Journal of Computational Physics 384 (2019) 134–150

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Efficient modelling of solute transport in heterogeneous media with discrete event simulation

Q. Shao [a,*], S.K. Matthäi [b], L. Gross [a]

[a] School of Earth and Environmental Sciences, The University of Queensland, Queensland 4067, Australia
[b] Reservoir Engineering Group, Peter Cook Centre for CCS, Department of Infrastructure Engineering, The University of Melbourne, Victoria 3010, Australia

ARTICLE INFO

Article history:
Received 27 July 2018
Accepted 29 January 2019
Available online 8 February 2019

Keywords:
Solute transport
Discrete event simulation
Parallel computation
Node-centred finite volumes
Heterogeneity

ABSTRACT

To address the problem of different time scales present in the simulation of solute transport through systems with a complex permeability structure such as fractured porous rocks, we propose a parallel discrete event simulation (DES) algorithm based on local time stepping criteria, specifically developed for the hybrid finite-element node-centred finite volume (FV) framework. A preemptive-event-processing (PEP) approach is applied to synchronise discrete events with sufficiently close time stamps, thereby facilitating the parallelisation for shared memory architectures. The accuracy of the presented DES-PEP scheme is first verified against the analytical solution of a 1D advection equation with spatially variable coefficients. The DES scheme is then applied to simulate tracer advection through a 3D model of highly heterogeneous fractured rock represented by an unstructured adaptively refined mesh with over 1 million elements. DES produces results comparable to those of a conventional time-driven simulation (TDS), but uses less than 1% of the execution time. Analysis of event distributions shows that updates occur almost exclusively in a small number of FV cells marked by order-of-magnitudes faster fluid flow and advection-dominated transfer, while slow-flowing cells remain inactive and excluded from computations. This focusing of the computational effort leads to high simulation efficiency while simultaneously diminishing round-off errors. Scalability tests with a parallel version of DES on shared memory demonstrate further computational speedups mirroring the increased number of threads. With the use of 20 threads, execution time is reduced from 42.5 days (with TDS) to only 1.5 hours, equivalent to a speedup of over 670. This parallel DES algorithm therefore enables efficient multi-core simulation of solute transport in heterogeneous geologically realistic systems.

© 2019 Elsevier Inc. All rights reserved.

1. Introduction

Understanding the transport of solutes (e.g., chemicals, tracers, minerals or contaminants) in porous geological formations is essential for a broad range of geo-engineering applications, including enhanced recovery of oil and gas from hydrocarbon reservoirs [1], remediation of groundwater pollution in aquifers [2], and storage of nuclear waste or CO_2 in the subsurface [3]. The high degree of heterogeneity in geological systems creates various numerical and computational challenges

* Corresponding author.
E-mail addresses: q.shao@uq.edu.au, bnushaoqi@gmail.com (Q. Shao).

https://doi.org/10.1016/j.jcp.2019.01.026
0021-9991/© 2019 Elsevier Inc. All rights reserved.

when solving fluid pressure equations and advection–diffusion equations describing the solute transport [4]. Various numerical methods have been proposed to discretise the governing equations and simulation variables in space and time. Typical examples are finite difference methods (FDM) [5,6] and finite element methods (FEM) [7,8], with the latter having more flexibility to represent complex geometries. Combinations of FEM and finite volume method (FVM) have become increasingly popular [4,9–12]. In this approach, a node-centred finite volume mesh is superimposed on the finite element mesh. The elliptic fluid pressure equation is solved by FEM and the hyperbolic transport equation by FVM. This combined approach shows improved accuracy and efficiency over the fully coupled FEM, while retaining the geometric flexibility of finite elements [10,11].

How to efficiently integrate these spatially discretised equations over time is still a fundamental challenge. Traditionally time advance is based on a global time step, which synchronously evolves the system at spatially uniform time intervals. These time-driven simulation (TDS) techniques fall into two categories: explicit and implicit schemes. In explicit schemes, unknown variables are computed at the current time level from quantities already available from previous times. The size of the integration time step is restricted by the most stringent Courant–Friedrichs–Levy stability criterion (also known as the Courant or CFL condition) in the system, which guarantees the capture of the fastest transient change, and ensures numerical stability. This becomes problematic in simulations of solute transport in highly heterogeneous porous media where flow properties, such as permeability, can vary over many orders of magnitude [11,13,14], leading to dramatic variations in transport velocities. This problem is further exacerbated in models constructed with variable mesh sizes where the smallest elements and control volumes tend to represent areas of highest permeability (e.g. rock fractures) and therefore coincide with regions of enhanced fluid flow [4]. The resultant infinitesimal global time step leads to prohibitively slow TDS simulations.

Stability of implicit schemes is not restricted by the severe CFL constraint permitting larger time increments. However, this is at the expense of accuracy and an increased computational burden from solving a system of equations at each iteration step. Matthäi, Nick, Pain and Neuweiler [4] found that when simulating solute transport through a 3D stochastic discrete fracture model containing 2000 disc-shaped fractures, a single implicit transport step required approximately 20 times the CPU time needed for a single explicit step, even though an efficient algebraic multigrid solver, SAMG [15], was used. Furthermore, implicit schemes introduce numerical smoothing and tend to be too diffusive [16]. Omelchenko and Karimabadi [17] argue that implicit integrations conducted with large time increments are unable to correctly represent local physical phenomena with characteristic process frequencies higher than the inverse of the time-step size.

To get around global time step restrictions, a number of local time-stepping approaches have been proposed to allow the time step to vary spatially satisfying a local, rather than global CFL condition. In these approaches which are also known as adaptive or multiple time-stepping schemes, the solutions at different elements or cells are updated with different time increments determined from local CFL conditions [18–21]. Because of the asynchronous update of different elements, it is difficult for these methods to correctly and efficiently calculate numerical fluxes across element interfaces. Some use time interpolation for flux computation, however, these violate local conservation of fluxes, leading to reduced numerical accuracy and stability [17]. Local time-stepping approaches are often used in combination with adaptive mesh refinement, where the spatial grid is locally refined in order to represent complex geometries and better capture more active regions [22–27]. Spatial elements on each level of refinement are clustered into a logically rectangular patch, leading to a hierarchy of nested patches with different temporal resolutions that locally satisfy the CFL conditions. In order to preserve conservation laws, global synchronization steps are required to impose flux corrections at patch interfaces. For efficiency, a fixed local time step size, computed at the beginning of each global synchronization step, is used throughout the global time step. This is problematic for highly complex systems as the CFL conditions may change rapidly and significantly during a global synchronization step, making the pre-determined time integration sequence inaccurate and unstable.

Discrete event simulation (DES), also called event-driven simulation, provides an alternative to conventional time-stepping schemes. Unlike TDS where the simulation progresses in pre-defined global or local time steps, in DES temporal evolution of the global system is driven by discrete events. This forces transition from one state to another at irregular (or asynchronous) points in time [28,29]. Such discrete events, representing effective units of information in DES, are simulation objects characterised by a process function responsible for updating the object state, and a time stamp indicating when the process function is scheduled to be executed. A DES simulation begins with scheduling events based on local rates of change in individual elements. All scheduled but not yet executed events are listed into an event queue sorted by their time stamps in increasing order. The top event always corresponds to an event with the earliest time stamp. The DES simulation then progresses by repeatedly removing the top event from the priority queue, executing it by calling its process function and adding a new event with an updated time stamp to the queue. Consequently, DES avoids using pre-determined time steps and integration sequence as in the TDS schemes, but allows individual parts of the simulation domain to evolve based on their own physically determined temporal scales, which may vary in space and over the progress of the simulation. This effectively removes the global CFL restriction and, at the same time, ensures numerical accuracy and stability. DES focuses updates on active parts of the system, while eliminating the computational overhead associated with idling processors.

DES was originally developed for naturally discrete systems, as found in operations research, management science, games and telecommunications [28–30]. It was extended to modelling continuous systems [31,32] with reported speedups by a factor of 300 in some cases [33]. The DES scheme has been applied to solve flux-conservative partial differential equations with source terms [17] by introducing a self-adaptive predictor–corrector scheme. The predictor schedules events with a time delay estimated from the local variation rate, and the corrector ensures that pending events are processed in time, if

their causality constraints undergo a significant change. This self-adaptive paradigm, in combination with a synchronization mechanism for updating neighbouring events, effectively preserves causality and guarantees numerical stability. Later work by Stone, Geiger and Lord [34] associated an event with the mass transfer at element interfaces, rather than the evolution within an element. In order to improve event processing by preventing unnecessary inter-element synchronisations in parts of the computational domain where the solution properties are fairly homogeneous, Omelchenko and Karimabadi [35] introduced a preemptive-event-processing (PEP) technique, which automatically enforces synchronous execution of events with sufficiently close timestamps. So far the advances in DES simulation of continuous systems have been demonstrated on low-dimensional uniform meshes. In this paper we will investigate the use of DES to complex and realistic problems at larger scales. We will also make use of the advantage that the PEP technique can be used to easily parallelise DES at least for shared memory architectures – a fact which was not realised in the original publication.

The objective of this paper is to develop a parallel DES algorithm for a finite-element node-centred finite volume scheme. The implementation uses the Complex Systems Modelling Platform (CSMP++) [36], an object-oriented application programmer interface designed for simulation of complex geologic processes and their interactions [37]. It provides an efficient simulation framework for solute transport through heterogeneous porous media on high-dimensional, complex and unstructured meshes. The rest of this paper is structured as follows. In section 2, the governing equations for solute transport in porous media are presented, followed by a brief description of the discretization of these equations in the CSMP++ FEM-FVM framework. We then describe in detail the algorithmic and programming aspects of the parallel DES-PEP for the node-centred FV transport scheme. In section 3, the proposed DES algorithm is verified with analytical solutions of the 1-D advection equation with spatially variable coefficients. In section 4, we apply it to tracer advection through a heterogeneous 3D model of fractured rock, and compare its results and performance with conventional TDS simulations. We also conduct a scalability analysis for multiple threads on this model. Section 5 concludes this paper with a summary of the most important results obtained.

2. Methodology and implementation

2.1. Governing equations

Ignoring diffusion which is readily modelled using the finite element approach, advective transport of a non-reactive solute through a porous medium is governed by the general advection equation

$$\phi \frac{\partial c}{\partial t} + \nabla \cdot (\mathbf{v}c) = q, \tag{1}$$

where c (kg/m^3) represents the solute concentration, ϕ refers (m^3/m^3) to the porosity of the medium, t (s) is time, and q (kg/m^3 s) describes external sinks and sources of the solute. The transport velocity $\mathbf{v}$ (m/s) is given by Darcy's law

$$\mathbf{v} = -\frac{\mathbf{k}}{\mu}(\nabla p - \rho \mathbf{g}), \tag{2}$$

in which $\mathbf{k}$ (m^2) refers to the hydraulic conductivity tensor, μ (Pa s) and ρ (kg/m^3) are the dynamic viscosity and the density of water respectively, and $\mathbf{g}$ (m/s^2) represents the acceleration of gravity vector. Pore water pressure, p (Pa), is computed from the solution of the transient fluid pressure equation for a slightly compressible flow problem

$$\phi c_t \frac{\partial p}{\partial t} = \nabla \cdot \left[\frac{\mathbf{k}}{\mu}(\nabla p - \rho \mathbf{g}) \right] + \hat{Q}. \tag{3}$$

Here $\hat{Q}$ (1/s) represents external fluid sinks and sources, and c_t (m s^2/kg) is the total system compressibility calculated from a linear combination of the compressibility of the water, β_w (m s^2/kg), and that of porous material, β_s (m s^2/kg):

$$c_t = \phi \beta_w + (1 - \phi)\beta_s. \tag{4}$$

These governing equations are spatially discretised using the hybrid FEM-node-centred FVM framework implemented in CSMP++ (Fig. 1). The equations are solved sequentially using operator splitting. First, fluid pressure is computed by solving the parabolic pressure equation (3) using FEM. Flow velocity field is then found by solving (2). Finally, FVM is used to solve the hyperbolic transport equation (1) explicitly [11,12,38].

The work presented in this paper is focused solving the transport equation (1) on a FV mesh constructed around the nodes of the finite element mesh (Fig. 1). Based on Green's theorem, the integration of the divergence of flux over each finite volume, V_i, can be transformed to the integration of the flux over its surface area, S_i, leading to:

$$\int_{V_i} \phi \frac{\partial c}{\partial t} dV = -\int_{V_i} \nabla \cdot \mathbf{v} c dV + q_i V_i = -\int_{S_i} \mathbf{n}_j \cdot \mathbf{v} c dS + q_i V_i, \tag{5}$$

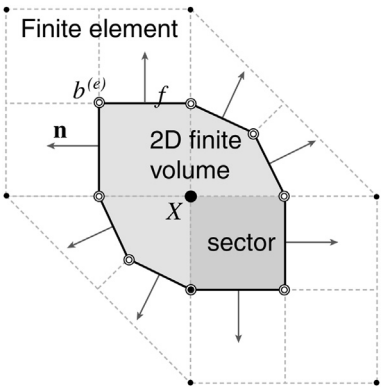

Fig. 1. 2D node-centred finite volumes implemented in CSMP++ (taken from Fig. 7a in [12]). Four neighbouring triangles and quadrilaterals share node X around which a FV cell shaded in grey is built using FE element barycentres $b^{(e)}$ and midpoints of element faces. FE elements are subdivided into sectors delimited by FV facets, f, with outward pointing normals, $\mathbf{n}$, and boundaries of finite elements.

where $\mathbf{n}_j$ is the outward pointing unit normal to facet j and q_i is the mean value of source q over V_i. Discretising equation (5) explicitly in time using a forward Euler approximation gives

$$c_i^{(t+\Delta t)} = c_i^{(t)} - R_i^{(t)} \Delta t + q_i^{(t)} \Delta t, \tag{6}$$

where R_i (kg/m^3 s) is the rate of change of concentration at FV cell V_i. Summing the solute flux f_j^c (kg/s) over all facets j leads to solute balance FB_i^c (kg/s) of FV cell, based on which R_i is computed as:

$$R_i = \frac{FB_i^c}{V_i^p} = \frac{\sum_j f_j^c}{V_i \phi_i} = \frac{\sum_j \mathbf{n}_j \mathbf{v}_j S_j c_j^{upstream}}{V_i \phi_i}, \tag{7}$$

where V_i^p (m^3) and ϕ_i (m^3/m^3) are the pore volume and porosity of cell V_i, and $\mathbf{v}_j$ (m/s), A_j (m^2) and $c_j^{upstream}$ (kg/m^3) are the flow velocity, facet area and upstream concentration for facet j. Upstream concentration is the concentration in the FV cell from where the fluid enters the current cell.

In this explicit first-order formulation, the size of the global time increment, Δt, has to be sufficiently small to satisfy the CFL condition, which ensures that the total outflow, $F_i^{out} = \sum_j \max(\mathbf{n}_j \mathbf{v}_j S_j, 0)$, does not exceed the pore volume of a FV cell:

$$\Delta t \leq \frac{V_i^p}{F_i^{out}}. \tag{8}$$

As discussed previously, meeting this condition can lead to prohibitively small time steps and as a consequence to long simulation runs. To address this problem, we apply DES to solve the transport equation. Its implementation is presented in detail in the following section.

2.2. Implementation of the DES transport scheme

In the DES scheme, solute concentration in each FV cell is associated with a so-called event, which holds properties such as local rate of concentration change, current time, and time scheduled for the processing of the event. Scheduled time is determined from local conditions, varying from cell to cell as a special feature of DES. The simulation is progressed by processing the sequence of events ordered from smallest to largest time stamp. Processing an event updates the concentration in the associated cell and also synchronises the states of its neighbouring cells to honour conservation laws. To ensure strict causality, a neighbouring event is allowed to be pre-empted for processing earlier than its scheduled time if a significant change to its concentration has occurred. All events with updated concentration are collected in a preemptive-event-processing (PEP) list to compute new local variation rates. Processed events are re-scheduled to calculate new schedule times based on updated local variation rates. After re-sorting the events according to updated schedule times, the simulation is progressed by processing the earliest events. In all studied cases, calculation of the change rate is the most time-consuming task but is easily parallelised when concentrations of neighbouring cells are available.

In the following we present this algorithm and its implementation in more details. To simplify notation, the subscript i indicating cell number is dropped.

2.2.1. Variables and data structures

Table 1 summarises all the variables used in the implemented DES algorithm. Solute concentration and other physical variables associated with the FV cell, are placed on either the centre node of the cell or the integration points of each FV

Table 1
Variables used in DES algorithm.

Group	Variable name	Symbol	Unit (in 3D)	Placement
FV nodal variables	Solute concentration	c	kg/m^3	FV centre node
	Concentration at upstream FV cell	$c^{upstream}$	kg/m^3	
	Volume of a FV cell	V	m^3	
	Porosity of a FV cell	ϕ	m^3/m^3	
	Pore volume of a FV cell	V^p	m^3	
	Total outflow from a FV cell	F^{out}	m^3/s	
	Concentration flux balance	FB^c	kg/s	
	Rate of concentration change	R	kg/m^3 s	
	Concentration source	q	kg/m^3 s	
FV facet variables	Area of FV facet	S	m^2	FV facet
	Unit normal vector of FV facet	n	m	
	Facet fluid flux	f^f	m^3/s	
	Facet concentration flux	f^c	kg/s	
Event variables	Current time stamp	$t_{current}$	s	Event object
	Scheduled time stamp for event processing	$t_{schedule}$	s	
	Target time increment for next event processing	Δt_{target}	s	
	Local CFL time increment	Δt_{CFL}	s	
	Target concentration change over Δt_{target}	Δc_{target}	kg/m^3	
	Cumulative concentration change	Δc_{cum}	kg/m^3	
	Flag variable for (re-)computing $t_{schedule}$	$Flag\_T$	bool	
	Flag variable for (re-)computing R	$Flag\_R$	bool	
Simulation variables	Simulation clock	T_{clock}	s	Simulation model
	Simulation end time	T_{end}	s	
	PEP time window	Δt_{PEP}	s	
	CFL multiplier	ω_{CFL}	–	
	PEP multiplier	ω_{PEP}	–	
	machine epsilon	ε	–	
	maximum finite value	∞	–	

facet or sector (Fig. 1). Each ***event*** object, corresponding to the concentration value in each FV cell, is a simulation unit in the DES algorithm. Via its member variables, the ***event*** object grants the solution variable the properties relevant to timing, changes and status. In the main iteration loop, simulation variables, including the PEP time window Δt_{PEP} and simulation clock T_{clock}, are used to track and monitor time progress.

A Fibonacci heap data structure [39], the ***EventQueue***, is created to contain the ***event*** objects sorted by their scheduled time stamps $t_{schedule}$ in increasing order. A sequence container, ***PEPList***, is used to store those active events with time stamps sufficiently close to T_{clock} and available for parallel computations.

2.2.2. DES algorithms

The implemented DES algorithm is outlined in the flowchart as displayed in Fig. 2, with its functions explained by the pseudocode as shown in Table 2. It consists of the following four basic computational steps.

Step 1: Initialisation of simulation

When the simulation starts, an *Initialisation*() function (Table 2) is called. ***event*** objects are created and added to the ***PEPList***, with their initial flags set to $Flag\_R = true$ and $Flag\_T = true$ to indicate that their local variation rates R and scheduled time stamps $t_{schedule}$ need to be computed as they are not initially available. The ***PEPList*** thus contains a full list of events at this initial stage of simulation. Simulation clock T_{clock} is initialised as 0.

Step 2: Variation rate computation and event scheduling

A new variation rate R and, if required, a new scheduled time $t_{schedule}$ for event processing, are calculated in this step. Each event e_{PEP} in ***PEPList*** has a status of $Flag\_R = true$ and thus the call *ComputeVariationRate* (e_{PEP}) (Table 2) is executed to compute the new local variation rate R and CFL time increment Δt_{CFL}, based on the concentration flux balance FB^c and the total outflow F^{out} of the associated FV cell. $Flag\_R$ is turned to *false* afterwards. If the time stamp of e_{PEP} needs to be (re-)scheduled as indicated by $Flag\_T = true$, then the call *Schedule* (e_{PEP}) (Table 2) computes a new target time increment $\Delta t_{target} = \omega_{CFL}\Delta t_{CFL}$ taking into account a user defined CFL coefficient ω_{CFL} ($0 < \omega_{CFL} \leq 1$), and a new scheduled time for event processing $t_{schedule}$. The target change of concentration Δc_{target} is set as a threshold value to determine whether e_{PEP} needs to be pre-empted for processing during the neighbour synchronisation procedure as described later in Step 4. If the computed $t_{schedule}$ is less than the proposed simulation end time T_{end}, e_{PEP} is added into the ***EventQueue*** for further computations. The ***PEPList*** is cleared at the end of this step.

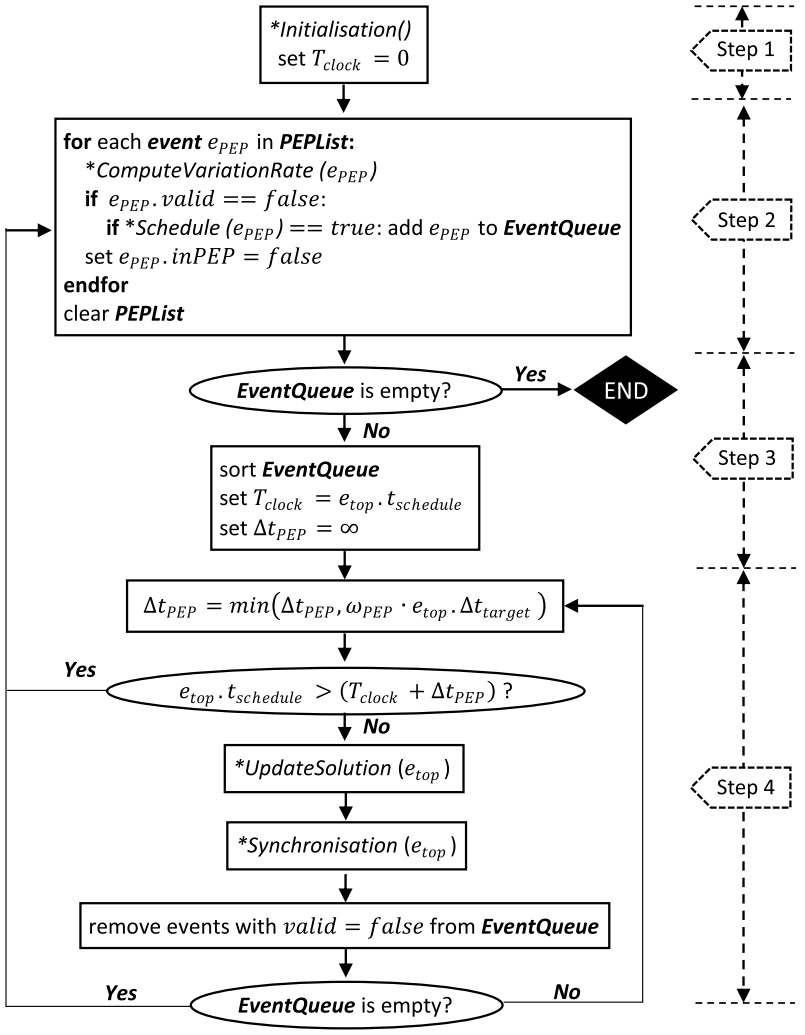

Fig. 2. Flowchart for implemented DES algorithm. Variables are explained in Table 1, and pseudocode of the functions (marked with *) is displayed in Table 2.

Step 3: Event queue sorting and simulation clock advancing

Active events, which are scheduled with a valid $t_{schedule}$ for processing in the future, have been added to the ***EventQueue*** in Step 2. If the ***EventQueue*** is empty, this means that no active event will be processed before T_{end}, and the simulation is terminated. Otherwise, the events in ***EventQueue*** are sorted based on their $t_{schedule}$ in increasing order. In this way, the event e_{top} at the top of the queue corresponds to the earliest event and needs to be processed first. The new T_{clock} is advanced to the scheduled time stamp of e_{top}. The PEP time increment Δt_{PEP}, which defines the width of the time window for projecting scheduled events with close timestamps to the current T_{clock} level, is initialised to positive infinity. The size of Δt_{PEP} is to be adaptively adjusted as stated in the next step.

Step 4: Event processing and synchronisation

The event e_{top} at the top of the ***EventQueue*** is extracted from the queue and Δt_{PEP} is updated to a smaller value between itself and the product of Δt_{target} of e_{top} and a tuning coefficient ω_{PEP} ($0 \leq \omega_{PEP} \leq 1$). If the scheduled time $t_{schedule}$ of e_{top} is later than $T_{clock} + \Delta t_{PEP}$ then all events scheduled for processing within the time window T_{clock} to $T_{clock} + \Delta t_{PEP}$ have been extracted from the ***EventQueue*** and added to the ***PEPList***. Computation is then continued with Step 2. Otherwise, the simulation will proceed to execute the function call *Process* (e_{top}) (Table 2) which first calls *UpdateSolution* (e_{top}) to update the concentration and cumulative concentration changes in the cell associated with e_{top} and set its current time stamp $t_{current}$ to T_{clock}. The event is then added to ***PEPList*** (with its status changed to $Flag\_R = true$) to update its local variation rate later. The neighbouring events of e_{top} need to be synchronised to preserve the conservation laws. In order to make sure

Table 2
Pseudocode for functions used in the DES algorithm.

function *Initialisation* ():
1: **for** each FV cell:
2: compute cell volume V and pore volume $V^p = V\phi$
3: initialise an associated ***event*** object and add it to ***PEPList***
4: set initial event flags $Flag\_R = true$ and $Flag\_T = true$
5: **endfor**
endfunction

function *ComputeVariationRate* (***event*** e):
1: **for** each facet of the FV cell associated to e:
2: compute facet fluid flux $f^f = \mathbf{v} \cdot \mathbf{n}S$
3: **if** $f^f > 0$: update FV total outflow $F^{out} \leftarrow F^{out} + f^f$
4: determine upstream concentration $c^{upstream}$
5: compute facet concentration flux $f^c = f^f c^{upstream}$
6: update FV concentration flux balance $FB^c \leftarrow FB^c + f^c$
7: **endfor**
8: compute FV variation rate $R = FB^c / V^p$
9: **if** $F^{out} < \varepsilon$: set CFL time increment $\Delta t_{CFL} = \infty$
10: **else**: compute $\Delta t_{CFL} = V^p / F^{out}$
11: set e status $Flag\_R = false$
endfunction

function *Schedule* (***event*** e):
1: Compute target change of solution $\Delta c_{target} = \omega_{CFL} \Delta t_{CFL} |R + q|$
2: **if** $\Delta c_{target} < \varepsilon$: set $\Delta c_{target} = \varepsilon$ and target time increment $\Delta t_{target} = \infty$
3: **else**: compute $\Delta t_{target} = \omega_{CFL} \Delta t_{CFL}$
4: set e status $Flag\_T = false$
5: **if** $(t_{current} + \Delta t_{target}) \geq T_{end}$: **return** *false*
6: **else**: compute scheduled time stamp $t_{schedule} = t_{current} + \Delta t_{target}$ and **return** *true*
endfunction

function *UpdateSolution* (***event*** e):
1: compute time increment $\Delta t = T_{clock} - t_{current}$
2: compute solution change $\Delta c = -R\Delta t + q\Delta t$
3: update solution value $c \leftarrow c + \Delta c$
4: update cumulative solution change $\Delta c_{cum} \leftarrow \Delta c_{cum} + \Delta c$
5: set current time stamp $t_{current} = T_{clock}$
6: add e to ***PEPList*** and set e status $Flag\_R = true$
endfunction

function *Process* (***event*** e):
1: **if** e status $Flag\_R = false$: execute *UpdateSolution* (e)
2: **for** each neighbouring ***event*** e_{nb} of e:
3: **if** e_{nb} status $Flag\_R = false$:
4: execute *UpdateSolution* (e_{nb})
5 **if** $|e_{nb}.\Delta c_{cum}| \geq |e_{nb}.\Delta c_{target}|$: execute *Process* (e_{nb})
6: **endif**
7: **endfor**
8: reset cumulative solution change $\Delta c_{cum} = 0$
9: set e status $Flag\_T = true$
endfunction

that all neighbouring cells hold the concentration at the simulation time T_{clock}, the *UpdateSolution* (e_{nb}) function is applied for each neighbour event e_{nb} that has not been updated. If accumulated concentration change Δc_{cum} of e_{nb} exceeds the threshold value, its target concentration change Δc_{target}, e_{nb} is pre-empted for processing and the function call *Process* (e_{nb}) is performed to synchronise its neighbour events. This synchronisation process is iterated until no neighbour event would be pre-empted for processing. Once an event is processed, its accumulated concentration change Δc_{cum} is reset to 0, and its scheduled time $t_{cheduled}$ becomes invalid and needs to be re-computed by setting its $Flag\_T$ to *true*. All processed events as marked by $Flag\_T = true$ are removed from ***EventQueue***. If ***EventQueue*** becomes empty which means all stored events have been processed, then the simulation goes back to Step 2. Otherwise, this step is repeated to process the next available top event in ***EventQueue***.

DES events are exchanged between ***PEPList*** and ***EventQueue*** during the simulation. ***PEPList*** is a dynamically changing list that contains active events at a certain time level which are to be updated for rate of concentration change and, if required, (re-)scheduled. Only valid events with scheduled time less than the simulation end time are passed to ***EventQueue*** for further computations. ***EventQueue*** contains all valid events in the simulation, sorted on their scheduled time stamps. The ***PEPList*** is always emptied before the simulation clock is advanced to the first event stored in ***EventQueue***. Events falling into the current PEP time window are processed and removed from ***EventQueue*** and added to the ***PEPList*** for re-computation of local variation rate and re-scheduling, together with their synchronised neighbour events. In this way, all other inactive or

invalid events are excluded from the simulation, limiting the expense of computational resources to only a small fraction of the whole collection of events. Simulation efficiency is therefore expected to be greatly improved.

Notably, in Step 2, during execution of the *ComputeVariationRate*() and *Schedule*() functions, the events stored in the ***PEPList*** are independent of each other. This provides an opportunity to incorporate parallel computations into Step 2 without any causality issues. Here we utilise the OpenMP shared memory multithreading application programming interface [40].

2.3. Performance analysis

In the following two sections, we evaluate the performance of our new unstructured-grid hybrid FEM-FVM implementation of DES with two models. First, it is verified with an analytical solution of the advection equation on a simple 1D model. Then, its computational accuracy and efficiency are evaluated for a geometrically complex field data-based three dimensional model of fractured rock with discrete fracture representations.

To quantify simulation accuracy, we use two standard statistical metrics, the Nash-Sutcliffe model efficiency (EF) [41] and the root mean square error (RMSE). EF describes the predictive accuracy of a model. It varies from 1 (perfect fit) to negative infinity. RMSE measures the difference between predicted and observed values, and a value close to 0 indicates good agreement. Corresponding mathematical expressions are:

$$EF = 1 - \frac{\sum_{i=1}^{n}(O_i - P_i)^2}{\sum_{i=1}^{n}(O_i - \overline{O})^2}, \tag{9}$$

and

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(O_i - P_i)^2}{n}}, \tag{10}$$

where n is number of observations, O_i is reference value for the concentration at the FV cell i, P_i is corresponding DES simulated value, and $\overline{O}$ is arithmetic mean of the reference values.

3. Verification of DES scheme for 1D models

Analytical techniques for the solution of the advection equation are generally restricted to simple problems with specific initial and boundary conditions. Here we adopt the analytical solutions to the one dimensional advection equation with spatially variable coefficients [42], where the velocity field is set as a linear function of distance, $u(x,t) = u_0 x$. Based on different initial and boundary conditions, we have conducted an analysis for two scenarios, a steady point source and a quasi-Gaussian initial profile.

3.1. Steady point source

For this case, a steady point concentration c_0 is applied at location $x = x_0$. It models the practice problem of pollutant transport in an open channel, where the polluted water is diluted by the unpolluted lateral inflow entering the channel, and the concentration of pollutant decreases with distance. By setting initial and boundary conditions to $c(x,0) = 0$ and $c(x_0,t) = c_0$, the evolution in time of the concentration profile is given by Zoppou and Knight [42]:

$$c(x,t) = \frac{c_0}{x} H\left[u_0 t - ln(x/x_0)\right] \tag{11}$$

where H is the Heaviside function.

Fig. 3a illustrates the comparison of DES simulation results (cell size = 0.1) and the analytical solution at $t = 5$, with $u_0 = 1$, $x_0 = 1$ and $c_0 = 100$. Simulated results agree well with the analytical solution. They both reflect a gradual decrease of concentration with distance from x_0. The close-to-one EF value (0.9982) and small RMSE value (0.7049) further indicate close agreement between the results, thereby verifying the accuracy of the implemented DES scheme for this case.

Fig. 3a only shows the DES and analytical results for $\omega_{CFL} = 0.5$ and $\omega_{PEP} = 0.5$. We conducted simulations with ω_{CFL} and ω_{PEP} values ranging from 0.1 to 0.9. Simulation accuracy is not affected by the chosen value of either adjusting coefficient in this testing case, as is indicated by the unchanged EF and RMSE values in Tables 3 and 4.

The cumulative number of DES events that occurred for each FV cell are plotted against the cell flow velocities in Fig. 4a. The diagram shows that the number of cumulative events is positively and nearly linearly correlated with flow velocity. This verifies that FV cells with a larger local variation rate are updated with higher frequencies, which is a fundamental characteristic of DES. When the value for the control parameter ω_{CFL} is decreased, the cumulative event count increases, attributed to the reduced time stamps as computed by the *Schedule*() function (Table 2). The resulting decrease of the number of iteration steps is shown in Table 3.

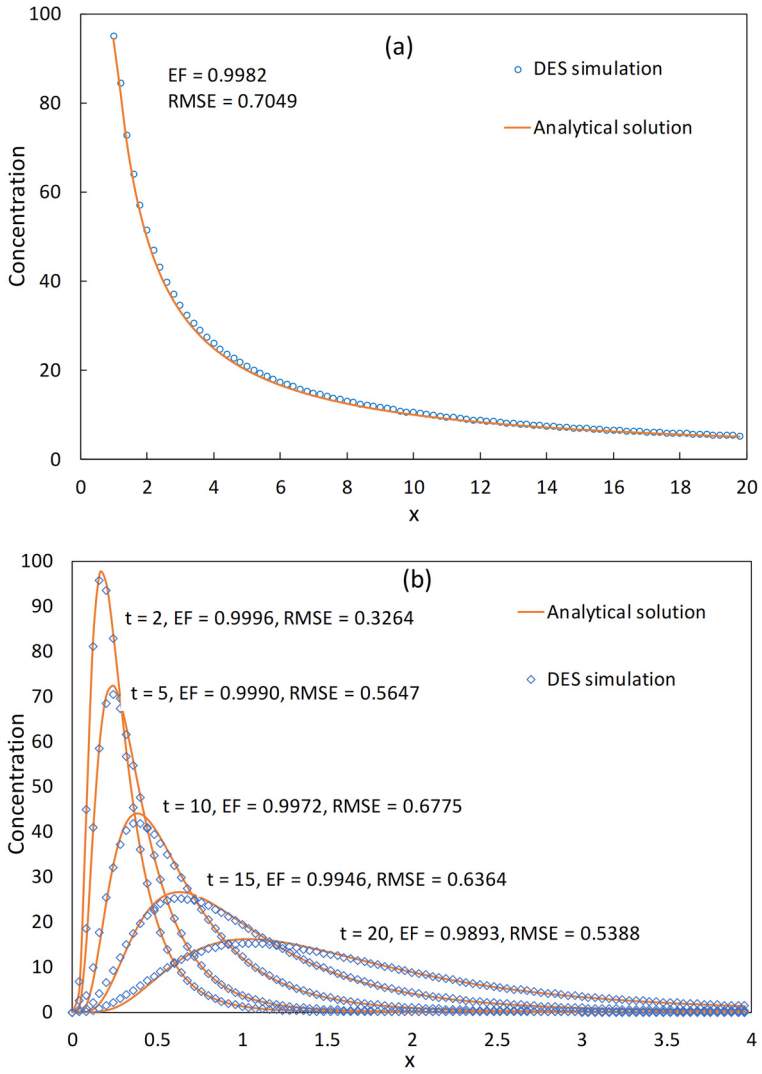

Fig. 3. Comparison of concentration profiles computed by DES and analytical solutions for 1D tracer advection in a non-uniform flow field featured by (a) a steady point source (Equation (11)) with $u_0 = 1$, $x_0 = 1$ and $c_0 = 100$ at $t = 5$; (b) a quasi-Gaussian initial profile (Equation (12)) with $u_0 = 0.1$, $x_0 = 0.2$, $\sigma = 0.2$ and $c_0 = 100$ at different time points. Only DES results computed with $\omega_{CFL} = 0.5$ and $\omega_{PEP} = 0.5$ are displayed, but the simulated concentration profiles are not sensitive to either coefficient with values ranging from 0.1 to 0.9 (Tables 3 and 4).

Table 3
Statistical results for simulation performance with varied ω_{CFL} values and a fixed ω_{PEP} value of 0.5, on 1D models.

ω_{CFL}	Mean *PEPList* size ($\times 10^3$)	Iteration steps ($\times 10^3$)	Total DES events ($\times 10^6$)	Execution time (s)*	EF	RMSE
1D model with steady point source						
0.1	1.09	16.77	18.29	155.2	0.998	0.705
0.3	1.08	5.50	5.96	51.2	0.998	0.705
0.5	1.04	3.19	3.32	30.3	0.998	0.705
0.7	0.97	2.19	2.13	19.2	0.998	0.705
0.9	0.99	1.61	1.60	13.9	0.998	0.705
1D model with initial quasi-Gaussian profile						
0.1	1.33	7.92	10.53	110.4	0.998	0.691
0.3	1.30	2.64	3.44	36.3	0.999	0.564
0.5	1.28	1.58	2.03	21.6	0.999	0.481
0.7	1.28	1.13	1.44	15.3	0.998	0.583
0.9	1.28	0.88	1.12	11.7	0.998	0.702

*Execution time is obtained on a single thread.

Table 4
Statistical results for simulation performance with varied ω_{PEP} values and a fixed ω_{CFL} value of 0.5, on 1D models.

ω_{PEP}	Mean *PEPList* size ($\times 10^3$)	Iteration steps ($\times 10^3$)	Total DES events ($\times 10^6$)	Execution time (s)*	EF	RMSE
1D model with steady point source						
0.1	0.29	13.40	3.84	32.5	0.998	0.705
0.3	0.64	4.93	3.14	27.0	0.998	0.705
0.5	1.04	3.19	3.32	30.3	0.998	0.705
0.7	1.26	2.22	2.80	27.0	0.998	0.705
0.9	1.43	1.68	2.41	24.9	0.998	0.705
1D model with initial quasi-Gaussian profile						
0.1	0.56	7.39	4.10	34.1	0.999	0.484
0.3	0.98	2.59	2.54	24.6	0.999	0.515
0.5	1.28	1.58	2.03	21.6	0.999	0.481
0.7	2.18	1.13	2.46	25.9	0.999	0.500
0.9	2.55	0.88	2.23	23.8	0.999	0.520

*Execution time is obtained on a single thread.

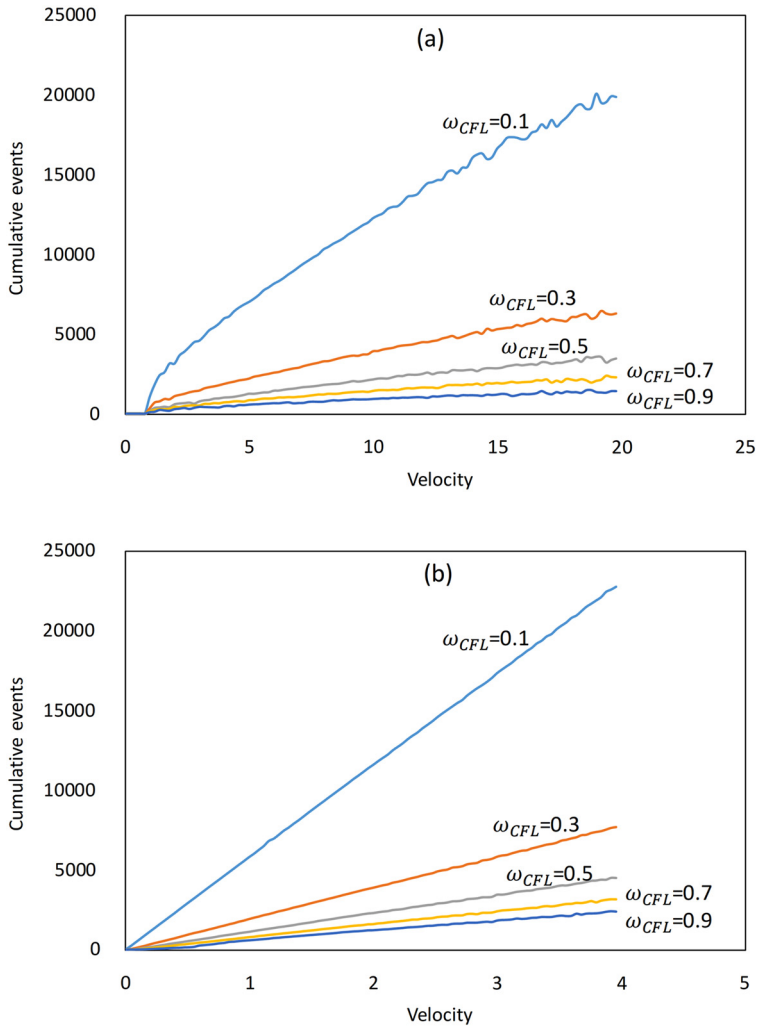

Fig. 4. Cumulative DES events in FV cells plotted against flow velocity with different ω_{CFL} values for 1D tracer advection in a non-uniform flow field featured by (a) a steady point source with $u_0 = 1$, $x_0 = 1$ and $c_0 = 100$ at $t = 5$; (b) a quasi-Gaussian initial profile with $u_0 = 0.1$, $x_0 = 0.2$, $\sigma = 0.2$ and $c_0 = 100$ at $t = 20$. Only the results with $\omega_{PEP} = 0.1$ are displayed, but the resultant cumulative events are not sensitive to this coefficient for values between 0.1 and 0.9 (Tables 3 and 4).

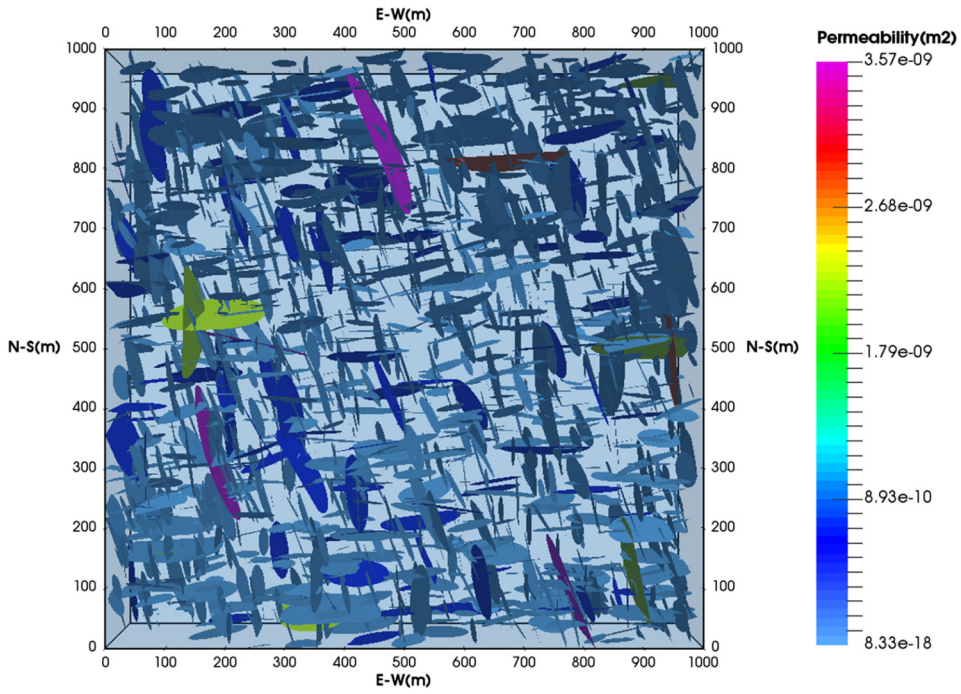

Fig. 5. Permeability distribution of model FRACS2000 embedding 2000 disc-shaped fractures. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

3.2. Quasi-Gaussian profile

By using the same velocity field as calculated from the linear function of distance, but setting the initial condition as a quasi-Gaussian concentration profile, the analytical solution of the advection equation becomes:

$$c(x,t) = \frac{M_0}{x\sigma\sqrt{2\pi}} exp\left\{\frac{-[ln(x/x_0) - u_0 t]^2}{2\sigma^2}\right\} \tag{12}$$

where σ is standard deviation, and $M_0 = c_0 x_0 \sigma\sqrt{2\pi}$ is the mass contained in the profile [42].

Fig. 3b displays the results at different time points, with $u_0 = 0.1$, $x_0 = 0.2$, $\sigma = 0.2$ and $c_0 = 100$. According to the analytical solution, peak concentration decays exponentially with time while solute mass is conserved. These behaviours are captured precisely by the DES simulation (cell size = 0.01). Again, near unity EF values and small RMSE values quantitatively verify the accuracy of the simulated results.

Similar to the simulation case with a constant point source, the simulated concentration profiles are almost unaffected by the chosen value of either ω_{CFL} or ω_{PEP}, as is indicated by the EF and RMSE values in Tables 3 and 4. A decrease of the ω_{CFL} value increases the number of iteration steps, leading to increased total number DES events and execution time (Table 3). A reduction of ω_{PEP} values also contributes to an increase of the iteration count, but leads a decreased size of the ***PEPList***, which represents the active events at any iteration step (Table 4). These combined effects lead to a small variation of the total DES event count and execution time with varying ω_{PEP} values.

A positive correlation is also observed between the number of cumulative events and flow velocity within FV cells in this simulation case (Fig. 4b). This once again demonstrates the ability of DES to asynchronously update different cells according to their local variation rates, focusing simulation effort on the processing of the most rapidly evolving model regions.

4. Performance evaluation on a complex 3D fracture model

The previous simple 1D tests verified the accuracy and correctness of the implemented DES algorithm. In this section we further evaluate its computational performance on a complex field data-based model.

4.1. Model setup

The 3D disc-shaped fracture model, FRACS2000 (Fig. 5), is based on a stochastically generated fracture geometry (DFN) produced by Paul LaPointe (Golder Associates, Inc.) for the San Andres formation in west Texas. It was converted into a volumetric mesh with lower dimensional fracture representations by Matthäi, Mezentsev and Belayneh [43] and has been used

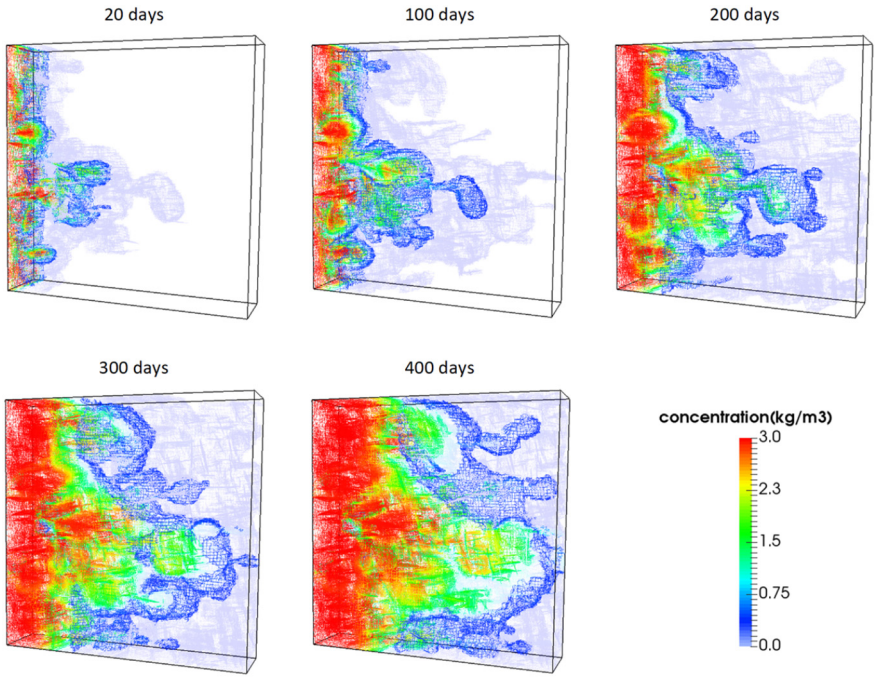

Fig. 6. Snapshots of DES simulation results ($\omega_{CFL} = 0.5$, $\omega_{PEP} = 0.5$) of tracer advection on model FRACS2000. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

Table 5
Statistical results for simulation performance with varied ω_{CFL} values and a fixed ω_{PEP} value of 0.5, on 3D model FRACS2000.

ω_{CFL}	mean *PEPList* size ($\times 10^3$)	iteration steps ($\times 10^6$)	total DES events ($\times 10^9$)	execution time ($\times 10^3$ s)*	EF**	RMSE**
0.1	0.93	2.93	2.74	26.76	0.998	0.011
0.3	0.92	0.98	0.89	8.80	0.982	0.033
0.5	0.91	0.59	0.53	5.28	0.951	0.055
0.7	0.90	0.42	0.38	3.77	0.905	0.076
0.9	0.89	0.33	0.29	2.95	0.858	0.095

*Execution time is obtained from the parallel simulation with 20 threads.
**EF and RMSE values are averaged from the results at 3 central lines along the axes as shown in Fig. 7.

previously for numerical simulations of single-phase and multiphase flows in fractured geological formations [4,12,38,44]. This model ($1 \times 1 \times 0.2$ km) contains 2000 fractures making up sets with 2 prominent orientations. The fracture diameter distribution is log normal and the fracture aperture (separation of the 2 walls) varies between 0.1 and 3.5 mm. The resulting permeability values for fractures and the rock matrix vary over 9 orders of magnitude across the domain (Fig. 5). This strongly heterogeneous and spatially adaptively refined model consists of 223,705 nodes and 1,113,580 finite elements in total. Fractures are represented by locally refined lower dimensional surface elements. Due to local variation of cell size and permeability leading to small local time step sizes, the FRACS2000 model is particularly challenging for TDS methods and therefore ideal for performance evaluation of DES.

During simulations, a constant pressure differential of 20 MPa was applied between the opposing left and right boundaries, amounting to a hydrostatic far-field fluid pressure gradient giving rise to a time-invariant flow field. Due to the strongly spatially-correlated permeability structure and large range of permeability values, flow velocity in the domain varied from 5.36×10^{-12} to 1.54×10^{-2} m/s. With no flow conditions set at all other model boundaries, non-reactive tracer with a concentration of 3.0 kg/m^3 was injected continuously through the left boundary for 400 days. The tracer advection was simulated with both TDS and DES methods.

4.2. Simulation accuracy

Fig. 6 displays snapshots of the tracer distribution through time, as simulated by DES with ω_{CFL} and ω_{PEP} both set to 0.5. As expected, tracer is transported gradually from left to right while the concentration front feathers out into the fractures. Fast advection and high concentrations are observed in clusters of larger fractures with high permeability values. Note that the larger a fracture is, the more likely it is interconnected with others.

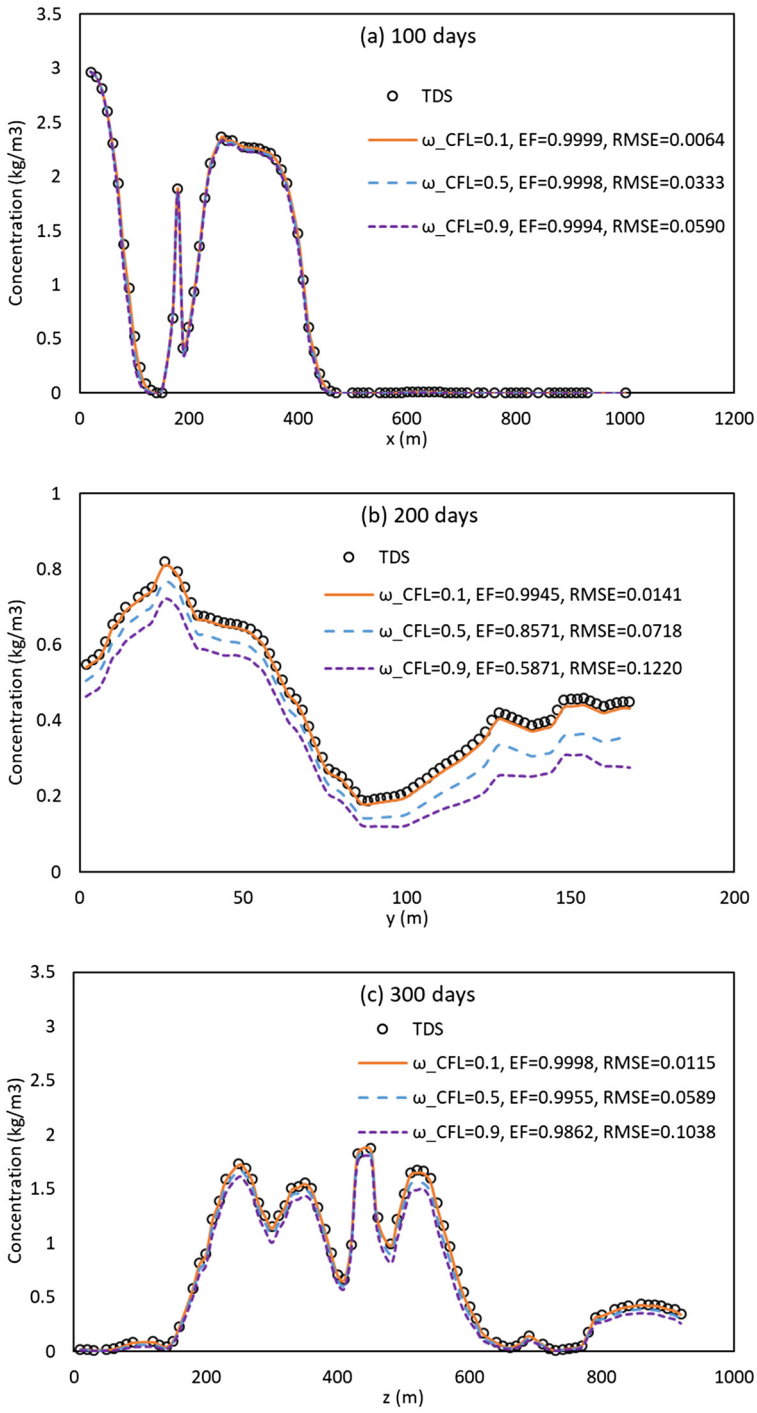

Fig. 7. Concentration profiles for the central line along (a) x axis at 100 days, (b) y axis at 200 days and (c) z axis at 300 days on model FRACS2000, as computed with TDS and DES methods with varied ω_{CFL} values.

To compare the DES simulation results with those of the TDS simulation in an efficient way, we have graphed concentration profiles at selected time points derived from both methods along the central line parallel to each of the three coordinate axes in Fig. 7. For comparison, the DES results with different ω_{CFL} values are shown. In general, a close agreement is observed between DES and TDS results for all concentration profiles, especially for simulation cases with a small ω_{CFL} value of 0.1. As the ω_{CFL} value increases, simulation accuracy tends to decrease by underestimating the concentrations, as seen in Fig. 7(b) and 7(c). This reduced accuracy is also reflected in decreased EF and increased RMSE values (Table 5). Nevertheless, a satisfactory accuracy (EF $>$ 0.85 and RMSE $<$ 0.01) in comparison to the TDS solution is achieved even with

Table 6
Statistical results for simulation performance with varied ω_{PEP} values and a fixed ω_{CFL} value of 0.5, on 3D model FRACS2000.

ω_{PEP}	Mean *PEPList* size ($\times 10^3$)	Iteration steps ($\times 10^6$)	Total DES events ($\times 10^9$)	Execution time ($\times 10^3$ s)*	EF**	RMSE**
0.1	0.31	1.70	0.53	5.63	0.952	0.055
0.3	0.60	0.89	0.53	5.43	0.952	0.053
0.5	0.91	0.59	0.53	5.28	0.951	0.055
0.7	1.25	0.42	0.53	5.27	0.950	0.055
0.9	1.60	0.33	0.52	5.61	0.955	0.054

*Execution time is obtained from the parallel simulation with 20 threads.
**EF and RMSE values are averaged from the results at 3 central lines along the axes as shown in Fig. 7.

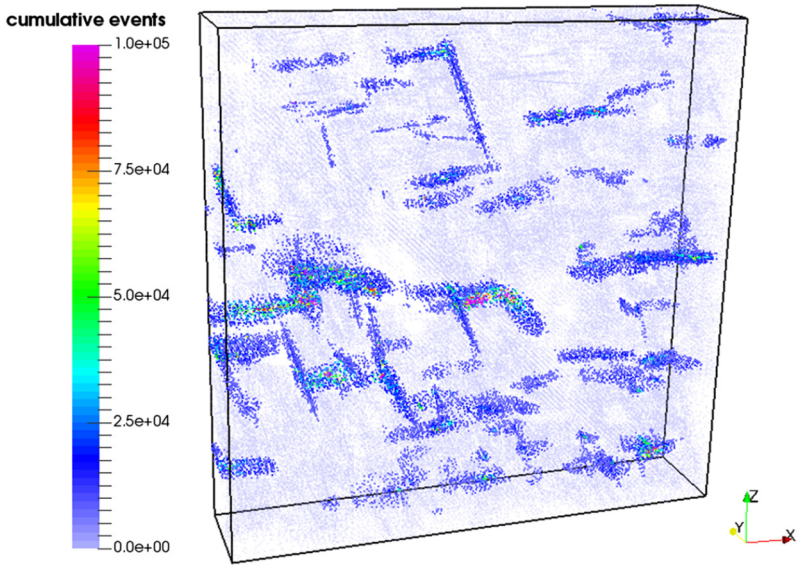

Fig. 8. Distribution of cumulative DES events on model FRACS2000.

a ω_{CFL} value as large as 0.9. Simulation accuracy is not sensitive to the input value of ω_{PEP} in this test case, see results shown in Table 6.

4.3. Computational efficiency

While obtaining highly similar simulation results, DES shows significantly improved computational performance characteristics by comparison with TDS. When ω_{CFL} and ω_{PEP} are both set to 0.5, the total events processed by DES ($= 5.07 \times 10^8$) are only 0.76% of those processed by TDS ($= 6.69 \times 10^{10}$). The underlying reason is evident from Fig. 8 which shows the spatial distribution of cumulative events in the DES simulation. The number of events accumulated by different FV cells varies strongly from 0 to 10^5 across the domain. Evidently, the FV cells representing highly permeable fractures are processed far more frequently due to the orders-of-magnitude greater flow velocities, while the frequency of updates in the rock matrix and in low permeability fractures are considerably lower. This reflects the focusing of the computational effort on rapidly evolving regions in the domain, thereby reducing the waste of computational work on the inactive parts. Conversely, during TDS simulation, a tiny time step (2.6 min), corresponding to the tightest local CFL condition in the domain, is forced on all FV cells, leading to a large total number of iterations (2.2×10^5) for each FV cell. As a result, for a comparative simulation accuracy, the execution time consumed by DES (9.5 hours) is only 0.93% of that required by TDS (1018.6 hours). This leads to a very significant speedup of around 107.

By examining the accuracy measures shown in Tables 5 and 6, we are able to further qualify the influence of the input ω_{CFL} and ω_{PEP} values on simulation performance. Similar to our findings for the 1D test cases, increasing the value of ω_{CFL} from 0.1 to 0.9 leads to reduced iteration count and therefore decreased execution time, at the cost of slightly reduced simulation accuracy (Table 5). Increasing the value of ω_{PEP} from 0.1 to 0.9 would also reduce the iteration step count, but as it is accompanied by an increase of the size of the ***PEPList***, the overall impact on total DES events, execution time and simulation accuracy is small (Table 6).

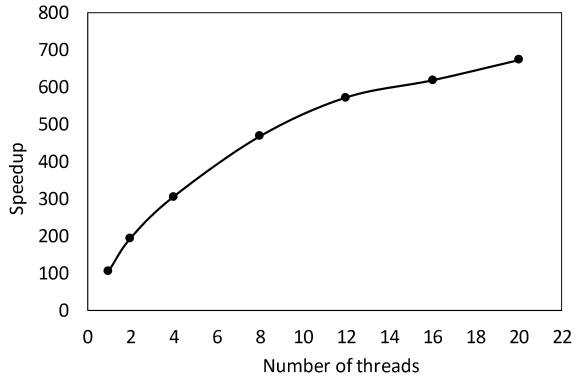

Fig. 9. Speedups (ratio between TDS and DES execution times) achieved by DES for the simulations of tracer advection through model FRACS2000 with different numbers of threads and input coefficients $\omega_{CFL} = 0.5$ and $\omega_{PEP} = 0.5$.

4.4. Scalability analysis on parallel computations

Thus far, all results were obtained with a serial DES program without relying on any parallel computations, except for the algebraic multigrid solver that was used once to compute the steady-state fluid pressure distribution at the onset of the simulation. Despite dramatically improved simulation efficiency, a simulation run on such a large 3D model still requires a considerable amount of time to complete. Further inspection of the runtime of each of the DES procedures, shows that 91.24% of the execution time is spent on Step 2 (Fig. 2), computing local variation rates with the function *ComputeVariationRate*() as well as the scheduling of events stored in the ***PEPList*** with the function *Schedule*(). As mentioned earlier, events in the ***PEPList*** are independent from each other, and therefore can be executed in parallel with multithreads without any causality issues. A parallelisation of this heaviest calculation effort in the simulation holds the biggest promise for further improvement of computational efficiency.

For parallelisation, we apply the OpenMP paradigm [40] and conduct a strong scalability test on the model FRACS2000 with numbers of threads varying from 1 to 20. The tests are performed on a Linux-based HPC cluster at The University of Queensland containing 66 compute nodes. We utilise a single node of this supercomputer, which is comprised of two 10-core Intel® Xeon® E5-2660 v3 2.60 GHz CPUs. Results are presented in Fig. 9. The speedup is calculated by dividing the TDS execution time on a single thread by the DES execution time with different number of threads. We see a general increase in speedup with increased thread number. A more linear and rapid increase from 107.2 to 466.5 is observed when initially increasing the threads from 1 to 8, followed by a more gradual increase to 671.6 when further increasing the threads to 20. This slowdown may be explained by the limited number of active events (908 on average) available in the ***PEPList***, which results in a decrease of workload per thread for larger number of threads and hence the dominance of synchronisation costs. It is expected that transport equations requiring computationally more expensive updates or problems with a larger ***PEPList*** will achieve better parallel speedups. This needs further investigation in future studies. Nevertheless, with the use of parallel DES computations, we have achieved remarkable performance improvement by further reducing the execution time from 9.5 hours to 1.5 hours.

5. Conclusions

In this paper we present a new parallel DES algorithm for hybrid FEM–FVM simulations performed on unstructured grids. It is implemented in the node-centred FV framework of the CSMP++ modelling platform, for the efficient simulation of solute transport through heterogeneous porous media in a stationary flow velocity field. Compared with conventional synchronous time-driven simulation, the implemented DES scheme has the following advantages: (1) efficiently removing the global CFL restriction via asynchronous updates of individual FV cells based on their physically determined local variation rates and temporal scales; and (2) focusing computational effort and resources on the active cells where fast solute transport occurs while excluding inactive or idle cells from computations by means of event sorting and synchronisation operations. DES greatly improves computational efficiency while retaining numerical stability and accuracy, as demonstrated by test cases. The PEP (preemptive-event-processing) method is incorporated in the presented new implementation of DES to facilitate parallel simulations with OpenMP. For this purpose, a dynamically changing ***PEPList*** is created containing active events with sufficiently closely scheduled time stamps. In this way, parallel computations can be applied to proximal events for the calculations of variation rates and time stamps, accelerating these most costly computations of the simulation.

We have demonstrated with a complex 3D unstructured model consisting of over 1 million adaptively refined elements, that execution time is significantly reduced from 1018.6 hours for TDS simulation to 9.5 hours for serial DES simulation, and further to only 1.5 hours for parallel DES simulation on 20 threads. It follows that the presented parallel DES–PEP scheme enables efficient simulation of solute transport in realistic and complex systems at large scales.

The focus of this paper is on simulating solute transport, but the presented parallel DES framework is generic and can be applied to other problems, such as the simulation of two phase flows through porous media. This work is in progress and will be reported in our future publications.

Acknowledgements

The presented research is funded by the Carbon Capture and Storage Research Development and Demonstration Fund (CCS49356) "Australian subsurface carbon sequestration Simulator" awarded by the Department of Industry, Innovation and Science, Australian Government. Dr. Yuri Omelchenko is thanked for providing additional technical context about his publications on the subject and for sharing his enthusiasm about the potential of DES as a fundamentally new and powerful computational method. The authors also thank Dr. Andrea Codd for the assistance in English polishing.

References

[1] L.W. Lake, Enhanced Oil Recovery, Prentice Hall, Englewood Cliffs, N.J, 1989.
[2] O.A. Cirpka, Intrinsic remediation in natural-gradient systems, in: P.K. Kitanidis, P.L. McCarty (Eds.), Delivery and Mixing in the Subsurface: Processes and Design Principles for In Situ Remediation, Springer, New York, NY, 2012, pp. 217–238.
[3] K. Pruess, J.S.Y. Wang, Y.W. Tsang, On thermohydrologic conditions near high-level nuclear wastes emplaced in partially saturated fractured tuff, 1: simulation studies with explicit consideration of fracture effects, Water Resour. Res. 26 (1990) 1235–1248.
[4] S.K. Matthäi, H.M. Nick, C. Pain, I. Neuweiler, Simulation of solute transport through fractured rock: a higher-order accurate finite-element finite-volume method permitting large time steps, Transp. Porous Media 83 (2010) 289–318.
[5] D.D. Laumbach, A High-Accuracy Finite-Difference Technique for Treating the Convection–Diffusion Equation, 1975.
[6] T.N. Narasimhan, P.A. Witherspoon, An integrated finite difference method for analyzing fluid flow in porous media, Water Resour. Res. 12 (1976) 57–64.
[7] P.S. Huyakorn, G.F. Pinder, The Computational Methods in Subsurface Flow, Academic Press, New York, USA, 1983.
[8] J. Istok, Groundwater Modeling by the Finite Element Method, American Geophysical Union, 1989.
[9] L.J. Durlofsky, A triangle based mixed finite element – finite volume technique for modeling two phase flow through porous media, J. Comput. Phys. 105 (1993) 252–266.
[10] R. Huber, R. Helmig, Multiphase flow in heterogeneous porous media: a classical finite element method versus an implicit pressure–explicit saturation-based mixed finite element–finite volume approach, Int. J. Numer. Methods Fluids 29 (1999) 899–920.
[11] S. Geiger, S. Roberts, S.K. Matthäi, C. Zoppou, A. Burri, Combining finite element and finite volume methods for efficient multiphase flow simulations in highly heterogeneous and structurally complex geologic media, Geofluids 4 (2004) 284–299.
[12] A. Paluszny, S.K. Matthäi, M. Hohmeyer, Hybrid finite element–finite volume discretization of complex geologic structures and a new simulation workflow demonstrated on fractured rocks, Geofluids 7 (2007) 186–208.
[13] G. Dagan, Solute transport in heterogeneous porous formations, J. Fluid Mech. 145 (1984) 151–177.
[14] M. Sahimi, Flow and Transport in Porous Media and Fractured Rock, Wiley–VCH Verlag GmbH & Co. KGaA, 2011.
[15] K. Stüben, Algebraic Multigrid (AMG): An Introduction with Applications, GMD-Forschungszentrum Informationstechnik, 1999.
[16] T. Unfer, J.-P. Boeuf, F. Rogier, F. Thivet, An asynchronous scheme with local time stepping for multi-scale transport problems: application to gas discharges, J. Comput. Phys. 227 (2007) 898–918.
[17] Y.A. Omelchenko, H. Karimabadi, Self-adaptive time integration of flux-conservative equations with sources, J. Comput. Phys. 216 (2006) 179–194.
[18] W.L. Kleb, J.T. Batina, M.H. Williams, Temporal adaptive Euler/Navier-Stokes algorithm involving unstructured dynamic meshes, AIAA J. 30 (1992) 1980–1985.
[19] X.D. Zhang, J.Y. Trepanier, M. Reggio, R. Camarero, Time-accurate local time stepping method based on flux updating, AIAA J. 32 (1994) 1926–1929.
[20] C. Dawson, R. Kirby, High resolution schemes for conservation laws with locally varying time steps, SIAM J. Sci. Comput. 22 (2001) 2256–2281.
[21] E.M. Constantinescu, A. Sandu, Multirate timestepping methods for hyperbolic conservation laws, J. Sci. Comput. 33 (2007) 239–278.
[22] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (1984) 484–512.
[23] R.D. Hornung, J.A. Trangenstein, Adaptive mesh refinement and multilevel iteration for flow in porous media, J. Comput. Phys. 136 (1997) 522–545.
[24] C.C. Chueh, M. Secanell, W. Bangerth, N. Djilali, Multi-level adaptive simulation of transient two-phase flow in heterogeneous porous media, Comput. Fluids 39 (2010) 1585–1596.
[25] M. Gedeon, D. Mallants, Sensitivity analysis of a combined groundwater flow and solute transport model using local-grid refinement: a case study, Math. Geosci. 44 (2012) 881–899.
[26] B. Amaziane, M. Bourgeois, M. El Fatini, Adaptive mesh refinement for a finite volume method for flow and transport of radionuclides in heterogeneous porous media, Oil Gas Sci. Technol. – Rev. IFP Energies nouvelles 69 (2014) 687–699.
[27] A. Dell'Oca, G.M. Porta, A. Guadagnini, M. Riva, Space–time mesh adaptation for solute transport in randomly heterogeneous porous media, J. Contam. Hydrol. (2017).
[28] R.M. Fujimoto, Parallel and distributed simulation systems, in: Proceeding of the 2001 Winter Simulation Conference (Cat. No. 01CH37304), 2001, pp. 147–157, vol. 141.
[29] J. Banks, Handbook of Simulation, John Wiley & Sons, Inc., 2007.
[30] B.P. Zeigler, T.G. Kim, H. Praehofer, Theory of Modeling and Simulation, Academic Press, Inc., 2000.
[31] J. Nutaro, Parallel Discrete Event Simulation with Application to Continuous System, Department of Electrical and Computer Engineering, University of Arizona, 2003.
[32] J. Nutaro, B.P. Zeigler, R. Jammalamadaka, S. Akerkar, Discrete Event Solution of Gas Dynamics Within the EVS Framework, Springer, Berlin, Heidelberg, 2003, pp. 319–328.
[33] H. Karimabadi, J. Driscoll, Y.A. Omelchenko, N. Omidi, A new asynchronous methodology for modeling of physical systems: breaking the curse of Courant condition, J. Comput. Phys. 205 (2005) 755–775.
[34] D. Stone, S. Geiger, G.J. Lord, Asynchronous discrete event schemes for PDEs, J. Comput. Phys. 342 (2017) 161–176.
[35] Y.A. Omelchenko, H. Karimabadi, A time-accurate explicit multi-scale technique for gas dynamics, J. Comput. Phys. 226 (2007) 282–300.
[36] S.K. Matthäi, S. Geiger, S.G. Roberts, The Complex Systems Platform CSP5.0: User's Guide, ETH, Zurich, Switzerland, 2004.
[37] S.K. Matthäi, S. Geiger, S.G. Roberts, A. Paluszny, M. Belayneh, A. Burri, A. Mezentsev, H. Lu, D. Coumou, T. Driesner, C.A. Heinrich, Numerical simulation of multi-phase fluid flow in structurally complex reservoirs, Geol. Soc. (Lond.) Spec. Publ. 292 (2007) 405–429.

[38] S.K. Matthäi, A.A. Mezentsev, M. Belayneh, Finite element – node-centered finite-volume two-phase-flow experiments with fractured rock represented by unstructured hybrid-element meshes, SPE Reserv. Eval. Eng. 10 (2007) 740–756.
[39] M.L. Fredman, R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, J. Assoc. Comput. Mach. 34 (1987) 596–615.
[40] OpenMP Architecture Review Board, OpenMP Application Programming Interface: Version 4.5, November 2015.
[41] J.E. Nash, J.V. Sutcliffe, River flow forecasting through conceptual models, part I — a discussion of principles, J. Hydrol. 10 (1970) 282–290.
[42] C. Zoppou, J.H. Knight, Analytical solutions for advection and advection–diffusion equations with spatially variable coefficients, J. Hydraul. Eng. 123 (1997) 144–148.
[43] S.K. Matthäi, A.A. Mezentsev, M. Belayneh, Finite element – node-centered finite-volume two-phase-flow experiments with fractured rock represented by unstructured hybrid-element meshes, in: SPE Reservoir Simulation Symposium, Society of Petroleum Engineers, Houston, Texas U.S.A., 2005.
[44] S. Geiger, Q. Huangfu, F. Reid, S.K. Matthäi, D. Coumou, M. Belayneh, C. Fricke, K.S. Schmid, Massively parallel sector scale discrete fracture and matrix simulations, in: SPE Reservoir Simulation Symposium, Society of Petroleum Engineers, The Woodlands, Texas, 2009.