J. Math. Study
doi: 10.4208/jms.v48n2.15.02

Vol. **48**, No. 2, pp. 106-124
June 2015

Stochastic Domain Decomposition for Time Dependent Adaptive Mesh Generation

Alexander Bihlo[1,2], Ronald D. Haynes[1,*], Emily J. Walsh[3]

[1] *Department of Mathematics and Statistics, Memorial University of Newfoundland, St. John's (NL), A1C 5S7, Canada.*
[2] *Department of Mathematics, University of British Columbia Vancouver (BC), V6T 1Z2, Canada.*
[3] *Department of Engineering Design and Mathematics, University of Western England, Bristol BS16 1QY, United Kingdom*

Received 1 April, 2015; Accepted (in revised version) 22 May, 2015

Abstract. The efficient generation of meshes is an important component in the numerical solution of problems in physics and engineering. Of interest are situations where global mesh quality and a tight coupling to the solution of the physical partial differential equation (PDE) is important. We consider parabolic PDE mesh generation and present a method for the construction of adaptive meshes in two spatial dimensions using stochastic domain decomposition that is suitable for an implementation in a multi– or many–core environment. Methods for mesh generation on periodic domains are also provided. The mesh generator is coupled to a time dependent physical PDE and the system is evolved using an alternating solution procedure. The method uses the stochastic representation of the exact solution of a parabolic linear mesh generator to find the location of an adaptive mesh along the (artificial) subdomain interfaces. The deterministic evaluation of the mesh over each subdomain can then be obtained completely independently using the probabilistically computed solutions as boundary conditions. A small scaling study is provided to demonstrate the parallel performance of this stochastic domain decomposition approach to mesh generation. We demonstrate the approach numerically and compare the mesh obtained with the corresponding single domain mesh using a representative mesh quality measure.

AMS subject classifications: 65N50, 65M50, 65L50, 65C05,65N55, 65M55

Key words: Mesh generation, Domain decomposition, Monte Carlo methods.

1 Introduction

The numerical solution of many partial differential equations (PDEs) benefits from the construction of an adaptive grid automatically tuned by the solution itself. The quasi–

*Corresponding author. *Email addresses:* abihlo@mun.ca (A. Bihlo), rhaynes@mun.ca (R. D. Haynes), emily3.walsh@uwe.ac.uk (E. J. Walsh)

 ©2015 Global-Science Press

Lagrangian (QL), *r*–refinement, approach used here keeps the number of mesh points and the mesh topology fixed, moving the mesh continuously in time using a moving mesh PDE (MMPDE). The solution of the mesh PDE gives a continuous mesh transformation between an underlying computational co-ordinate and the required physical co-ordinate. Both the mesh and the solution are obtained at each time step. The QL approach can be implemented in either an alternating or simultaneous manner. The simultaneous QL approach treats the MMPDE and physical PDE as one large coupled system. At each time the new mesh and new solution on that mesh are found concurrently. Hence the mesh reacts instantly to changes in the physical solution. This highly nonlinear coupling may destroy exploitable structure which exists in the discretization of the physical PDE alone. The alternating approach uses the current mesh and physical solution to update the mesh alone, this new mesh then facilitates the computation of the updated physical solution. This introduces a time lag in the mesh as the new mesh is based only on the current physical solution. Computationally, however, this decoupling reduces the size of the discrete problem. Furthermore, the solver becomes more modular; the mesh and physical solvers can be called in alternating fashion; each solver can be designed to take advantage of the structure inherent in each subsystem. The simultaneous approach is generally thought to be more difficult and expensive to solve and hence the alternating method (or a variant thereof) is typically used in two or more spatial dimensions. As we will see below, the alternating approach fits well with the stochastic domain decomposition approach we describe to parallelize our computations.

The general QL approach has shown great promise in recent years, solving problems in meteorology [7], relativistic magnetohydrodynamics [14], combustion and convection in a porous medium [9], groundwater flow and transport of nonaqueous phase liquids [16], Stefan problems [4], semiconductor devices [30], and viscoelastic flows [31], phase change problems [3], multiphase flows [26], and low speed viscous flow [18], to name just a few. A thorough overview of PDE based moving mesh methods may be found in [15].

Recently, motivated by the alternating solution method, one of the authors has studied the parallel solution of the nonlinear MMPDE alone using a Schwarz based domain decomposition approach. In [12], Haynes and Gander propose and analyze classical, optimal and optimized Schwarz methods in one spatial dimension at the continuous level. A numerical study of classical and optimized Schwarz domain decomposition for 2D nonlinear mesh generation has been presented in [13]. In [10], a *monolithic* domain decomposition method, simultaneously solving a linear mesh generator coupled to the physical PDE, was presented for a shape optimization problem. The authors used an overlapping domain decomposition approach to solve the coupled problem.

In this paper, we present an efficient, parallel strategy for the solution of the moving mesh PDE based on a stochastic domain decomposition method proposed by Acebrón *et al.* [1]. The motivation is two–fold. First, we wish to reduce (by parallelization) the potential burden of having to solve an additional (mesh) PDE. Second, it is often mesh

quality, not an extremely accurate solution of the mesh PDE, which is important. As we will see, the stochastic domain decomposition approach is a means to these ends.

A stochastic domain decomposition (SDD) method to find adaptive meshes for steady state elliptic problems was presented in [5]. The SDD approach, as originally formulated in [1], uses a probabilistic form of the point–wise solution for linear elliptic boundary value problems. The point–wise solution is evaluated only at the introduced subdomain interfaces. The approximation of the solution at each interface point is obtained independently using Monte–Carlo simulations and these evaluations are then used as Dirichlet boundary conditions for the (deterministic) subdomain solves which can be computed in parallel. The mesh PDEs are generally not solved to high accuracy. Mesh quality, allowing an accurate representation of the physical PDE, is what is generally required. The parallel algorithm and lower accuracy requirement makes the proposed SDD method computationally attractive. The lower accuracy requirement allows one to terminate the Monte–Carlo simulations well before convergence.

The existence of a stochastic representation of the exact solution of linear parabolic problems allowed us to extend the SDD approach to (linear) parabolic mesh generators in [5]. There we considered the time relaxed form of the Winslow–Crowley variable diffusion mesh generation method, first described in [29]. Only the solution of the mesh generator for a specified analytic mesh density function was considered.

Here we consider the generation of time dependent meshes where the mesh PDE is coupled to a physical PDE of interest. The coupling to the physical solution u, is provided by, for example, an arc-length type function $\rho = \sqrt{1+\alpha(u_x^2+u_y^2)}$. The parameter α is chosen to provide a balanced coupling with the physical PDE. Furthermore, due to a stochastic representation for the solution of PDEs subject to periodic boundary conditions, we give, for the first time, a method to generate meshes for periodic problems using a stochastic domain decomposition approach.

In Section 2 we describe the time dependent mesh generator used in this paper and how the coupling between the physical PDE and mesh PDE is handled for the global, single domain solution. Furthermore, we describe a mesh generator for a spatially periodic problem. Section 3 provides some background on the stochastic solution of linear time dependent, periodic and non-periodic PDEs and describes how this is used to generate a non-iterative domain decomposition algorithm. We precisely illustrate how to obtain approximations to the point–wise solution of the mesh PDEs using the stochastic approach and finally how to generate the meshes using the stochastic domain decomposition framework. In Section 4 we illustrate the algorithm for various examples including Burgers' equation in both the non-periodic and periodic situations and the shallow water equations on a periodic domain. A small scaling study is provided to show the potential of the approach. We conclude with some observations and items for further study in Section 5.

2 Mesh generation approach

As discussed extensively in the references above, in 1D the QL *r*–refinement approach generates a physical mesh $x \in [0,1]$, via a continuous mesh transformation $x(\xi)$, where $\xi \in [0,1]$ is an underlying computational variable. A guiding principle is the equidistribution principle of de Boor (in 1D) [8, 11, 28], which finds a mesh transformation $x(\xi)$ by enforcing

$$\int_0^{\xi} \rho(t,u,\tilde{x})\,d\tilde{x} = \xi \int_0^1 \rho(t,u,\tilde{x})\,d\tilde{x},$$

where the mesh density function ρ gives a measure of the level of difficulty or error in the physical solution u. The physical solution u may be given analytically or as the solution of a physical PDE. In differential form, the mesh transformation may be found as the solution of the quasilinear BVP

$$\frac{d}{d\xi}\left(\rho(x)\frac{dx}{d\xi}\right) = 0, \quad x(0) = a, x(1) = b. \tag{2.1}$$

Here we have suppressed the t and u dependency in ρ. This gives the physical co–ordinates $x_i = x(\xi_i)$ as a function of a (typically uniform) computational grid ξ_i. This BVP can be written as a linear BVP in terms of the inverse mesh transformation $\xi(x)$ as

$$\frac{d}{dx}\left(\frac{1}{\rho(x)}\frac{d\xi}{dx}\right) = 0, \quad \xi(a) = 0, \xi(b) = 1. \tag{2.2}$$

The linearity of the BVP for $\xi(x)$ makes it easier to solve (in some sense) and in higher dimensions has the additional benefit that it is easy to say concrete things about the well–posedness of the mesh transformation. As will be discussed below, the linearity of a mesh generator is also a prerequisite for the stochastic domain decomposition method. The obvious disadvantage is that the solution of the BVP for $\xi(x)$ does not give the mesh locations in the physical variables x. One alternative is to transform the physical PDE from the physical variables to the new computational co-oordinates. Alternatively, inverse linear interpolation could be used to find the x locations by projecting $\xi(x)$ onto a uniform ξ grid.

Indeed, in [5] we constructed our stochastic DD method for steady state mesh generation using the natural 2D extension of the linear BVP (2.2). In the time dependent case considered in this paper we will construct a stochastic DD method directly in the physical co-ordinates. The time stepping will provide a natural linearization as we will show below.

A natural way to derive the BVPs above (and which extends to higher dimensions) is as the Euler–Lagrange equations whose solutions minimize certain functionals of the required mesh transformations. For example minimizing the functional

$$I[x] = \frac{1}{2}\int_0^1 \left(\rho(x)\frac{dx}{d\xi}\right)^2 d\xi$$

leads to the quasi–linear BVP (2.1) above. A minimizer $\xi(x)$ of the functional

$$I[\xi] = \frac{1}{2}\int_a^b \frac{1}{\rho(x)}\left(\frac{d\xi}{dx}\right)^2 dx$$

satisfies the linear BVP (2.2) above.

For time dependent PDEs it is useful to use a formulation which involves the mesh speed, such a mesh equation is called a moving mesh PDE (MMPDE). If $\rho = \rho(t,x)$ the functional $I[\xi]$ becomes

$$I[\xi] = \frac{1}{2}\int_a^b \frac{1}{\rho(t,x)}\left(\frac{d\xi}{dx}\right)^2 dx.$$

As described in [15], the direction ξ which reduces $I[\xi]$ is given by the gradient flow equation

$$\frac{d\xi}{dt} = \frac{P}{\tau}\frac{d}{dx}\left(\frac{1}{\rho(t,x)}\frac{d\xi}{dx}\right),$$

where $\tau > 0$ is a user specified constant which determines the response of the mesh to changes in ρ (or u). Here P is a positive definite differential operator that we choose with some flexibility.

We can switch the roles of x and ξ in the gradient flow equation to get a mathematically equivalent moving mesh PDE in the physical variables x:

$$\frac{\partial x}{\partial t} = \frac{1}{\tau}\frac{\partial x}{\partial \xi}P\left(\rho\frac{\partial x}{\partial \xi}\right)^{-2}\left(\frac{\partial x}{\partial \xi}\right)^{-1}\frac{\partial}{\partial \xi}\left(\rho\frac{\partial x}{\partial \xi}\right).$$

Hence this resulting equation would retain the nice well–posedness properties.

If we choose $P = (\rho x_\xi)^2$, then we get

$$\frac{\partial x}{\partial t} = \frac{1}{\tau}\frac{\partial}{\partial \xi}\left(\rho\frac{\partial x}{\partial \xi}\right), \tag{2.3}$$

which is referred to as MMPDE5. Our focus here is the generation of time dependent meshes by using this nonlinear parabolic mesh generator.

The coupling to the physical PDE and physical solution u is provided by the mesh density function $\rho(x,u)$. In a typical deterministic implementation, the mesh and physical solution are updated by discretizing MMPDE5 and the physical PDE in time and either solving the resulting large nonlinear system of algebraic equations for both the mesh and physical solution simultaneously or proceeding in an alternating fashion. The alternating or MP approach [15] used in this paper freezes ρ in the time discretized mesh equation at the current u^n to compute the next mesh x^{n+1} and then integrates the physical PDE, using mesh x^{n+1}, to obtain u^{n+1}. The stochastic solution representation given below requires the PDE to be linear. This MP procedure effectively linearizes the MMPDE (2.3).

In two spatial dimensions we use the mesh generator

$$\begin{aligned} &x_t = \frac{\nabla_{\boldsymbol{\xi}}\rho}{\rho}\cdot\nabla_{\boldsymbol{\xi}} x + \nabla^2_{\boldsymbol{\xi}} x, \quad y_t = \frac{\nabla_{\boldsymbol{\xi}}\rho}{\rho}\cdot\nabla_{\boldsymbol{\xi}} y + \nabla^2_{\boldsymbol{\xi}} y, \\ &x(0,\boldsymbol{\xi}) = x_0 = L_x\xi, \quad y(0,\boldsymbol{\xi}) = y_0 = L_y\eta, \end{aligned} \tag{2.4}$$

where $\boldsymbol{\xi}=(\xi,\eta)$ and $\nabla_{\boldsymbol{\xi}}=(\partial/\partial\xi,\partial/\partial\eta)$, which we solve over the square computational domain $\Omega_{\rm c}=[0,1]\times[0,1]$. For the sake of simplicity we assume a rectangular domain $\Omega_{\rm p}=[0,L_x]\times[0,L_y]$, where $L_x>0$, $L_y>0$. At the actual boundaries of the computational domain $\Omega_{\rm c}$, we employ the fixed boundary conditions $x(t,0,\eta)=0$, $x(t,1,\eta)=L_x$, $y(t,\xi,0)=0$ and $y(t,\xi,1)=L_y$. The remaining boundary conditions, $x(t,\xi,0)$, $x(t,\xi,1)$, $y(t,0,\eta)$ and $y(t,1,\eta)$ are found by solving the respective one-dimensional forms of the mesh generator (2.4) along these boundaries.

Note that the mesh generator (2.4) is obtained by *first* dividing the two dimensional version of (2.1) by ρ and *then* relaxing the equation in time. We have found in practice that this form of a relaxed mesh generator gives better meshes than by relaxing the original version (2.1). In the discrete formulation, these two possible time relaxed mesh generators are related by a scaled time step.

We are also interested in mesh generation on periodic domains. It has been pointed out in [27], that in order to use the mesh generator (2.4) on a rectangular periodic domain of physical dimensions $\Omega_{\rm p}=0,L_x[\times[0,L_y[$,

$$x(t,1,\eta)=x(t,0,\eta)+L_x, \quad y(t,\xi,1)=y(t,\xi,0)+L_y,$$

one should express the mesh and physical PDEs in terms of the displacements $\mathbf{X}=\mathbf{x}-L\boldsymbol{\xi}$, where $L=\mathrm{diag}(L_x,L_y)$, $\mathbf{x}=(x,y)$ and $\mathbf{X}=(X,Y)$, which are directly periodic on $\Omega_{\rm c}$. In this new set of variables, the mesh generator (2.4) becomes

$$X_t = \frac{\nabla_{\boldsymbol{\xi}}\rho}{\rho}\cdot\nabla_{\boldsymbol{\xi}} X + \nabla^2_{\boldsymbol{\xi}} X + \frac{\rho_\xi}{\rho}(1+L_x), \quad Y_t = \frac{\nabla_{\boldsymbol{\xi}}\rho}{\rho}\cdot\nabla_{\boldsymbol{\xi}} Y + \nabla^2_{\boldsymbol{\xi}} Y + \frac{\rho_\eta}{\rho}(1+L_y).$$

An alternative to the procedure proposed in [27] works directly in the original variables x and y and properly extends the computational domain using the periodicity of the grid. This allows one to evaluate the derivatives of $\mathbf{x}$ on the boundaries without having to take into account the actual size of the physical domain $\Omega_{\rm p}$. As this approach allows one to work with the physical coordinates $\mathbf{x}$ directly, we use this second method in this paper.

3 Stochastic domain decomposition and mesh generation

In this section we describe stochastic domain decomposition methods for mesh generation on periodic and non-periodic domains and describe an implementation which couples the mesh generator to the solution of a physical PDE.

3.1 Stochastic domain decomposition for linear PDEs

The main idea of stochastic domain decomposition as proposed in [1] (see also [2]) rests on the stochastic representation of the exact solution to *linear elliptic* (resp. *parabolic*) *boundary value problems*. This now classical connection between stochastic analysis and boundary value problems was first uncovered by Kakutani [19,20] for the Dirichlet problem using Brownian motion. The monographs [21,24] provide a more recent exposition.

Numerically evaluating this stochastic representation of the exact solution of a linear boundary value problem using Monte–Carlo methods enables one to compute the point-wise numerical solution to the underlying PDE. From the practical point of view, this is fundamentally different to solving a PDE using, say, finite differences, which requires the computation of the numerical solution over the entire domain even if it is only needed in a single point.

Notoriously, Monte–Carlo methods converge slowly, with convergence rates proportional to $N^{-1/2}$ where N is the number of Monte–Carlo simulations, if pseudo-random numbers are used [25]. Hence they play a role mostly for higher dimensional problems, where they can be shown to outperform deterministic methods.

An alternative is to use them in the context of domain decomposition. Namely, splitting the entire domain into non-overlapping subdomains, the stochastic solution can be used to compute the point-wise interface solutions between the subdomains. Once these interface solutions are determined with sufficient accuracy, they act as Dirichlet boundary values for the individual subdomains. The PDE solution over each subdomain is computed deterministically using an appropriate discretization of the underlying PDE. The main advantage of this method is that iteration, as is required in classical domain decomposition methods (such as Schwarz methods), can be completely avoided. Also, the solutions over each subdomain can be obtained in parallel and thus the method is suitable for massively parallel computing architectures.

In [5, 6] we have shown that the stochastic domain decomposition technique is an effective way for the parallel generation of adaptive meshes. A main motivator for the approach is that it is, in general, not necessary to compute the meshes with high accuracy. What is important is to obtain meshes with high mesh quality. It was shown in [5, 6] that even meshes that are not accurate solutions to the mesh PDEs can have good mesh quality. This characteristic of mesh generation enables an increase in the efficiency of the stochastic domain decomposition method.

3.2 Stochastic analysis for Dirichlet boundary value problems

In this section we give the specifics of the stochastic analysis required to generate the solution of linear parabolic boundary value problems. Specifically, we consider system (2.4) where the unknowns $x = x(t,\xi,\eta)$ and $y = y(t,\xi,\eta)$ are required on the time-space domain given by $[0,T]\times\Omega_c$, with T being some finite final time. System (2.4) is supplemented with the boundary conditions $x(t,\xi,\eta)|_{\partial\Omega_c} = f(t,\xi,\eta)$ and $y(t,\xi,\eta)|_{\partial\Omega_c} = g(t,\xi,\eta)$,

for given continuous functions f and g. The initial values are $x(0,\xi,\eta)=x_0(\xi,\eta)$ and $y(0,\xi,\eta)=y_0(\xi,\eta)$.

It is important to stress here that (2.4) is nonlinear if the mesh density function is a function on the physical domain, that is $\rho=\rho(t,x,y)$. However, as was indicated in Section 2, in the practical implementation it is possible to freeze ρ at time layer t^n when computing the mesh at time t^{n+1}. This effectively boils down to a linearization of the mesh generator (2.4). It is thus appropriate to assume that ρ in (2.4) is not a function of x and y but of some auxiliary variables $\tilde{x}$ and $\tilde{y}$ (and time), i.e. we assume that $\rho=\rho(t,\tilde{x},\tilde{y})$. Then this mesh generator becomes linear and allows for a stochastic representation of its exact solution [23], given by

$$\begin{aligned} x(t,\xi,\eta)&=\mathrm{E}\left[x_0(\mathbf{\Phi}(t))\mathbf{1}_{[\tau_{\partial\Omega_c}>t]}\right]+\mathrm{E}\left[f(t-\tau_{\partial\Omega_c},\mathbf{\Phi}(\tau_{\partial\Omega_c}))\mathbf{1}_{[\tau_{\partial\Omega_c}<t]}\right],\\ y(t,\xi,\eta)&=\mathrm{E}\left[y_0(\mathbf{\Phi}(t))\mathbf{1}_{[\tau_{\partial\Omega_c}>t]}\right]+\mathrm{E}\left[g(t-\tau_{\partial\Omega_c},\mathbf{\Phi}(\tau_{\partial\Omega_c}))\mathbf{1}_{[\tau_{\partial\Omega_c}<t]}\right], \end{aligned} \tag{3.1a}$$

where the stochastic process $\mathbf{\Phi}(t)$ satisfies the stochastic differential equation (SDE)

$$\mathrm{d}\mathbf{\Phi}(t)=\frac{1}{\rho}\nabla_\xi\rho\,\mathrm{d}t+\sqrt{2}\mathrm{d}\mathbf{W}(t). \tag{3.1b}$$

In (3.1), $\mathrm{E}[\cdot]$ is the expected value, $\tau_{\partial\Omega_c}$ is the first exit time of the stochastic process $\mathbf{\Phi}(t)$ starting at (ξ,η), $\mathbf{W}$ is two-dimensional Brownian motion and $\mathbf{1}$ is the indicator function. See [21, 23, 24] for a more extensive discussion of this subject. We note that the stochastic solution (3.1a) has contributions from both the specific initial condition and boundary values of the mesh generator.

3.3 Stochastic analysis for periodic problems

Freezing ρ the mesh generator (2.4) is in the form of a system of linear, second order, parabolic PDEs. In Section 3.2 we wrote the stochastic point–wise solution assuming a Dirichlet boundary value problem. On periodic domains, the celebrated Kac–Feynman formula can be used to obtain the stochastic representation of the mesh generator (2.4), see e.g. [23]. In this case, only initial values $x(0,\xi,\eta)=x_0(\xi,\eta)$ and $y(0,\xi,\eta)=y_0(\xi,\eta)$ are given. The solution to (2.4) can then be written as

$$x(t,\xi,\eta)=\mathrm{E}[x_0(\mathbf{\Phi})],\quad y(t,\xi,\eta)=\mathrm{E}[y_0(\mathbf{\Phi})], \tag{3.2}$$

where $\mathbf{\Phi}$ satisfies the same stochastic differential equation as given in (3.1b).

3.4 Stochastic domain decomposition on periodic and non-periodic domains

While the stochastic solutions (3.1) and (3.2) can in principle be used to obtain the solution to the parabolic mesh generator at any time $t>0$, our mesh generation problem is slightly

more complicated. The mesh density function ρ is linked to the solution of the physical PDE system, which changes over time. For this reason, in practice we use (3.1) and (3.2) only to advance the mesh over a single time step, Δt, from t^n to t^{n+1}.

We discretize the SDE (3.1b) using the Euler–Maruyama method,

$$\mathbf{\Phi}^{k+1} = \mathbf{\Phi}^k + \frac{\nabla_\xi \rho}{\rho}\bigg|_{(t^n,\mathbf{\Phi}^k)} \Delta t_s + \sqrt{2\Delta t_s}\mathbf{W}(0,1), \tag{3.3}$$

with constant time step $\Delta t_s = \Delta t/M$, $k=0,\dots,M-1$, where $\mathbf{W}$ is a two-dimensional vector of Gaussian distributed random numbers with zero mean and variance one. In other words, one time step of size Δt is split into M sub-time steps. This splitting is necessary so that the use of an excessively small time step Δt for the solution of the physical differential equation can be avoided. The mesh density function ρ remains fixed at time step t^n. The derivatives $\nabla_\xi \rho$ are approximated with finite differences.

In practice, the starting points of the stochastic process at time t^n, $\mathbf{\Phi}^0$, coincide with the grid points where the stochastic solution is required. Once the new values $\mathbf{\Phi}^{k+1}$ at time t^{k+1} are computed, both ρ and $\nabla_\xi \rho$ have to be approximated at $\mathbf{\Phi}^{k+1}$. For this, bi-linear interpolation is used. The procedure is repeated until $\mathbf{\Phi}^M$ is computed, which coincides with the value of the stochastic process at time t^{n+1}. We then evaluate the initial values of x_0 and y_0 given at time t^n at the new location $\mathbf{\Phi}^M$ using bi-cubic interpolation. This gives the values $\tilde{x}_0(\mathbf{\Phi}^M)$ and $\tilde{y}_0(\mathbf{\Phi}^M)$, which approximate the actual values $x_0(\mathbf{\Phi}^{n+1})$ and $y_0(\mathbf{\Phi}^{n+1})$ that are needed in both the solutions (3.1a) (for the first term) and (3.2).

Solving Dirichlet boundary value problems stochastically, a boundary test has to be applied to determine whether the process $\mathbf{\Phi}^{k+1}$ has left the domain within the sub-time step $[t^k, t^{k+1}]$. A linear Brownian bridge is used as interpolating process as discussed in [17]. If the process left the domain before time t^{n+1}, the integration can be stopped and the second term in (3.1a) can be evaluated. No such boundary test is needed for periodic domains.

In order to estimate the expected value using the arithmetic mean, the procedure is repeated N times.

3.5 Local subdomain problem

In the domain decomposition solution to the problem (2.4), we only use the stochastic solutions (3.1) and (3.2) to generate the subdomain interface solutions for the Dirichlet and periodic problems. Once these solutions are computed, the solution to the mesh generation problem on the individual subdomains becomes a Dirichlet boundary value problem. In order to solve this problem, the original parabolic mesh generator (2.4) has to be solved with the Dirichlet boundary conditions

$$\mathbf{x}|_{\partial\Omega_c^i} = f^i,$$

where $\partial\Omega_c^i$ denotes the boundary of the i–th subdomain of the computational domain and f^i are the values obtained either from the stochastic representation (3.1) or the boundary conditions on the global mesh where $\partial\Omega_c^i \bigcap \partial\Omega_c \neq \emptyset$.

For the local subdomain solver we discretize (2.4) using centered finite differences for the spatial derivatives and an implicit Euler method for the time stepping.

3.6 Domain decomposition solution

The domain decomposition strategy relies on combining the stochastic evaluation along the interfaces with the deterministic subdomain solver. At each time step, the stochastic solution procedure provides the Dirichlet boundary conditions for the deterministic single-domain solver. This allows the computation of the new mesh at the time step t^{n+1}. The solution values can either be evaluated stochastically at each (ξ_i,η_j) along the artificial interfaces or a sample can be evaluated with the rest obtained by interpolation. An optimal placement strategy for the interpolation nodes was presented in [5].

Once the new mesh is computed, the mesh density function $\rho(t^{n+1},x,y)$ is evaluated on the new mesh and the solution procedure is repeated with the new mesh density function.

3.7 Mesh quality

As mentioned previously, there is a crucial difference between the SDD method as proposed in [1] for the computation of numerical solutions of general linear elliptic boundary value problems and for the mesh generation case. The latter does not necessarily require a very accurate solution of the mesh equation; here, mesh quality is more important.

There are several ways of assessing the quality of an adaptive mesh and different mesh quality measures based on properties such as equidistribution and alignment have been derived. See [15] for an extensive discussion of mesh quality measures. In [5] it was proposed to use the geometric mesh quality measure for the assessment of the quality of meshes generated using the SDD method. The geometric mesh quality measure is defined by

$$Q(K)=\frac{1}{2}\frac{\mathrm{tr}(J^{\mathrm{T}}J)}{\sqrt{\det(J^{\mathrm{T}}J)}}, \tag{3.4}$$

where J is the Jacobian of the transformation $x=x(\xi,\eta), y=y(\xi,\eta)$ and K is a mesh element in Ω_c. The quantity $Q(K)$ measures how far the mesh cell is from being equilateral, that is we have $Q(K)\geq 1$ with $Q(K)=1$ precisely for an equilateral cell.

In [5] we argued that using this measure is appropriate as meshes computed using stochastic methods show several kinks when they are far away from convergence. Thus, the values of $Q(K)$ are in general larger for such meshes compared to grids that are computed using deterministic methods. Of course, the absolute value of $Q(K)$ depends on

the mesh density function used and the underlying problem for which an adaptive grid is computed. We are therefore not interested in the absolute value of $Q(K)$ but only in the ratio between $Q^{\mathrm{SD}}(K)$, the geometric mesh quality measure of the reference mesh obtained on a single domain, and $Q^{SDD}(K)$, the geometric mesh quality measure of the grid obtained using the SDD method. If this ratio $R(K)=Q^{\mathrm{SD}}/Q^{\mathrm{SDD}}$ is close to one, the mesh obtained from the SDD method is a good approximation to the single domain mesh.

The quantity R is a function of each individual mesh cell. In the next section, for the sake of convenience, we only list the maximum and mean values, $R_{\max}$ and R_{mean} over all mesh cells.

4 Numerical results

In this section, numerical experiments in 2D are presented to demonstrate the effectiveness of the domain decomposition algorithm and its suitability for problems with both Dirichlet and periodic boundary conditions.

4.1 Burgers' equation with Dirichlet boundary conditions

The first test problem considered is the scalar form of the two-dimensional Burgers' equation

$$u_t+(u^2/2)_x+(u^2/2)_y+\nu(u_{xx}+u_{yy})=0, \tag{4.1}$$

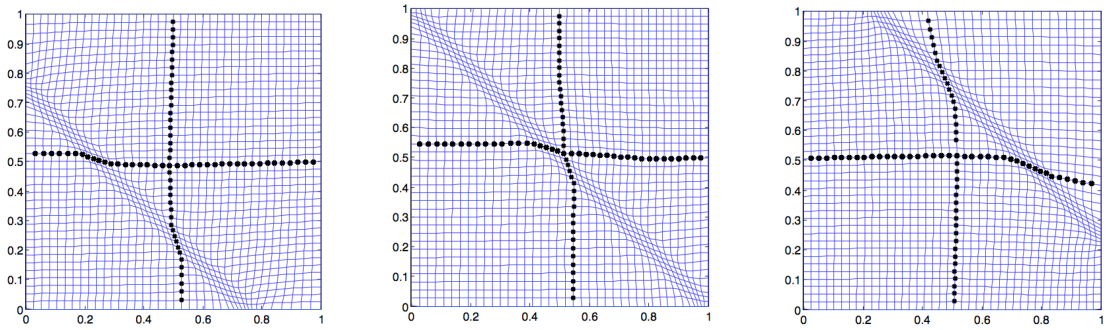
$[x,y]\in\Omega_p=[0,1]\times[0,1]$. The initial and Dirichlet boundary conditions are chosen such that the exact solution is

$$u=\left(1+\exp\left(\frac{x+y-t_0}{2\nu}\right)\right)^{-1},$$

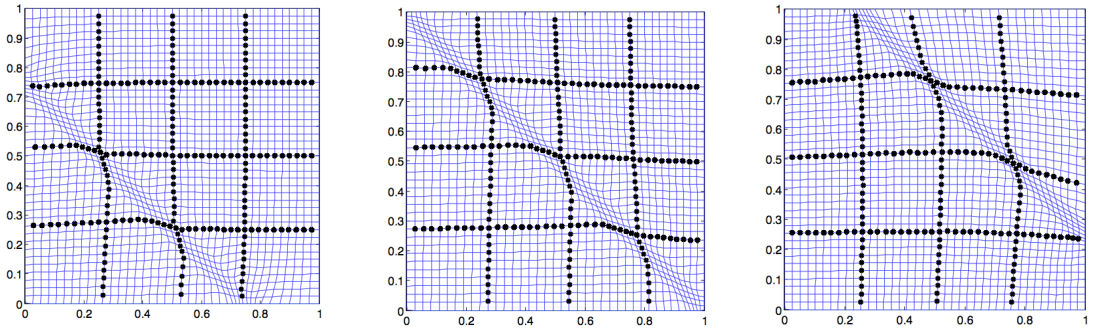
and we consider the case of a moderately small diffusion coefficient $\nu=0.005$. In this problem the smaller ν, the more convection dominates, and large gradients develop and move to the boundaries for $t>0$, thus requiring a higher concentration of mesh points to resolve the oblique shock that propagates with time. This is a common test problem in the moving mesh literature [22,32]. The coordinate transformation $(x,y)=(x(\xi,t),y(\xi,t))$ is used to rewrite the 2D Burgers' equation in QL form in computational coordinates, and this is discretized in the computational domain using centered differences in space and a trapezoidal rule for the time integration. This is coupled to the mesh generator (2.4). Both (2.4) and (4.1) are then solved alternately in time (the MP procedure) for each subdomain using an arc-length mesh density function

$$\rho=\sqrt{1+\alpha(u_x^2+u_y^2)},$$

with $\alpha=10/\|u_x^2+u_y^2\|_\infty$. The Dirichlet boundary conditions (at the subdomain interfaces) are found by evaluating (3.1a) using (3.3). At the physical boundary a 1D version of (2.4)



(a) t=0.75 (b) t=1.00 (c) t=1.25



(d) t=0.75 (e) t=1.00 (f) t=1.25

Figure 1: Evolution of mesh for Burgers' equation using the stochastic DD method with 2×2 (top) and 4×4 (bottom) subdomains.

is solved. We integrate the finite difference discretization of (4.1) using 41×41 grid points on the physical domain $\Omega_c=[0,1]\times[0,1]$ with a time step $\Delta t=0.001$. The initial time is $t_0=0.25$. We use $N=10000$ Monte–Carlo simulations with a time step $\Delta t_s=\Delta t/10$ for the solution of the stochastic differential equation (3.3). In Fig. 1 the evolution of the mesh is shown when the physical domain is split into 2×2 and 4×4 subdomains.

We report the mesh qualities and the l_∞-norm comparing the exact solution to the numerical solution at times $t_f=0.75$, $t_f=1$ and $t_f=1.25$ in Table 1. The geometric mesh quality measure is computed for the single domain solution and compared to the DD solution obtained by splitting the physical domain into 2×2, 3×3 and 4×4 subdomains.

Table 1 shows that all the l_∞-errors on meshes generated using SDD are approximately equal to the errors found on the associated single domain meshes. Also, the ratios of the geometric mesh quality measures of the single domain and DD solutions are close to one for all times, indicating that the DD solutions are a good approximation to the single domain solution.

Table 1: Mesh quality and l_∞-errors for Burgers' equation with Dirichlet boundary conditions.

t_f	l_∞^{SD}	$l_\infty^{2\times2}$	$R_{\mathrm{max}}^{2\times2}$	$R_{\mathrm{mean}}^{2\times2}$	$l_\infty^{3\times3}$	$R_{\mathrm{max}}^{3\times3}$	$R_{\mathrm{mean}}^{3\times3}$	$l_\infty^{4\times4}$	$R_{\mathrm{max}}^{4\times4}$	$R_{\mathrm{mean}}^{4\times4}$
0.75	0.027	0.023	0.99	0.99	0.031	0.95	0.98	0.026	0.95	0.98
1	0.031	0.033	0.99	0.99	0.320	0.97	0.99	0.034	0.97	0.99
1.25	0.030	0.031	1	1	0.35	0.92	1	0.24	1	1

The scaling properties of the SDD algorithm are reported in Table 2. For this study we solved Burgers' equation with Dirichlet boundary conditions on a varying number of subdomains, including 2×2, 4×4, 8×8 and 16×16 subdomains. Three series of experiments with a total of $N\times N=81\times81$, $N\times N=97\times97$ and $N\times N=113\times113$ points covering the physical domain were carried out.

Table 2: Time in seconds required per time step to construct the mesh for Burgers' equation with Dirichlet boundary conditions.

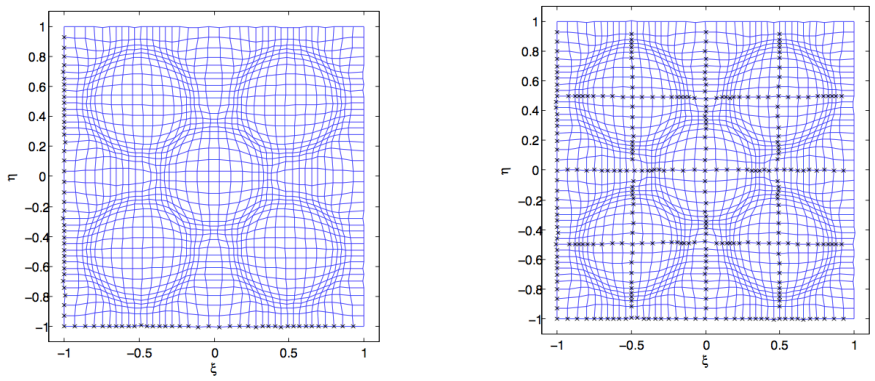
$N_{\mathrm{total}}\times N_{\mathrm{total}}$	SD	2×2	4×4	8×8	16×16
81×81	0.22	0.45	0.33	0.24	0.12
97×97	0.34	0.56	0.40	0.27	0.16
113×113	0.84	0.68	0.51	0.35	0.17

It can be seen from Table 2 that increasing the number of processors (and thus the number of subdomains) indeed leads to a decreased computational time per time step required to construct the adaptive mesh. For a total of $N\times N=81\times81$ mesh points, the solution on 16×16 subdomains is computationally cheaper than the single domain solution. Increasing the number of total points to $N\times N=97\times97$ makes both the solution on 8×8 and 16×16 subdomains cheaper compared to the single domain solution. If $N\times N=113\times113$ total points are used, the domain decomposition solution is computationally faster than the single domain solution for all numbers of subdomains.

The results of Table 2 thus demonstrate the potential of the stochastic domain decomposition algorithm to be significantly more efficient in generating adaptive meshes for problems with a large number of total points provided that enough parallel compute cores are available.

4.2 Periodic mesh generation

In order to demonstrate the generation of meshes over periodic domains we first study the case of a prescribed mesh density function. In particular, we revisit the five ring problem in the form considered in [5]. That is, we consider an analytically specified



(a) single domain solution (b) solution with 4×4 subdomains

Figure 2: A comparison of the single domain mesh and the mesh obtained for 4×4 subdomains for the five ring problem.

velocity function of the form

$$u=\tanh\left[R\left(x^2+y^2-\frac{1}{8}\right)\right]+\tanh\left[R\left(\left(x-\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2-\frac{1}{8}\right)\right]$$
$$+\tanh\left[R\left(\left(x-\frac{1}{2}\right)^2+\left(y+\frac{1}{2}\right)^2-\frac{1}{8}\right)\right]+\tanh\left[R\left(\left(x+\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2-\frac{1}{8}\right)\right]$$
$$+\tanh\left[R\left(\left(x+\frac{1}{2}\right)^2+\left(y+\frac{1}{2}\right)^2-\frac{1}{8}\right)\right],$$

on the domain $\Omega_p=[-1,1[\times[-1,1[$, with $R=30$. A simple arc-length mesh density function

$$\rho=\sqrt{1+\alpha(u_x^2+u_y^2)},$$

with $\alpha=0.2$ is used. The physical domain is discretized using 41×41 grid points, the final integration time is $t=0.05$ with a time step $\Delta t=5\times10^{-4}$. Since this time step is quite small, we use $\Delta t_s=\Delta t$ for the solution of the discretized SDE (3.3) as well. Again, $N=10000$ Monte–Carlo simulations were used. In Fig. 2 the mesh generated on a single domain is compared to the mesh generated by splitting the physical domain into 4×4 subdomains.

On inspection these meshes appear to be of similar quality and the computed measures of mesh quality confirm this, with $R_{\max}=1$ and $R_{\text{mean}}=1$ for the 2×2, 3×3 and 4×4 configurations. The mesh quality is identical to the single domain reference case.

4.3 Burgers' equation on a periodic domain

We repeat here the integration of Burgers' equation (4.1) but now using a doubly periodic domain of size $\Omega_p = [0,2\pi[\times[0,2\pi[$. The initial condition is

$$u = u_0 + A\sin(x+y-2\pi),$$

where $u_0 = 0.75$ and $A = 0.5$. The diffusion coefficient is $\nu = 0.001$. Again, 41×41 grid points are used and the time step of the integration was $\Delta t = 0.005$, which coincides with the time step used in the SDE (3.3). Only $N = 1000$ Monte–Carlo simulations were used. The results of the integration at times $t = 0.85$ and $t = 1$ are collected in Table 3.

Table 3: Mesh qualities for Burgers' equation with periodic boundary conditions.

t_f	$R^{2\times2}_{\max}$	$R^{2\times2}_{\text{mean}}$	$R^{3\times3}_{\max}$	$R^{3\times3}_{\text{mean}}$	$R^{4\times4}_{\max}$	$R^{4\times4}_{\text{mean}}$
0.85	0.99	1	1	1	0.99	1
1	0.99	1	0.99	1	0.99	1

The results in Table 3 confirm that the domain decomposition solution leads to meshes with almost the same geometric mesh quality as the single domain reference solution.

4.4 Shallow water equations on a periodic domain

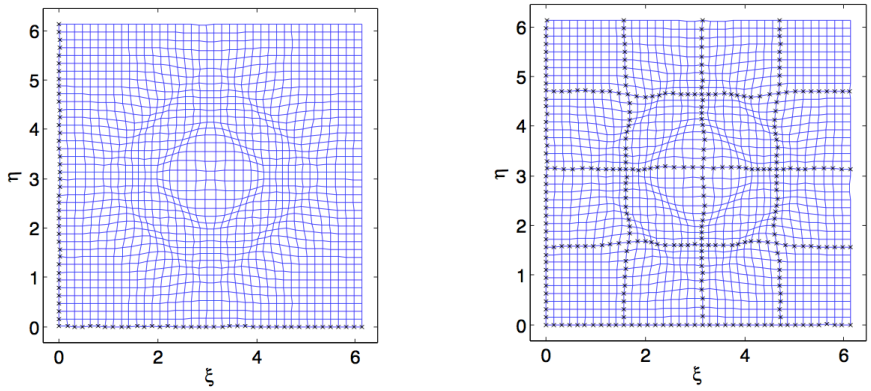
In this final example we solve the system of shallow-water equations in nondimensional form

$$\begin{aligned} &u_t + uu_x + vu_y + h_x = 0,\\ &v_t + uv_x + vv_y + h_y = 0,\\ &h_t + uh_x + vh_y + h(u_x + v_y) = 0, \end{aligned} \tag{4.2}$$

where (u,v) is the two-dimensional velocity field and h is the height of a water column over a constant reference level. For this problem, periodic boundary conditions are considered.

We discretize system (4.2) in computational coordinates using centered differences for the spatial derivatives and a trapezoidal rule for the time integration. The physical domain is of size $\Omega_p = [0,2\pi[\times[0,2\pi[$, which is discretized using 41×41 grid points. The time step of the integration was $\Delta t = 0.005$, which again coincides with the time step used for the solution of the SDE (3.3). We found that using $N = 1000$ Monte–Carlo simulations gives sufficiently good results.

The initial condition is a pile of water of height $h = 12.5$ in the center of the domain over a base level at height $h = 10$. This initial condition simulates the breaking of a dam, i.e. the evolution of the water level once the walls holding the pile of water have been removed.



(a) single domain solution (b) solution with 4×4 subdomains

Figure 3: A comparison of the single domain solution and that obtained for 4×4 subdomains for the shallow water equations at $t=0.15$.

Since the initial pile of water decays rapidly into gravity waves, we kept the integration times short and report our mesh quality results only at $t=0.05$ and $t=0.15$. A comparison of the mesh for the single domain solution and that computed for 4×4 subdomains can be seen in Fig. 3.

As we saw in the previous examples the meshes appear to be of similar quality and this is substantiated by the computed mesh quality measures which can be found in Table 4. The mesh qualities obtained from the domain decomposition solution are very close to that of the single domain reference solution.

Table 4: Mesh quality for the shallow-water equations with periodic boundary conditions.

t_f	$R^{2\times2}_{\max}$	$R^{2\times2}_{\text{mean}}$	$R^{3\times3}_{\max}$	$R^{3\times3}_{\text{mean}}$	$R^{4\times4}_{\max}$	$R^{4\times4}_{\text{mean}}$
0.05	0.99	1	0.97	1	0.99	1
0.15	0.99	1	0.99	1	0.99	1

5 Conclusion

Originally, the SDD method has been proposed in [1] for the parallel solution of linear elliptic boundary value problems. In [5, 6] we have extended the method to the parallel generation of adaptive meshes for prescribed mesh density functions. In this paper we have for the first time demonstrated the use of stochastic domain decomposition for the generation of adaptive moving meshes for time dependent PDE problems. Furthermore, we have provided the first stochastic domain decomposition method for the generation of meshes on periodic domains.

Our algorithm hinges on the alternating MP procedure for PDE based mesh generation in higher dimensions. A fully parallel algorithm would result if the physical PDE was also solved in parallel using domain decomposition or other approaches.

Indeed the linearization provided by the MP procedure has the undesirable affect of decoupling the physical solution from the mesh. This effect can be reduced using a M^kP procedure. This produces a mesh which more closely satisfies the equidistribution principle. In this case, we approximate x^{n+1} by a sequence of *sub-meshes* $x^{n+1,k}$, where $x^{n+1,k+1}$ is obtained from $x^{n+1,k}$ by using a step of $\Delta t_n/K$ with a linearized MMPDE. In this case ρ is approximated by $\rho^{n+1,k}$ obtained by constructing a piecewise linear interpolant of (x^n,ρ^n) values onto $x^{n+1,k}$. A M^vP algorithm, which updates from t_n to t_{n+1} using variable time steps can also be used. The time lag between the mesh and physical solution can be reduced by iterating between the mesh and physical PDE l times, resulting in a $(MP)^l$ algorithm. The algorithm $(MP)^\infty$ is equivalent to the simultaneous solution (if the iteration converges). These solution variants are discussed at length in [15]. The implementation of these variants and a study of their efficacy within the stochastic domain decomposition framework is a topic of current investigation.

During our experiments we also detected that fewer Monte–Carlo simulations are needed to generate quality periodic meshes. This seems to be due to the absence of an exit time test for the periodic case. Work to fully understand and utilize this is also underway.

Acknowledgements

The authors thank Weizhang Huang for helpful discussions. AB is a recipient of an APART Fellowship of the Austrian Academy of Sciences. This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] J. A. Acebrón, M. P. Busico, P. Lanucara and R. Spigler. Domain decomposition solution of elliptic boundary-value problems via Monte Carlo and quasi-Monte Carlo methods. SIAM J. Sci. Comput., 27: 440–457, 2005.

[2] J. A. Acebrón and R. Spigler. A new probabilistic approach to the domain decomposition method. In Domain Decomposition Methods in Science and Engineering XVI, Springer, Berlin Heidelberg, 473–480, 2007.

[3] M. J. Baines, M. E. Hubbard, P. K. Jimack and R. Mahmood. A moving-mesh finite element method and its application to the numerical solution of phase-change problems. Commun. Comput. Phys. 6: 595–624, 2009.

[4] G. Beckett, J. A. Mackenzie and M. L. Robertson. A moving mesh finite element method for the solution of two-dimensional stefan problems. J. Comput. Phys. 168: 500–518, 2001.

[5] A. Bihlo and R. D. Haynes. Parallel stochastic methods for PDE based grid generation. Comput. Math. Appl. 68: 804–820, 2014.

[6] A. Bihlo and R. D. Haynes. A stochastic domain decomposition method for time dependent mesh generation. In Domain Decomposition Methods in Science and Engineering XXII, Springer, in press, 2015.

[7] C. J. Budd, M. J. P. Cullen and E. J. Walsh. Monge-Ampére based moving mesh methods for numerical weather prediction, with applications to the Eady problem. J. Comput. Phys. 236: 247–270, 2013.

[8] H. G. Burchard. Splines (with optimal knots) are better. Appl. Anal. 3: 309–319, 1974.
[9] W. Cao, W. Huang and R. D. Russell. An *r*-adaptive finite element method based upon moving mesh PDEs. J. Comput. Phys., 149: 221–244, 1999.
[10] R. Chen and X. C. Cai. Parallel One-Shot Lagrange-Newton-Krylov-Schwarz Algorithms for Shape Optimization of Steady Incompressible Flows. SIAM J. Sci. Comput. 34, 584–605, 2012.
[11] C. de Boor. Good approximation by splines with variable knots, in Spline Functions and Approximation Theory. Springer, Basel, 57–72, 1973.
[12] M. J. Gander and R. D. Haynes. Domain decomposition approaches for mesh generation via the equidistribution principle. SIAM J. Numer. Anal. 50: 2111–2135, 2012.
[13] R. D. Haynes and A. J. M. Howse. Generating equidistributed meshes in 2d via domain decomposition. In Domain Decomposition Methods in Science and Engineering XXI, Springer, 776–797, 2012.
[14] P. He and H. Tang. An adaptive moving mesh method for two-dimensional relativistic magnetohydrodynamics. Comput. Fluids, 60: 1–20, 2012.
[15] W. Huang and R. D. Russell. Adaptive Moving Mesh Methods, Springer, New York, 2010.
[16] W. Huang and X. Zhan. Adaptive moving mesh modeling for two dimensional groundwater flow and transport. Amer. Math. Soc., Providence, RI, 239–252, 2005.
[17] K. M. Jansons and G. D. Lythe. Exponential timestepping with boundary test for stochastic differential equations. SIAM J. Sci. Comput. 24: 1809–1822, 2003.
[18] C. Jin , K. Xu and S. Chen. A three dimensional gas-kinetic scheme with moving mesh for low-speed viscous flow computations. Adv. Appl. Math. Mech 2: 746–762, 2010.
[19] S. Kakutani. On Brownian motions in n-space. Proc. Imp. Acad., Tokyo, 20: 648–652, 1944.
[20] S. Kakutani. Two-dimensional Brownian motion and harmonic functions. Proc. Imp. Acad., Tokyo, 20: 706–714, 1944.
[21] I. Karatzas and S. E. Shreve. Brownian motion and stochastic calculus. Vol. 113 of Graduate Texts in Mathematics, Springer, New York, 1991.
[22] J. Lang, W. Cao, W. Huang and R. D. Russell. A two-dimensional moving finite element method with local refinement based on a posteriori error estimates. Appl. Numer. Math., 46: 75–94, 2003.
[23] G. Milstein and M. Tretyakov. Stochastic numerics for mathematical physics. Springer, Berlin Heidelberg, 2004.
[24] B. Øksendal. Stochastic differential equations: an introduction with applications. Springer, Heidelberg, 2010.
[25] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery. Numerical recipes 3rd edition: The art of scientific computing. Cambridge University Press, Cambridge, UK, 2007.
[26] S. Quan. Simulations of multiphase flows with multiple length scales using moving mesh interface tracking with adaptive meshing. J. Comput. Phys., 230: 5430–5448, 2011.
[27] Z. Tan, K. Lim and B. Khoo. An adaptive moving mesh method for two-dimensional incompressible viscous flows. Commun. Comput. Phys., 3: 679–703, 2008.
[28] A. B. White. On selection of equidistributing meshes for two-point boundary-value problems. SIAM J. Numer. Anal., 16: 472–502, 1979.
[29] A. M. Winslow. Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh. J. Comput. Phys., 1: 149–172, 1966.
[30] Y. Yuan. The characteristic finite element alternating-direction method with moving meshes for the transient behavior of a semiconductor device. Int. J. Numer. Anal. Model., 9: 86–104, 2012.
[31] Y. Zhang and T. Tang. Simulating three-dimensional free surface viscoelastic flows using

moving finite difference schemes. Numer. Math. Theory Methods Appl., 4: 92–112, 2011.
[32] Z. Zhang and T. Tang, An adaptive mesh redistribution algorithm for convection-dominated problems, Department of Mathematics, Hong Kong Baptist University, 2002.