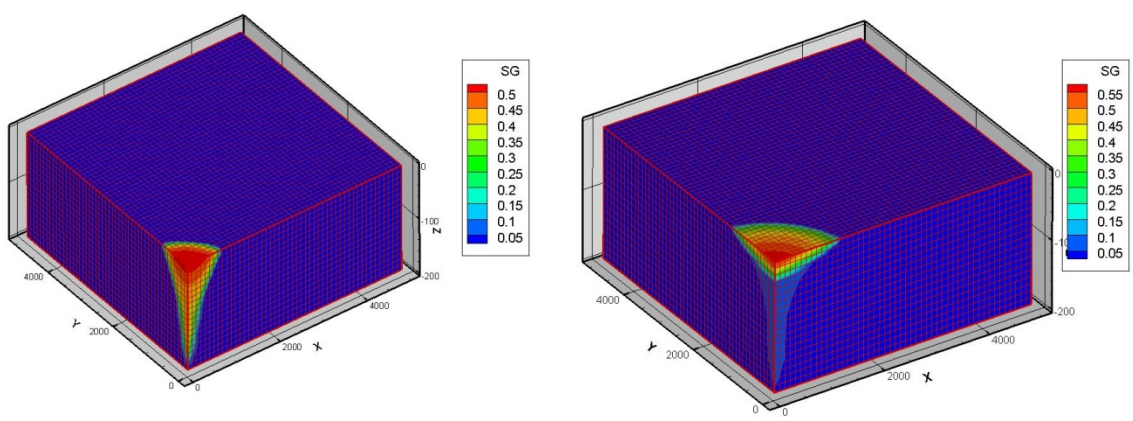

Figure 28 Isotropic case, 3D sg controur: Left, t=10 years; right, t=50 years

As expected, the CO_2 rises fast towards the top boundaries, during and after the injection. Only a small part of the domain sees the spread of the circular CO_2 plume, however a much greater area is expected to be affected when refining the vertical discretization from dz=10m to dz=2m.

Computational performance – comparison PFLOTRAN/TOUGH-ECO2N

For the number of equations solved and the corresponding computational effort, the same considerations reported in the PFLOTRAN/TOUGH-EWASG comparison can be repeated. The same initial and maximum time step sizes have been used in the two software (3.17 10^{-8} y, and 0.1 y respectively).

Instead, two different tolerance values have been adopted for the convergence of the Newton-Raphson solution algorithm, 10^{-5} for TOUGH-ECO2N (default) and 10^{-6} for PFLOTRAN-MPHASE. In both cases, solutions without notable numerical noise have been achieved. It is worth observing that using the same tolerance value would not guarantee the same solution procedure in terms of number of non-linear iterations and time steps. The reason for this is that different convergence criteria are implemented in the two software. While both software check the residual infinite norm against the tolerance, the residual is scaled differently: in TOUGH-ECO2N the residual is divided by the accumulation term (only when this is larger than 1), instead in PFLOTRAN-MPHASE the residual is divided by the cell volume (and multiplied by the time step if this is smaller than 1).
The software default tolerances have been adopted for the linear solvers (TOUGH-ECO2N rtol=10^{-5}; PFLOTRAN, rtol=10^{-5}).

Keeping in mind these differences, the following execution times have been recorded compiling both software in optimization-for-speed mode (-O0 option), and running the simulation in the same machine (Intel® Core™ i7-4770K CPU @ 3.50GHz × 8, 16 Gib of memory):

Table 2 Execution time TOUGH-ECO2N / PFLOTRAN-MPHASE comparison time in minutes

	TOUGH-ECO2N	PFLOTRAN str. grid	PFLOTRAN unstr. grid
Injection	32.13	5.60	5.24
Post-injection	55.46	13.50	13.78

Table 3 Execution time comparison PFLOTRAN-MPHASE in sequential and using 4 cores, time in minutes. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, np number of mpi-core processes

	PFL sg – np =1	PFL sg – np = 4	PFL ug - np = 1	PFL ug - np =4
Injection	5.60	3.41	5.24	3.45
Post-injection	13.50	8.14	12.45	8.08

Each simulation has been performed by two separate runs using restart files, since some issues have been encountered when defining a step change for the injection mass rate in TOUGH-ECO2N (from the value imposed during the injection, to zero in the post injection phase). The first run simulates only the injection (t=1, 10 years), while the second starts when the injection is over and continues until the end of the simulation (t=10, 50 years).

For the injection simulation, PFLOTRAN is nearly 6 times faster, while the speed up is reduced to x4 for the post-injection. To better analyze this behavior the number of time steps and iterations of each simulation are reported in the tables below.

Table 4 Number of time steps comparison. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, np number of mpi-core processes

	TOUGH-ECO2N	PFL sg - np =1	PFL sg - np =4	PFL ug - np =1	PFL ug - np =4
Injection	155	165	165	163	163
Post-injection	440	431	431	430	430

Table 5 Number of iteration comparison. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, np number of mpi-core processes.

	TOUGH-ECO2N	PFL sg - np =1	PFL sg - np =4	PFL ug - np =1	PFL ug - np =4
Injection	856	181	181	180	180
Post-injection	902	431	431	430	430

In Table 2 to Table 5, “PFL sg” and “PFL ug” refers to runs performed by PFLOTRAN, using the structured and the explicit unstructured grid formats respectively. Although the computational grids used in these tests are structured, TOUGH-ECO2N considers any mesh as being fully unstructured. Since PFLOTRAN can work with both structured and unstructured grids, both formats have been tested to have a meaningful comparison with TOUGH-ECO2N, and to assess the PFLOTRAN unstructured grid implementation at the same time. In the results shown above, some preliminary parallel computing tests have been carried out for PFLOTRAN using the same workstation adopted for the sequential run (np indicated the number of core-mpi processes used).

To carry out a fair comparison, another PFLOTRAN parameter was tuned to get more or less the same number of time steps, this is the maximum change allowed in the gas saturation, which, for this problem, controls the time step size. As it can be seen from the table above, the number of time steps needed to complete a simulation is very similar for all cases simulated. Each time step occurs in 2-3 seconds, and the computational time recorded would not differ significantly if a perfect match in the number of time steps had be achieved.

For the injection simulation, for which the improvement in performance brought by PFLOTRAN is greater, TOUGH-ECO2N reduces 5 times the time step for missed convergence. This never occurs in PFLOTRAN. In the relaxation simulation, the maximum time step imposed seems rather small, and causes the PFLOTRAN solver to converge at the first Newton iteration for the entire run. The same doesn’t happen in TOUGH where in several time steps, up to 4-5 Newton iterations are executed before advancing the time step. For this case, TOUGH-ECO2N needs a larger number of nonlinear iteration to converge, resulting in substantially longer simulation times. Several parameters have been tested in the attempt to reduce the number of Newton iterations in TOUGH-ECO2N: loosening the tolerances for the non-linear iterative algorithm (from 10^{-5} to 10^{-4}) and for the linear solver (rtol from 10^{-6} to 10^{-5}), and switching between the different linear solvers available in TOUGH. However, none of the tests has brought a significant improvement.

The saving in simulation time observed when using PFLOTRAN, is mainly due to a more robust solver able to converge with a smaller number of non-linear iterations. However, this reason alone is not sufficient to explain the better computational efficiency, because the speed up factor does not correspond to the reduction in number of Newton iterations:

- injection: speed up factor, 32.13/5.6 = 5.7; Newton iteration reduction, 856/181 = 4.7;
- post-injection: speed up factor, 55.46/13.50 = 4.1; Newton iteration reduction, 902 / 431 = 2.09

Although the matrix size solved by the two software is the same, the non-linear system being solved differs for one of three equations (Energy for PFLOTRAN, salt transport for TOUGH-ECO2N). The effects on the computational efficiency due to this difference have not been investigated. In addition, the two software use different libraries for the solution of the linear solvers (PETSc for PFLOTRAN, internal implementations for TOUGH2-V2, Aztec parallel linear equation solver for TOUGH2-MP). The efficiency of these linear solver packages, which has not been investigated separately from the PFLOTRAN and TOUGH implementation, is also expected to have an important impact.

Finally, before moving to the parallel computing analysis, another sequential case was run introducing anisotropic values for the media permeability: kx=ky=10^{-12}, kz=10^{-13}. Below the 3d contour of the gas saturations are shown, for two times: t=10 years (end of the injection), and t=50 years (end of the simulation), obtained by PFL ug. The results obtained with PFL sg and TOUCH-ECO2N are nearly identical.

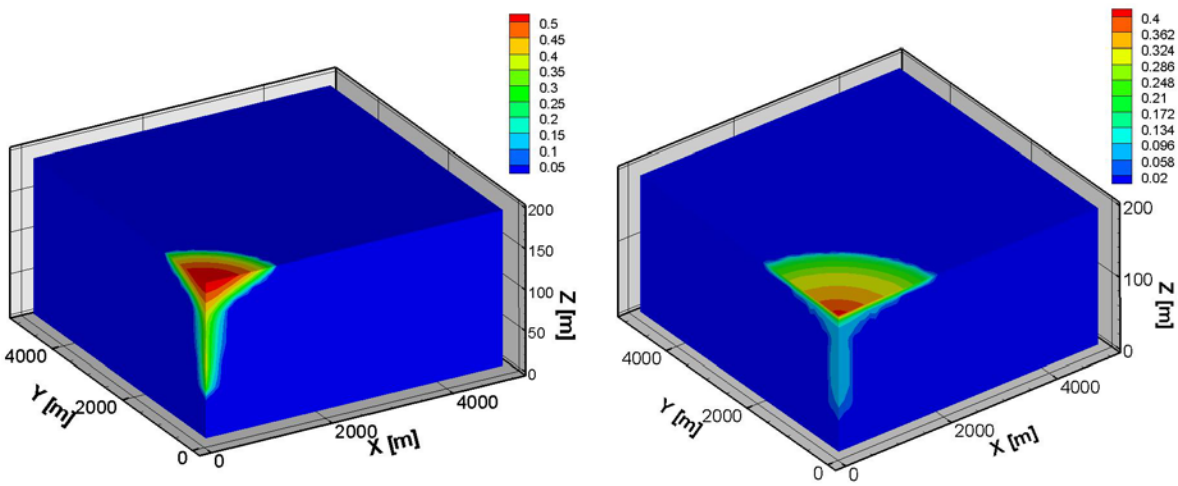

Figure 29 Anisotropic case, 3D sg contour: Left, t=10 years; right, t=50 years

As expected, the plume is more horizontally spread compared to the isotropic case, and slightly smaller computational times are needed to complete the runs for both PFLOTRAN and TOUGH-ECO2N. The differences in the software performance remain of the same magnitude.

4.2.3 Parallel run results – computational grid 2

Physical Solution

The problem used for the parallel computing tests is assigned a homogeneous and anisotropic material (kx=ky= 10^{-13} m^2, kz=10^{-12} m^2). The simulations are performed only for the injection period (t=10 years), which is enough to test the two software scaling capacities, avoiding unnecessary long computational times. Below, the 3d contour of the gas saturations are shown for t=10 years (end of the injection), for the coarse (grid 1) and refined (grid 2) meshes. The results are obtained by PFL-ug only, since PFL-sg and TOUCH-ECO2N produce nearly identical results.

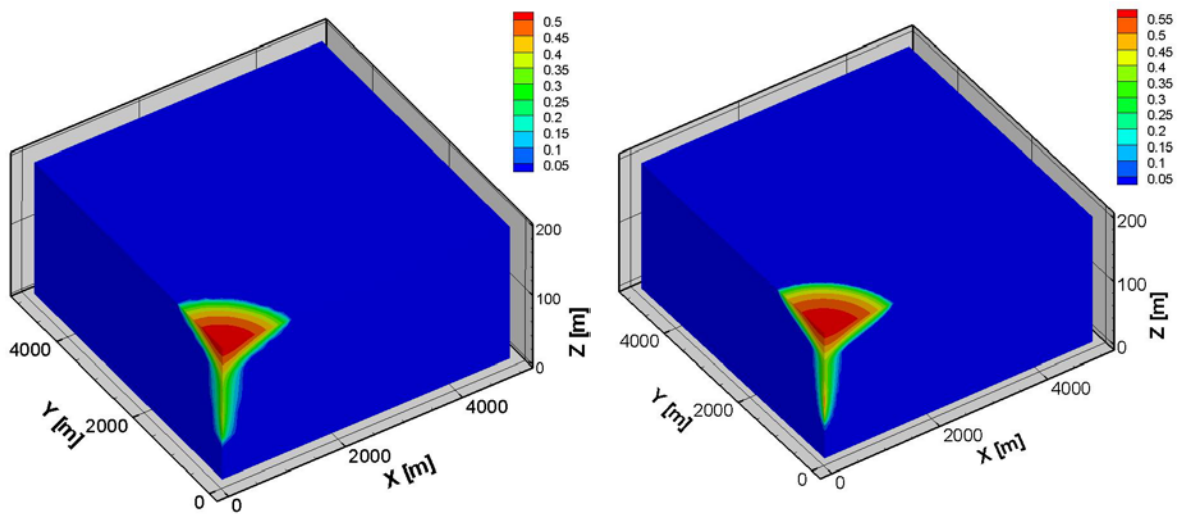

Figure 30 Anisotropic case, t=10 years, 3D sg contour: Left, grid 1 (50 k cells), right grid 2 (1M cells)

As expected, the finer grid predicts a wider horizontal spread of the CO_2 plume (about 100m more in radius).

Computational performance – comparison PFLOTRAN/TOUGH-ECO2N

The same numerical parameters adopted in the sequential runs have been employed for the parallel tests (initial and maximum time steps, and solver tolerances). To assess the best overall performance of the two software, no attempt was done to match the same number of time steps, and the PFLOTRAN time stepping was controlled by only imposing a maximum gas saturation change of 0.05, which should guarantee a good compromise between stability and computing performance. For TOUGH-ECO2N default parameters have been used.

The initial idea was to increase the number of cores using a power of 2 law (16, 32, 64, 128, 256, 512, 1024). However, since the Titan queuing policy imposes a maximum simulation time of 2h for job up to 1984 cores, several slower simulations (i.e. np < 128) had to be cut from the test. The situation was even more dramatic for TOUGH-ECO2N, which could not complete the 10 years simulation even using 1024 cores. For a comparison with PFLOTRAN, the simulation times for the first year of injection (t=1 year) were considered, and one run was executed using 2048 processor cores to complete the 10 year injection.
The number of MPI processes coincides with the number of cores, which in turn coincide with the number of computational sub-domains, for both TOUGH-ECO2N and PFLOTRAN. It is also worth observing that increasing the number of processors with a power-2 law fits well the Titan computing architecture, occupying at each run all the cores available in a computing node. In fact, each Titan's node is made of two Numa nodes, each featuring 8 cores, for a total of 16 cores per node.

Below are the tables that report the computational time, the number of time steps and the number of non-linear (Newton) and linear iterations for all the simulations run.

Simulation time up to 1 year

Table 6 Simulation up to 1 year - execution time in minutes. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

np	128	256	512	1024	2048

PFL - sg	26	13	6	4	-
PFL - ug	21	11	6	4	-
ECO2N	-	-	68	47	35

Table 7 Simulation up to 1 year – number of time steps. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

Np	128	256	512	1024	2048
PFL - sg	165	166	167	169	-
PFL - ug	165	174	167	166	-
ECO2N	-	-	229	229	229

Table 8 Simulation up to 1 year – number of Newton iterations. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

Np	128	256	512	1024	2048
PFL – sg	415	419	417	424	
PFL - ug	410	428	412	414	
ECO2N	-	-	-	-	

Table 9 Simulation up to 1 year – number of linear iterations. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

Np	128	256	512	1024	2048
PFL – sg	107360	106869	106260	140480	-
PFL - ug	82789	91869	94897	102550	-
ECO2N	-	-	-	-	-

Table 10 Simulation up to 1 year – number of time step cuts. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

Np	128	256	512	1024	2048
PFL – sg	0	0	0	0	-
PFL – ug	0	0	0	0	-
ECO2N	-	-	5	5	5

Simulation time up to 10 years

Table 11 Simulation up to 10 years - execution time in minutes. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

Np	128	256	512	1024	2048
PFL – sg	84	43	18	12	-
PFL – ug	67	37	19	13	-
ECO2N	-	-	-	-	283

Table 12 Simulation up to 10 years – number of time steps. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

Np	128	256	512	1024	2048
PFL – sg	459	462	455	460	-
PFL – ug	449	455	452	461	-
ECO2N	-	-	-	-	1414

Table 13 Simulation up to 10 years – number of Newton iterations. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

Np	128	256	512	1024	2048
PFL – sg	1281	1294	1264	1281	-
PFL – ug	1241	1238	1249	1277	-
ECO2N	-	-	-	-	7073

Table 14 Simulation up to 10 years – number of linear iterations. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

Np	128	256	512	1024	2048
PFL – sg	352495	354541	345463	454014	-
PFL – ug	265420	280868	305841	331114	-
ECO2N	-	-	-	-	2014662

Table 15 Simulation up to 10 years – number of time step cuts. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

Np	128	256	512	1024	2048
PFL – sg	4	2	3	4	-
PFL – ug	5	1	4	3	-
ECO2N	-	-	-	-	28

The dashes indicate that data was not available, either because the simulation was not run, or because the data are not readily available in the output files.

In general, TOUGH-ECO2N requires a much larger number of time steps, non-linear and linear iterations. However, in the comparison between the two software, it is clear at first glance that ECO2N requires many more non-linear and linear iterations to reach the same physical time. In turn, the computational time, when using the same number of cores, is roughly an order of magnitude higher.

It must be said that in the tables above only the wall clock times are reported, this includes setting up time, and writing of the output files. This time should be properly assessed, however, it is not expected to play an important role in the comparison between the two software. In fact, only 3 text output files are printed during the simulations, and the setting up time should be only a small fraction of the simulation time. For example, it takes about 50 seconds for PFLOTRAN to write a single output file.

With the current set up, and for the cases presented in this report, PFLOTRAN is much faster than TOUGH-ECO2N, with nearly linear scaling up to 512 cores, while increasing np to 1024 is still speeding

up the simulation, but with a factor that is not any longer linear. This is in somehow expected because for np=1024, only 3k DOF per MPI-process (i.e. per core) are available, and the communication time becomes important. This is true also for TOUGH-ECO2N, however its scalability for this test case could not be tested for lower np numbers, due to the Titan 2h-limit policy. The problem could be avoided by running smaller test problems.

Finally, it must be observed that PFLOTRAN shows different number of time steps depending on the domain decomposition (ie the number of CPUs used). This should not happen, because domain-decomposition theory guarantees the same number of time steps regardless of the number of sub-domains. Thus, this phenomena should be further investigated but was out-of-the scope of this project.

4.2.4 PFLOTRAN/TOUGH-ECO2N – detailed parallel performance analysis for computational grid 2 case

Only simulations up to 1 year have been considered, and detailed data are provided for three numerical tests, obtained increasing the number of cores: 512, 1024 and 2048. See table below.

Table 16 Simulation up to 1 year - execution time in minutes. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

np	512	1024	2048
PFL - sg	6	4	2.5
PFL - ug	6	4	3.8
ECO2N	68	47	35

Table 17 Simulation up to 1 year – number of time steps. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

Np	512	1024	2048
PFL - sg	167	169	167
PFL - ug	167	166	166
ECO2N	229	229	229

Table 18 Simulation up to 1 year – number of Newton iterations. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

Np	512	1024	2048
PFL – sg	417	424	421
PFL - ug	412	414	411
ECO2N	1084	1084	1084

Table 19 Simulation up to 1 year – number of linear iterations. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

Np	512	1024	2048
PFL – sg	106260	140480	140636
PFL - ug	94897	102550	108446
ECO2N	255873	283669	311842

Table 20Simulation up to 1 year – number of time step cuts. PFL sg= PFLOTRAN structured grid; PFL ug= PFLOTRAN unstructured grid, Np number of mpi-core processes.

Np	512	1024	2048
PFL – sg	0	0	0
PFL – ug	0	0	0
ECO2N	5	5	5

From the comparison the following differences are observed:

- ECO2N maintains the same number of Netwon and time steps as the number of cores increases, the same doesn't occur for PFLOTRAN.
- The ECO2N simulations need more than twice the Newton iterations required by PFLOTRAN.
- The ECO2N simulations need nearly three times the linear iterations required by PFL-ug (same mesh format adopted in TOUGH-ECO2N)

As mentioned previously, further analysis is needed to understand why PFLOTRAN changes the time step size, thus the number of time steps as the number of processors core (i.e. sub-domains) increases.

4.2.5 PFLOTRAN scaling test on a real reservoir problem – Sleipner CO_2 storage site

The Sleipner project is a commercial scale CO_2 capture and storage site operated by STATOIL. CO_2 is separated from the Sleipner gas condensate and is injected for permanent storage in the Utsira formation. This is a brine bearing formation shallower than the natural gas reservoir and comprises nine layers. The CO_2 is injected in a deep layer at around 1000 m below sea level (b.s.l) through a deviated well, and it slowly flows upwards to the uppermost layer (layer-9, at circa 800 b.s.l) where it is trapped by a caprock. The injection started in 1996, and the first CO_2 reached layer-9 about three years later. For more details see [12].

The parallel computing tests reported here use a simulation case that focuses on layer-9, for which the time dependent mass flow rate of CO_2 entering the layer has been taken from the literature. The computational times in the table below refers to the first 30 days of simulation, during which the CO_2 moves vertically from the injection point towards the caprock.

The computational mesh is passed to PFLOTRAN using the implicit unstructured format, and it features circa 260k cells.

Table 21PFLOTRAN performance on the Sleipner CO_2 storage site case

np	16	32	64	128	256	512	1024
# time steps	95	110	109	121	168	113	186
Wall clock mins	39.47	20	11	7	4.34	3.05	6.62
Time flow sec	2303.4	1150.3	588.9	349.6	202.5	124.3	151.5
Newton it.	280	298	263	339	278	217	310
Linear it.	94549	96547	108357	160842	166051	152696	219808
dt cuts	8	12	11	17	27	13	35

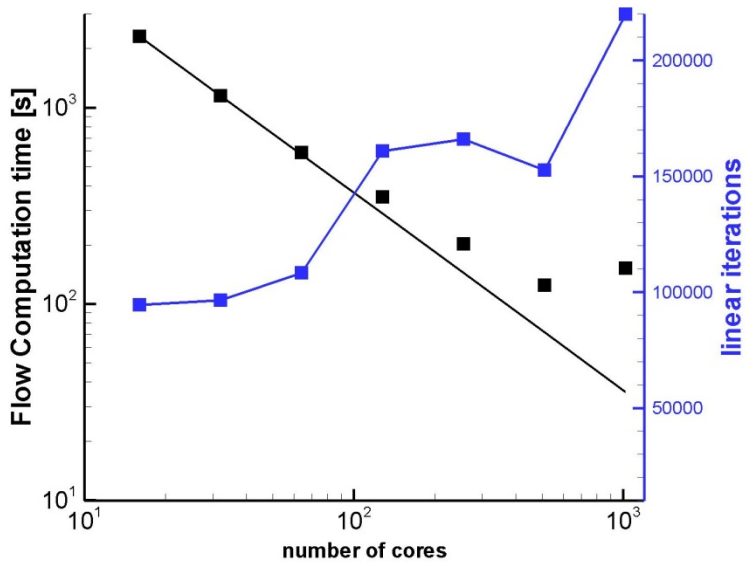

Figure 31PFLOTRAN parallel performance for the Sleipner CO_2 storage case

The change in number of time steps and Newton iterations observed in the simpler case (computational grid-2), is even more noticeable here where the model is characterized by heterogeneous rock properties and an irregular mesh.

4.3 CH_4 EXAMPLES

4.3.1 PFLOTRAN scaling test on a reservoir problem

The implementation of the CH_4 into PFLOTRAN was tested using a similar approach as that done for the CO_2 case. The properties for CH_4 that were implemented into MPHASE were obtained from NIST data and verified using the paper by Duan and Mao [8]. The computational domain for the testing of the speedup was similar to that used for the testing of CO_2. The domain is 200m high and extends 10,000m in each horizontal direction at a depth location of 2000m. Again, an injection well was placed at a depth of 2100m to 2140m with a constant injection rate of 0.4kg/s of CH_4. Because of the limits on computational time imposed on TITAN, a simulation of 2.25 years was calculated with a constant injection for the duration of the simulation. The constant injection negates the requirement of a restart of the TOUGH2-MP simulations.

As with the CO_2, the problem's symmetry was utilized requiring the simulation of only a quarter of the domain with symmetry boundaries that cross the vertical well. The remaining vertical boundaries are subsequently treated as open boundaries with a hydrostatic pressure imposed. The top and bottom boundaries are impermeable. Initially, there is no concentration of CH_4 and the media has an initial temperature of 50 ^{0}C at a depth of 2000m and a thermal gradient of 0.025 ^{0}C/m.

The constant porosity and saturated permeability properties in the layer were the same as the CO_2 case:

```
POROSITY 0.12d0
PERM_X 1.d-12  (m²)
```

```
PERM_Y 1.d-12  (m²)
PERM_Z 1.d-13  (m²)
```

The rock density was slightly changed:

```
ROCK_DENSITY 2500   (kg/m³)
```

The thermal rock properties remained the same (assuming thermal equilibrium):

```
SPECIFIC_HEAT 1000          (J/Kg/K)
THERMAL_CONDUCTIVITY 0.5    (W/m/K)
```

The capillary pressure was calculated using the van Genuchten relation and the relative permeability function used the modified Brooks-Corey relations.

The computational grid used for the speedup test utilizes the same discretization as that done for the finer grid with the CO_2 case giving approximately 3M DOF: dx=dy=50m, dz=2m.

In order to verify that PFLOTRAN and TOUGH2-MP had similar implementations of the CH_4, an initial run using a much coarser grid was evaluated for an injection of 10 years. The results of the CH_4 gas are provided in Figure 32. These similar results give confidence that the implementation is correct and the refined grid with 3M DOF was evaluated.

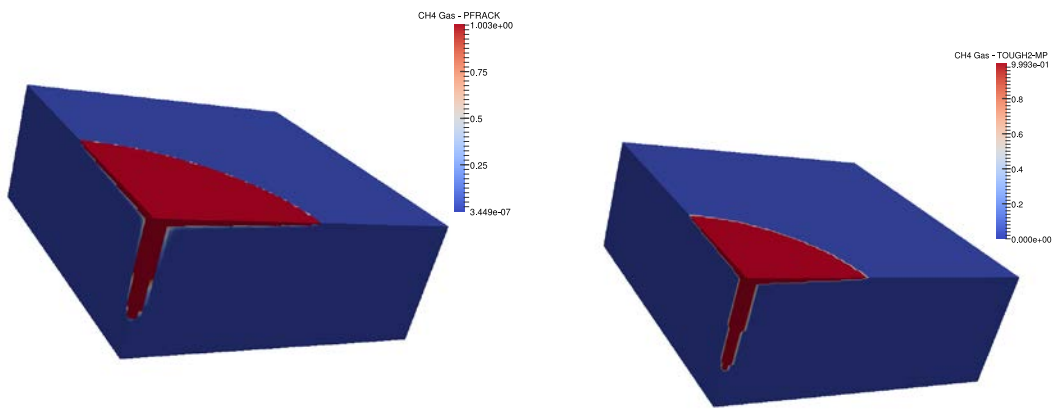

Figure 32 CH_4 gas plume for t=10 years and a coarse grid. The PFLOTRAN results are on the left and the TOUGH2-MP results are on the right.

As with the CO_2 evaluation and comparison, the number of cores used for the speedup comparison follows a power of 2 law starting with 128 cores. The simulation times for each run is presented in Table 22 and Figure 33.

Table 22 Simulation time in seconds of the finer grid for a CH_4 injection of 2.25 years.

Processors	128	256	512	1024	2048
PFLOTRAN	1287	790	311	164	146
TOUGH2-MP	6732	3431	1709	966	839

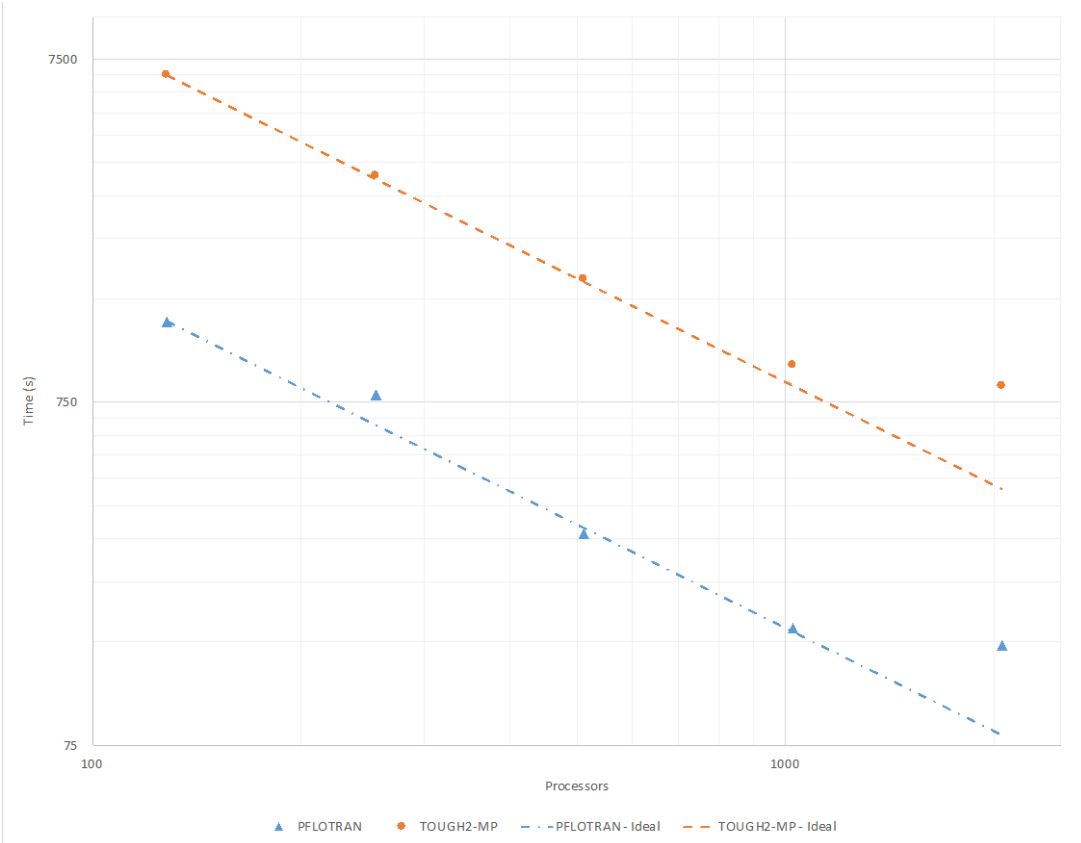

Figure 33 Parallel performance for the CH_4 injection for 2.25 years.

Both of the codes scale at a fairly similar pace and begin to deviate from the ideal speedup after 1024 cores. For the injection runs of CH_4, PFLOTRAN is approximately 6 times faster than TOUGH2-MP which is similar to the results seen with the injection of CO_2.

4.3.2 TOUGH-EWASG and PFLOTRAN model comparison on Kissinger example

The models are based on the scenario 3 reported in Kissinger et al [13]. There are still some parameters that need to be clarified to replicate the exact same case. However, in case of uncertainty, assumptions have been made to set up the PFLOTRAN and TOUGH2-EWASG models, ensuring that the same parameters are used for a fair comparison. Both PFLOTRAN and TOUGH gave very similar results as Kissinger et al[13] , but some slight differences were observed which are explained in details below.

4.3.2.1 Parameters that differs from Kissinger

In Kissinger et al. [13], a Leverett J-Function is used to scale the parameters of the Brook and Corey capillary pressure and relative permeability curves (see page 3868, equation 2). Assuming that all the layers are made of the same type of rock, the same Leverett J-Function can be assumed. With this hypothesis, the following relationship holds between two layers:

$$\frac{P_{c1}}{\sigma cos\theta}\sqrt{\frac{k_1}{\emptyset_1}} = \frac{P_{c2}}{\sigma cos\theta}\sqrt{\frac{k_2}{\emptyset_2}} \quad \rightarrow \quad P_{c1}\sqrt{\frac{k_1}{\emptyset_1}} = P_{c2}\sqrt{\frac{k_2}{\emptyset_2}}$$

This equation can be used, for example, to compute the critical capillary pressure (P_{cc}), when the media is fully saturated. However, in [13], the following questions remained unanswered:

- Which is the reference layer to be used for k and φ? And which critical capillary pressure? The 1800 Pa indicated in the paper?
- Is there any scaling for the λ parameter?

For now the same critical capillary pressure is assumed for all layers, with a value of 19.6 kPa (note that this is an order of magnitude larger than the 1800 Pa indicated in [13] which seems to be a typo)

Different capillary pressure and relative permeability curves have also been used: Van Genutchen has been used for the capillary pressure, and a modified version of the original Brook and Corey relations have been adopted for the relative permeability of brine and gas. The equations used are those described in [14]. These functions have been chosen to facilitate the comparison between PFLOTRAN and TOUGH, because these are the only capillary pressure and relative permeability functions implemented both in PFLOTRAN and TOUGH.

Although all these parameters can affect the solution, the only one with a significant impact for this particular problem is likely to be the critical capillary pressure P_{cc}.

4.3.2.2 Boundary, initial conditions, and computational mesh

The initial condition is generated assuming hydrostatic pressure before the CH_4 leakage occurs. In general, an equilibrium run is required to generate the initial condition. An initial temperature that stays constant during the simulation is assigned, imposing T=15 °C at the ground surface (top boundary), and a geothermal gradient equal to 0.03°C/m.

The symmetry conditions, required to reduce the domain to a quarter of the original, are satisfied imposing no flux conditions. These are the default boundaries in both PFLOTRAN and TOUGH. No flux conditions are used also for the bottom and top boundaries. For the two sides opposite to the symmetry planes, hydrostatic pressure boundary conditions are imposed.

Note on the hydrostatic boundary conditions

In PFLOTRAN, Dirichlet conditions can be applied directly from the input file, thanks to an internal implementation that transforms the fixed boundary values in fluxes, using a ghost cell technique. This functionality is not available in TOUGH2, where Dirichlet boundary conditions are applied indirectly adding a layer of infinite volume cells to the boundary (see TOUGH2 manual). The infinite volume freezes the variable initial values of the boundary cells, which act as ghost cells manually added by the users. The hydrostatic boundary condition is effectively a Dirichet boundary condition for the pressure, and in TOUGH2 can be applied with the following procedure:

- Including a layer of cells parallel to the boundary where the hydrostatic pressure boundary condition must be applied
- Generate an initial condition by a gravity equilibration run, after removing any sink/source from the domain
- Assign infinite volume to the additional cells forming the layer parallel to the boundary

In PFLOTRAN, the default method to apply hydrostatic boundary conditions uses its automatic functionality to create and handle ghost cells, with imposed pressure values that can be adjusted at each time step to account for the changes of the boundary cell variables (e.g. brine density, etc).

Although both approaches represent physically an open boundary that allows fluids to flow out the computational domain, they can behave differently from a numerical point of view. Since the objective of the comparison between TOUGH2-EWASG and PFLOTRAN presented here is the PFLOTRAN validation, the PFLOTRAN simulations have been carried out using the same TOUGH2 approach to the hydrostatic boundary condition. This was possible using the PFLOTRAN explicit unstructured mesh format.

The computational grid is very similar to the one described in the Kissinger paper.

4.3.2.3 Mathematical models

Numerical method
The numerical formulation for the solution of the multiphase systems is identical for the two software, which both adopt a control volume discretization, and a variable switch technique combined to a Newton Raphson solution algorithm. No dispersion effects are modeled, and the diffusivity present in the results is entirely artificial and due to discretization errors.

Thermal effects
Although available in TOUGH2-EWASG, no energy equation is solved, and the temperature imposed as initial condition is used throughout the simulation to compute fluid properties as done in[13]. On the other hand, PFLOTRAN solves an energy equation as default, and turning this off would require some programming effort: analyze the MPHASE code to be sure the temperature initial values are taken to compute fluid properties. Unfortunately, it seems there is a bit more to do than turning off a flag in the initialization. The TOUGH2-EWASG conditions are replicated imposing a methane injection temperature equal to that of the injection layer. The Joule-Thompson effect is not modeled because the gas enthalpy is computed assuming the methane as an ideal gas, while the heat of dissolution is negligible due to the low solubility of methane in water. As a result, the temperature field remains nearly identical to the initial condition. Finally, from a computational point of view, the additional effort required for the solution of the energy equation in PFLOTRAN, compensates the computational effort required for the solution of the salt transport equation in TOUGH-EWASG, which cannot be switched off (see below).

Salinity effects
In [13], a constant salt concentration is imposed in the aqueous phase. In PFLOTRAN, this is modeled in the same way. However, in TOUGH-EWASG the salt transport equation cannot be switched off, and the condition of constant salinity is imposed assigning an initial salt mass fraction constant in space. In fact, for the process described in this model, no salt precipitation occurs, while the salt transport is negligible. From a computational point of view, the solution of the salt transport equation taking place in TOUGH-EWASG compensates the computational load needed in PFLOTRAN to solve the energy equation.

Methane dissolution in brine
The dissolution of CH_4 in brine is described by different models in the two software:

- PFLOTRAN: [8]
- TOUGH-EWASG: Henry constant as a function of temperature, using a polynomial regression of data [15].

Both models account for the salting out effect. For the problem simulated, PFLOTRAN returns a higher CH_4 solubility.

Fluid properties computation
Both software use:

- Methane density: perfect gas law (as in [13])
- Methane viscosity: Irvine and Liley model[7]
- Pure water properties: IFC (1967)

The main difference is found in the model adopted for the computation of the brine density. PFLOTRAN uses a model by Duan [8]that takes account of salt and CH_4 concentrations, while TOUGH2-EWASG uses a model by Haas[16], with a correction for high temperature (>350 °C), which accounts only for salinity effects.

4.3.3 Results

Only simulation 1 of scenario 3 from [13] has been considered. The first validation case considers pure water, i.e. zero salt concentration. Figure 34shows the contour plots for the static pressure and brine density obtained from the TOUGH2-EWASG equilibrium run, which was stopped after a few time steps, when the solution had reached steady-state.

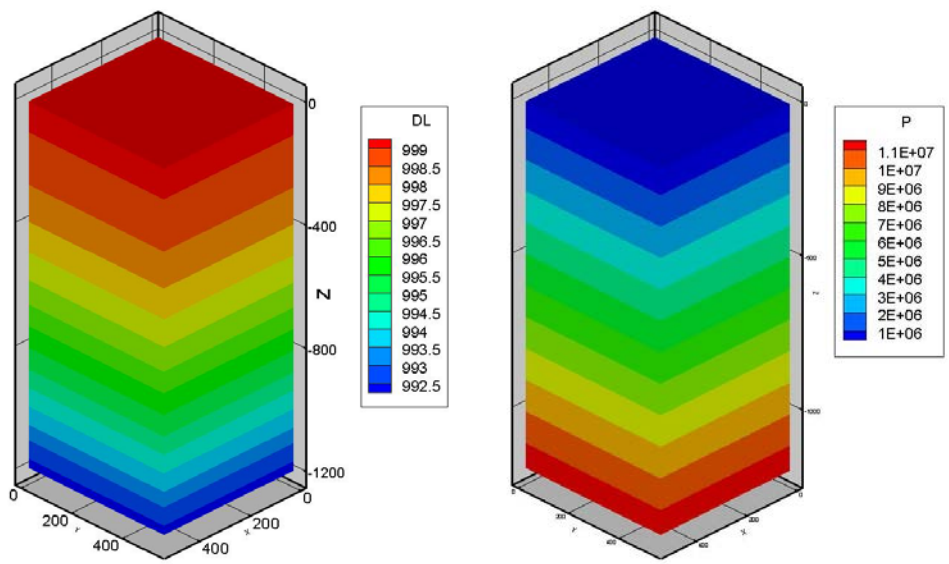

Figure 34 Contour plots for the static pressure and brine density obtained from the TOUGH2-EWASG equilibrium run

The equilibrium run has not been performed with PFLOTRAN, since its hydrostatic condition initialization functionality, is enough, at least for this simple case, to produce an initial condition very close to the one computed by the TOUGH2-EWASG equilibration run.

At the end of the CH_4 leakage from the gas reservoir, after 7.76 years, the distribution of the gas saturation, and the concentration of CH_4 dissolved in water are shown in Figure 35 and Figure 36.

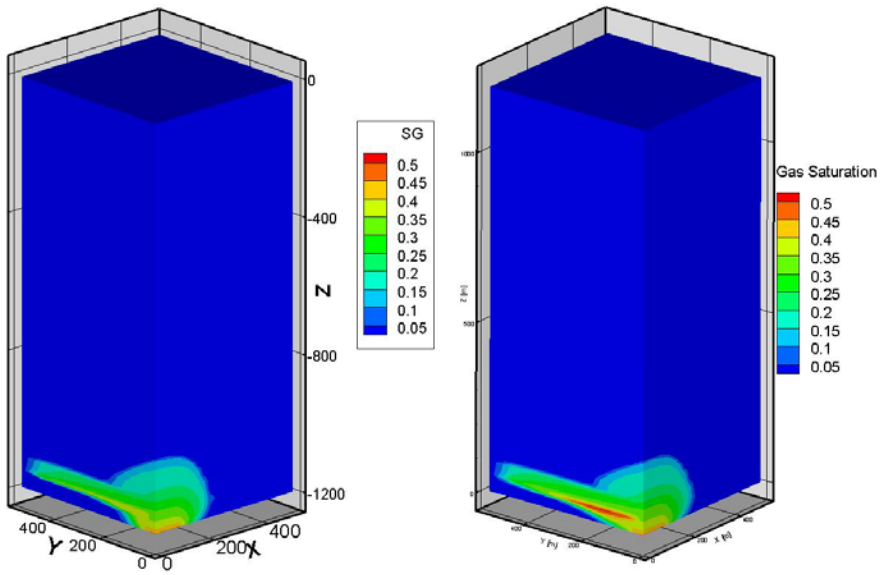

Figure 35 Gas saturation at the end of leakage – t=7.76 years: left TOUGH2-EWASG, right PFLOTRAN

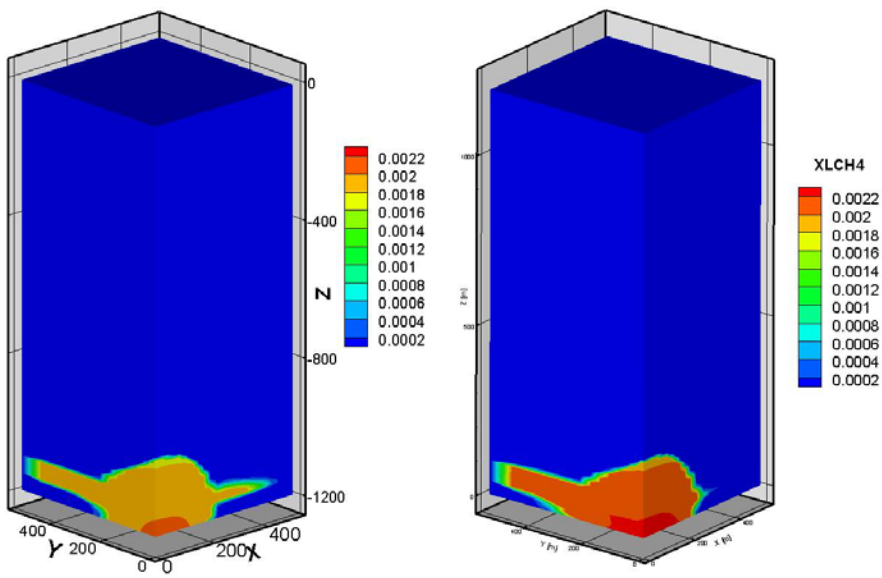

Figure 36 CH_4 liquid molar fraction at the end of leakage – t=7.76 years: left TOUGH2-EWASG, right PFLOTRAN

Although from the figures it appears that there is no CH_4 escaping the computational domain, this is not the case. In fact, the cells adjacent to the two side boundaries opposite to the symmetry corner, are characterized by an infinite ($1x10^{40}$) volume, and their gas saturation remains zero. This is better visible in the horizontal cross sections located in the first high permeable layer from the bottom (Upper Jurassic 1), where a large amount of CH_4 escapes the computational domain, see Figure 37 and Figure 38.

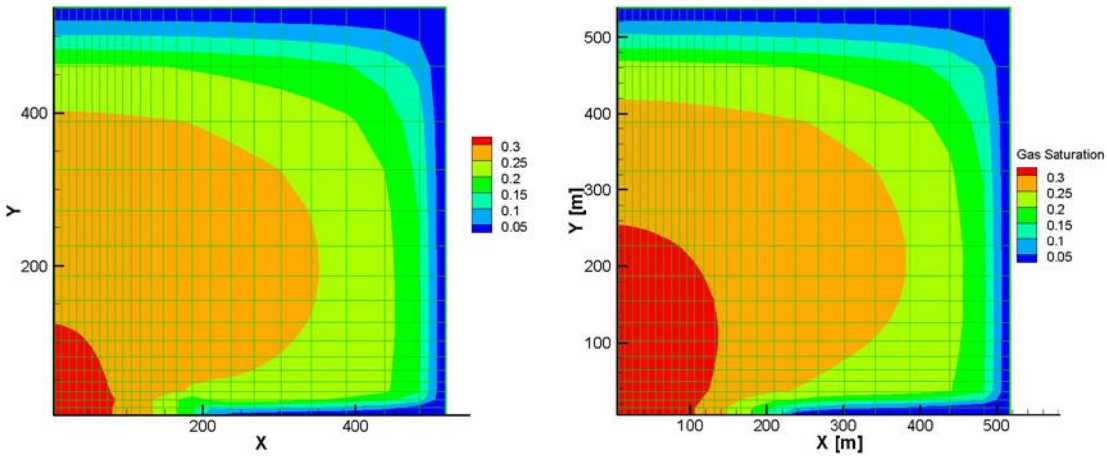

Figure 37 Gas saturation at z=-1135 (first high permeability layers from the bottom) at the end of leakage – t=7.76 years: left TOUGH2-EWASG, right PFLOTRAN

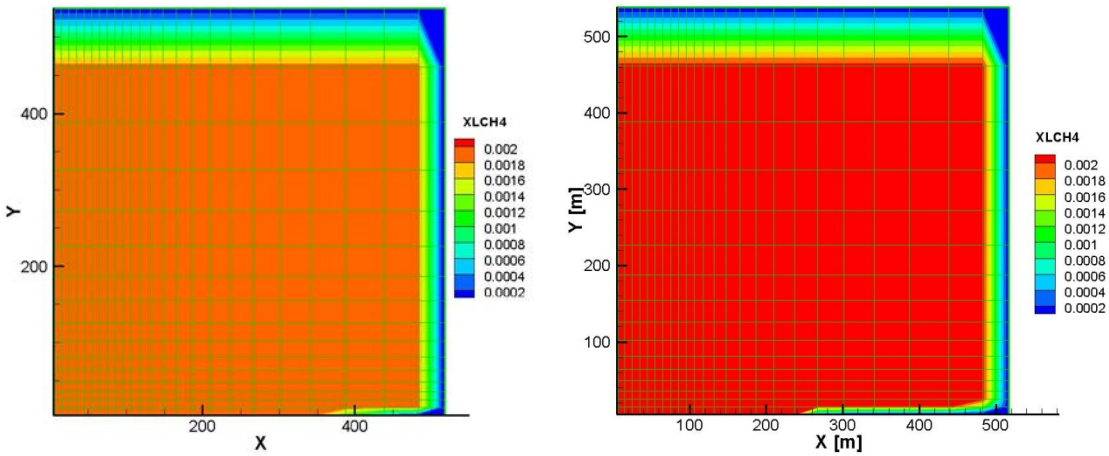

Figure 38 CH_4 liquid molar fraction at z=-1135 (first high permeability layers from the bottom) at the end of leakage – t=7.76 years: left TOUGH2-EWASG, right PFLOTRAN

PFLOTRAN shows higher values of gas saturation and CH_4 dissolution in the faulted zone but also in the high permeable layer. Pressure contours are not reported because no remarkable differences were observed. The different CH_4 dissolution behaviors can be explained by the fact that the two software use different CH_4 solubility models. One possible reason could be that the PFLOTRAN takes into account the liquid phase density dependence on dissolved CH_4. In turn, heavier brine might result in stronger buoyancy forces pushing the gas more upwards than horizontally.

The gas saturation and CH_4 molar fraction at the end of the simulation, t=100 years, are shown in Figure 39 and Figure 40.

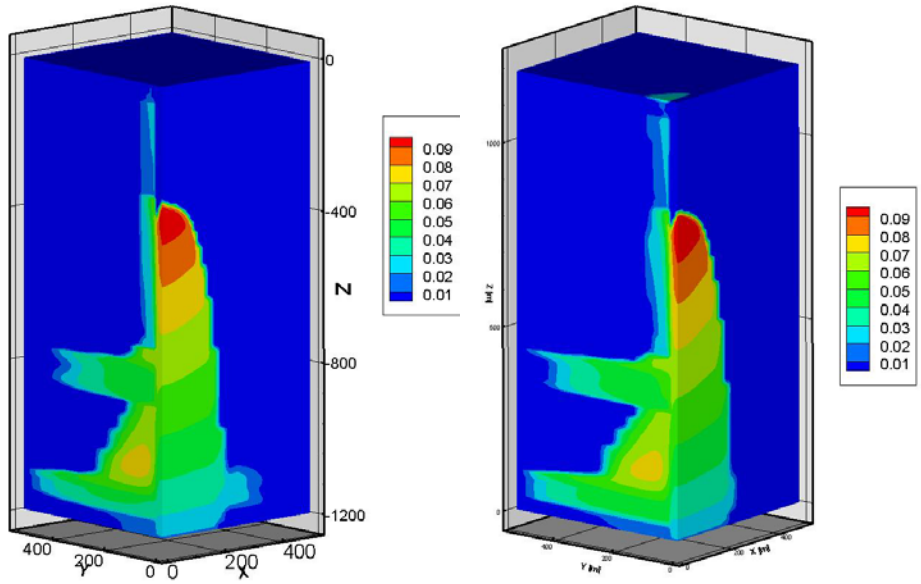

Figure 39 Gas saturation at the end of the simulation – t=100 years: left TOUGH2-EWASG, right PFLOTRAN

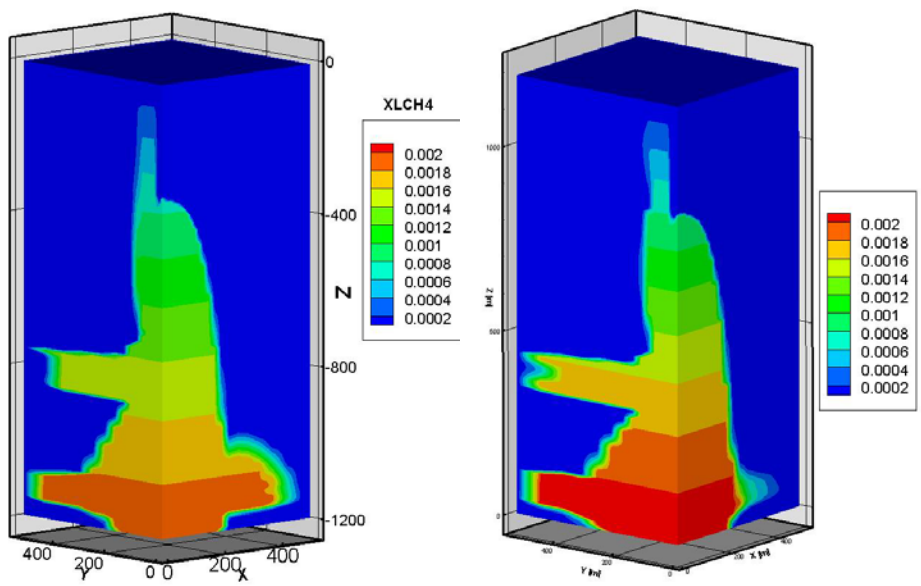

Figure 40 CH_4 liquid molar fraction at the end of the simulation – t=100 years: left TOUGH2-EWASG, right PFLOTRAN

The first notable difference is that the gas reaches the top surface only in the PFLOTRAN model, however, the amount of CH_4 reaching the ground surface is small. In this second phase of the simulation, when the gas is driven upwards only by buoyancy forces, the effect of dissolved CH_4 on the liquid phase density can be even more important, and it is probably why PFLOTRAN, which account for this process, shows higher amount of gas reaching the top surface.

The differences found between TOUGH-EWASG and PFLOTRAN for the case above (C_{NaCl}=0), magnify when a salt concentration different than zero is considered. In the next case, the same salinity taken in [13] is assigned to the brine, i.e. a salt mass fraction equal to 0.1. The equilibrium run, which is performed with PFLOTRAN and TOUGH-EWASG, is shown in Figure 41 and Figure 42.

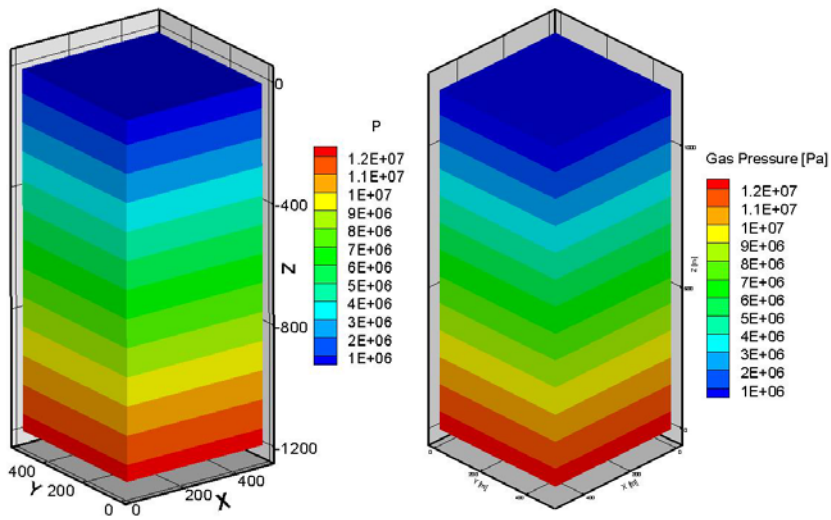

Figure 41 Pressure contours for equilibration run with an initial salt mass fraction equal to 0.1. Left TOUGH2-EWASG, right PFLOTRAN

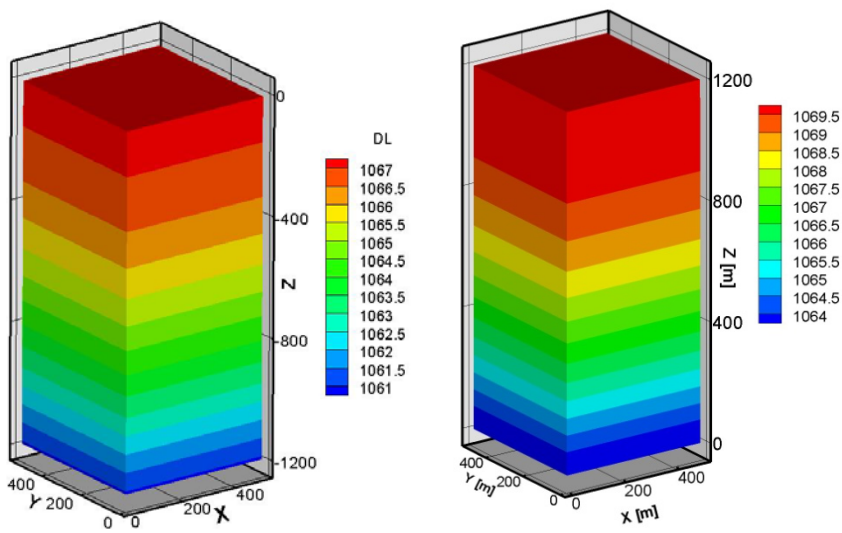

Figure 42 Liquid phase density contour for equilibration run with an initial salt mass fraction equal to 0.1. Left TOUGH2-EWASG, right PFLOTRAN

Both pressure and initial brine density contours are very similar. In this case, both models show methane leakages up to the top surface, but there is a significant difference between the amounts of CH_4 that

leaked to the top (see Figure 43 and Figure 44). TOUGH2_EWASG and PFLOTRAN give very similar results that are also very close to the results presented in [13].

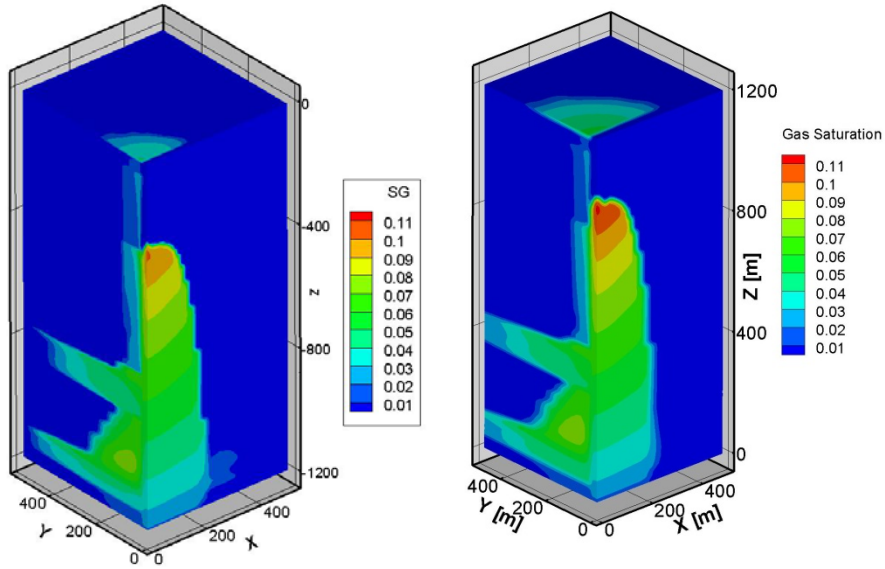

Figure 43 Gas saturation at the end of the simulation, XS=0.1 – t=100 years: left TOUGH2-EWASG, right PFLOTRAN.

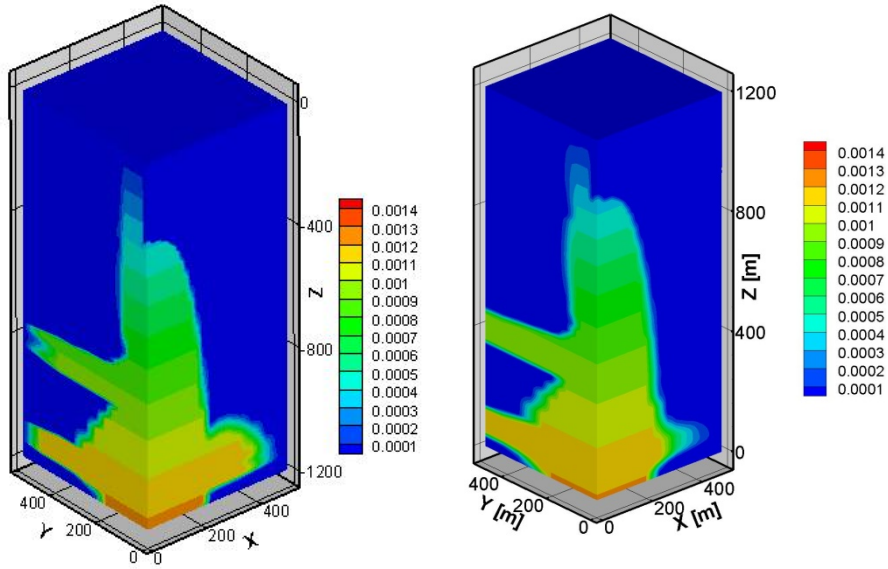

Figure 44 CH4 liquid molar fraction at the end of the simulation, XS=0.1 – t=100 years: left TOUGH2-EWASG, right PFLOTRAN.

For this last case ($C_{NaCl} \neq 0$), PFLOTRAN uses built-in hydrostatic boundary conditions, with no need of additional vertical cells layers. Therefore, the plots for the PFLOTRAN simulation show lateral escape of gas. As explained before, this occurs also for the TOUGH2-EWASG run, however the escape is hidden by the infinite volume cells used to apply the hydrostatic pressure.

5. PROPOSED FUTURE WORK

5.1 MOVING FROM MPHASE TO GENERAL PHASE IN PFLOTRAN

All the implementations were done for the MPHASE mode in PFLOTRAN. However, recently a more general multiphase mode has been implemented in PFLOTRAN, the GENERAL mode. There have been discussions with PFLOTRAN developers on how our work could be integrated into the GENERAL mode. Some new classes need first to be implemented in GENERAL mode to allow a correct implementation of a CH_4-brine module, but it will allow to give access to the scientific community to our work in the latest version of PFLOTRAN.

5.2 DEVELOP NEW CODE THAT LEVERAGE MORE PETSC LIBRARIES

PFLOTRAN has long utilized PETSc to provide scalable linear and nonlinear solvers for its many simulation modes. Much of the code written in the PFLOTRAN library addresses grid management and time stepping. As PETSc has matured, this functionality is now available in the library and can be factored out of PFLOTRAN. To this end we wrote a dimension and topology independent code which provides functionality analogous to Richards mode in PFLOTRAN and solved a benchmark problem with an analytic solution proposed by Tracy [14]. The additional functionality of PETSc allows us to write this code in approximately 400 lines of code. While the capabilities of the demonstration code are far from those of PFLOTRAN, it demonstrates that converting PFLOTRAN to use these new structures can make PFLOTRAN a smaller and therefore simpler codebase to both learn and support.

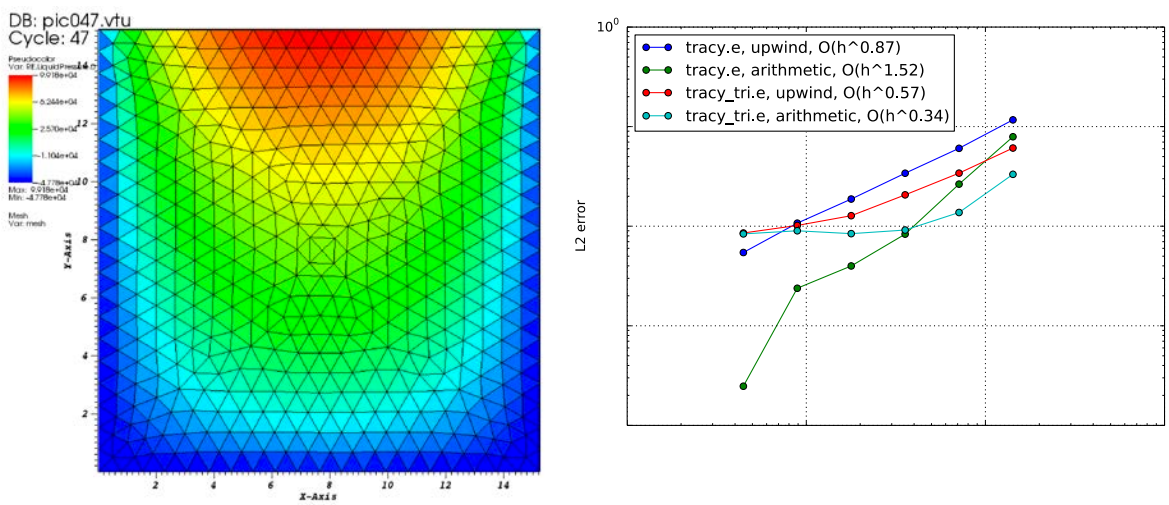

Figure 45 Results of the convergence study proposed by Tracy against a variably saturated analytic solution. On the left panel, a picture of the steady solution on the well-shaped but not orthogonal triangular mesh corresponding to the tracy_tri.e runs in the right panel.

In Figure 45 we present the results of the benchmark study. In the left panel, we show the liquid pressure of the exact solution. The blue levels correspond to a saturation state near the residual amount and red to a fully saturated state. In the right panel we show convergence results reporting the L2 norm of the relative true error in various approximations. We use two meshes, the first a structured grid of quadrilaterals (trace.e) and the second a well-shape mesh of triangular elements (tracy_tri.e). We also used this solution

to investigate the various methods of treating material discontinuities at cell interfaces. While it is common for various forms of upwinding to be used, arithmetic averaging is more defensible mathematically. We notice two effects which are important to note. First, upwinding causes a loss of convergence rate. In the quadrilateral cases, we see that the arithmetic averaging method approaches the expected 2nd order converge rate while upwinding degrades that rate to sub-linear. However, the triangular meshes cause a more severe problem in that we fail to converge to the correct solution no matter what averaging method is used. This is because of errors in the two-point flux approximation used to approximate the Darcy velocity from one cell to another.

5.3 EXPAND GEOMECHANICS MODULE

The geomechanics module is still at its infancy and need some work to catch up with the state-of-the art. For instance, the boundary conditions available in the geomechanics module need to be extended: boundary condition such as distributed load (or traction/compression) will remove all the restrictions with the nodal forces.

5.4 DISCRETE FRACTURE NETWORK DFN

Recently, a DFN package, dfnWorks [17][18][19]has been developed in Los Alamos National laboratory where the flow through the fracture network is simulated using PFLOTRAN. This capability and the power of today's super computer could lead to some very unique simulations.

6. CONCLUSIONS

PFLOTRAN was successfully adapted for shale formation simulation and proved to be a powerful numerical tool with better HPC efficiency than the current state-of-the art. Fast and efficient UQ methods were developed to tackle the uncertainties in subsurface properties and can be expanded to other applications than shale formation.

7. BIBLIOGRAPHY

[1] U. E. P. Agency, "Study of the Potential Impacts of Hydraulic Fracturing on Drinking Water Resources," http://www2.epa.gov/sites/production/files/documents/hf-report20121214.pdf, 2012.

[2] TOUGH. [Online]. Available: http://esd.lbl.gov/research/projects/tough/.

[3] R. Mills, L. Chuan, P. Lichtner and G. Hammond, "Mills, R. T., Lu, C., Lichtner, P. C., & Hammond, G. E. (2007, July). Simulating subsurface flow and transport on ultrascale computers using PFLOTRAN. In Journal of Physics: Conference Series (Vol. 78, No. 1, p. 012051). IOP Publishing.," *Journal of Physics: Conference series*, vol. 78, no. 1, p. 012051, 2007.

[4] R. Mills, V. Sripathi, G. H. G. Mahinthakumar, P. Lichtner and B. Smith, "Experiences and challenges scaling PFLOTRAN, a PETSc-based code for subsurface reactive flow simulations, towards the petascale on Cray XT systems.," in *Proc. 51st CUG meeting*, Atlanta, GA, USA, 2009.

[5] "www.pflotran.org," [Online].

[6] "http://www.mcs.anl.gov/petsc/," [Online].

[7] T. Irvine and P. Liley, Steam and gas tables with computer equations, New York: Academic press.,

1984.

[8] Z. Duan and S. Mao, "A thermodynamic model for calculating methane solubility, density and gas phase composition of methane-bearing aqueous fluids from 273 to 523 K and from 1 to 2000 bar," *Geochimica et Cosmochimica Acta,* vol. 70, pp. 3369-3386, 2006.

[9] A. Ghassemi, S. Kelkar and M. McClure, “Influence of Fracture Shearing on Fluid Flow and Thermal Behavior of an EGS Reservoir - Geothermal Code Comparison Study,” in *Fourtieth Workshop on Geothermal Reservoir Engineering,* Stanford University, Stanford, California, January 26-28, 2015.

[10] D. Peaceman, “ Interpretation of well-block pressure in numerical reservoir simulation with non-square grid blocks and anisotropic permeability,” *SPE,* pp. 531-543, 1983.

[11] Y. Jiang, “Techniques for modeling complex reservoirs and advanced wells,” Stanford University, 2007.

[12] A. Cavanagh, “Benchmark Calibration and Prediction of the Sleipner CO_2 plume from 2006 to 2012,” *Energy Procedia,* vol. 37, pp. 3529-3545, 2013.

[13] A. e. A. Kissinger, “Hydraulic fracturing in unconventional gas reservoirs: risks in the geological system, part 2,” *Environmental earth sciences,* vol. 70, no. 8, pp. 3855-3873, 2013.

[14] C. Doughty, “Modeling geologic storage of carbon dioxide: comparison of non-hysteretic and hysteretic characteristic curves,” *Energy Conversion and Management* , vol. 48, no. 6, pp. 1768-1781, 2007.

[15] A. Battistelli, C. Calore and K. Pruess, "The Simulator TOUGH2/EWASG for modelling geothermal reservoirs with brines and noncondensable gas," *Geothermics,* vol. 26, no. 4, pp. 437-464, 1997.

[16] J. Haas, “hysical properties of the coexisting phases and thermochemical properties of the H_2O component in boiling NaCl solutions,” US Govt. Print. Off., 1976.

[17] dfnWorks, "http://www.lanl.gov/expertise/teams/view/dfnworks," [Online].

[18] J. D. Hyman, S. Karra, N. Makedonska, C. W. Gable, S. L. Painter and H. S. Viswanathan, "dfnWorks: A discrete fracture network framework for modeling subsurface flow and transport," *Computers & Geosciences,* vol. 84, pp. 10-19, 2015.

[19] S. Karra, N. Makedonska, H. S. Viswanathan, S. L. Painter and J. D. Hyman, "Effect of advective flow in fractures and matrix diffusion on natural gas production," *Water Resources Research,* 2015.