A Multicore Numerical Framework for Assessing the Permeability of Reservoir Rocks

[*]Peter G. Tilke, [§]Christopher R. Leonardi, [†]David W. Holmes, and [¶]John R. Williams

[*]*Department of Mathematics and Modeling, Schlumberger-Doll Research Center, 1 Hampshire Street, Cambridge, MA 02139*
[§]*Civil and Environmental Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139*
[†]*Department of Mechanical Engineering, James Cook University, Angus Smith Drive, Douglas, QLD 4811, Australia*
[¶]*Civil and Environmental Engineering and Engineering Systems, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139*

Abstract. This paper presents a numerical framework that is capable of simulating multiphase flow in reservoir rocks at the pore scale. The framework combines a suite of numerical methods, including smooth particle hydrodynamics (SPH) and the lattice Boltzmann method (LBM), with shared-memory, multicore parallel processing to increase the flexibility and scalability of solutions. By incorporating a suite of methods in the numerical framework, each with their own relative strengths, the range of problems that can be solved is greatly increased. The utilized parallel programming model exploits the large memory as well as the low latency of processor caches available in contemporary multicore servers. Maximized cache performance is achieved by taking a fine-grained approach to domain decomposition and also taking advantage of the spatial locality of data in the solvers. This results in scalable speed-up efficiency, whilst the asynchronous distribution of fine-grained, parallel work tasks results in natural load balancing. Both the SPH and LBM solvers are applied to determine the permeability of reservoir rocks from x-ray microtomographic images of samples. Predictions of the absolute permeability of West Texas Dolomite and Berea Sandstone samples are presented, with both comparing well with experimental data.

Keywords: permeability, dolomite, Berea sandstone, lattice Boltzmann, smoothed particle hydrodynamics.
PACS: 47.56.+r

INTRODUCTION

The permeability of a hydrocarbon reservoir is an important input parameter in predictive production simulations. In reservoir simulators each cell is prescribed a permeability tensor which is statistically interpolated from a limited number of data points. However, as the size of cells decreases due to advances in computational hardware, there is motivation to honor the increasing resolution of the simulator with a more accurate description of the permeability distribution in the reservoir.

Determining the permeability of reservoir rocks via displacement experiments on core samples can be expensive and time consuming. It can also be impractical when suitable core samples are not readily available. For these reasons, a number of different modeling approaches have been developed which facilitate the prediction of absolute and relative permeability (as well as other petrophysical properties) of reservoir rocks from high resolution images of small physical samples.

The most straightforward way to predict the permeability of a porous rock is via an empirical relationship based on porosity, hydraulic radius or another combination of parameters, such as the Kozeny-Carman equation [1]. More recently, micron-scale x-ray tomography (μCT) has been employed to generate high resolution, 3D images of millimeter scale samples [2]. These images can then serve as the primary input for a range of numerical experimentation procedures for the determination of permeability and other properties. These procedures include pore network models [3], and direct numerical simulation of pore-scale fluid flow using the finite difference (FD) method [4], the lattice Boltzmann method (LBM) [5, 6], and smoothed particle hydrodynamics (SPH) [7].

The numerical modeling strategies that have been applied to the problem of multiphase flow in porous media each have their own strengths and weaknesses. In an attempt to leverage the capabilities of more than one technique, this paper presents a generalized numerical framework that incorporates a range of

these numerical methods. The framework combines the LBM, SPH, FD and the discrete element method (DEM), with shared-memory, multicore parallelism to increase the flexibility and scalability of computations.

Using both LBM and SPH the framework has been employed to determine the absolute permeability of two oil reservoir rock samples. The numerical permeability predictions compare well with published experimental data for similar rock types.

GENERALIZED MULTI-SOLVER FRAMEWORK

The development of a generalized multi-solver framework has been motivated by the need to simulate a variety of fluid-solid interaction phenomena which occur in hydrocarbon reservoirs. The creation of the framework has included development in three key areas, namely, a plug-in solver framework for a range of numerical methods, a distribution technique to facilitate asynchronous, parallel computation on multicore hardware, and a library of methods including LBM, SPH, FD and DEM.

The numerical methods incorporated into the multi-solver framework (i.e. LBM, SPH, FD, DEM) are well established. Depending on their formulation, each method is more suited to application in some problems than others. Including each of them in the framework provides flexibility when deciding the best approach for solving reservoir flow problems. For example, SPH excels at naturally reproducing free surface flows and the interfaces of multiple phases. With the LBM, the mapping of complex boundaries is straightforward, and it can also be readily coupled with DEM and or FEM to solve fluid structure interaction problems.

The plug-in solver framework acts as a coordination layer for the execution of the chosen numerical method. For example, SPH can be 'plugged into' the framework, which then manages explicit time integration and task distribution for parallel computing. Additionally, the plug-in solver allows for more than one method to be coupled together to simulate, for example, particle suspensions using the LBM and DEM, as shown schematically in Figure 1.

The multi-solver framework is underpinned by three important implementation features, namely, global time integration, function inheritance for each solver, and a common data structure format. Global time integration is realized using a single, explicit time stepping scheme which controls the progression of each of the solvers that is plugged into the framework. Function inheritance for the solvers is made possible by the use of a common base class of processing methods (i.e. functions). The method class of the chosen solver(s) inherits from the base class and exposes its methods for the processing of information. Finally, the use of a single data structure format for each solver facilitates a common strategy for data decomposition and thread-safe parallel processing.

The implementation of the multi-solver is aided by inheritance and polymorphism, both of which are attributes of high-level, object-oriented languages such as C#.NET. In the discussion of the parallel efficiency of the framework it will be shown that the use of a fine-grained domain decomposition strategy can largely circumvent the cost of memory management, which is also a feature of C#.NET.

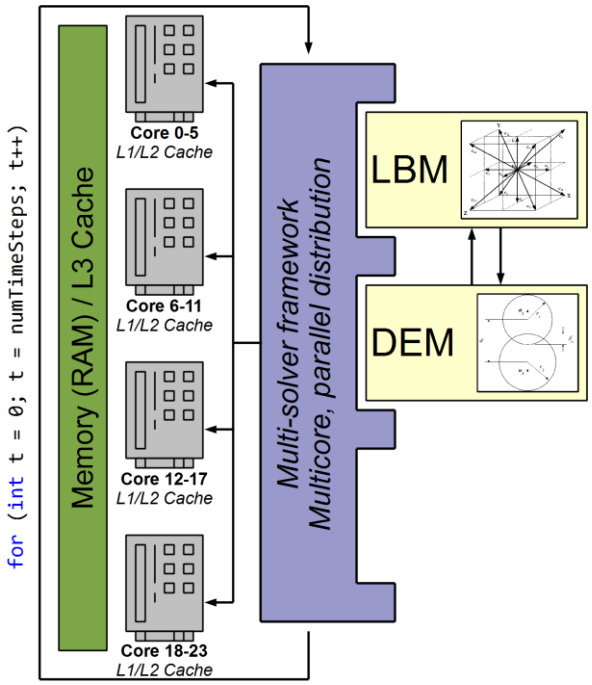

FIGURE 1. Schematic of the generalized multi-solver framework, showing the LBM and DEM solvers 'plugged in' and fully coupled. The framework manages time integration via a global, explicit scheme. Computations are parallelized via shared-memory, multicore hardware.

Asynchronous Parallelism Using Shared-Memory, Multicore Hardware

The most common approach to parallel processing of numerical methods utilizes a distributed memory cluster as the underlying hardware. In this approach the computational domain is decomposed into the same number of sub-domains as there are nodes available in the cluster. Each cluster node processes a single sub-domain at each time step and, when all sub-domains have been processed (in a time step), global solution data is synchronized.

The synchronization of solution data at the end of each time step involves the communication of *ghost*

regions between nodes. These ghost regions correspond to neighboring sections of the problem domain (resident in memory on other cluster nodes) which are required on a cluster node for the processing of its own sub-domain. In the LBM this is typically a 'layer' of grid points that encapsulates the local sub-domain, but in SPH the layer of neighboring particles required is equal to the radius of the compact support zone. In 3D particle-based methods, such as SPH, it is possible for the amount of data communicated in ghost regions to be of the same order as the amount of stationary data. As a consequence of Amdahl's Law [8], and the fact that communication between nodes with packages is a serial process, this can significantly degrade the scalability of the implementation.

Another challenge with distributed memory parallelism can be sub-optimal load balancing, which also degrades parallel efficiency. Simply decomposing the problem into equal-sized sub-domains may not result in an equitable distribution of work between the cluster nodes. This is particularly relevant when the computational work of a problem is heterogeneously distributed throughout the domain. For example, in porous media flows the work at *rock* locations can be less than that at *fluid* locations, and in particle suspensions, more work is required (due to hydrodynamic coupling) in areas of above average particle concentration than in other areas.

The issues of data communication and load balancing are addressed in the multi-solver framework by employing shared-memory, multicore hardware and fine-grained domain decomposition, respectively.

Using shared-memory, multicore hardware for parallel processing removes the need for ghost regions and the transfer of data over relatively slow connections between cluster nodes. Instead, all data is accessible to all cores from either local caches (i.e. L1, L2) or global memory (i.e. L3 cache, RAM). Access times for these data stores are many orders of magnitude shorter than cross-machine communication [9]. When used with an optimum cache-blocking strategy these access times significantly reduce the latency associated with data reads and writes.

Cache-blocking in the multi-solver framework is optimized by utilizing fine-grained domain decomposition. Instead of partitioning the domain into one sub-domain per core, a collection of significantly smaller sub-domains is created. These sub-domains, or computational tasks, are sized to fit in the low-level cache (i.e. L1, L2) of a processing core, which minimizes the time spent reading and writing data as a task is processed. This minimization of access time is important, particularly in *memory-bound* numerical methods, such as the LBM and SPH.

In the LBM, the tasks created by fine-grained domain decomposition could be cubic bundles of nodes. In SPH these tasks could be bins populated with particles via spatial hashing. On a multicore server with a core count on the order of 10^1~10^2 the number of tasks could be in the order of 10^3~ 10^4. Multicore distribution of these tasks requires the use of a coordination tool to manage them onto processing cores in a load balanced way. While such tasks could easily be distributed using a traditional approach like scatter-gather, here the H-Dispatch distribution model [10] has been used because of the demonstrated advantages for performance and memory efficiency.

The H-Dispatch distribution model maintains a thread on each core in the underlying multicore server. These threads remain active throughout the duration of the analysis so that local, temporary memory required for processing tasks can be reused (thereby minimizing garbage collection). The novel feature of H-Dispatch is the way in which tasks are distributed to threads. Rather than a scatter or push of tasks from the manager to threads, here threads request values when free. H-Dispatch manages these events-based, asynchronous requests and distributes cells to the requesting threads accordingly. It is this pull mechanism that enables the use of a single thread per core as threads only request a value when free, thus, there is never more than one task at a time associated with a given enduring thread (and its associated local variable memory). Additionally, when all tasks in the problem space have been dispatched and processed, H-Dispatch identifies step completion (i.e. synchronization) and the process can begin again. By using many more tasks than cores, and events-based distribution of these tasks, the computational workload of the numerical method is naturally balanced.

The Lattice Boltzmann Method

The LBM (see [11] for a review) has emerged in the last 20 years as a powerful numerical method for the simulation of fluid flows. It has found application in a vast array of problems including flows in porous media, multiphase flows, and particle suspensions.

The LBM differs from conventional approaches to computational fluid dynamics in that it does not involve the discretization and solution of the governing hydrodynamic equations (i.e. Navier-Stokes). Instead, the LBM can be interpreted as a discrete form of the Boltzmann equation at the mesoscopic scale. The primary variables in the LBM are particle distribution functions, which exist at each of the lattice nodes that comprise the fluid domain. These functions relate the probable amount of fluid 'particles' moving with a discrete speed in a discrete direction at each lattice node at each time increment. The particle distribution functions are evolved at each

time step via a two-stage, collide-stream process. The collision process monotonically relaxes the particle distribution functions towards their respective equilibria, and in doing so governs the viscous properties of the fluid. The redistributed (i.e. collided) functions are then adjusted by a body force term (if applicable), after which the streaming process propagates them to their nearest neighbor nodes.

Spatial discretization in the LBM is typically based on a square (2D) or cubic (3D) array of nodes. In the present work the D3Q15 lattice is employed, which includes a zero velocity vector, the six nearest-neighbor velocity vectors, and the eight furthest-neighbor velocity vectors.

The macroscopic fluid variables, density and momentum flux, are calculated at each lattice node as velocity moments of the particle distribution functions.

In this study the immersed moving boundary (IMB) method [12] is employed to handle the hydrodynamic coupling of the fluid and structure. Further details of the IMB method and its coupling to the FEM/DEM can be found in Owen et al. [13].

The obvious choice for decomposition of the LBM domain data is to use cubic nodal bundles. An important advantage of this approach is the ease with which the bundle size can be used to optimize cache blocking. By adjusting the bundle size, the associated computational task can be re-sized to fit in cache close to the processor (e.g. L1 or L2). This minimizes cache misses and the associated retrieval of data from RAM, which occurs with significantly greater latency.

To ensure thread safety, two copies of the LBM particle distribution functions at each node are stored. Nodal processing is undertaken using a pull-collide sequence. Incoming functions are read from neighbor nodes (non-local read), collided, and then written to the future set of functions for the current node (local write). On cache-sensitive multicore hardware, this sequence of operations outperforms collide-push, which requires local reads and non-local writes [9].

The benefit of optimized cache blocking is found by varying the bundle size and measuring the speed-up. This was performed for a simple, 200^3 problem [14] and the optimal performance point (92% speed-up efficiency) was found at a side length of 20. Additionally, it was found that this optimal side length could be applied to larger domains (300^3 and 400^3) and still yield maximum speed-up efficiency. This suggests that the optimum bundle size for the LBM can be determined in an a priori fashion for specific hardware.

Smoothed Particle Hydrodynamics

SPH is a mesh-free, Lagrangian particle method which was first proposed for the study of astrophysical problems [15, 16], but is now widely applied to fluid mechanics problems [17]. A key advantage of particle methods such as SPH is their ability to advect mass with each particle, thus removing the need to explicitly track phase interfaces for problems involving multiple fluid phases or free surface flows. However, the management of free particles brings with it the associated computational cost of performing spatial reasoning at every time step.

SPH theory has been detailed widely in the literature with various formulations having been proposed. The methodology of authors such as Tartakovsky and Meakin [18, 19] and Hu and Adams [20] has been shown to perform well for the case of multi-phase fluid flows. Their particle number density variant of the conventional SPH formulation removes erroneous artificial surface tension effects between phases and allows for phases of significantly differing densities. Such a method has been used for the performance testing in this work.

Solid boundaries in the simulator are defined using rows of virtual particles [21], and no-slip boundary conditions are enforced for low Reynolds number flow simulations using an artificially imposed boundary velocity method [22].

The data associated with each particle in the SPH implementation is stored in a collection of bins. These bins are analogous to the bundles used in the LBM, and form the collection of SPH tasks that is distributed to cores using H-Dispatch. Particles are assigned to bins using spatial hashing based on their Cartesian coordinates. By storing particle data based on its physical location the number of bins accessed while determining the interaction of a particle and its neighbors is minimized.

As in the LBM, care must be taken to avoid common shared memory problems such as race conditions and thread contention. To circumvent the problems associated with using locks [23] the SPH data can be structured to remove the possibility of thread contention altogether. By storing the present and previous values of the SPH field variables (e.g. position, velocity etc.), necessary gradient terms can be calculated as functions of values in previous memory, while updates are written to the current value memory. This reduces the number of synchronizations per time step from two (if the gradient terms are calculated before synchronizing, followed by the update of the field variables) to one, and a rolling memory algorithm switches the index of previous and current data with successive time steps.

By definition, the SPH bins can be used to maximize cache blocking just as the bundles were in the LBM, and test results showed similar speed-up efficiency to that of the LBM [14]. The SPH results also highlighted the influence that read/write access

times have on the performance of memory-bound numerical methods such as SPH. The parallel SPH implementation was tested in both single-search (perform spatial reasoning once per time step and store the results in memory for use twice per time step) and double-search forms (perform the spatial search twice per time step, as needed). The results showed that the double-search code scaled well, but the single-search code scaled poorly. On first consideration this outcome seems counterintuitive, as the code with more computation performs better. However, it actually highlights the penalty associated with reading/writing more data than is necessary in an algorithm that is already memory-bound.

ABSOLUTE PERMEABILITY OF RESERVOIR CORE SAMPLES

The ability of the SPH and LBM solvers to predict the permeability of porous media has been extensively benchmarked. In [18] SPH was used to determine the drag coefficient and permeability of fluid in a periodic cubic array of spheres for porosity values between 10% and 99%. The correlation of the SPH predictions with published Stokes flow solutions [23] was excellent. This range of test problems was repeated with the LBM, yielding almost identical results.

The multi-solver framework was then applied to numerically determine the porosity-permeability relationship of two rock samples, namely West Texas Dolomite and Berea Sandstone. For both samples, the structural model geometry was generated from a μCT image which had been segmented to classify voxels (i.e. pixels) as either rock/clay or pore. The Dolomite image measured 500^3 (125 million voxels, total) and was scanned at a resolution of 1μm, corresponding to a $0.5mm^3$ sample. The Berea image was taken from a cylindrical core sample measuring 4.95mm in diameter and 5.43mm in length at resolution of 2.8μm. This resulted in a voxelated image set that measured 1840×1840×1940 (6.568 billion voxels, total) in size. Current hardware limitations prevented either of the full images from being analyzed in one numerical experiment. Instead, a set of sub-blocks with voxel dimensions between 200^3 and 400^3 was taken from each of the full images for flow testing. A rendered representation of the pore space in one of the 300^3 Dolomite sub-blocks is shown in Figure 2. The results of the sub-block tests were then used to generate a porosity-permeability relationship for each rock type.

Spatial discretization of the permeability analyses was handled similarly for both the SPH and LBM solvers. In the LBM the grid spacing was set equal to the voxel size. In SPH the particles were initialized with a density of approximately one per voxel, which convergence tests showed to be adequate for capturing the dominant flow channels.

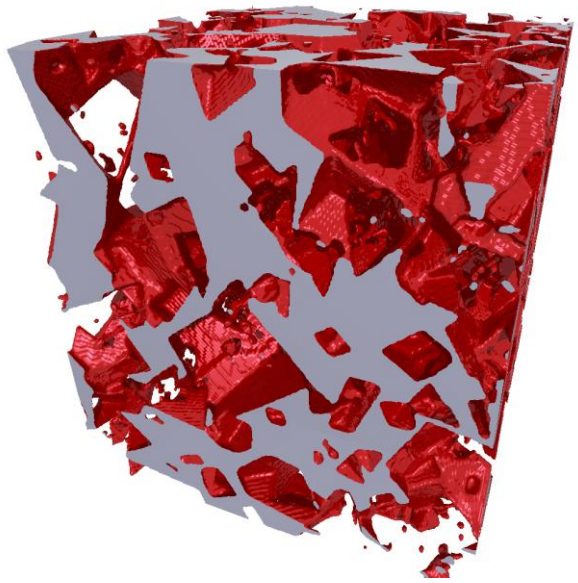

FIGURE 2. A rendered representation of the pore volume in a 300^3 Dolomite sub-block (the rock is not shown).

Water ($\rho = 10^3 kgm^{-3}$, $\nu = 10^{-6} m^2 s^{-1}$) was used as the driving fluid in all analyses. Flow was driven through the porous media by a constant body force in the negative z-direction. The four domain surfaces parallel to the direction of flow were designated no-flow boundaries and the in-flow and out-flow surfaces were specified as periodic. Due to the incompatibility of the two rock surfaces at these boundaries, periodicity could not be applied directly. Instead, the experimental arrangement was replicated by adding a narrow volume of fluid at the top and bottom of the domain.

The results of the Dolomite permeability tests are graphed in Figure 3. Experiments [25] have shown that the permeability-porosity relationship for Dolomite is strongly dependent on the crystal size of the rock. These crystal-size bands have been included in the plot of numerical results. The mean crystal size of the tested sample placed it in Band III, which is where the LBM and SPH predictions predominantly lie.

The results of the Berea permeability tests (using the LBM only) are also graphed in Figure 3. The experimentally determined permeability for the sample is included for comparison. It can be seen that the numerical predictions are slightly higher than measured value, although the correspondence between the two measurements is reasonable. Both rock test results suggest that the presented numerical procedure is appropriate for determining rock permeability.

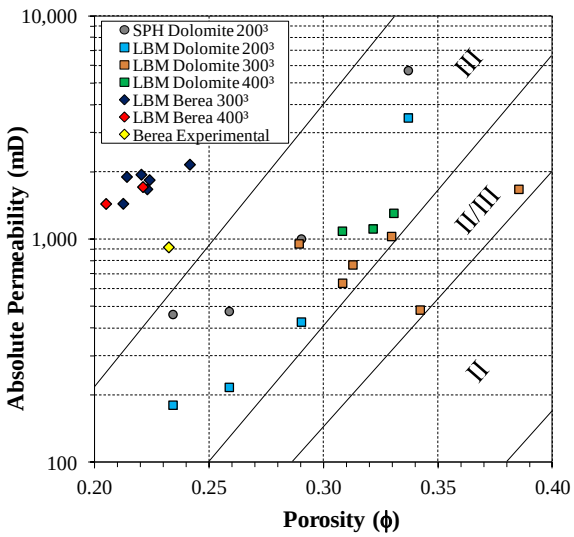

FIGURE 3. Results of the Dolomite and Berea permeability tests undertaken. Guidelines from experimental data [25] for Dolomite are included for comparison.

DISCUSSION

This paper has presented a multi-solver framework that is capable of leveraging a range of numerical methods to solve coupled problems in hydrocarbon reservoirs. As the range of implemented and verified modeling capabilities increases, so too will the scope of applications. For example, current research is testing the permeability characterization process for applicability in low porosity rock samples, and the combination of non-Newtonian fluids and suspended particles is being applied to simulate the transport of proppants in hydraulic fractures.

The LBM and SPH solvers in the framework were used to predict the absolute permeability of West Texas Dolomite and Berea Sandstone samples. The numerical results compared well with experimental data. However, it should be noted that the 200^3 Dolomite results showed that the SPH predictions were consistently greater than those from the LBM (for identical sub-samples). This discrepancy warrants further investigation.

With the completion of a robust procedure for predicting the absolute permeability of rocks, immediate development will focus on quantitative predictions of the relative permeability of samples under two-phase (i.e. oil and water) flow.

ACKNOWLEDGMENTS

The authors are grateful to the Schlumberger-Doll Research Center for their support of this research, and for supplying the μCT images analyzed in this study.

REFERENCES

1. Bernabé, Y., and Bruderer, C., *J. Geophys. Res.* **103**(B1), 513-525 (1998).
2. Arns, C. H., Bauget, F., Ghous, A., Sakellariou, A., Senden, T. J., Sheppard, A. P., Sok, R. M., Pinczewski, W. V., Kelly, J. C., and Knackstedt., M. A., *Petrophysics* **46**(4), 260-277 (2005).
3. Blunt, M. J., Jackson, M. D., Piri, M., and Valvatne., P. H., *Adv. Water Resour.* **25**(8-12), 1069-1089 (2002).
4. Zhan, X., Schwartz, L. M., Toksöz, M. N., Smith, W. C., and Morgan, F. D., *Geophysics* **75**(5), F135-F142 (2010).
5. Arns, C. H., Bauget, F., Limaye, A., Sakellariou, A., Senden, T. J., Sheppard, A. P., Sok, R. M., Pinczewski, W. V., Bakke, S., Berge, L. I., and Øren, P.-E., *SPE Journal* **10**(4), 475-484 (2005).
6. Kameda, A., Dvorkin, J., Keehm, Y., Nur, A., and Bosl, W. *Geophysics* **71**(1), N11-N19 (2006).
7. Williams, J. R., Holmes, D., and Tilke, P., *Comp. Meth. Appl. Sci.* **25**, 113-134 (2011).
8. Herlihy, M., and Shavit, N., *The Art of Multiprocessor Programming*, Burlington:MorganKaufman, 2008, pp13.
9. Pohl, T., Kowarschik, M., Wilke, J., Iglberger, K., and Rüde, U., *Parallel Process. Lett.* **13**(4), 549-560 (2003).
10. Holmes, D. W., Williams, J. R., and Tilke, P. G., *Comput. Phys. Commun.* **181**(2), 341-354 (2010).
11. Chen, S., and Doolen, G. D., *Annu. Rev. Fluid Mech.* **30**, 329-364 (1998).
12. Noble, D. R., and Torczynski, J. R., *Int. J. Mod. Phys. C* **9**(8), 1189-1201 (1998).
13. Owen, D. R. J., Leonardi, C. R., and Feng, Y. T., *Internat. J. Numer. Methods Engrg.* **87**(1-5), 66-95 (2011).
14. Leonardi, C. R., Holmes, D. W., Williams, J. R., and Tilke, P. G., "A Multi-Core Numerical Framework for Characterizing Flow in Oil Reservoirs" in *Spring Simulation Multiconference-2011*, edited by L. T. Watson et al., Simulation Series 43(2), Red Hook: Simulation Councils, 2011, pp. 166-174.
15. Lucy, L. B., *Astro. J.*, **82**, 1013-1024 (1977).
16. Gingold, R. A., and Monaghan, J. J., *Mon. Not. R. Astron. Soc.* **181**, 375-389 (1977).
17. Liu, G. R., and Liu, M. B., *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*, Singapore: World Scientific, 2007.
18. Tartakovsky, A. M., and Meakin, P., *J. Comput. Phys.* **207**, 610-624 (2005).
19. Tartakovsky, A. M., and Meakin, P., *Adv. Water Resour.* **29**, 1464-1478 (2006).
20. Hu, X. Y., and Adams, N. A., *J. Comput. Phys.* **213**, 844-861, (2006).
21. Morris, J. P., Fox, P. J., and Zhu, Y., *J. Comput. Phys.* **136**, 214-226 (1997).
22. Holmes, D. W., Williams, J. R., and Tilke, P. G., *Int. J. Numer. Anal. Met.* **35**(4), 419-437 (2011).
23. Adl-Tabatabai, A. R., Kozyrakis, C., and Saha, B., *Queue* **4**(10), 24-33 (2007).
24. Larson, R. E., and Higdon, J. J. L., *Phys. Fluids A* **1**(1), 38-46 (1989).
25. Sneider, R. M., and Sneider, J. S., *Search and Discovery* **10007**, (2000).