Asynchronous Discrete Event Schemes for PDEs

D. Stone[a,*], S. Geiger[b], G. J. Lord[a]

[a]*Dept. Mathematics, Heriot-Watt University, Edinburgh*
[b]*Institute of Petroleum Engineering, Heriot-Watt University, Edinburgh*

Abstract

A new class of asynchronous discrete-event simulation schemes for advection-diffusion-reaction equations is introduced, based on the principle of allowing quanta of mass to pass through faces of a (regular, structured) Cartesian finite volume grid. The timescales of these events are linked to the flux on the face. The resulting schemes are self-adaptive, and local in both time and space. Experiments are performed on realistic physical systems related to porous media flow applications, including a large 3D advection diffusion equation and advection diffusion reaction systems. The results are compared to highly accurate reference solutions where the temporal evolution is computed with exponential integrator schemes using the same finite volume discretisation. This allows a reliable estimation of the solution error. Our results indicate a first order convergence of the error as a control parameter is decreased, and we outline a framework for analysis.

Keywords: Asynchronous, Adaptive, Discrete-Event-Simulation, PDE, Conservation Laws

1. Introduction

We develop new schemes for the simulation of porous media flow based on an asynchronous simulation methodology. By asynchronous it is meant different parts of the spatial domain are allowed to exist at different times simultaneously during the course of the simulation. Numerous different categories of numerical schemes fall under this broad description; here we are interested in schemes based on the Discrete Event Simulation (DES) methodology. This methodology is essentially the idea of evolving a system forward in time by discrete events, which are local in space, with each event having its own local timestep determined by the physical activity in that region, see [1, 2, 3, 4]. In this way more active regions of the spatial domain receive more events, in principle leading to more efficient distribution of computational effort. A full description and algorithm is presented in Section 2.

[*]Corresponding Author

Traditionally DES schemes were developed for naturally discrete systems, such as war games and telecommunications (see e.g. [5]), not continuous physical systems such as models describing fluid flow or solute transport in porous media. The use of DES concepts applied to continuous physical systems was introduced by [1] for plasma simulation where an event is the motion of an ion particle between two cells. The same authors then presented in [2] an asynchronous method for conservation law PDES with sources, in one dimension, based on evolving the PDE model at different rates in different cells. See also [4] and the references therein. Our methods are all, by contrast, face based, in that an event is always the transfer of mass between cells, not evolution within a cell.

These schemes are self-adaptive in the sense that during each event, the local state of the system is evolved forward in time by an appropriately sized timestep, which is chosen automatically. The size of a timestep can be limited by accuracy requirements, for example, or Courant Friedrichs Lewy (CFL) conditions (see for example Section 4.2 of [6]). In the simulation of the evolution of physical systems, and especially porous media flow applications, the appropriate size for a timestep will often vary significantly in both space and time, see for example Figure 1. Classical explicit and implicit timestepping methods have the disadvantage of using a global timestep size, which is limited to the smallest appropriate timestep anywhere in domain. We now briefly discuss existing non-global timestepping methods for comparison with our schemes.

There is some similarity with the modelling philosophy of the reaction diffusion master equation (RDME) [8] and the Gillespie method [9] and its derivatives, see [10, 11] for example. However, the methods here are deterministic and approximate the bulk behaviour modelled by a PDE instead of operating on the scale of molecules. The goal of asynchronous schemes may be compared with, for example, adaptive time stepping [12] schemes, Space-Time Discontinuous Galerkin methods (see e.g. [13]) or local timestepping [14, 15, 16] schemes (LTS), where the spatial grid is refined in space in order to better capture more active regions, and a corresponding local timestep is used to ensure a local CFL condition. Local timestepping schemes also exist where the grid is not refined spatially and the local timesteps are varied to better capture activity according to local rates. See for example [17], where a binary tree is used to schedule the order in which cells will update, but full asynchronicity is avoided (unlike here) by implementing a standard LTS interpolation procedure between adjacent cells at different times, when approximating spatial derivatives. Another method with conceptual overlap is wavefront tracking, see [18].

While traditional PDE solvers rely on efficient linear algebra solvers, and exponential integrators (eg [19, 20, 21, 22, 23, 24]) rely on efficient approximation of the matrix exponential, asynchronous schemes rely on an efficient way of ordering the pending events. The list of pending events is typically stored in a binary tree or custom priority queue, adding some additional complexity to the implementation. A custom type of priority queue is described in [2] and we use our own implementation of this description. More details on the implementation and comparison to other schemes are discussed in [25]. Initially we focus on the simulation of linear conservation laws in the absence of reaction

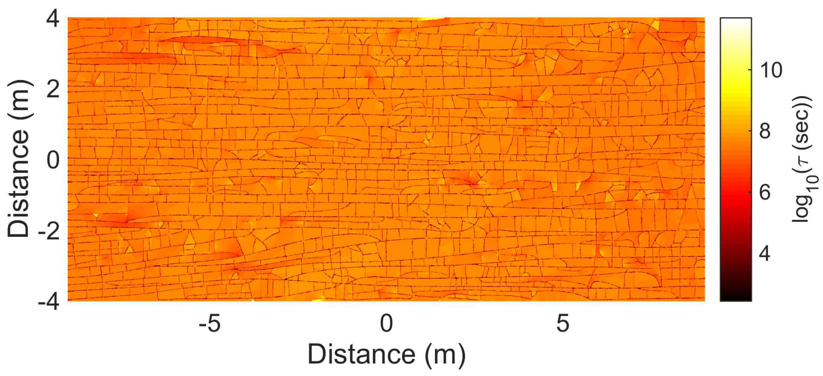


Figure 1: Simulated distribution of the residence time τ (the average time which a particle of fluid is expected to remain in a certain control volume) in a fractured porous media. Note that the fracture permeabilities are significantly higher than the matrix permeability. This permeability contrast together with the connectivity of the fractures cause the extreme variation of residence times. Areas with low residence times indicate very fast flow, which will require a significantly different time-step Δt to resolve the physical processes in a numerical simulation compared to regions where the residence time is large. See [7] for further details.

terms, describing for example the transport of a non-reactive tracer in porous media,

$$\frac{\partial c(\mathbf{x},t)}{\partial t} = \nabla f(c(\mathbf{x},t)), \qquad t \in \mathbb{R}, \qquad \mathbf{x} \in \Omega \subset \mathbb{R}^d, \tag{1}$$

$d = 1, 2, 3$, where $c(\mathbf{x}, t)$ is a concentration and f is a given flux function. An initial condition $c(\mathbf{x}, 0) = c_0(\mathbf{x})$ is provided. For simplicity, we consider 'no flow' boundary conditions, that is, Neumann type boundary conditions with zero flux on external faces. Other types of boundary conditions could easily be added in this framework. We are primarily interested in advection diffusion systems, where the flux is of the form

$$f(c(\mathbf{x},t) = D(\mathbf{x})\nabla c(\mathbf{x},t) - \mathbf{v}(\mathbf{x})c(\mathbf{x},t), \tag{2}$$

where D is the diffusivity and $\mathbf{v}$ is a given velocity. The combination of (1) and (2) is the PDE

$$\frac{\partial c(\mathbf{x},t)}{\partial t} = \nabla(D(\mathbf{x})\nabla c(\mathbf{x},t)) - \nabla(\mathbf{v}(\mathbf{x})c(\mathbf{x},t)). \tag{3}$$

In Section 3.3 we describe the incorporation of a reaction term in to (3) for our schemes. The spatial domain Ω is discretised into cells, as in a standard finite volume approach (see for example [26, 27, 28, 29] and references therein), and equation (1) is discretised in space over the grid. To describe our new schemes we start by focusing on a simple system; a conservation law (1) without sources with flux given by, for example, (2).

2. The Basic Face Based Asynchronous Scheme (BAS)

Our schemes make use of the flux on each face of a computational grid arising, from a traditional finite volume (FV) discretisation. The spatial domain Ω is divided to a grid of cells each with a unique index $j \in \{1, 2, \ldots, J\}$. Similarly every face also has a unique index $k \in \{1, 2, \ldots, K\} = \mathcal{F}$. For a cell with index $j \in \mathcal{C}$, define the set $\mathcal{F}_j$ of faces belonging to the cell, where $\mathcal{F}_j \subset \mathcal{F}$. Also, For a given face k, we define the set of associated faces of k $\tilde{\mathcal{F}}_k$ of a face k as follows. If face k is adjacent to cells $j_1, j_2 \in \mathcal{C}$, then

$$\tilde{\mathcal{F}}_k = \mathcal{F}_{j_1} \cup \mathcal{F}_{j_2},$$

i.e., the set of all the faces of the two cells which face k is adjacent to. That is, if k is the face in common to two cells j_1 and j_2, then the set of associated faces is the set of all the faces belonging to either cell j_1 or j_2. See also Figure 2 for a depiction of some of the notation with respect to the grid. The finite volume discretisation of (1) is based on the approximation of the flux across faces in the grid.

Let f_k be the approximation of the flux on a face k, which depends upon the concentration values c_{j_1}, c_{j_2} in the two cells with indexes j_1, j_2 adjacent to face k. The concentration c_j of a cell j is assumed constant throughout the cell, and is derived from the mass in the cell m_j and its volume V_j as $c_j = \frac{m_j}{V_j}$. The flux f_k on a face is assumed constant and defines the flow of mass across the face between its two adjacent cells, i.e., the flow of mass from cell j_1 due to face k will be $\pm f_k A_k$; and into cell j_2 will be be $\mp f_k A_k$ (with the signs depending on the direction of net flow), where A_k is area of the face k. The direction of mass flow depends on the sign on f_k. To be explicit, the equations for mass flow *across a single face* k, are

$$\frac{dm_{j_1}}{dt} = \sum_{k=\text{ faces of cell } j_1} \pm f_k A_k, \qquad \frac{dm_{j_2}}{dt} = \sum_{k=\text{ faces of cell } j_2} \mp f_k A_k.$$

Later we will be interested in the contribution to the mass accumulation in a cell *due to a single face*. Let $\left(\frac{dm_j}{dt}\right)_k$ be the contribution to the mass accumulation in cell j due to face k, i.e.,

$$\left(\frac{dm_{j_1}}{dt}\right)_k = \pm f_k A_k, \qquad \left(\frac{dm_{j_2}}{dt}\right)_k = \mp f_k A_k. \tag{4}$$

For an advection-diffusion system, one of the two cells will be the upwind cell; without loss of generality let this be cell j_1. Then the flux across a face k may be approximated by finite differences such as,

$$f_k = \frac{\bar{D}_k \left(\frac{m_{j_2}}{V_{j_2}} - \frac{m_{j_1}}{V_{j_1}} \right)}{\Delta x_k} - \frac{m_{j_1}}{V_{j_1}} v, \tag{5}$$

where $\bar{D}_k$ is an approximation of the diffusivity at the face based on the diffusivity in the two cells, typically the harmonic mean of D_{j_1} and D_{j_2}, Δx_k is the distance between the two cell centroids; and v is the scalar product of the velocity at the centre of face k with the unit vector in the direction of the line from the centre of cell j_1 to cell j_2.

The total rate of change of mass, and thus concentration in a cell j is found from (4) for each $k \in \mathcal{F}_j$. This can be expressed as a matrix, L which gives the finite volume semidiscretisation of (1) as a system of ODEs,

$$\frac{d\mathbf{c}}{dt} = L\mathbf{c}, \qquad L \in \mathbb{R}^{J \times J} \tag{6}$$

where $\mathbf{c} = (c_1, c_2, \ldots, c_J)^T$ is the vector of concentrations in cells. In a standard finite volume based implementation (6) is then discretised in time, resulting in the fully discrete approximation.

Face based asynchronous schemes are based on events involving the transfer of mass across a single face but do not form the global system (6). Instead they can be defined in terms of much smaller local matrices which we call "connection matrices" and introduce in Section 5.1. They proceed in discrete events approximating the effect of (4). The outline of the algorithm is as follows.

- Every face has an individual time t_k and a projected update time $\hat{t}_k$.
- The face with the lowest update time $\hat{t}_k$ is chosen for an event.
- During an event, the two cells adjacent to face k are updated by having a fixed amount of mass, ΔM, passed between them.
- A timestep Δt_k is associated with this event, and after the event the time on face k is updated to $t_k + \hat{t}_k$.
- After the event the update time $\hat{t}$ is recalculated for every face of the two cells involved in the event.
- This repeats until all faces are synchronised at a final time T.

The full algorithm is given in Algorithm 1. We now consider the details missing from the above outline, specifically how $\hat{t}_k$ is calculated and its relation to ΔM and the local flux across a face. Note that choosing an appropriate value of the global mass unit ΔM to balance accuracy and efficiency is of great importance in using this method.

For a face k, the projected update time $\hat{t}_k$ is calculated so that in the interval $\Delta t_k \equiv \hat{t}_k - t_k$, at most ΔM units of mass pass through the face. The Basic Asynchronous Scheme (BAS) calculates the update time $\hat{t}_k$ as,

$$\hat{t}_k = \begin{cases} t_k + \frac{\Delta M}{|f_k|A_k} \text{ if this} \leq T \\ T \text{ otherwise.} \end{cases} \tag{7}$$

This is derived from a standard Euler-type approximation of the flow of flux through the face, ignoring the effect of the other faces in the cell. That is, we want a face to have passed an amount of mass ΔM in the time interval $\hat{t}_k - t_k$, and this leads to an approximation of the derivative in (4),

$$\text{Mass flow through single face } k \approx \frac{\Delta M}{\hat{t}_k - t_k} = |f_k|A_k, \tag{8}$$

from which the first case in (7) follows. The absolute value of the flux is used to ensure that the calculated time values are positive. The direction is irrelevant when calculating the always positive $\hat{t}_k$, thus the only magnitude of the flux is important.
When (7) calculates $\hat{t} > T$, the value of T is used instead. In this way the simulation finishes with every face at the desired final time T; it is an Euler-type approximation using the imposed timestep $T - t_k$. The mass transferred during this final synchronisation step is not ΔM. Let δm be the mass transferred in an event for face k. Then, again following from a simple Euler-type approximation,

$$\delta m = \begin{cases} \Delta M \text{ if } t^k + \frac{\Delta M}{|f_k|A_k} \leq T \\ |f_k|(T - t_k)A_k \text{ otherwise.} \end{cases} \tag{9}$$

```
   Data: Grid structure, Initial concentration values, $\Delta M$, $T$
1  Initialise: $t = 0$ ; Calculate $f_l$ from (5) and $\hat{t}_l$ from (7) $\forall$ faces $k$ ;
2  while $t \leq T$ do
3      Find face $k$ s.t. $\hat{t}_k = \min_{l \in \mathcal{F}} \hat{t}_l$ ;
4      Get cells $j_1$ and $j_2$ adjacent to $k$;
5      Calculate $\delta m$ from (9) ;
6      $m_{j_1} \leftarrow m_{j_1} - \text{sign}(f_k)\delta m$ ;
7      $m_{j_2} \leftarrow m_{j_2} + \text{sign}(f_k)\delta m$ ;
8      $t = t_k \leftarrow \hat{t}_k$ ;
9      for $l \in \tilde{\mathcal{F}}_k$ do
10         Recalculate $f_l$ from (5) ;
11         Recalculate $\hat{t}_l$ from (7) ;
12     end
13 end
```
Algorithm 1: Pseudo code for the basic asynchronous scheme (BAS).

Algorithm 1 describes the BAS method. After initialising the required values on all faces, the update loop is run until every face is synchronised to the desired final time of T. Each iteration of the loop is a single event and proceeds as follows. First the face with the lowest projected update time $\hat{t}$ is found (line 3) Then the two cells adjacent to this face are located from the grid structure (line 4). The amount of mass to transfer between these cells is calculated (line 5). This equation simply returns the global mass unit ΔM in most cases, except when the face is being forced to use an update time T; see equations (7) and (9). Mass is transferred between the cells in the correct direction (lines 6-7). A loop (lines 8-12) updates set of associated faces of k (the faces of cells j_1 and j_2); recalculating their fluxes and update times based on the new mass values. The loop then continues by finding the next face with the lowest update time (back to line 3). In Figure 2 we show a schematic of two cells undergoing the mass transfer and time update parts of a single event, corresponding to lines 6 - 8 in Algorithm 1.
This is the simplest face based asynchronous scheme we can conceive. In all our experiments we have observed, that as the mass unit ΔM decreases to zero, the approximation produced by this scheme converges to the exact solution of the linear ODE system produced by applying the corresponding finite volume discretisation to the corresponding PDE (1).

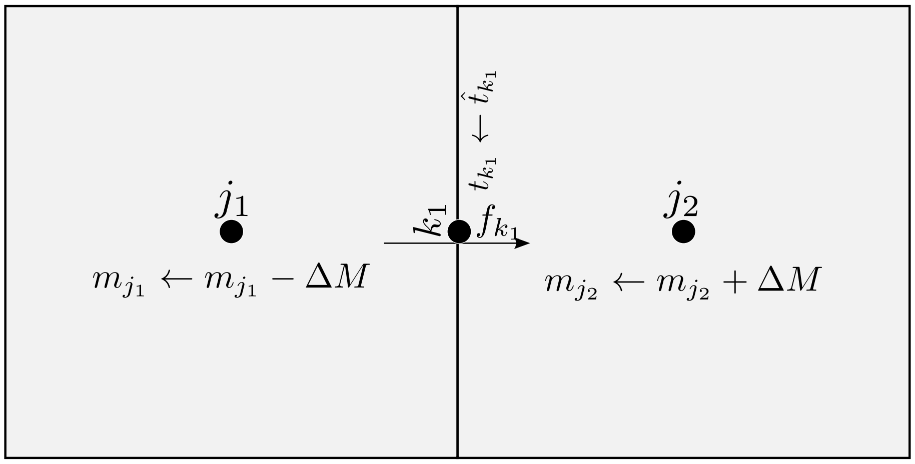

$t_{k_1} \leftarrow \hat{t}_{k_1}$
j_1
k_1
f_{k_1}
j_2
$m_{j_1} \leftarrow m_{j_1} - \Delta M$
$m_{j_2} \leftarrow m_{j_2} + \Delta M$

Figure 2: A schematic of two cells undergoing an event as described in lines 6 - 8 in Algorithm 1. The two cells are labeled j_1 and j_2 and their common face is k_1. The flux across face k_1 is labeled as f_{k_1}. The schematic shows the mass of ΔM being deducted from the mass in cell j_1 (i.e. $m_{j_1} \leftarrow m_{j_1} - \Delta M$) and being added to the mass in cell j_2 ($m_{j_1} \leftarrow m_{j_1} - \Delta M$); this is lines 6 and 7 in Algorithm 1. The time on the face k_1 is then updated to its update time ($t_{k_1} \leftarrow \tilde{t}_{k_1}$; corresponding to line 8 in Algorithm 1.) Note that the set of associated faces of k_1, defined as all the faces of the two cells belonging to the two cells adjacent to k_1, is the set of all the faces shown in this figure.

We now briefly discuss the cost per event of the algorithm. Line 5 costs 3 flops. Lines 6 and 7 are another 3 flops (two additions, and a single multiplication by -1). Lines 10 and 11 cost 12 flops in total, per face. Over all 11 faces updated (lines 9 to 12), this is a cost of 132 flops. The total for direct calculation in a single event is then 138 flops. The updating of the priority queue for each of the associated faces is also necessary (consider this abstracted into line 3). The priority queue we used was implemented as a a binary heap in the classical way, modified to allow recomputing the priority (i.e., resorting) of any element. The classical cost of an insert operation for a binary heap is $O(\log(N))$, where N is the length of the queue (i.e., the same as the number of faces in the grid for us). The bound $O(\log(N))$ comes from the depth of the binary tree represented by the binary heap being $\log(N)$ deep, so that the loop for a resort operation will contain at most $\log(N)$ operations. Looking at our implementation of the priority queue, each iteration costs 2 flops and 9 variable overwrites (there is some unwieldy bookkeeping involved in allowing every entry to be indexed by its associated face index). We can then assign an estimated flop cost to line 3 of the algorithm of $22\log(N)$ ($22 = 11$ faces times 2 flops).

The priority queue is clearly a significant contributor to the overall flop cost per event, but it is closely matched by the cost of the flux and time updates of lines 9 to 12 in the algorithm. Consider a grid with 10^6 faces. Then the flop cost of updating the priority queue is at most $22\log_2(10^6) \approx 439$, while the cost from lines 9 to 12 is still 138 flops, which is not trivial in comparison. We note that the cost of line 10 can be reduced by storing some values for each face, reducing the number of flops required for equation (5).

We see that the cost of bookkeeping, whether it is flux updates or priority queue resorts, have the potential overwhelm the cost for system updates in the for this scheme, and in principle, asynchronous schemes in general.

3. Modifications to BAS

Here we describe some modifications that can be made to the basic asynchronous scheme. First we describe the concept and implementation of a mass-tracking parameter, which increases the accuracy of the scheme by reducing the asynchronicity error. Second we describe a method for 'cascading' events which bypass the bookkeeping and priority queue update parts of an event, first introduced in [2] and applied to our new class of schemes. Finally we describe a modification to allow reaction terms to be simulated.

3.1. Using A Mass-Passed Tracking Value - the BAST Scheme

When an event occurs in the basic scheme, only one face is updated, while the associated faces are not, which naturally introduces inaccuracy from the decoupling of activity on one face from the activity of nearby faces. Modifications can be sought to reduce this asynchronicty error while still retaining the essential asynchronous characteristics of the scheme. We describe one such successful modification here.

Consider adding an extra parameter to each face k, which is intended to track the mass that the face should have passed during an event of an associated face

(earlier defined at the start of Section 2). Let this parameter be called the mass passed value, $\Delta M_{p,k}$. The larger the mass passed value the more likely the face is to be updated.
We describe the implementation of $\Delta M_{p,k}$ to illustrate its intended function. First, for every face it is initialised to zero, and reset to zero when the face has an event. In Algorithm 1, during the loop of (lines 9-12) over each face l in the set of associated faces $\tilde{\mathcal{F}}_k$ of the active face k, *except k itself*, the mass passed value is updated as

$$\Delta M_{p,l} = \Delta M_{p,l} + (\hat{t}_k - t_l) A_l f_l. \tag{10}$$

Compare this to the second equation in (9). In (10), the mass-passed tracking value $\Delta M_{p,l}$ is incremented by the amount of mass that would have passed through face l during a timestep of length $\hat{t}_k - t_l$. Also, in the modified scheme every face of the cells j_1, j_2 has its time updated at this point,

$$t_l = \hat{t}_k,$$

as though these faces have also had events, although no transfer has occurred for these faces. The mass passed value effectively tracks the mass the faces would have passed in the time $[t_l, \hat{t}_k]$.
The mass passed value then adjusts the next event for a face k, depending on the size of $\Delta M_{p,k}$, as follows. The timestep approximation (8) is replaced with

$$\frac{\Delta M - \Delta M_{p,k}}{\hat{t}_k - t_k} \approx \left(\frac{dm}{dt}\right)_k = |f_k| A_k,$$

leading to the modified version of (7), the equation for $\hat{t}_k$,

$$\hat{t}_k = \begin{cases} t_k + \frac{\Delta M - \Delta M_{p,k}}{|f_k| A_k} & \text{if this} \leq T \\ T & \text{otherwise.} \end{cases} \tag{11}$$

Thus, faces will have increased priority for events if they have greater mass passed values.

3.2. The Cascading or 'Flux Capacitor' Concept of [2]

A crucial innovation in [2] is allowing cells to trigger their own events if they have been subject to too much activity without an event - each cell has a 'flux capacitor' value assigned, which is incremented each time a neighbouring cell has an event, and reset to zero when the cell itself has an event. Instead of affecting the update time of faces (or cells), the job of the flux capacitor value is, if and when it exceeds a certain threshold, to trigger a new event, *independent of its update time and the priority queue.*
In a situation such as an advancing front, or simply a region of high activity, this can lead to cells (or faces) constantly triggering their neighbours, following the path of high activity and ignoring the costly update time and priority queue calculations temporarily. This further emphasises the objective of DES methods

to focus attention on the most active parts of the domain.
We have also implemented this concept in our face-based Asynchronous schemes. Consider the Mass-Passed Tracking scheme described in Section 3.1 with the following modifications. First, the dependence of update time on $\Delta M_{p,k}$ is removed (i.e., instead of (11), we use the basic (7)). Second, when some face j has its $\Delta M_{p,j}$ incremented as part of an event on an associated face k, then an event is automatically triggered on j if $\Delta M_{p,j} > \Delta M$. We note there is potential to use other threshold values than the mass unit ΔM, but it seems to work well, see Section 4.1.
Because these schemes cascade 'cheap' events along fronts or regions of very high activity, we refer to them as cascading schemes, however the underlying concept is of course nothing but the 'flux capacitor' of [2]. We have the basic scheme, BAS, augmented with the concept, which we will abbreviate as BAS-casc.

3.3. Adding a Reaction Term

It is possible to include a reaction term, so that we may simulate conservation equations of the form

$$\frac{dc(\mathbf{x},t)}{dt} = \nabla f(c(\mathbf{x},t)) + r(c(\mathbf{x})), \qquad t \in \mathbb{R}, \qquad \mathbf{x} \in \Omega, \tag{12}$$

i.e., (1) with a reaction term r. It is in principle possible to also apply our method to non-autonomous reaction terms $r = r(c\mathbf{x}, t)$. We have found that a standard Strang operator splitting implementation of the reaction during events is effective.
This method has been applied to BAS and BAST. First we assign to each cell its own independent time t_j. The resulting algorithm is just Algorithm 1 added to and modified. Algorithm Algorithm 2 shows this modified algorithm. We describe the key lines of this algorithm point by point as follows:

1. A new step, line 6: Calculate timestep values for each of the two cells, as $\Delta t_{j_1} = \hat{t}_k - t_{j_1}$ and $\Delta t_{j_2} = \hat{t}_k - t_{j_2}$.
2. A new step, lines 7 and 8: Update the mass in both the cells according to the reaction term, using an Euler type step. For each cell use half the timestep for the cell. That is, perform the update,

$$m_{j_1} \leftarrow m_{j_1} + V_{j_1}\frac{\Delta t_{j_1}}{2} r\left(\frac{m_{j_1}}{V_{j_1}}\right),$$

$$m_{j_2} \leftarrow m_{j_2} + V_{j_2}\frac{\Delta t_{j_2}}{2} r\left(\frac{m_{j_2}}{V_{j_2}}\right).$$

3. The mass transfer across the face proceeds exactly as in the original scheme; lines 5-7 of Algorithm 1. Now lines 8 - 10 in Algorithm 2.
4. Repeat step 2 (lines 11 and 12).

```
   Data: Grid structure, Initial concentration values, $\Delta M$, $T$
1  Initialise: $t = 0$ ; Calculate $f_l$ from (5) and $\hat{t}_l$ from (7) $\forall$ faces $k$ ;
2  while $t \leq T$ do
3      Find face $k$ s.t. $\hat{t}_k = \min_{l \in \mathcal{F}} \hat{t}_l$ ;
4      Get cells $j_1$ and $j_2$ adjacent to $k$;
5      $\Delta t_{j_1} = \hat{t}_k - t_{j_1}$, $\Delta t_{j_2} = \hat{t}_k - t_{j_2}$ ;
6      $m_{j_1} \leftarrow m_{j_1} + V_{j_1} \frac{\Delta t_{j_1}}{2} r\left(\frac{m_{j_1}}{V_{j_1}}\right)$ ;
7      $m_{j_2} \leftarrow m_{j_2} + V_{j_2} \frac{\Delta t_{j_2}}{2} r\left(\frac{m_{j_2}}{V_{j_2}}\right)$ ;
8      Calculate $\delta m$ from (9) ;
9      $m_{j_1} \leftarrow m_{j_1} - \text{sign}(f_k)\delta m$ ;
10     $m_{j_2} \leftarrow m_{j_2} + \text{sign}(f_k)\delta m$ ;
11     $m_{j_1} \leftarrow m_{j_1} + V_{j_1} \frac{\Delta t_{j_1}}{2} r\left(\frac{m_{j_1}}{V_{j_1}}\right)$ ;
12     $m_{j_2} \leftarrow m_{j_2} + V_{j_2} \frac{\Delta t_{j_2}}{2} r\left(\frac{m_{j_2}}{V_{j_2}}\right)$ ;
13     $t = t_k \leftarrow \hat{t}_k$ ;
14     for $l \in \tilde{\mathcal{F}}_k$ do
15         Recalculate $f_l$ from (5) ;
16         Recalculate $\hat{t}_l$ from (7) ;
17     end
18 end
```

Algorithm 2: Pseudo code for the basic asynchronous scheme (BAS) with reaction term modification.

Also, at line 8 of Algorithm 1, we set the cell times to be $\hat{t}_k$ alongside the face time. That is, $t_{j_1} \leftarrow \hat{t}_k$, $t_{j_2} \leftarrow \hat{t}_k$.
We note again that this can and has been applied to both BAS and BAST.
We now discuss this process. First consider step 2. The first point to note is that it can be expressed in terms of concentration instead of mass simply as,

$$c_{j_1} \leftarrow c_{j_1} + \frac{\Delta t_{j_1}}{2} r\left(c_{j_1}\right),$$

$$c_{j_2} \leftarrow c_{j_2} + \frac{\Delta t_{j_2}}{2} r\left(c_{j_2}\right).$$

Ignoring for the moment the half timesteps, this is a single step of the Euler method (though with different timesteps for each cell) for the system consisting only of cells j_1 and j_2, and governed by the reaction-term-only PDE,

$$\frac{dc(\mathbf{x},t)}{dt} = r(c(\mathbf{x})), \qquad t \in \mathbb{R}, \qquad \mathbf{x} \in \Omega. \tag{13}$$

This is analogous to how in BAS we consider the system consisting only of the two cells j_1 and j_2 (and the internal face k), governed by the flux PDE (4). Step 4 corresponds to step 2 by 'completing' the halved Euler step. Steps 2 through 4 are thus simply an operator splitting method, applied to the tiny two cell subsystem considered by each event. Specifically it a Strang method, the simplest form of operator splitting.

This is an extension of the concept of our face based schemes to systems with reaction or source terms; the only technicality is the introduction of time values assigned to cells as well as faces, which is required to define timesteps for the reaction steps in a sensible way. This method retains the interesting property of not needing to be explicitly based on a PDE. Indeed, the scheme can be implemented based on (4) for the faces and (13) for the cells, without any use or reference to (12). The modifications described here can also be applied to BAST.
We can imagine this method having difficulty with situations or parts of a domain where there is no flux between cells but still reactions within the cells changing the concentration values there - in this case the reaction activity could be 'missed' by the lack of events. It is likely possible to find other modifications which allow the schemes to handle reaction terms.
We note that this is of course not the only possible modification for nonlinear reaction terms that could be implemented; simply one which seems to work. Another natural approach would be to consider separate events for the reaction term inside cells.

4. Numerical Results

Before we outline some some steps towards convergence analysis in Section 5 we present some numerical experiments with the new schemes which demonstrate convergence and the relationship between scheme parameters such as N

the total number of events, ΔM the mass unit and the average time step $\bar{\Delta} t$. Our first test systems are linear PDEs, which produce ODE systems of the form $\frac{dc}{dt} = Lc$ after a finite volume discretisation. In Section 3.3 we add in a reaction term. Reference solutions for comparison are computed below using a first order exponential integrator, see [19] and the references therein. The error is measured in the discrete $L^2(\Omega)$ norm. The Matlab Reservoir Simulation Toolkit (MRST [30]) was used to generate the (regular, structured, Cartesian) grids for the experiments but the discretisation and solver routines were implemented by us.

4.1. Fracture System with Varying Diffusivity

In this example a single layer of cells is used, making the problem effectively two dimensional. The domain is $10 \times 10 \times 10$ metres, divided into 100×100 cells of equal size. We specify the velocity field to be $\mathbf{v}(\mathbf{x}) = (1, 0)^T$ (a constant velocity field may not be realistic however this example still provides an interesting test case). The initial condition was $c(\mathbf{x}) = 0$ everywhere except at $\mathbf{x_0} = (4.95, 9.95)^T$ where $c(\mathbf{x_0}) = 1$. A fracture in the domain is represented by having a line of cells which we give certain properties. These cells were chosen by a weighted random walk through the grid (weighted to favour moving in the positive y-direction so that the fracture would bisect the domain). This process started on an initial cell which was marked as being in the fracture, then randomly chose a neighbour of the cell and repeated the process. This was done once to prepare the grid before the main tests. We set the diffusivity to be $D = 100$ on the fracture and $D = 0.1$ elsewhere. Figure 3 a) shows the diffusivity of the system.

In Figure 3 we show in b) the reference solution of (3) at $T = 2.4$ which, since this is a linear system solved with an exponential integrator, is a very accurate approximation to the true solution. In Figure 4 a) we plot the solution at $T = 2.4$ using BAS with $\Delta M = 10^{-6}$ and in c) with $\Delta M = 10^{-9}$. Visually the $\Delta M = 10^{-9}$ solve agrees well with the comparison solve Figure 3 b). In Figure 4 b) and d) we have plotted maps of the number of events on each cell on a log scale for each of the respective solves. We observe that the updates and hence computational work, is concentrated in the regions of most physical activity - i.e. in the high diffusivity region and to right of it (due to the advection). It seems that with the smaller value of ΔM BAS is better able to concentrate computational activity where it is needed - note the greater spread in events over the system in Figure 4 b) compared to d).

In Figure 5 a) we show the convergence of the schemes with ΔM. The estimated error is plotted against the mass unit ΔM, and we clearly observe that the error for all our schemes is $O(\Delta M)$ for sufficiently small ΔM. In Figure 5 b) the estimated error is plotted against average timestep, Δt. Interestingly the error of the asynchronous schemes seems to be first order with respect to the average timestep. Plot c) shows the total number of events N against ΔM. For Figure 5 c), we observe that for both schemes the relationship between N and ΔM is the same for sufficiently small ΔM, as we see clearly $N = O(\Delta M^{-1})$. For larger mass unit values we observe that N is not changing with respect to

ΔM for BAS, although from plot Figure 5 a) we can see that the error is still decreasing for that range of mass unit values.

Note that a relation $\Delta t = O(\Delta M)$ logically follows from the relations implied in plots c) and d) (results for this can also be seen in [25]). This relation may be naively inferred from (7), from which it follows that $\Delta t_k = \frac{\Delta M}{|f_k|A_k}$. However, we cannot take this for granted since after any number of events the mass vector $\mathbf{m}$, and thus the flux across any given face f_k, can be expected to be different if a different value of ΔM is used for the solve. Thus we cannot rule out a priori that the denominator $|f_k A_k|$ in (7) has some dependence on ΔM.

Plot c) indicates that for sufficiently small ΔM, the total number of events over the solve, for a given ΔM, is the same or almost the same, for both BAST and BAS-Casc (although not for BAS). This could possibly indicate the existence of some 'preferred path' of events. That is, an ordering of faces on which events occur, which, in the limit $\Delta M \to 0$, all the discussed schemes tend toward following.

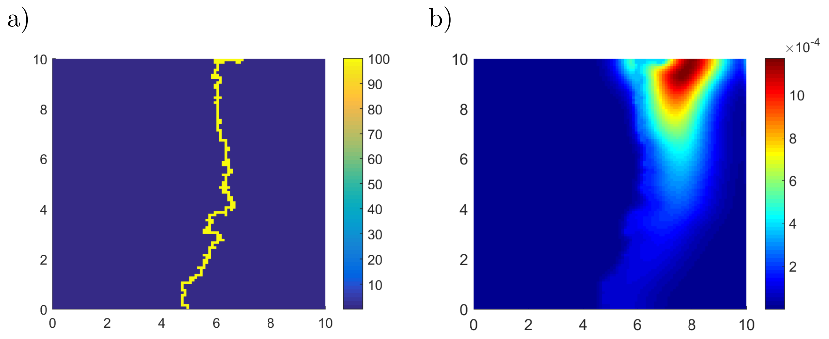


Figure 3: For the system described in Section 4.1. a) The diffusivity of the system, showing the fracture. b) The reference solution was computed with an exponential integrator.

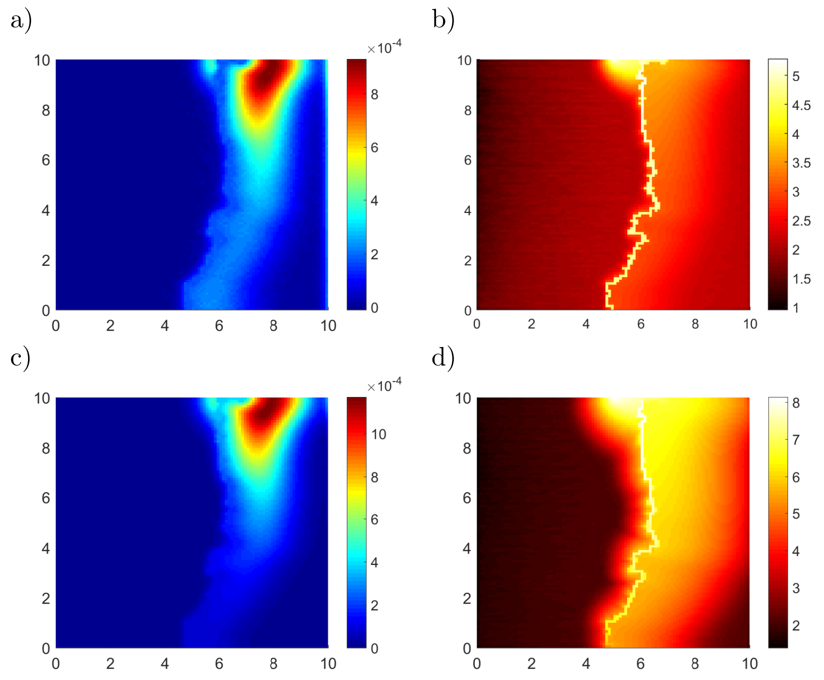


Figure 4: For the system described in Section 4.1. a) Solution produced by BAS with $\Delta M = 10^{-6}$; here ΔM is too great for excellent agreement with comparison solve, although the qualitative properties of the flow have clearly been captured well. b) Shows logarithm of number of events experienced by each cell for the same run as a). c) Solution produced by BAS with $\Delta M = 10^{-9}$; this solution is in close agreement with the comparison solve - compare to Figure 3 plot a). d) Shows logarithm of number of events experienced by each cell for the same run as d).

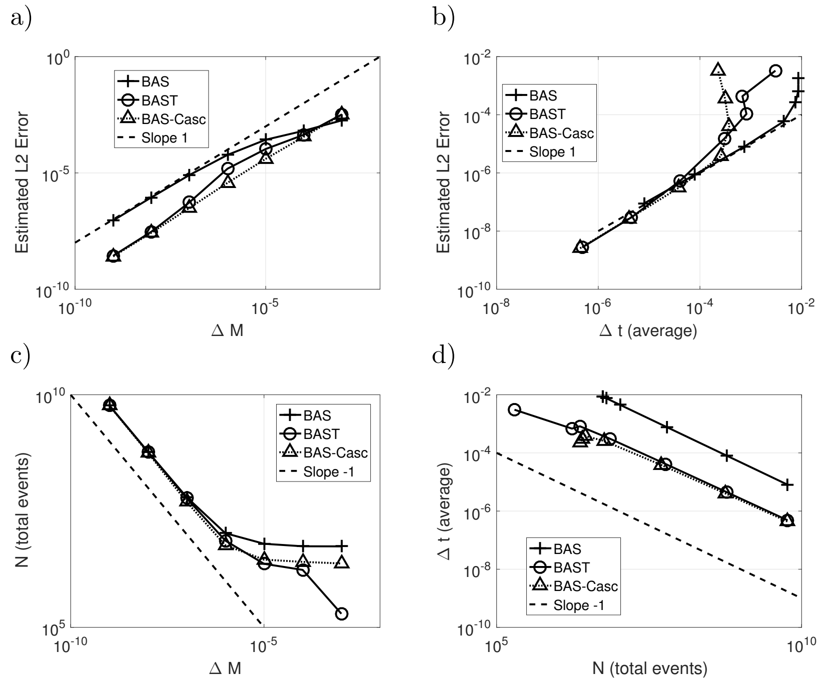


Figure 5: Results for the experiment described in Section 4.1. a) Estimated error against ΔM to indicate convergence. b) Estimated error against average event timestep. c) Total number of events N, against ΔM. d) Total number of events N against average event timestep.

4.2. Uniform constant diffusivity example

In this example the domain is $\Omega = 10 \times 10 \times 10$ metres again, discretised into $40 \times 40 \times 32$ cells, for a total of 51200 cells in the system. We solve (3) with a diffusivity field that is uniformly $D(\mathbf{x}) = 2$ and a constant velocity field $\mathbf{v}(\mathbf{x}) = (0.1, 1.1, 0)^T$. The initial condition is sinusoidal, varying between 0 and 1, on the line of cells where $y = z = 0$, and zero elsewhere. The final time was $T = 2.4$.

We show the state of the system at the final time T in Figure 6 c), as produced by BAST with $\Delta M = 1.9532 \times 10^{-10}$. This solution is seen to agree well with the reference solution in Figure 6 a). Plot d) in Figure 6 shows the logarithm of the number of events experienced by each cell during the same BAST solve. We see again how the scheme automatically focuses more computational effort, in the form of transfer events, at different areas of the domain according to local rate of activity. There is about a difference of five orders of magnitude in number of events between the least and most active cells in Figure 6 d).

In Figure 7 we present comparisons of various parameters for the schemes; the format is the same as Figure 5, and many of the conclusions are similar.

Plots b) through d) in Figure 7 indicate relationships between the parameters ΔM, N (total number of events), average Δt, and error, of the schemes.

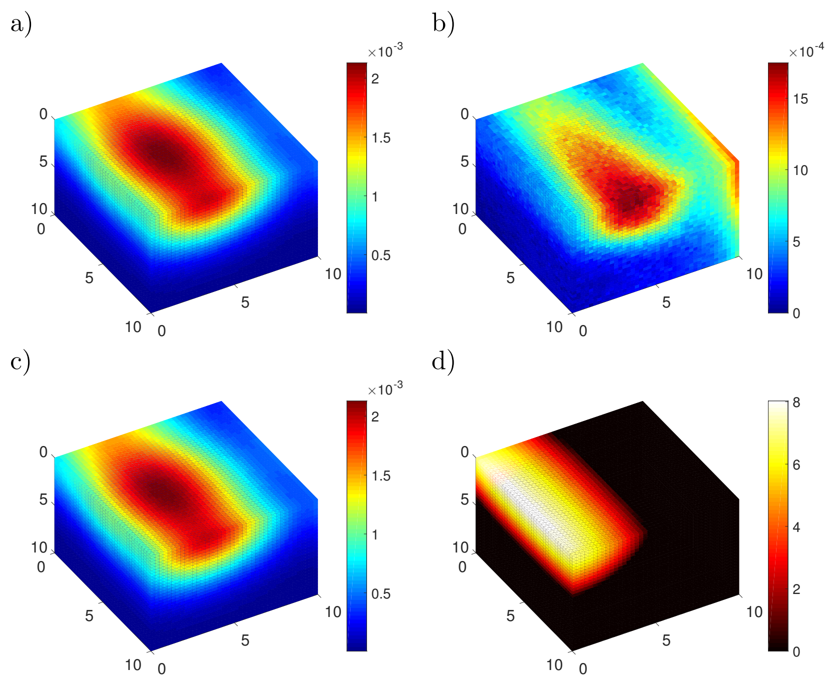


Figure 6: For the system described in Section 4.2. a) The reference solution was computed with an exponential integrator. b) Solution produced by BAS with $\Delta M = 1.953 \times 10^{-6}$; here ΔM is too great to allow close agreement with the comparison solve and we can observe 'chequerboard' effects. c) Solution produced by BAS with $\Delta M = 1.953 \times 10^{-9}$; this solution is in close agreement with the comparison solve. d) Shows logarithm of number of events experienced by each cell for the run with BAS and $\Delta M = 1.953 \times 10^{-9}$.

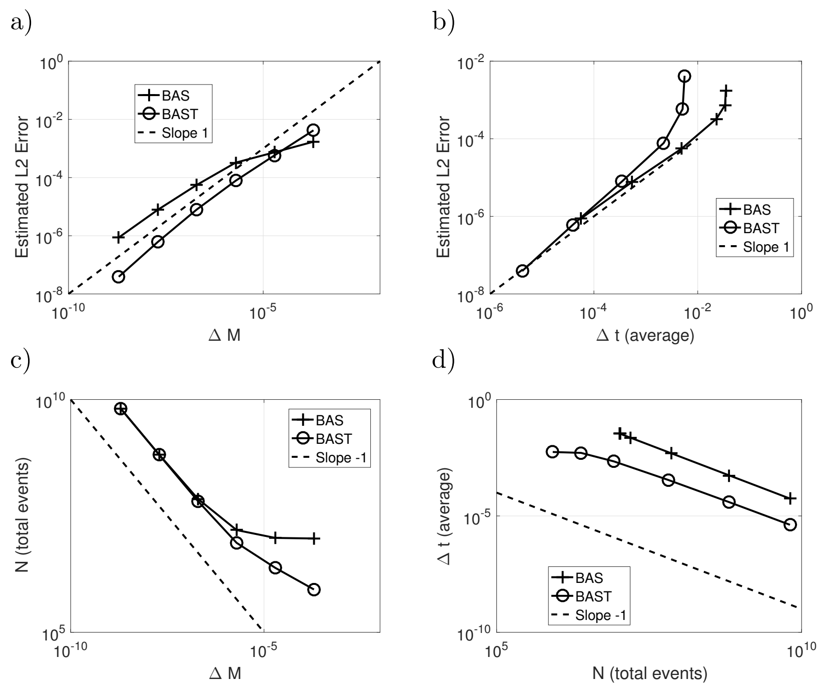


Figure 7: Results for the experiment described in Section 4.2. a) Estimated error against ΔM to indicate convergence. b) Estimated error against average event timestep. c) Total number of events N, against ΔM. d) Total number of events N against average event timestep.

4.3. Reaction-Diffusion Example

This experiment is a reaction-diffusion system in two dimensions with a (regular, structured) Cartesian grid, intended to test the leapfrog type reaction term implementation described in Section 3.3. The velocity field is set to be uniformly zero. The reaction term used is

$$r(c) = -\frac{c}{1+c},$$

which is a Langumiur-type reaction term, that can be used to model mass in the system adsorbing to the walls of the porous medium and thus being lost (see for example, [31]). In our example a region of high concentration in the centre of the domain diffuses outwards (the diffusivity field is uniform) while reacting according to the above Langmuir adsorption term. The final time is $T = 1$. The domain is again $\Omega = 10 \times 10 \times 10$ metres and discretised into $100 \times 100 \times 1$ cells. For this test the concentration was $c(\mathbf{x}) = 0$ for all $\mathbf{x}$ except at $\mathbf{x_0} = (4.95, 5.05)^T$ where $c(\mathbf{x_0}) = 1$. The boundary conditions were no-flow on all boundaries.

Figure 8 a) shows the reference solution, (b) is produced by BAS with $\Delta M = 10^{-6}$ and (c) with $\Delta M = 10^{-9}$. We see that with $\Delta M = 10^{-6}$ the accuracy is noticeably worse than for $\Delta M = 10^{-9}$. In (c) we plot the logarithm of number of events in each cell for $\Delta M = 10^{-9}$. We see that the computational effort largely follows the diffusion process of the solution.

In Figure 9 we show the convergence and parameter relations of the schemes. The parameter relations revealed in Figure 9 plots b) through d) are the same as those from the experiments in previous sections. Again, Figure 9 plot a) shows that error of the schemes converge to zero as ΔM decreases to zero. The schemes are still roughly first order with the addition of a reaction term.

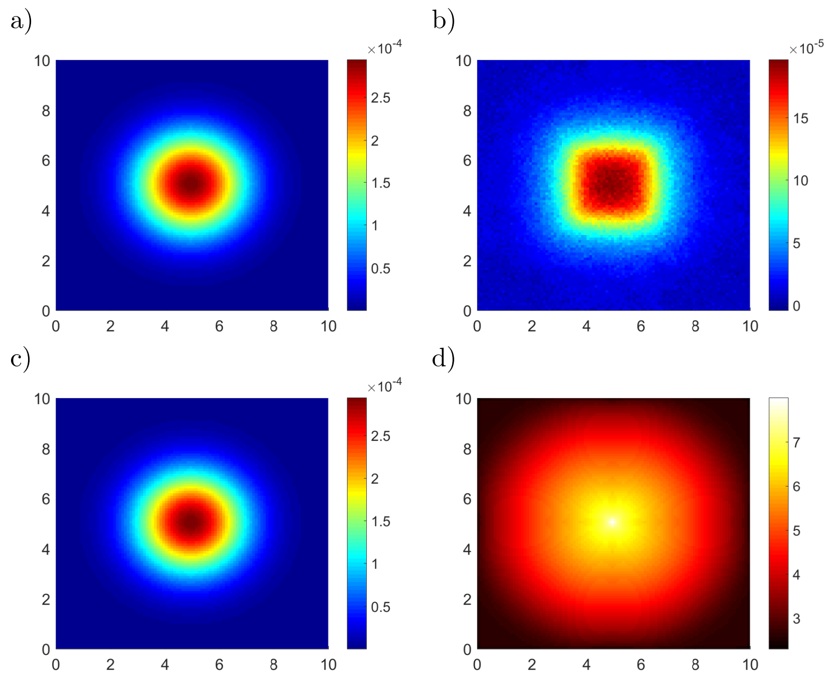
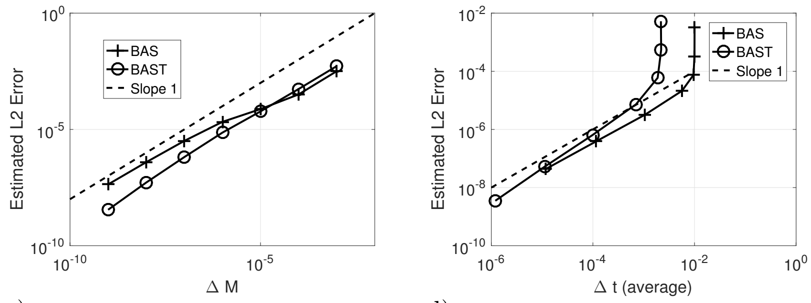


Figure 8: For the system described in Section 4.3. a) The reference solution was computed with an exponential integrator. b) Solution produced by BAS with $\Delta M = 10^{-6}$; here ΔM is too great to allow close agreement with the comparison solve and we can observe 'chequerboard' effects c) Solution produced by BAS with $\Delta M = 10^{-9}$; this solution is in close agreement with the comparison solve. d) Shows logarithm of number of events experienced by each cell for the run with BAS and $\Delta M = 10^{-9}$.

a)

b)



c)

d)

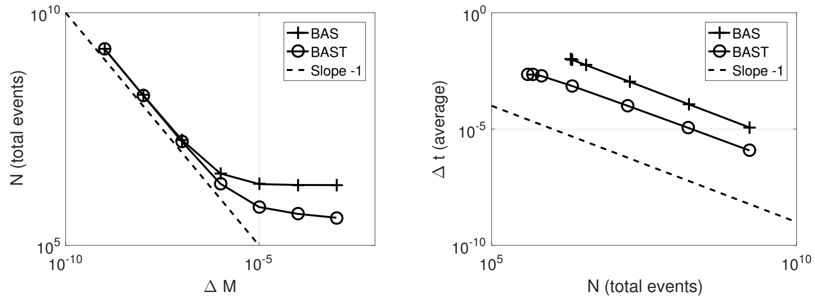


Figure 9: Results for the experiment described in Section 3.3. a) Estimated error against ΔM to indicate convergence. b) Estimated error against average event timestep. c) Total number of events N, against ΔM. d) Total number of events N against average event timestep.

5. Towards a General Convergence Result for BAS

5.1. Connection Matrices

One event in the scheme of Algorithm 1 is the transfer of mass across the active face k, between the two cells j_1 and j_2 adjacent to k. In effect, during the event, the face k and the two cells are being considered as an independent system from the rest of the domain. The only free variables are the masses m_{j_1} and m_{j_2} in the two cells. Thus, with the finite volume discretisation of the flux in place, the local flow of mass across the face may be considered as a 2×2 ODE system. Consider (6) for only two cells, after multiplying out each cell equation by the volume V_j, we have

$$\begin{pmatrix} \left(\frac{dm_{j_1}}{dt}\right)_k \\ \left(\frac{dm_{j_2}}{dt}\right)_k \end{pmatrix} = \begin{pmatrix} -a_k & b_k \\ a_k & -b_k \end{pmatrix} \begin{pmatrix} m_{j_1} \\ m_{j_2} \end{pmatrix} = \begin{pmatrix} A_k f_k \\ -A_k f_k \end{pmatrix}. \tag{14}$$

The non-negative scalars a_k, b_k are functions of the diffusivity D_j and velocity v_j of the two cells, the distance between their centres, and the area of the face k. Recalling equations (4) and (5), we can see that, if j_1 is the upwind cell, then a and b are,

$$a_k = \bar{D}_k \frac{1}{V_{j_1} \Delta x_k} + v, \;\; b_k = \bar{D}_k \frac{1}{V_{j_2} \Delta x_k},$$

or, if j_2 is the upwind cell,

$$a_k = \bar{D}_k \frac{1}{V_{j_1} \Delta x_k}, \;\; b_k = \bar{D}_k \frac{1}{V_{j_2} \Delta x_k} + v,$$

where v is the scalar product of the velocity at the centre of the face, with the unit vector in the direction of the line connecting the centres of the two cells, pointing from the upwind into the downwind cell. Thus we see that a_k and b_k are indeed non-negative, since $\bar{D}_k$ and v are both non-negative.

The matrix in (14) is an example of what we henceforth refer to as a local *connection matrix* $\tilde{L}_k$. The corresponding global connection matrix L_k is the sparse matrix with nonzero elements only at (j_1, j_1), (j_1, j_2), (j_2, j_1) and (j_2, j_2);

$$L_k \equiv \begin{pmatrix} & & & \\ & -a_k & b_k & \\ & & & \\ & a_k & -b_k & \\ & & & \end{pmatrix} \in \mathbb{R}^{J \times J}. \tag{15}$$

The structure of the connection matrix reflects the conservation of mass between the two adjacent cells (since the column sum is zero). The connection matrix L_k associated with face k describes the relationship between the two cells j_1 and j_2 adjacent to face k in the discretisation (6), and thus has nonzero entries only in columns and rows j_1 and j_2.

Let $\mathbf{m}$ be the vector of all mass values in the system and $\mathbf{c}$ the vector of all concentration values in the system, related by $\mathbf{c} = \mathbf{m}\mathbf{V}$, where $\mathbf{V}$ is the diagonal matrix with entries $\frac{1}{V_j}$, i.e., the inverse of the volume in each cell. The global ODE system for $\mathbf{m}$ can be accumulated from the connection global matrices on each face, that is,

$$\frac{d\mathbf{m}}{dt} = \sum_{k \in \mathcal{F}} L_k \mathbf{m}.$$

Right multiplying by $\mathbf{V}$ gives

$$\frac{d\mathbf{c}}{dt} = \sum_{k \in \mathcal{F}} L_k \mathbf{c},$$

and we see that the system discretisation matrix L in (6) is accumulated from the connection global matrices on every face, that is,

$$L = \sum_{k \in \mathcal{F}} L_k. \tag{16}$$

Consider the local description of an event across face k with adjacent cells j_1, j_2. Lines $5-7$ in Algorithm 1 describe an update that is equivalent to an Euler type step for solving (14), i.e.,

$$\begin{pmatrix} m_{j_1} \\ m_{j_2} \end{pmatrix} \leftarrow (I + \Delta t_k \tilde{L}_k) \begin{pmatrix} m_{j_1} \\ m_{j_2} \end{pmatrix}, \tag{17}$$

where I is the identity matrix. Alternatively, using the $J \times J$ connection matrix L_k, then we can express event updates in terms of the entire system. The full system version of (17) is

$$\mathbf{m} \leftarrow (I + \Delta t_k L_k)\mathbf{m}. \tag{18}$$

Due to the sparsity of L_k, clearly only the cells j_1, j_2 are affected by (18) even though the equation describes the entire system.

We now describe properties of global connection matrices. A connection matrix acting on any vector produces a vector pointing in only one direction in the solution space. That is, the action of a connection matrix L_k on any vector $\mathbf{x}$ is a scalar multiple of a vector $\hat{\mathbf{z}}_k$, determined by L_k. Consider a connection matrix L_k with non-empty columns and rows j_1, j_2, then

$$L_k \mathbf{x} = (b_k x_{j_2} - a_k x_{j_1})\hat{\mathbf{z}}_k, \tag{19}$$

where $\hat{\mathbf{z}}_k = (0, \ldots, 0, 1, 0, \ldots, 0, -1, 0, \ldots, 0)^T$, where the non-zero entries are at j_1 and j_2. It follows that $\hat{\mathbf{z}}_k$ is an eigenvector of L_k and the corresponding eigenvalue can be found,

$$L_k \hat{\mathbf{z}}_k = \lambda_k \hat{\mathbf{z}}_k \qquad \lambda_k = -(a_k + b_k), \tag{20}$$

thus the eigenvalue λ_k is negative.

5.2. Framework for analysis

Here we present a framework for the analysis of BAS based on the connection matrix formulation. In particular we use the fact that the action of a connection matrix L_i on any vector $\mathbf{y}$ produces a scalar, determined by $\mathbf{y}$, multiplying a direction vector $\hat{z}_i$. For two connection matrices L_i and L_j, with corresponding direction vectors $\hat{z}_i$, $\hat{z}_j$, define $c_{i,j}$ to be such that

$$L_i \hat{z}_j = c_{i,j} \hat{z}_i,$$

and vice versa for $c_{j,i}$. The eigenvalue of L_i from (20) is then $\lambda_i = c_{i,i}$. Define the matrix C as having the entries $(C)_{i,j} = c_{i,j}$. We use the fact that L is the sum of K connection matrices, $L = \sum_{k=1}^{K} L_k$. Let $\hat{Z}$ be the matrix whose kth column is $\hat{z}_k$, i.e. $\hat{Z} \equiv (\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_K)$.

The following relationship between L, $\hat{Z}$ and C will be useful.

Lemma 5.1. *Given L, $\hat{Z}$ and C as defined, $L\hat{Z} = \hat{Z}C$ and thus, $L^n\hat{Z} = \hat{Z}C^n$ for any positive integer n.*

Proof. For the first relation we note that

$$L\hat{Z} = \sum_{k=1}^{K} L_k \hat{Z} = \sum_{k=1}^{K} (c_{k,1}\hat{z}_k, c_{k,2}\hat{z}_k, \ldots, c_{k,K}, \hat{z}_k)$$

given the definitions of L, $\hat{Z}$, and $c_{k,i}$. Then consider

$$\hat{Z}C = \left(\sum_{k=1}^{K} c_{k,1}\hat{z}_k, \sum_{k=1}^{K} c_{k,2}\hat{z}_k, \ldots, \sum_{k=1}^{K} c_{k,K}\hat{z}_k \right),$$

from which we see that $L\hat{Z} = \hat{Z}C$ follows. The second relation in the lemma is obtained by inductive application of the first. $\square$

Consider the action of L on the system's initial mass vector $\mathbf{m_0}$,

$$L\mathbf{m}_0 = \sum_{k=1}^{K} f_k \hat{z}_k,$$

where we have defined f_k by $L_k\mathbf{m}_0 = f_k\hat{z}_k$, using (19). Let $\mathbf{f}_0$ be the vector whose kth entry is f_k, then $L\mathbf{m}_0 = \hat{Z}\mathbf{f}_0$. From this and Lemma 5.1 we have $L^n\mathbf{m}_0 = \hat{Z}C^{n-1}\mathbf{f}_0$ which we can use to re-express $e^{tL}\mathbf{m}_0$, the exact solution to $\frac{d\mathbf{m}}{dt} = L\mathbf{m}$ (equivalent to (6) after multiplying through each cell j by its volume V_j), as

$$e^{tL}\mathbf{m}_0 = \mathbf{m}_0 + \hat{Z} \sum_{i=1}^{\infty} \frac{t^i C^{i-1}}{i!} \mathbf{f}_0.$$

Using the series definition of $\varphi_1(z) = \sum_{i=0}^{\infty} z^i/(i+1)!$, we can rewrite this as

$$e^{tL}\mathbf{m}_0 = \mathbf{m}_0 + t\hat{Z}\varphi_1(tC)\mathbf{f}_0. \tag{21}$$

For the scheme BAS, after some total number of events n, let n_k be the number of events experienced by face k. Let $\mathbf{n}$ be the vector whose kth entry is n_k. Then the state of BAS can be expressed as,

$$\mathbf{m}_n = \mathbf{m}_0 + \Delta M \hat{Z}\mathbf{n}. \tag{22}$$

Note that we are assuming that the direction of mass transfer is consistent across each face across the whole solve (i.e. so that the direction of transfer is never reversed from a previous step), which may not be completely justified in all cases. Comparing (21) and (22), we have a sufficient condition for convergence.

Lemma 5.2. *Assuming the direction of mass transfer is consistent across each face across the whole solve, BAS will converge if*

$$\Delta M \hat{Z}\mathbf{n} \to t\hat{Z}\varphi_1(tC)\mathbf{f}_0 \qquad \text{as } \Delta M \to 0, \tag{23}$$

when $\mathbf{n}$ *evolves according to the rules of the scheme in Algorithm 1.*

A sketch proof of (23) is as follows. First we approximate $\Delta M\mathbf{n}$ by a continuous variable, $\mathbf{x} = \Delta M\mathbf{n}$. We assume that in the limit $\Delta M \to 0$ this is justifiable, as ΔM becomes so small that integer multiples of ΔM become effectively continuous. We wish to argue that

$$\frac{d\mathbf{x}}{dt} = C\mathbf{x} + \mathbf{f}_0. \tag{24}$$

Given that $\mathbf{x}(0) = 0$, the solution to this is

$$\mathbf{x}(t) = t\varphi_1(tC)\mathbf{f}_0,$$

from which (23) would follow. Note that the right hand side of (24) is the flux. To see this consider the action of a L_k on $\mathbf{m}_n$, using (22)

$$L_k\mathbf{m}_n = \Delta M(c_{k,1}, \ldots c_{k,K})\mathbf{n}\hat{z}_k + L_k\mathbf{m}_0. \tag{25}$$

(For this we used $L_k\hat{Z}\mathbf{n} = (c_{k,1}\hat{z}_k, c_{k,2}\hat{z}_k, c_{k,3}\hat{z}_k, \ldots c_{k,K}\hat{z}_k)\mathbf{n} = (c_{k,1}, \ldots c_{k,K})\mathbf{n}\hat{z}_k$.) The vector on the right hand side of (25) has only two nonzero entries, the positive and negative of the flux across face k. Since $\hat{z}_k$ has only nonzero entries -1 and 1, the flux across face k is the coefficient of the right hand side. Accumulating over every face k, we have,

$$\text{total flux in each cell} = L\mathbf{m}_n = \Delta M C\mathbf{n} + \mathbf{f}_0.$$

We must interpret the t in the derivative in (24) as the system time, i.e. the time of the face which has most recently updated. Since $\mathbf{m}_n = \mathbf{m}_0 + \hat{Z}\mathbf{x}$, $\mathbf{x}$ is the vector of displacements along each direction vector $\hat{z}$, from the starting point of $\mathbf{m}_0$. Thus (24), if true, implies that the rate of change of the solution in the direction of a $\hat{z}$ associated with a face k, with respect to the system time, is equal to the flux across face k.

We can ask if anything in the construction of the scheme indicates the potential for this behaviour. We can examine (7), the equation for determining update time for a face. We restate it here for convenience,

$$\hat{t}_k = t_k + \frac{\Delta M}{f_k},$$

where f_k is the flux across the face k, $\hat{t}_k$ is the update time of the face, and t_k is the time of the face. If the face is chosen for an event (by having a lowest update time), then it updates with timestep $\Delta t_k = \hat{t}_k - t_k$. We may re-arrange to $\frac{\Delta M}{\Delta t} = f_k$. Heuristically, in the limit $\Delta M \to 0$ we may replace the fraction with $\frac{dx_k}{dt}$, and write

$$\frac{dx_k}{dt} = f_k.$$

A vector of these values would give (24). There is however the need to bridge the gap between the asynchronous nature of the algorithm and the synchronous nature of the ODE (24). For this we would have to assume or demonstrate that in the limit $\Delta M \to 0$, the individual face times t_k tend towards being equal or arbitrarily close to the entire system time t. This is a potential subject of further work.

6. Concluding Remarks

New simulation methods based on discrete asynchronous events have been developed and tested. The schemes were BAS, the simplest implementation of the methodology, and BAST, which adds a mass-tracking feature which reduces error due to asynchronicity. In addition, a cascading modification similar to that introduced in [2] was tested for one example, and reaction-term modifications were implemented and tested for BAS and BAST. From these tests we see that the new asynchronous schemes converge in error, for fixed spatial grids, as $\Delta M \to 0$. The order of convergence is appears to be approximately $O(\Delta M)$ according to the numerical results. There also seems to be a regime of sufficiently low ΔM in which parameter relationships emerge. These relationships are, Error $= O(\bar{\Delta t})$ (where $\bar{\Delta t}$ denotes the average of Δt over the solve), $N = O(\Delta M^{-1})$, $\bar{\Delta t} = O(\Delta M)$, and $\bar{\Delta t} = O(N^{-1})$. The convergence results also indicate the basic viability of the face based asynchronous schemes, and the fact that the same conclusions can be drawn for BAS and the different modified schemes, implies the existence of a large space of possible viable schemes of this class.

We note that the relation $\bar{\Delta t} = O(N^{-1})$ can be explained a priori, following from the fact that every face will have timesteps summing to T, and that N is the sum of the number of events on each face. The way that $\bar{\Delta t}$ is calculated for the non-tracking schemes BAS is then equivalent to $\bar{\Delta t} = \frac{TK}{N}$, where K is the number of faces. This must be modified for the mass-tracking scheme BAST but a similar a priori relation can certainly be found. We note further that then the relationship $\bar{\Delta t} = O(\Delta M)$ is equivalent to $O(N^{-1}) = O(\Delta M)$, so that

these two observed relations are equivalent. Also, the relations $\bar{\Delta}t = O(\Delta M)$ and Error $= O(\Delta M)$ together imply Error $= O(\bar{\Delta}t)$.
This leaves the observations Error $= O(\Delta M)$ and $N = O(\Delta M^{-1})$ as independent and requiring theoretical explanation. The relation $N = O(\Delta M^{-1})$ may seem to follow naturally from the construction of the schemes, but showing this rigorously while taking account of the asynchronous nature of the schemes is nontrivial.
For the first order error relation, Error $= O(\Delta M)$, we may observe that first order is the best we could expect, given that the flux update relations are first order, which would impose an order barrier on the scheme as a whole. The fact that the order of the scheme is not worse than that of the flux updates, i.e., that the asynchronicity does not reduce the order, is what is interesting. While care was taken in optimizing our codes, they remain essentially demonstration pieces and so we do not compare the efficiency against well established methods. For obvious reasons the implementation of DES based schemes for continuous systems such as these is not as well understood as for classical schemes. The new schemes demonstrate convergence and can be applied to large scale problems in three dimensions and offer complete adaptivity.

Acknowledgments
The work of Dr D. Stone was funded by the SFC/EPSRC(EP/G036136/1).

References

[1] H. Omelchenko and H. Karimabadi. Event-driven, hybrid particle-in-cell simulation: A new paradigm for multi-scale plasma modeling. *Journal of Computational Physics*, 216:153–178, 2006.

[2] H. Omelchenko and H. Karimabadi. Self-adaptive time integration of flux-conservative equations with sources. *Journal of Computational Physics*, 216:179–194, 2006.

[3] T. Unfer, Jean-Pierre Boeuf, F. Rogier, and F. Thivet. An asynchronous scheme with local time stepping for multi-scale transport problems: application to gas discharges. *Journal of Computational Physics*, 227(2):898–918, 2007.

[4] Yuri A Omelchenko and Homa Karimabadi. Hypers: A unidimensional asynchronous framework for multiscale hybrid simulations. *Journal of Computational Physics*, 231(4):1766–1780, 2012.

[5] Richard M Fujimoto. *Parallel and distributed simulation systems*, volume 300. Wiley New York, 2000.

[6] K. W. Morton and D. F. Mayers. *Numerical solution of partial differential equations.* Cambridge University Press, Cambridge, second edition, 2005.

[7] S Geiger, A Cortis, and JT Birkholzer. Upscaling solute transport in naturally fractured porous media with the continuous time random walk method. *Water Resources Research*, 46(12), 2010.

[8] C. W. Gardiner. Handbook of stochastic methods for physics, chemistry and the natural sciences. *Applied Optics*, 25:3145, 1986.

[9] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.

[10] R. Erban, J. Chapman, and P. Maini. A practical guide to stochastic simulations of reaction-diffusion processes. *arXiv preprint arXiv:0704.1908*, 2007.

[11] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The journal of physical chemistry A*, 104(9):1876–1889, 2000.

[12] L. F. Shampine. Error estimation and control for ODEs. *Journal of Scientific Computing*, 25(1-2):3–16, 2005.

[13] R. B. Lowrie, P. L. Roe, and B. Van Leer. Space-time methods for hyperbolic conservation laws. In *Barriers and Challenges in Computational Fluid Dynamics*, pages 79–98. Springer, 1998.

[14] S. Osher and R. Sanders. Numerical approximations to nonlinear conservation laws with locally varying time and space grids. *Mathematics of Computation*, 41(164):321–336, 1983.

[15] B. F. Sanders. Integration of a shallow water model with a local time step. *Journal of Hydraulic Research*, 46(4):466–475, 2008.

[16] V. Savcenco, W. Hundsdorfer, and J. G. Verwer. A multirate time stepping strategy for stiff ordinary differential equations. *BIT*, 47(1):137–155, 2007.

[17] N. F. Otani. Computer modeling in cardiac electrophysiology. *Journal of Computational Physics*, 161(1):21–34, 2000.

[18] Helge Holden and Nils Henrik Risebro. *Front tracking for hyperbolic conservation laws*, volume 152. Springer, 2015.

[19] M. Hochbruck and A Osterman. Exponential integrators. *Acta Numerica*, pages 209–286, 2010.

[20] M. Hochbruck and A. Ostermann. Exponential Runge-Kutta methods for parabolic problems. *Appl. Numer. Math.*, 53(2-4):323–339, 2005.

[21] S. M. Cox and P. C. Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176:430–455, 2002.

[22] A. Tambue, G. J. Lord, and S. Geiger. An exponential integrator for advection-dominated reactive transport in heterogeneous porous media. *Journal of Computational Physics*, 229(10):3957–3969, 2010.

[23] A. Tambue, I. Berre, and J. M. Nordbotten. Efficient simulation of geothermal processes in heterogeneous porous media based on the Exponential Rosenbrock–Euler and Rosenbrock-type methods. *Advances in Water Resources*, 53:250–262, 2013.

[24] J. Niesen and W. Wright. A Krylov subspace algorithm for evaluating the ϕ-functions in exponential integrators. *arXiv:0907.4631v1*, 2009.

[25] D. Stone. *Asynchronous and exponential based numerical schemes for porous media flow.* PhD thesis, Heriot-Watt, 2015.

[26] J. Droniou. Finite volume schemes for diffusion equations: introduction to and review of modern methods. *Math. Models Methods Appl. Sci.*, 24(8):1575–1619, 2014.

[27] S. Patankar. *Numerical heat transfer and fluid flow.* CRC Press, 1980.

[28] H. K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics: the finite volume method.* Pearson Education, 2007.

[29] R. J. LeVeque. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002.

[30] K. A. Lie, S. Krogstad, I. S. Ligaarden, H. M. Natvig, J. R.and Nilsen, and B. Skaflestad. Open-source matlab implementation of consistent discretisations on complex grids. *Computational Geosciences*, 16(2):297–322, 2012.

[31] R. I. Masel. *Principles of adsorption and reaction on solid surfaces*, volume 3. John Wiley & Sons, 1996.