Adaptive finite element method assisted by stochastic simulation of chemical systems

Simon L. Cotter
Tomáš Vejchodský
Radek Erban

Preprint No. 14-2011

PRAHA 2011

ADAPTIVE FINITE ELEMENT METHOD ASSISTED BY STOCHASTIC SIMULATION OF CHEMICAL SYSTEMS

SIMON L. COTTER*, TOMÁŠ VEJCHODSKÝ†, AND RADEK ERBAN*

Abstract. Stochastic models of chemical systems are often analysed by solving the corresponding Fokker-Planck equation which is a drift-diffusion partial differential equation for the probability distribution function. Efficient numerical solution of the Fokker-Planck equation requires adaptive mesh refinements. In this paper, we present a mesh refinement approach which makes use of a stochastic simulation of the underlying chemical system. By observing the stochastic trajectory for a relatively short amount of time, the areas of the state space with non-negligible probability density are identified. By refining the finite element mesh in these areas, and coarsening elsewhere, a suitable mesh is constructed and used for the computation of the probability density.

1. Introduction.

Stochastic simulation algorithms (SSAs) have been successfully used in recent years to understand a number of biochemical models [3, 30, 39]. However, a systematic analysis of these models is challenging because of the computational intensity of SSAs. A suitable alternative to stochastic simulation is a solution of the chemical Fokker-Planck equation [18, 25]. Consider a well-mixed system of N chemical species and denote by $\mathbf{x} = (x_1, \dots, x_N)$ a vector of concentrations of these species. The stationary chemical Fokker-Planck equation is a drift-diffusion partial differential equation (PDE) for an N-dimensional probability distribution function $p \equiv p(\mathbf{x})$ where $\mathbf{x} \in \Omega \subset \mathbb{R}^N$, which can be written in the following form:

$$\operatorname{div}\left[D(\mathbf{x})\nabla p(\mathbf{x}) - \mathbf{v}(\mathbf{x})p(\mathbf{x})\right] = 0 \tag{1.1}$$

where $D \equiv D(\mathbf{x}) : \Omega \to \mathbb{R}^{N\times N}$ is the diffusion matrix, $\mathbf{v} \equiv \mathbf{v}(\mathbf{x}) : \Omega \to \mathbb{R}^N$ is the drift term and div is the divergence operator. There have been several methods developed in the literature to solve (1.1) for moderately large N. They include adaptive finite element methods (FEMs) which are commonly used for $N \leq 3$ [4] and sparse grid approaches which are applicable for larger values of N [40]. Adaptive FEMs can be used to identify a suitable mesh which is refined in crucial regions and not in others. Although these approaches can be used for the chemical Fokker-Planck equation, they do not exploit the fact that its solution is a probability distribution of a stochastic process.

In this paper, we present an adaptive mesh construction which is suitable for the FEM solution of (1.1) if this equation arises from modelling of stochastic chemical systems. The main idea is to exploit the fact that stochastic trajectories spend a significant amount of time in parts of the state space where the mesh refinement is needed. Since adaptive FEMs are mostly applicable for systems up to $N = 3$, we will focus on systems of 3 chemical species. However, the presented methodology can be modified for larger (multiscale) chemical systems provided that they have up to $n \leq 3$ important (slow) variables. In [13, 20], a method for estimation of coefficients of an effective Fokker-Planck equation is presented. This effective equation is of the same form as (1.1) but it is written in the dimension n which is smaller than a total number N of chemical species. If one has a suitable SSA for simulating the low dimensional slow dynamics [9, 10, 17], the presented mesh refinement can be applied. However,

*Mathematical Institute, University of Oxford, 24-29 St. Giles', Oxford, OX1 3LB, United Kingdom (cotter@maths.ox.ac.uk; erban@maths.ox.ac.uk).

†Institute of Mathematics, Czech Academy of Sciences, Žitná 25, 115 67 Praha 1, Czech Republic (vejchod@math.cas.cz).

```
[1] Calculate propensity functions $\alpha_k(\mathbf{X}(t))$, $k = 1, 2, \dots, M$.
[2] Waiting time $\tau$ till next reaction is given by (2.1).
[3] Choose one $j \in \{1, 2, \dots, M\}$, with probability $\alpha_j(\mathbf{X}(t))/\alpha_0(\mathbf{X}(t))$, and perform
    reaction $R_j$, by adding $\nu_{ji}$ to each $X_i(t)$ for all $i = 1, 2, \dots, N$.
[4] Continue with step [1] with time $t = t + \tau$.
```

TABLE 2.1
The pseudo code for the Gillespie SSA.

since the main aim of this paper is to present this novel numerical methodology, we will not consider any dimensional reduction and restrict to systems which are directly written in terms of $N = 3$ chemical species. In the following paper [12], we will show how this mesh refinement can be used to study bifurcation behaviour of stochastic chemical systems.

The paper is organised as follows. In Section 2, we introduce our notation, the Gillespie SSA and the chemical Fokker-Planck equation. In Section 3, we introduce the standard finite element framework that we will use throughout the paper. We also formulate the problem in its weak form in order to define the problem to be solved. In Section 4, we introduce the stochastic simulation assisted adaptive Finite Element Method (saFEM). In Section 5, we offer some insight into how the algorithmic parameters of the saFEM may be chosen in practise. In Section 6, implementational issues related to the algorithm are addressed. Numerical results concerning convergence of the method are presented in Section 7.

2. Chemical Fokker-Planck Equation.

Let us consider a well-mixed system of N chemical species X_i, $i = 1, 2, \dots, N$, which are subject to M chemical reactions R_k, $k = 1, 2, \dots, M$. The state of the system is described by the state vector $\mathbf{X}(t) = [X_1(t), X_2(t), \dots, X_N(t)]$ where $X_i(t)$ denotes the number of molecules of the corresponding chemical species. Each chemical reaction is described by its propensity function and the stoichiometric vector [19, 23]. The propensity function $\alpha_k(\mathbf{x})$ is defined in such a way that $\alpha_k(\mathbf{x})\mathrm{d}t$ is the probability that the k-th reaction occurs in the infinitesimally small interval $[t, t+\mathrm{d}t)$ provided that $\mathbf{X}(t) = \mathbf{x}$. The stoichiometric vector is $\boldsymbol{\nu}_k = [\nu_{k1}, \nu_{k2}, \dots, \nu_{kN}]$ where ν_{ki} is the change in X_i during reaction R_k. Stoichiometric vectors form the corresponding stoichiometric matrix $\nu = (\nu_{ki})_{k,i=1}^{M,N}$.

The time evolution of the state vector $\mathbf{X}(t)$ is often simulated by the Gillespie SSA [23] which is described in Table 2.1. Given the values of the propensity functions (step [**1**]), the waiting time to the next reaction is given by:

$$\tau = -\frac{\log(u)}{\alpha_0(\mathbf{X}(t))}, \quad \text{where} \quad \alpha_0(\mathbf{X}(t)) = \sum_{k=1}^{M} \alpha_k(\mathbf{X}(t)), \tag{2.1}$$

and u is a uniformly distributed random number in $(0, 1)$. The reaction R_j is chosen in step [**3**] using another uniformly distributed random number.

The stationary probability distribution corresponding to the chemical system can be approximated by solving the chemical Fokker-Planck equation [25]. This equation can be written in the form (1.1) where the diffusion and drift coefficients are:

$$d_{ij}(\mathbf{x}) = \frac{1}{2}\sum_{k=1}^{M} \nu_{ki}\nu_{kj}\alpha_k(\mathbf{x}), \qquad i,j = 1,2,\dots,N, \tag{2.2}$$

$$v_i(\mathbf{x}) = \sum_{k=1}^{M} \nu_{ki}\alpha_k(\mathbf{x}) - \sum_{j=1}^{N} \frac{\partial d_{ij}}{\partial x_j}(\mathbf{x}), \qquad i = 1,2,\dots,N. \tag{2.3}$$

This equation is solved on a bounded domain $\Omega \subset [0,\infty)^N$ in which the vast majority of the invariant probability density sits [18]. On the boundary $\partial\Omega$ we introduce the homogeneous Neumann boundary condition:

$$\big[D(\mathbf{x})\nabla p(\mathbf{x}) + \mathbf{v}(\mathbf{x})p(\mathbf{x})\big]\cdot\mathbf{n}(\mathbf{x}) = 0, \qquad \text{for } \mathbf{x}\in\partial\Omega, \tag{2.4}$$

where $\mathbf{n}(\mathbf{x})$ is the outward facing normal at $\mathbf{x}\in\partial\Omega$. We seek a solution of (1.1) which corresponds to the probability distribution function. Therefore, we impose the following normalisation condition:

$$\int_\Omega p(\mathbf{x})\,\mathrm{d}\mathbf{x} = 1. \tag{2.5}$$

The choice of the computational domain Ω might be problematic if we have no a priori information about the problem (1.1) with (2.4) and (2.5). In this case, the stochastic simulations provide a reliable tool to determine the correct Ω as we will show in Section 4.2. However, before discussing the details of saFEM, we have to introduce some finite element terminology. This will be done in the next section.

3. Finite Element Method.

Equation (1.1) with boundary condition (2.4) can be numerically solved by the FEM. Since the finite element formulation is based on the corresponding weak formulation, we first introduce the weak solution $p \in H^1(\Omega)$ by the equality

$$a(p,\phi) = 0, \qquad \forall\phi\in H^1(\Omega). \tag{3.1}$$

The bilinear form $a(\cdot,\cdot)$ is naturally given by

$$a(p,\phi) = \int_\Omega \left(D(\mathbf{x})\nabla p(\mathbf{x}) + p(\mathbf{x})\mathbf{v}(\mathbf{x})\right)\cdot\nabla\phi(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{3.2}$$

The finite element formulation is obtained by projecting the weak formulation (3.1) into a finite dimensional subspace V_h of $H^1(\Omega)$. Thus, we seek $p_h \in V_h$ such that

$$a(p_h,\phi_h) = 0, \qquad \forall\phi_h\in V_h. \tag{3.3}$$

We note that taking a basis $\phi_1,\phi_2,\dots,\phi_m$ of V_h, we can express the finite element solution as

$$p_h(\mathbf{x}) = \sum_{i=1}^{m} p^i\phi_i(\mathbf{x}), \quad \mathbf{x}\in\Omega.$$

Here, the coefficients $p^i\in\mathbb{R}$ solve the system of linear algebraic equations

$$A\mathbf{p} = 0, \tag{3.4}$$

where $\mathbf{p} = (p^1, p^2, \ldots, p^m)^T$ and the stiffness matrix $A \in \mathbb{R}^{m \times m}$ is defined by its entries

$$A_{ij} = a(\phi_j, \phi_i), \quad i, j = 1, 2, \ldots, m. \tag{3.5}$$

We construct the finite element space V_h and the corresponding basis functions in the standard way [11]. In what follows, we will focus on computations in three dimensions, i.e. $N = 3$. We consider the finite element mesh $\mathcal{T}_h$ consisting of tetrahedral elements. The lowest-order finite element space V_h then consists of globally continuous and piecewise linear functions over the mesh $\mathcal{T}_h$:

$$V_h = \{\phi_h \in H^1(\Omega) : \phi_h|_K \in \mathbb{P}^1(K) \text{ for all } K \in \mathcal{T}_h\}, \tag{3.6}$$

where $\mathbb{P}^1(K)$ stands for the space of linear functions over the tetrahedron $K \in \mathcal{T}_h$. If $\mathbf{q}_j \in \mathbb{R}^3$, $j = 1, 2, \ldots, m$, stand for the nodes of the mesh $\mathcal{T}_h$ then the standard finite element basis functions ϕ_i are uniquely determined by the condition

$$\phi_i(\mathbf{q}_j) = \delta_{ij}, \quad i, j = 1, 2, \ldots, m,$$

where δ_{ij} stands for Kronecker's symbol.
An efficient solution to problem (3.3) can be obtained by employing adaptively refined meshes [38]. The optimally adapted mesh leads to an approximation with the smallest error, provided the number of degrees of freedom is fixed. Practically, the optimal mesh can be hard to determine, but meshes close to the optimal can be found. These meshes are fine in regions where the solution exhibits steep gradients, boundary layers, interior layers or singularities, and they are relatively coarse in the other regions.
The standard numerical approach for construction of nearly optimal meshes is the adaptive algorithm based on suitable a posteriori error estimators [4, 38]. This algorithm starts with an initial coarse mesh and refines it adaptively by a sequence of refinement steps. In each step, problem (3.3) has to be solved on the actual mesh, an error indicator has to be computed for each element, and based on these indicators the mesh is refined at suitable places. Using the mesh refinements assisted by stochastic simulations, we can avoid the sequence of refinement steps and construct a suitably adapted mesh at once.

4. Adaptive mesh refinement assisted by stochastic simulations.

Since the chemical Fokker-Planck equation (1.1) is a continuous approximation of the evolution of the probability density given by the Gillespie SSA [23], one can expect that trajectories simulated from the SSA will be informative. Once the trajectory has reached probabilistic equilibrium[1], regions surrounding the path of the trajectory are likely to be regions with non-negligible invariant density with respect to the steady state Fokker-Planck equation. We should be aiming to refine the finite element mesh in regions where the rate of change of the gradient of the invariant density is larger. The area in which we should be trying to refine our mesh can be well approximated by the region which has non-negligible invariant density. Therefore, stochastic simulations of the chemical system can be informative about a good choice of finite element mesh.
The construction of the locally adapted mesh assisted by stochastic simulations is done in three stages. In Stage I, we use the stochastic simulations to identify those regions

[1]Equilibrium can be reached simply by running the SSA until the state of the trajectory is sufficiently decorrelated from its starting position, or by starting at a steady state of the mean field approximation (4.1) of the chemical system.

[1] Identify steady states (stable and unstable) of the mean field approximation of the system, or in the case of oscillatory systems, one (or more) coordinates along the limit cycle. We denote these $S \in \mathbb{N}$ points by $\{\mathbf{y}_k\}_{k=1}^S$.

[2] Run one (or more) Gillespie simulations of length $B+T > 0$ for each of the initial conditions $\{\mathbf{y}_k\}_{k=1}^S$. Here, $B \geq 0$ is the length of a possible initial transient and $T > 0$. Take a subsample of $Q > 0$ points per unit time per trajectory in the time interval $[B, B+T]$ and denote them as in (4.2). We also denote by $x_i^{\max}$ (resp. $x_i^{\min}$) the maximal (resp. minimal) value of the i-th chemical species, $i = 1, 2, 3$, during time intervals $[B, B+T]$ of all S simulations.

[3] Use (4.3), to define a neighbourhood $\Gamma \subset [0,\infty)^N$ of the points (4.2) as the region of the domain in which we require the finest level of refinement of the mesh. This is made up of ellipsoids around each point with radii given by (4.5) with parameter $\beta_1 > 0$.

[4] Define the domain of solution Ω by (4.6), using parameter $\beta_2 > 0$.

[5] Start with the mesh as one single cuboid covering the whole of Ω.

[6] Loop over all cuboids in the mesh. If the cuboid is sufficiently close (as given by (4.7)) to Γ, then split the cuboid into eight equally sized cuboids. Update the list of hanging nodes/hang type (face/edge hanging node as shown in Figure 4.2 (left)).

[7] Repeat step **[6]** until the maximum resolution has been reached after $H \in \mathbb{N}$ iterations, or the maximum total number of cuboids has been reached.

Table 4.1
Mesh generation using SSA trajectories.

of the state space, where the system spends most of its time, and hence where we are interested to resolve the problem with the highest accuracy. In Stage II, we identify the computational domain Ω as a cuboid that covers the region from Stage I and its neighbourhood. In Stage III, we construct the actual mesh in the computational domain Ω based on the information from Stage I. In what follows, we provide detailed description of these three stages in the case of $N = 3$ chemical species. Following the notation of Section 2, the chemical species will be denoted as X_1, X_2 and X_3. The mesh generation algorithm is summarised in Table 4.1.

4.1. Stage I: Stochastic simulation. In step **[1]** of Table 4.1, we identify structures that a priori should exist in the probability density. An indication of the regions which may contain significant amounts of invariant density can be found by analysing the 3-dimensional system of ordinary differential equations (ODEs)

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = v_i(x_1, x_2, x_3), \qquad i \in \{1, 2, 3\}, \tag{4.1}$$

which is closely related the mean field approximation of the chemical system. Steady states of ODE system (4.1) will often coincide with regions of the solution of the steady state Fokker-Planck equation (1.1) that have large density. The most important things to identify are stable steady states of (4.1), which can be found by solving an algebraic system $[v_1(x_1, x_2, x_3), v_2(x_1, x_2, x_3), v_3(x_1, x_2, x_3)] = 0$. In oscillating systems, one can identify limit cycles. There are several tools in the literature for analysis of ODEs of the form (4.1), such as AUTO [15].

In step **[2]**, these steady states can then be used as starting points for SSA trajectories, once the numbers of molecules of each species have been rounded to the nearest integer. One might additionally wish to ensure that each chain is starting in probabilistic equilibrium by running a short simulation of length $B \geq 0$ from these points before starting the sampling procedure. The Gillespie SSA is detailed in Table 2.1. In the case of limit cycles, several points along the limit cycle can be used as starting points for the SSA. Either way, we denote the S starting positions for the trajectories by $\{\mathbf{y}_k\}_{k=1}^{S} \subset \mathbb{R}^3$. The trajectories simulated up to some time $B + T > 0$ using the SSA can help inform us about the regions of domain which will have non-negligible invariant density. Since the trajectories will contain many points, we cannot store all of the points that are simulated. Instead, we subsample from the trajectories at equidistant time points, at a rate $Q > 0$ points per unit time. This leaves us with a set of sampled points

$$\{\mathbf{z}_{k,l}\}_{k=1,l=1}^{S,\lfloor QT \rfloor} \subset \mathbb{R}^3 \tag{4.2}$$

from the invariant distribution, where $\lfloor QT \rfloor$ denotes the integer part of the real number QT. We hope to recover, from this set of points (4.2), information about what might be an optimal finite element mesh. We also denote by $x_i^{\max}$ (resp. $x_i^{\min}$) the maximal (resp. minimal) value of the i-th chemical species, $i = 1, 2, 3$, during time intervals $[B, B + T]$ of all S simulations. These numbers will be useful in (4.4)–(4.5). In step **[3]**, we make the approximation that the region which contains the majority of the invariant density, is a subset of the union of a set of ellipsoids centred at each point (4.2), namely

$$\Gamma \equiv \bigcup_{k,l} \mathcal{E}_{\mathbf{r}}(\mathbf{z}_{k,l}). \tag{4.3}$$

Here, $\mathcal{E}_{\mathbf{r}}(\mathbf{z}_{k,l})$ is an ellipsoid with radii $\mathbf{r} = [r_1, r_2, r_3]$ centred at point $\mathbf{z}_{k,l}$ for $k = 1, 2, \ldots, S$, $l = 1, 2, \ldots, \lfloor QT \rfloor$. These radii can be picked to be proportional to the range of each chemical species using the parameter $\beta_1 > 0$ and numbers $x_i^{\min}$, $x_i^{\max}$, $i = 1, 2, 3$, computed in step **[2]**. Namely, we define

$$x_i^{\text{range}} = x_i^{\max} - x_i^{\min}, \qquad\qquad i = 1, 2, 3, \tag{4.4}$$

and

$$\mathbf{r} = \beta_1 \left(x_1^{\text{range}},\, x_2^{\text{range}},\, x_3^{\text{range}} \right). \tag{4.5}$$

One should note that the parameters B and T might in principle be different for different initial conditions y_k, but to simplify the notation, we do not stress this fact by using the notation B_k and T_k and use simply B and T.

4.2. Stage II: Automatic detection of the domain. Next, we would like to identify the domain Ω on which we wish to solve (1.1). Since we already have an approximation of the region which contains the majority of the probability density, we can simply fit a cuboid around this region, and then extend it by a factor. The only other condition that we enforce is that the domain must be contained by the positive quadrant of the state space. In step **[4]**, we pick a parameter $\beta_2 > 0$ to define how much we would like to extend the cuboid:

$$\Omega = \mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{A}_3, \tag{4.6}$$

where

$$\mathcal{A}_i = \left(\max\{0, x_i^{\min} - \beta_2\, x_i^{\text{range}}\}, x_i^{\max} + \beta_2\, x_i^{\text{range}}\right), \qquad i = 1, 2, 3.$$

We then wish to solve (1.1) on the domain Ω with boundary conditions (2.4) on $\partial\Omega$.

4.3. Stage III: Construction of the adaptive mesh. In this stage, we construct the adaptively refined mesh. In the previous stage we naturally defined the computational domain as a cuboid (4.6). Therefore, we base our refinement approach on simple splitting of a cuboid into 8 congruent sub-cuboids, as shown in Figure 4.1. However, any other standard mesh refinement technique can be utilised instead.

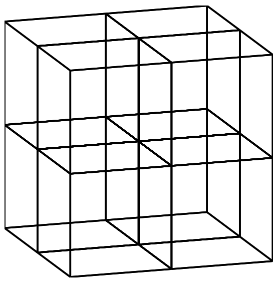


FIG. 4.1. *Example of a single cuboid after the proposed refinement into* 8 *sub-cuboids.*

An advantage of the refinement of cuboids into 8 sub-cuboids is its simplicity. A disadvantage is that these refinements produce so-called hanging nodes in the mesh, see Figure 4.2 (left).

In step **[5]** of the mesh generation, our mesh consists of a single large cuboid. We then, in step **[6]**, iteratively refine cuboids in the following fashion. In each iteration of the refinement procedure, we loop over all of the current cuboids that exist following the previous iteration of the method. For each of the cuboids $I_1 \times I_2 \times I_3$, where $I_1, I_2, I_3 \subset [0, \infty)$ are intervals, we compute the minimum distance between all points in this cuboid, and the set of points (4.3) in each coordinate, namely

$$\operatorname{dist}(\Gamma, I_i) = \inf\{|a_i - b_i| : [a_1, a_2, a_3] \in \Gamma,\ b_i \in I_i\}, \qquad i = 1, 2, 3.$$

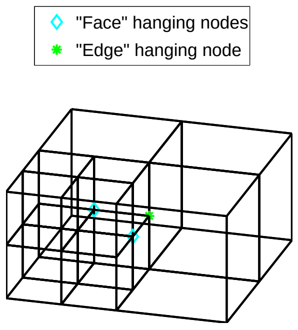
We refine the given cuboid $I_1 \times I_2 \times I_3$ if

$$\operatorname{dist}(\Gamma, I_i) < |I_i| \qquad \text{is satisfied for every } i = 1, 2, 3. \tag{4.7}$$

If a cuboid is to be refined, it is split into 8 cuboids of equal volume. As detailed in step **[7]**, this refinement condition can be iterated for a set number of times $H \in \mathbb{N}$, where $\mathbb{N} \equiv \{1, 2, 3, \ldots\}$, or until the number of cuboids present in the mesh is as large as we would like or can deal with given our computational resources.

Notice that the algorithm described above produces hanging nodes of order one only. This means that two neighbouring elements in the mesh are either of equal size or one of them has eight times greater volume than the other one. This is important for practical implementation, because the hanging nodes of order one are much easier to work with than the hanging nodes of higher orders. See [37] for more details and Figure 4.2(b) for an illustration in 2D. In order to guarantee the continuity of the

(a)



(b)

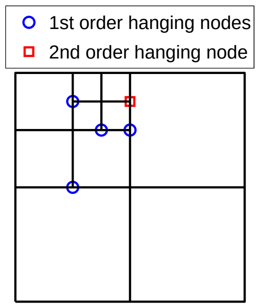
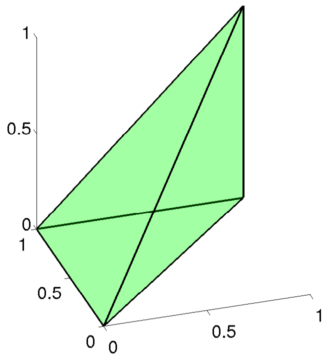


FIG. 4.2. (a) 1st *order hanging nodes of different types in cuboid meshes.* (b) *Hanging nodes of order* 1 *and order* 2 *on a* 2D *mesh.*

(a)



(b)

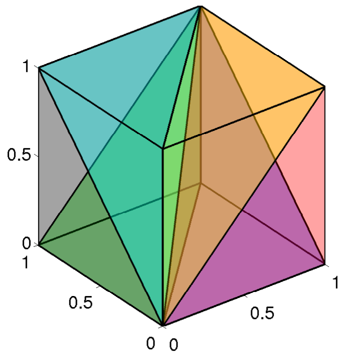


FIG. 4.3. (a) *Example of one element from the refinement scheme.* (b) *Example of a unit cube split into 6 elements of equal size and shape.*

approximation, the value of the function at the hanging node is necessarily decided by the values of the function at the vertices of the less refined cube. Therefore the hanging nodes are not in actual fact additional degrees of freedom in the problem. Thus, the dimension of problem (3.4) is equal to the total number of vertices in the generated mesh, less the number of hanging nodes. This special treatment of hanging nodes actually enforces on us a mesh which is highly refined in the regions which we wish it to be, and then the mesh becomes gradually coarser as we move away from those regions.

4.4. Tetrahedral mesh. Once the cuboid mesh has been generated by steps [**1**]–[**7**] of Table 4.1, we can then implement a finite element method on it. In the numerics shown in this paper, we further split each cuboid into 6 path tetrahedra. However there is nothing to say that we could not use cubic elements, but we use tetrahedra to simplify some implementational issues.

We choose a refinement regime in which the elements are clustered in groups of 6

non-obtuse[2] tetrahedra which together form each cuboid [7]. Figure 4.3(a) shows an example element from the refinement scheme. The idea is that if we map each cuboid to the unit cube, then each element is exactly of the same shape and size as all the others. Figure 4.3(b) shows how 6 of these elements can tessellate into a cube.
A lot of refinement methods in the literature involve splitting each element into several smaller elements [31]. However, if this is not done in a clever way, this can lead to degradation of the quality of the elements. That is, some of the angles of the elements may become too small, leading to long thin elements, which can lead to less accuracy in the approximation [8].

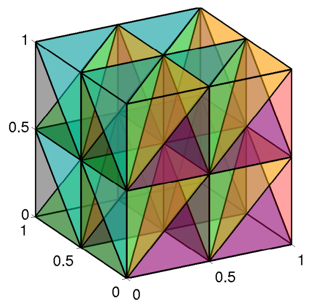


FIG. 4.4. *Example of a unit cube after a refinement iteration.*

In the method that we propose here, instead of splitting each element, we split each cuboid into 8 equally sized cuboids. Each cuboid is then split, as before, into 6 equally sized and shaped (after a linear transformation if not a cube) tetrahedral elements. Figure 4.4 shows a cube which has been refined once and been split into 48 elements, with 27 vertices.

5. Parameter Selection.

The algorithm has several parameters S, T, Q, B, β_1, β_2 and H whose values must be decided by the user. The parameters S, T, Q, B relate to the implementation of the Gillespie SSA, and the remaining three β_1, β_2, H relate to the selection of the domain and the mesh, given the recorded output of the SSA. In this section we suggest how one might go about selecting values for these parameters.

5.1. SSA parameters.

The purpose of simulating the SSA trajectories, as described in Subsection 4.1, is to get an indication of the regions of the domain which contain the majority of the invariant probability density. There are four parameters to consider, $S \in \mathbb{N}$, $T > 0$, $Q > 0$ and $B \geq 0$. The parameter B represents the length of the simulated trajectory that we ignore at the beginning of the simulation. This period of simulation is simply used to ensure that the Markov chain has entered probabilistic equilibrium. This ensures that it is highly likely the point in state space where the rest of the trajectory starts has non-negligible density with respect to the invariant distribution. The final parameter, Q, represents the rate (per unit time) at which we record samples from each trajectory in the time interval $[B, B+T]$. This parameter is necessary since if we recorded every single state that the trajectory passed through, we could quickly run out of memory.

[2]All six dihedral angles between its faces are less than or equal to $\pi/2$

Choosing sensible values for these parameters is not easy, since it is problem dependent. However, our aim is not to run such a long trajectory that simply using a histogram of the values would lead to an accurate approximation of the invariant density, as this would completely negate the need for the rest of the algorithm. Likewise, if the trajectory is too short, we will have a very poor approximation of the areas of the domain which contain the majority of the probability density. Our aim, therefore, is to get as good an approximation of this region as possible with the shortest possible trajectories.
As with many Monte Carlo estimates, due to the law of large numbers, the error decays at a rate proportional to $1/\sqrt{T}$. It is reasonable to expect that the approximation of the region which contains the majority of the invariant density should decay at a similar rate. Since convergence diagnostics relating to Markov chains are an open area with no hard and fast solution [14], we suggest that a sample trajectory be created a priori to ascertain the relaxation time of the system. The parameter B can simply be set to be a proportion of the length T, for instance $B = T/10$. The parameter Q should be picked according to how much memory one would like to set aside to store the sampled points. Since we are calculating minimum distances to any point in this set in step **[6]** of the algorithm, the more points there are, the longer the algorithm will take to run.

5.2. Domain and mesh generation parameters. The domain and mesh generation parameters are given by $\beta_1 > 0$, $\beta_2 > 0$ and $H \in \mathbb{N}$. The parameter β_2 defines the size of the domain as seen in (4.6). Unless the trajectories from the SSA are very short, the estimation of the domain should be relatively trivial, and a value of order 1 should suffice in most instances. The value of β_1, which indicates how much error we estimate there to be in our estimation of the region which contains the majority of the invariant density, should be directly proportional to $1/\sqrt{T}$. Too large a value will lead to unnecessary refinement in some areas, while too small a value will lead to not enough refinement and more error. The parameter H indicates the maximum number of splits that the initial cubes of the mesh may undergo. In practice this can be found at runtime, by capping the total number of elements we allow in the mesh due to memory and CPU constraints.

5.3. A numerical test of accuracy. Although the method has seven parameters which have to be specified, it is relatively easy to check that the computed results are numerically accurate. Given the parameter set $\{S, T, Q, B, \beta_1, \beta_2, H\}$, we can compare the computed results with the results obtained with the parameter set $\{S, 2T, 2Q, 2B, 2\beta_1, \beta_2, H+1\}$. If the result does not change significantly, then we can conclude that our parameters were chosen sufficiently large. Here, the parameters T, Q, B, and β_1, are multiplied by 2 because increasing any of these parameters will increase the accuracy of the solution. In a similar way H can be increased by one, which means that the finest level of refinement in the mesh becomes smaller by a factor of two in each coordinate. However, we do not propose to modify S and β_2 for the following reasons. The number of starting points S can be determined by the number of stable equilibria of (4.1). In this case, the parameter S does not have to be changed during this a posteriori test of accuracy. Furthermore, β_2 is used only to choose the size of the domain, and multiplying the size of the domain by 2 would lead to a smaller proportion of the domain being covered by Γ (given by (4.3)), and as such a more accurate solution would not be guaranteed. If one wishes to test whether β_2 is large enough, the parameter H may have to be increased as a function of β_2 in order to maintain the same level of refinement in Γ.

6. Implementation of the saFEM. As soon as the computational mesh is determined, the approximation $p_h \in V_h$ of the invariant distribution p is computed by the standard FEM approach [11, 37]. In order to compute the entries of the stiffness matrix (3.5), it is necessary to use suitable quadrature routines. In numerical examples below, we consider chemical reactions of at most second order. Therefore, the diffusion $D(\mathbf{x})$ and drift $\mathbf{v}(\mathbf{x})$ coefficients are polynomials of degree at most two. Since the finite element space consists of piecewise linear functions, it suffices to use a tetrahedral quadrature rule of order three which integrates cubic polynomials exactly. The optimal (Gauss) quadrature rule on tetrahedra of order three has 5 points [37].
Since the dimension of the resulting algebraic system (3.4) can be very large, even for relatively coarse approximations, parallelisation of the assembly process is of paramount importance. Several packages exist for constructing matrices in parallel. The Portable Extensible Toolkit for Scientific Computation (PETSc) is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modelled by PDEs [5,6]. The matrix is split into sections which are controlled by each processor. If the degrees of freedom are ordered in a sensible way, then Message Passing Interface (MPI) traffic between the processors can be kept to a minimum, leading to excellent scaling of processors vs. computation time [6].
Once the stiffness matrix (3.5) is assembled, we have to find a nontrivial solution of the algebraic system (3.4). Practically, we look for the eigenvector corresponding to the zero eigenvalue of the matrix A. To find this eigenvector in parallel, we use a sister package of PETSc, which is called the Scalable Library for Eigenvalue Problem Computations (SLEPc) [28]. In particular, we used the MUltifrontal Massively Parallel sparse direct Solver (MUMPS) [1, 2] package for the preconditioning, and a power method to solve the resulting eigenvalue problem [32].
Once we have calculated the eigenvector, it is then necessary to reconstruct the approximated function. For this, we use the information regarding the hanging nodes so that we reconstruct all of the vertices on the mesh. The function in question is a probability density and therefore we normalise the obtained finite element solution p_h such that it satisfies the condition (2.5).

7. Numerical Results. In this section, we first use a simple example chemical system from [13] and implement the saFEM on a range of different mesh sizes, and with different algorithmic parameters. Then we present the results of saFEM for the Oregonator [22].

7.1. Convergence of the numerical method. We will study the system of three chemical species X_1, X_2 and X_3 which are subject to the following system of five chemical reactions [13]:

$$\emptyset \xrightarrow{k_1} X_1 \underset{k_3}{\overset{k_2}{\rightleftarrows}} X_2 \xrightarrow{k_4} X_3 \xrightarrow{k_5} \emptyset. \tag{7.1}$$

Then the propensity functions are defined by

$$\alpha_1(t) = k_1 V, \quad \alpha_2(t) = k_2 X_1(t), \quad \alpha_3(t) = k_3 X_2(t),$$
$$\alpha_4(t) = k_4 X_2(t), \quad \alpha_5(t) = k_5 X_3(t),$$

where V is the volume of the reactor [13, 19]. We consider this system with the following set of non-dimensionalised parameters:

$$k_1 V = 100, \quad k_2 = k_3 = 5, \quad k_4 = k_5 = 1. \tag{7.2}$$

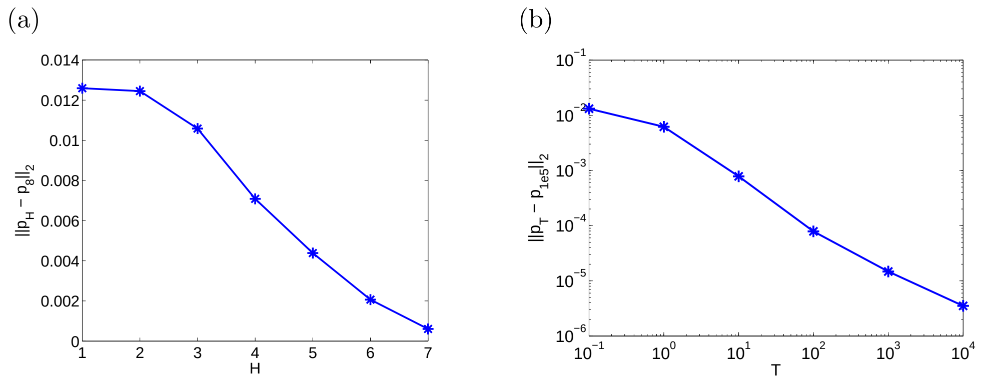


FIG. 7.1. (a) *Convergence of the saFEM with $S = 1$, $T = 10^5$, $Q = 0.1$, $B = 10^3$, $\beta_1 = 0.01$ and $\beta_2 = 0.55$ over a range of mesh refinements H, for the system* (7.1). *Errors are given by* (7.3). (b) *Convergence of the saFEM with $S = 1$, $Q = 10$, $B = 10^3$, $\beta_1 = 0.01$, $\beta_2 = 0.4$, and $H = 6$ over a range of SSA lengths T, for the system* (7.1). *Error is given by* (7.4).

As discussed in Section 5, there are seven different algorithmic parameters, whose values determine the accuracy and efficiency of the methods. In this section we will analyse the effects and convergence of the method due to altering the three most important of these parameters.

7.2. Convergence of approximation. First we consider how the error in the approximation of the invariant distribution p decays as we refine the mesh, each time using the same stochastic simulation, with $S = 1$, $T = 10^5$, $Q = 0.1$, $B = 10^3$, $\beta_1 = 0.01$ and $\beta_2 = 0.55$. We choose $S = 1$ since the corresponding ODE approximation given by (4.1) gives us:

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = 100 - 5x_1 + 5x_2,$$
$$\frac{\mathrm{d}x_2}{\mathrm{d}t} = 5x_1 - 6x_2 - 0.5,$$
$$\frac{\mathrm{d}x_3}{\mathrm{d}t} = x_2 - x_3,$$

which has a single steady state at $(119.5, 99.5, 99.5)$. Therefore we pick the single starting point for the SSA trajectory to be at $(120, 100, 100)$. Note that the constant term -0.5 in the second equation comes from the derivative of the diffusion terms as given by (2.3), while these derivatives sum to zero for the first and third equations. Figure 7.1(a) shows the convergence of the method as the number of splits H is increased from 1 to 7, where the error is calculated by comparison with an approximation using a mesh which has undergone up to 8 splits:

$$\mathrm{Error}(H) = \left(\int_\Omega |p_8(\mathbf{x}) - p_H(\mathbf{x})|^2 \mathrm{d}\mathbf{x} \right)^{1/2}. \tag{7.3}$$

When using a relatively coarse mesh, it is possible for the value of the approximation to be negative in regions. Since the function represents a probability density, which is strictly non-negative, this does not make sense. These negative areas can also complicate the normalisation of the function. We solve this problem by simply cutting off the negative values, i.e. we multiply the approximated function by the indicator function over the set where the function is non-negative, before normalising by (2.5).

7.3. Length of Stochastic Simulation. Next we show convergence of the method as T is increased, with $S = 1$, $Q = 10$, $B = 10^3$, $\beta_1 = 0.01$, $\beta_2 = 0.4$ and $H = 8$. As in the previous section, the SSA trajectory is given initial condition $(X_1, X_2, X_3) = (120, 100, 100)$.
As the system is ergodic, we would expect that as the length of stochastic simulation increases and the number of samples increases, that our samples becomes increasingly representative of the invariant density in question. This gives us more information about where we should be refining our mesh, which in turn should give us a more accurate approximation.
Figure 7.1(b) shows the convergence of approximation as the length of stochastic simulation is increased. The error as plotted in Figure 7.1 (b) is the L^2 difference between the approximations with varying T and the approximation calculated with $T = 10^5$, namely:

$$\text{Error}(T) = \left(\int_{\Omega} |p_{10^5}(\mathbf{x}) - p_T(\mathbf{x})|^2 \mathrm{d}\mathbf{x} \right)^{1/2}. \tag{7.4}$$

Note that in order to more easily compare the distributions, the domain selection for all of the data points was made using the longest stochastic simulation with $T = 10^5$. This shows that as the simulation length is increased we see a convergence of the chosen mesh to one which well represents the region containing the invariant density.

7.4. Parameter β_1. In certain situations the stochastic simulations may be expensive and we may only be able to take a few samples from the stochastic trajectory. In this case we still have a large amount of uncertainty about the region which contains the vast majority of the invariant density. Therefore we can only hope to approximate this by increasing the size of the ellipsoid around each sampled point that we include in the region Γ given by (4.3). Figure 7.2 shows the convergence of the method as β_1 is increased, with $S = 1$, $T = 10^5$, $Q = 0.1$, $B = 10^3$, $\beta_2 = 0.4$ and $H = 6$.

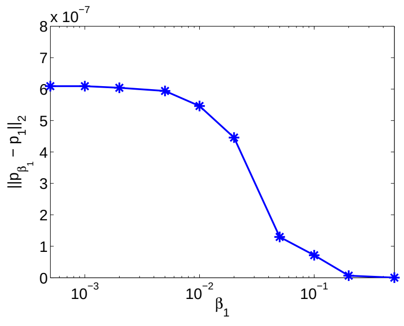


FIG. 7.2. *Convergence of the saFEM with $\beta_2 = 0.4$, $Q = 0.1$, $B = 10^3$, $T = 10^5$ and $S = 6$ over a range of radii of uncertainty β_1, for the system* (7.1). *The error is defined by* (7.5).

The error as plotted in Figure 7.2 is given by:

$$\text{Error}(\beta_1) = \left(\int_{\Omega} |p_1(\mathbf{x}) - p_{\beta_1}(\mathbf{x})|^2 \mathrm{d}\mathbf{x} \right)^{1/2}. \tag{7.5}$$

7.5. A numerical test of accuracy. In Section 5.3 a brief test for sufficient convergence was suggested, where two solutions were calculated, one with parameters

$$\{S, T, Q, B, \beta_1, \beta_2, H\},$$

and the second with parameters

$$\{S, 2T, 2Q, 2B, 2\beta_1, \beta_2, H+1\}. \tag{7.6}$$

If the two solutions are close up to some tolerance, then we can be fairly sure that the method has converged to a reasonable degree.
The solution for the system (7.1) calculated with the parameter set

$$\{S=1,\ T=10^5,\ Q=0.05,\ B=500,\ \beta_1=10^{-3},\ \beta_2=0.45,\ H=7\},$$

when compared with the solution using the parameters

$$\{S=1,\ T=2\times10^5,\ Q=0.1,\ B=10^3,\ \beta_1=2\times10^{-3},\ H=8\},$$

gave L^2 error of 1.99×10^{-4}, which is significantly less than the L^2 norm of the solution (4.51×10^{-3}). This gives a relative L^2 error of 4.41×10^{-2}. For ease of comparison the domain Ω chosen using parameters (7.6) was used for both meshes.

7.6. Oregonator. Our final example is the Oregonator [22] which is a three species chemical system motivated by the Belousov-Zhabotinsky reaction. It exhibits oscillatory behaviour and is traditionally given by the following system of ODEs:

$$\begin{aligned}\frac{\mathrm{d}x_1}{\mathrm{d}t} &= k_1x_2 - k_2x_1x_2 + k_3x_1 - 2k_4x_1^2,\\ \frac{\mathrm{d}x_2}{\mathrm{d}t} &= -k_1x_2 - k_2x_1x_2 + \frac{1}{2}k_cx_3,\\ \frac{\mathrm{d}x_3}{\mathrm{d}t} &= 2k_3x_1 - k_cx_3.\end{aligned} \tag{7.7}$$

We now construct a set of reactions whose behaviour is given by (7.7) in the thermodynamic limit:

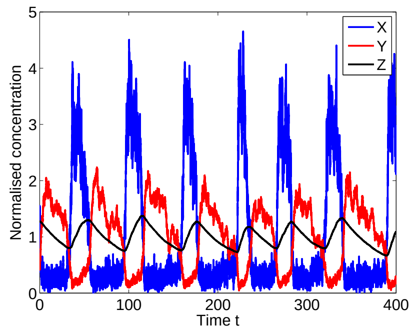
$$\begin{array}{ccc} X_2 \xrightarrow{k_1} X_1 & X_1 + X_2 \xrightarrow{k_2} \emptyset & X_1 \xrightarrow{k_3} 2X_1 + 2X_3 \\ 2X_1 \xrightarrow{k_4} \emptyset & X_3 \xrightarrow{k_5=k_c/2} \emptyset & X_3 \xrightarrow{k_6=k_c/2} X_2. \end{array} \tag{7.8}$$

Note that the reaction with parameter k_c has been split into two reactions R_5 and R_6 in order to get the factor of half in the equation for the rate of change of x_2, with $k_5 = k_6 = k_c/2$. We consider this system with the following set of dimensionless parameters:

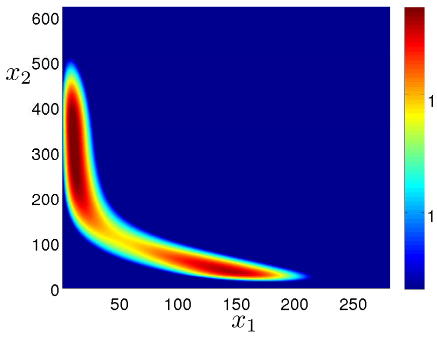
$$k_1 = 0.3, \quad k_2 = 4000, \quad k_3 = 5, \quad k_4 = 1200, \quad k_c = 0.02. \tag{7.9}$$

In this parameter regime the stochastic description of these reactions exhibits oscillatory behaviour. Figure 7.3(a) shows normalised trajectories of the Oregonator (7.8) with parameters given by (7.9), simulated using the Gillespie SSA. This figure demonstrates the oscillatory behaviour of the Oregonator in this parameter regime.
One thing of note should be mentioned at this point, regarding the ergodicity of this system. The zero state, at the origin, is an absorbing state for this system, and so trivially the invariant distribution for this system is a Dirac measure on this state. However, we are interested in the behaviour of this system conditioned on non-extinction of the species. Therefore we ensure that our domain of solution does not include this state, and thus the transient states involved in the oscillation behaviour now form the regions with non-zero invariant density.

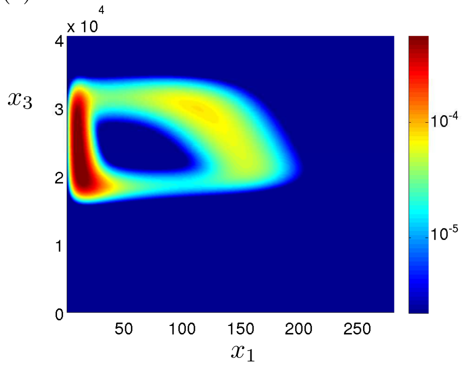
(a)



(b)



(c)



(d)

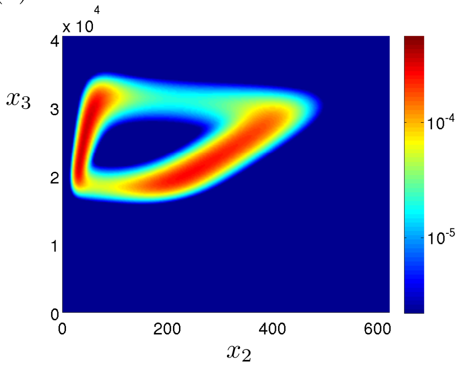


FIG. 7.3. (a) *Trajectories of the Oregonator* (7.8) *with parameters given by* (7.9)*, simulated using the Gillespie SSA. Plots show normalised trajectories, with actual molecule numbers divided by a time average* (47.7*,* 211.5 *and* 2.4×10^4 *resp.).* (b)–(d) *Approximation of the log of the marginal invariant distribution (conditioned on non-extinction of species X_1) of* (7.8) *with system parameters* (7.9) *in the* (b) *x_1-x_2 plane,* (c) *x_1-x_3 plane,* (d) *x_2-x_3 plane, by the saFEM with algorithmic parameters given by* (7.10)*.*

In the figures that follow, these algorithmic parameters were used to approximate the steady state distribution (conditioned on non-extinction of species X_1):

$$S = 1, \qquad T = 10^6, \qquad Q = 10^{-2}, \qquad B = 10^3, \qquad H = 8 \tag{7.10}$$

Since this system has a single limit cycle and is ergodic, step [**1**] of the algorithm (in Table 4.1) can be replaced by running the Gillespie SSA for a period of algorithm time until we are reasonably sure that the chain has entered probabilistic equilibrium, and using the endpoint of this simulation as the starting point for the SSA in step [**2**]. Since the system has a single stable orbit, we pick $S = 1$. Since a priori we do not know a specific point on the most likely orbit, the initial condition $(x_1, x_2, x_3) = (100, 100, 100)$ was chosen.

Since we wish to condition on non-extinction of all of the species, we must ensure that X_1 never drops below 1. We can do this by slightly altering the domain Ω of solution of the FPE. We do this by replacing the first line of (4.6) by $\Omega = \mathcal{A}_1^* \times \mathcal{A}_2 \times \mathcal{A}_3$, where

$$\mathcal{A}_1^* = \left(\max\{1, x_1^{\min} - \beta_2\, x_1^{\text{range}}\}, x_1^{\max} + \beta_2\, x_1^{\text{range}}\right).$$

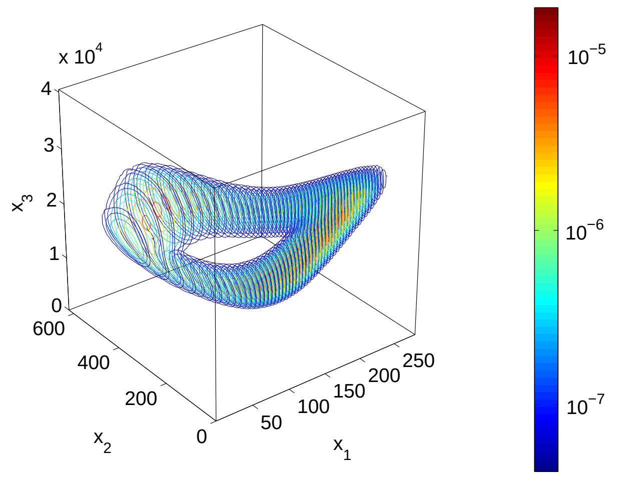


FIG. 7.4. *Approximation of the invariant distribution (conditioned on non-extinction) of* (7.8) *with system parameters* (7.9)*. Contours plots shown in several slices.*

The longer the simulation in the step **[2]** of the saFEM, the more sure we can be of the regions which have non-negligible probability density, and therefore the larger we can make β_2 and β_1. We choose $\beta_1 = 0.01$ and $\beta_2 = 0.1$.
Using these parameters, a mesh was created by the saFEM with 8.65×10^5 vertices, out of a possible 1.70×10^7 with up to 8 splits of each cuboid, with 1.17×10^5 hanging nodes. Therefore there were a total of 7.48×10^5 degrees of freedom, a mere 4.4% of the degrees of freedom that would have been required to fill the domain with cuboids of the finest refinement level. The mesh contained 4.79×10^6 individual tetrahedral elements.
Figure 7.4 shows several slices of the approximated invariant probability density conditioned on non-extinction of species X_1, represented by contour plots. Since visualisation of a three-dimensional function on a two-dimensional page is problematic, we present also in Figure 7.3(b)–(d) the marginal densities of the approximation of the invariant density of the Oregonator system with parameters (7.9) in 3 different planes.
To verify that the method is accurately representing the invariant distribution, we can test the approximation as suggested in Section 5.3. The approximation was compared with one with the parameters given as follows:

$$S = 1,\ T = 5 \times 10^5,\ Q = 5 \times 10^{-3},\ B = 5 \times 10^2,\ H = 7,\ \beta_1 = 5 \times 10^{-3}.$$

We omit β_2 here as the same domain as was chosen for the original parameter set was used for ease of comparison of the two densities. The L^2 difference between the two solutions was 2.89×10^{-11}, with the original solution having an L^2 norm of 1.74×10^{-8}, giving a relative L^2 error of 1.66×10^{-3}, verifying that our solution is well converged.

8. Discussion.
In this paper we have presented a method for solution of the steady-state Fokker-Planck equation for chemical systems. The method is based on the use of stochastic trajectories to estimate the region in which we must refine our

finite element mesh the most. This method is only valid for systems which do not have large regions of the domain with non-negligible invariant density that can be accurately approximated by a linear function. Our assumption that regions which have high enough probability density require high levels of refinement would fail in this case, and the mesh produced would be inefficient.

The use of the Fokker-Planck equation as a means to model chemical reactions of the mesoscale has been present in the literature for some time [24, 36]. As computational resources have improved, the solving of such equations in more than one dimension has become tractable. It has been shown that using appropriate methods (such as finite volume methods), the solution of the Fokker-Planck equation is far more efficient than using an SSA for the computation of both time-dependent and steady state solutions, and with a high degree of accuracy [21, 36].

The Fokker-Planck equation does suffer badly from the curse of dimensionality, and many different methods exist which attempt to find approximations of the solution of these equations efficiently in more than one dimension. Some involve dimension reduction of the equations themselves, through simplifying assumptions, and through coupling with macroscopic reaction rate equations for some of the species [33]. Sparse grid methods have also been used in an attempt to find the invariant density of the chemical master equation [27]. Sparse grids attempt to overcome the curse of dimensionality by reducing greatly the number of degrees of freedom considered. This CME solution can also be coupled to Fokker-Planck equations for the more abundant species using a hybrid method [27]. Other methods have been used to approximate the solution of the chemical master equation, including adaptive wavelet compression [29] and finite state projection [16, 34].

There are several other ways that stochastic trajectories could be calculated in the saFEM, other than the standard SSA as detailed in Table 2.1. The τ-leap method [26] approximates the trajectory by modelling several reactions in each iteration of the algorithm, up to a fixed time step. This can accelerate the simulation of the trajectory, but could incur some errors, including the possibility that the trajectory may become negative in one or more of the chemical species. Likewise, one could use numerical approximations of the chemical Langevin equation [25], a stochastic differential equation (SDE) with corresponding Fokker-Planck equation given by (2.2), to create trajectories. As with the τ-leap method, the trajectories from this SDE can become negative, and so with both of these methods proper treatment of boundary conditions is key to accurately approximating the region of the positive quadrant that contains the majority of the probability density.

More information could also be extracted from the stochastic trajectories. A rudimentary approximation of the density could be made using the samples taken from the simulations, and used as a preconditioner for the eigensolver. As the eigensolver used is iterative, even a rough guess of the form of the solution could help to reduce computation times. It has been previously shown that combining stochastic simulations and solutions of differential equations can be used for preconditioning of computations [35].

Acknowledgements: The research leading to these results has received funding from the European Research Council under the *European Community*'s Seventh Framework Programme (*FP7/2007-2013*)/ ERC *grant agreement* No. 239870. This publication was based on work supported in part by Award No KUK-C1-013-04, made by King Abdullah University of Science and Technology (KAUST). The work of Simon Cotter was also partially supported by a Junior Research Fellowship of St Cross College, University of Oxford. Tomáš Vejchodský would like to acknowledge the support by the Grant Agency of the Academy of Sciences (project. IAA100760702) and the Academy of Sciences of the Czech Republic (institutional research plan no. AV0Z0190503). Radek Erban would also like to thank Somerville College, University of Oxford, for a Fulford Junior Research Fellowship; Brasenose College, University of Oxford, for a Nicholas Kurti Junior Fellowship; the Royal Society for a University Research Fellowship; and the Leverhulme Trust for a Philip Leverhulme Prize. This prize money was used to support a research visit of Tomáš Vejchodský in Oxford.

REFERENCES

[1] P. Amestoy, I. Duff, J. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[2] P. Amestoy, A. Guermouche, J. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136–156, 2006.

[3] A. Arkin, J. Ross, and H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage l-infected Escherichia coli cells. *Genetics*, 149:1633–1648, 1998.

[4] I. Babuška and W. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis*, 15(4):736–754, 1978.

[5] S. Balay, J. Brown, K. Buschelman, W. Gropp, D. Kaushik, M. Knepley, L. McInnes, B. Smith, and H. Zhang. PETSc Web page, 2011. http://www.mcs.anl.gov/petsc.

[6] S. Balay, W. Gropp, L. McInnes, and B. Smith. Efficient management of parallelism in object oriented numerical software libraries. In *Modern Software Tools in Scientific Computing*, pages 163–202, 1997.

[7] L. Beilina, S. Korotov, and M. Křížek. Nonobtuse tetrahedral partitions that refine locally towards fichera-like corners. *Applications of Mathematics*, 50:569–581, 2005.

[8] J. Bey. Tetrahedral grid refinement. *Computing*, 55:355–378, 1995.

[9] Y. Cao, D. Gillespie, and L. Petzold. Multiscale stochastic simulation algorithm with stochastic partial equilibrium assumption for chemically reacting systems. *Journal of Computational Physics*, 206:395–411, 2005.

[10] Y. Cao, D. Gillespie, and L. Petzold. The slow-scale stochastic simulation algorithm. *Journal of Chemical Physics*, 122(1):14116, 2005.

[11] P. Ciarlet. *The finite element method for elliptic problems.* North-Holland Publishing Co., Amsterdam, 1978. Studies in Mathematics and its Applications, Vol. 4.

[12] S. Cotter, T. Vejchodský, and R. Erban. Stochastic bifurcations of biochemical systems. in preparation, 2011.

[13] S. Cotter, K. Zygalakis, I. Kevrekidis, and R. Erban. A constrained approach to multiscale stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 135:094102, 2011.

[14] M. Cowles and B. Carlin. Markov chain monte carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91(434):883–904, 1996.

[15] E. Doedel, A. Champneys, T. Fairgrieve, Y. Kuznetsov, B. Sandstede, and X. Wang. AUTO 97: Continuation and bifurcation software for ordinary differential equations (with homcont), 1997.

[16] B. Drawert, M. Lawson, L. Petzold, and M. Khammash. The diffusive finite state projection algorithm for efficient simulation of the stochastic reaction-diffusion master equation. *Journal of Chemical Physics*, 132:074101, 2010.

[17] W. E, D. Liu, and E. Vanden-Eijnden. Nested stochastic simulation algorithm for chemical kinetic systems with disparate rates. *Journal of Chemical Physics*, 123:194107, 2005.

[18] R. Erban, S. J. Chapman, I. Kevrekidis, and T. Vejchodsky. Analysis of a stochastic chemical system close to a sniper bifurcation of its mean-field model. *SIAM Journal on Applied Mathematics*, 70(3):984–1016, 2009.

[19] R. Erban, S. J. Chapman, and P. Maini. A practical guide to stochastic simulations of reaction-diffusion processes. 35 pages, available as http://arxiv.org/abs/0704.1908, 2007.

[20] R. Erban, I. Kevrekidis, D. Adalsteinsson, and T. Elston. Gene regulatory networks: A coarse-grained, equation-free approach to multiscale computation. *Journal of Chemical Physics*, 124:084106, 2006.

[21] L. Ferm, P. Lötstedt, and P. Sjöberg. Conservative solution of the fokker-planck equation for

stochastic chemical reactions. *BIT Numerical Analysis*, 46:61–83, 2006.

[22] R. Field and R. Noyes. Oscillations in chemical systems. IV. limit cycle behavior in a model of a real chemical reaction. *Journal of Chemical Physics*, 60(5):1877–1884, 1974.

[23] D. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[24] D. Gillespie. Approximating the master equation by Fokker-Planck equations for single-variable chemical systems. *Journal of Chemical Physics*, 72(10):5363–5370, 1980.

[25] D. Gillespie. The chemical Langevin equation. *Journal of Chemical Physics*, 113(1):297–306, 2000.

[26] D. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[27] M. Hegland, A. Hellander, and P. Lötstedt. Sparse grids and hybrid methods for the chemical master equation. *BIT Numerical Mathematics*, 48(2):265–283, 2008.

[28] V. Hernandez, J. Roman, and V. Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software*, 31(3):351–362, 2005.

[29] T. Jahnke and T. Udrescu. Solving chemical master equations by adaptive wavelet compression. *Journal of Computational Physics*, 229(16):5724 – 5741, 2010.

[30] S. Kar, W. Baumann, M. Paul, and J. Tyson. Exploring the roles of noise in the eukaryotic cell cycle. *Proceedings of the National Academy of Sciences USA*, 106:6471–6476, 2009.

[31] S. Korotov and M. Kek. Acute type refinements of tetrahedral partitions of polyhedral domains. *SIAM Journal on Numerical Analysis*, 39(2):724–733, 2002.

[32] V. Kublanovskaya. On some algorithms for the solution of the complete eigenvalue problem. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 1(3):555–570, 1961.

[33] P. Lötstedt and L. Ferm. Dimensional reduction of the fokker-planck equation for stochastic chemical reactions. *Multiscale Modelling and Simulation*, 5:593–614, 2006.

[34] B. Munsky and M. Khammash. The finite state projection algorithm for the solution of the chemical master equation. *The Journal of Chemical Physics*, 124(4):044104, 2006.

[35] L. Qiao, R. Erban, C. Kelley, and I. Kevrekidis. Spatially distributed stochastic systems: Equation-free and equation-assisted preconditioned computation. *Journal of Chemical Physics*, 125:204108, 2006.

[36] P. Sjöberg, P. Lötstedt, and J. Elf. Fokker-planck approximation of the master equation in molecular biology. *Computing and Visualization in Science*, 12:37–50, 2009.

[37] P. Šolín, J. Červený, and I. Doležel. Arbitrary-level hanging nodes and automatic adaptivity in the *hp*-FEM. *Mathematics and Computers in Simulation*, 77(1):117–132, 2008.

[38] R. Verfürth. *A review of a posteriori error estimation and adaptive mesh-refinement techniques*. Chichester: John Wiley & Sons. Stuttgart: B. G. Teubner., 1996.

[39] J. Villar, H. Kueh, N. Barkai, and S. Leibler. Mechanisms of noise-resistance in genetic oscillators. *Proceedings of the National Academy of Sciences USA*, 99(9):5988–5992, 2002.

[40] C. Zenger. Sparse grids. In W. Hackbusch, editor, *Parallel algorithms for partial differential equations*, volume 31 of *Notes on Numerical Fluid Mechanics*, pages 241–251. Vieweg-Verlag, 1991.