# A stochastic domain decomposition method for time dependent mesh generation

Alexander Bihlo[1] and Ronald D. Haynes[2]

## 1 Introduction

We are interested in PDE based mesh generation. The mesh is computed as the solution of a mesh PDE which is coupled to the physical PDE of interest. In [3] we proposed a stochastic domain decomposition (SDD) method to find adaptive meshes for steady state problems by solving a linear elliptic mesh generator. The SDD approach, as originally formulated in [1], relies on a numerical evaluation of the probabilistic form of the exact solution of the linear elliptic boundary value problem. Monte–Carlo simulations are used to evaluate this probabilistic form only at the sub-domain interfaces. These interface approximations can be computed independently and are then used as Dirichlet boundary conditions for the deterministic sub-domain solves. It is generally not necessary to solve the mesh PDEs with high accuracy. Only a good quality mesh, one that allows an accurate representation of the physical PDE, is required. This lower accuracy requirement makes the proposed SDD method computationally more attractive, reducing the number of Monte–Carlo simulations required.

Grid adaptation by a SDD approach does generate interesting issues in its own right. Grid quality should be monitored during the interface solves to give a suitable stopping criteria for the stochastic portion of the algorithm. Such a stopping criteria can be readily implemented by checking the mesh quality (as measured e.g. through mesh smoothness, alignment or equidistribution, see [5]) after every $n$th Monte–Carlo simulation. If the mesh quality is below a threshold, the additive nature of expected value computations allows one to resume the Monte–Carlo simulations and hence improve the mesh generation.

Memorial University of Newfoundland, St. John's, NL, Canada `abihlo@mun.ca` · Memorial University of Newfoundland, St. John's, NL, Canada `rhaynes@mun.ca`

As mentioned, in [3] only the steady grid generation problem was considered. Of course, in practice, the problem of grid generation is coupled with the process of solving the system of physical, often time dependent, PDEs. It is this latter issue that we begin to explore in this paper.

We are interested in time dependent PDEs whose solutions evolve on disparate space and time scales. The solution behaviour lends itself to the use of time dependent meshes which automatically adapt and evolve to efficiently resolve the solution features. The generation of these time dependent grids can be done either by statically applying an elliptic mesh generator using the physical solution obtained at the previous time step or by employing a time relaxation of the static mesh PDE resulting in a parabolic moving mesh PDE, as in [5]. The extension of the SDD approach to (linear) parabolic mesh generators is possible due to the existence of a stochastic representation of the exact solution of such linear parabolic problems. For the sake of illustration, we will work with the time-relaxed form of the Winslow–Crowley variable diffusion mesh generation method, first described in [9].

## 2 Winslow's method

The *equipotential* method of mesh generation in 2D, as described in [4], found the mesh lines in the physical co-ordinates $x$ and $y$ as the level curves of the potentials $\xi$ and $\eta$ satisfying Laplace's equations

$$\nabla^2\xi = 0, \qquad \nabla^2\eta = 0, \tag{1}$$

and appropriate boundary conditions which ensure grid lines lie along the boundary of the domain. Here derivatives are with respect to the physical co–ordinates. The mesh transformation, $x(\xi,\eta)$ and $y(\xi,\eta)$, in the physical domain $\Omega_p$, can be found by (inverse) interpolation of the solution of (1) onto a (say) uniform $(\xi,\eta)$ grid. In practice, the inversion to the physical co–ordinates is not necessary. Instead one could transform the physical PDE of interest to the computational co–ordinate system.

Winslow [10] generalized (1) by adding a diffusion coefficient $w(x,y)$ depending on the gradient or other aspects of the solution. This gives the linear elliptic mesh generator

$$\nabla\cdot(w\nabla\xi) = 0 \quad \text{and} \quad \nabla\cdot(w\nabla\eta) = 0. \tag{2}$$

The function $w(x,y)$, known as a mesh density function, characterizes regions where additional mesh resolution is needed and in general depends on the solution of the physical PDE. We assume $w$ and $1/w$ are strictly positive, bounded $C^2$-functions.

Here we assume the solution of the physical PDE is time dependent and hence the mesh density function is changing with time, $w = w(t,x,y)$. One could still use (2) to solve the mesh transformation at each time $t$. For time dependent PDEs this would result a system of differential–algebraic equations for the physical solution and the mesh. Instead, we choose to relax (2) to obtain a parabolic linear mesh generator of the form

$$\xi_t = \frac{1}{T}(\nabla w \cdot \nabla \xi + w \nabla^2 \xi) \quad \text{and} \quad \eta_t = \frac{1}{T}(\nabla w \cdot \nabla \eta + w \nabla^2 \eta). \tag{3}$$

This gives a mesh PDE which depends explicitly on the mesh speed and provides a degree of temporal smoothing for the mesh. In fact one can show the difference between the solution of (3) and the solution of (2) goes to zero as $T \to 0$, see [5]. In the following, we set $T = 1$.

Below we only work with prescribed functions for $w$. In practice, however, the monitor function would be linked to the solution of a physical PDE. For example, one could use an arc-length type function $\rho = \sqrt{1 + \alpha(u_x^2 + u_y^2)}$ and choose $w = 1/\rho$. We also note that our algorithm uses an interpolated form of $w$ instead of the analytic expression. In practice, this is necessary since $u$ is only known on the current grid as we alternately solve the mesh and physical PDEs.

## 3 Linear parabolic differential equations and stochastic domain decomposition

The system of mesh PDEs (3) is of the form

$$\xi_t = \mathrm{L}\xi, \quad \eta_t = \mathrm{L}\eta, \tag{4}$$

where $\xi(t, x, y)$ and $\eta(t, x, y)$ are the computational coordinates defined over $[0, T] \times \Omega_{\mathrm{p}}$. In system (4), L is a linear elliptic operator of the form

$$\mathrm{L} = a_{ij} \frac{\partial^2}{\partial x_i \partial x_j} + b_i \frac{\partial}{\partial x_i},$$

with continuous coefficient matrix $a(t, x, y) = (a_{ij})(t, x, y)$, $i, j \in \{1, 2\}$, and drift vector $\mathbf{b} = (b_1, b_2)^{\mathrm{T}}(t, x, y)$. Here we employ the summation convention over repeated indices. System (4) is accompanied by smooth boundary and initial conditions $\xi|_{\partial\Omega_{\mathrm{p}}} = f(t, x, y)$, $\eta|_{\partial\Omega_{\mathrm{p}}} = g(t, x, y)$, $\xi(0, x, y) = \xi_0(x, y)$, and $\eta(0, x, y) = \eta_0(x, y)$.

The solution of such linear parabolic problems can be described using the tools of stochastic calculus [2, 7]. Provided that $\xi$ and $\eta$ are $C^1$-functions in $t$ and $C^2$ in $(x, y)$, the point–wise solution of system (4) at $(t, x, y) \in [0, T] \times \Omega_p$ is given probabilistically as

$$\xi(t,x,y) = \mathrm{E}\left[\xi_0(\mathbf{X}(t))\mathbf{1}_{[\tau_{\partial\Omega_\mathrm{p}}>t]}\right] + \mathrm{E}\left[f(t-\tau_{\partial\Omega_\mathrm{p}},\mathbf{X}(\tau_{\partial\Omega_\mathrm{p}}))\mathbf{1}_{[\tau_{\partial\Omega_\mathrm{p}}<t]}\right], \quad (5)$$

where the process $\mathbf{X}(t) = (x(t), y(t))^\mathrm{T}$ satisfies, in the Îto sense, the stochastic differential equation (SDE)

$$\mathrm{d}\mathbf{X}(t) = \mathbf{b}(t,\mathbf{X}(t))\mathrm{d}t + \sigma(t,\mathbf{X}(t))\mathrm{d}\mathbf{W}(t).$$

The relation between $\sigma$ and $(a_{ij})$ is given through

$$\frac{1}{2}\sigma(t,x,y)\sigma(t,x,y)^\mathrm{T} = a(t,x,y)$$

for all $(t,x,y) \in [0,T]\times\mathbf{R}^2$. The solution for $\eta(t,x,y)$ is completely analogous.

In (4), the $\mathrm{E}[\cdot]$ denotes the expected value, $\tau_{\partial\Omega_\mathrm{p}}$ is the time when the stochastic path starting at $(x,y)$ first hits the boundary of the physical domain $\Omega_\mathrm{p}$, $\mathbf{W}$ is two-dimensional Brownian motion and $\mathbf{1}$ is the indicator function. See [7] for a proper definition of the required probability space.

The time dependent mesh generator (3) is a special case of the general form (4) with

$$a(t,x,y) = wI_2, \quad b_1(t,x,y) = w_x, \quad b_2(t,x,y) = w_y, \quad (6)$$

where $I_2$ is the $2\times 2$ identity matrix.

For our two dimensional mesh generator we choose the initial conditions $\xi(t=0,x,y) = \xi_0(x,y) = x$ and $\eta(t=0,x,y) = \eta_0(x,y) = y$, corresponding to an initial uniform mesh, and the static boundary conditions $\xi(t,x_\mathrm{l},y) = 0$, $\xi(t,x_\mathrm{r},y) = 1$, $\eta(t,x,y_\mathrm{l}) = 0$ and $\eta(t,x,y_\mathrm{u}) = 1$. This ensures we use the standard computational domain $\Omega_\mathrm{c} = [0,1]\times[0,1]$ and the rectangular physical domain $\Omega_\mathrm{p} = [x_\mathrm{l},x_\mathrm{r}]\times[y_\mathrm{l},y_\mathrm{u}]$. The remaining boundary conditions for $\xi(t,x,y_\mathrm{l}), \xi(t,x,y_\mathrm{u}), \eta(t,x_\mathrm{l},y)$ and $\eta(t,x_\mathrm{r},y)$ are determined by solving the 1D version of (2) along the boundaries. Collectively, we use $f$ and $g$ to denote these boundary conditions for $\xi$ and $\eta$ as in Eq. (5).

Hence we have to solve the SDE

$$\mathrm{d}\mathbf{X}(t) = \nabla w\,\mathrm{d}t + \sqrt{2}w\,\mathrm{d}\mathbf{W}(t), \quad (7a)$$

for the single path $\mathbf{X}(t)$. The stochastic form of the exact solution of Eq. (3) for $\xi$ is then obtained by evaluating

$$\xi(t,x,y) = \mathrm{E}\left[\xi_0(\mathbf{X}(t))\mathbf{1}_{[\tau_{\partial\Omega_\mathrm{p}}>t]}\right] + \mathrm{E}\left[f(\mathbf{X}(\tau_{\partial\Omega_\mathrm{p}}))\mathbf{1}_{[\tau_{\partial\Omega_\mathrm{p}}<t]}\right]. \quad (7b)$$

The point–wise solution for $\eta(t,x,y)$ is obtained in an analogous fashion.

In principle, the probabilistic solution (7) allows one to determine the computational coordinates $\xi$ and $\eta$ at each point in the space–time domain $[0,T]\times\Omega_\mathrm{p}$. However, this is prohibitively expensive (unless a sufficiently large number of compute cores is available). A more efficient approach is to evaluate the solution (7) only at points along artificially imposed interfaces. These solutions serve as boundary values for the DD implementation. Moreover, one can reduce the number of stochastic solves along the interfaces even further as described at the end of the next section, cf. [2].

In the mesh generation context it is not possible to obtain the solution of (5) at all times, as the solution of the mesh PDE is coupled to the physical solution. That is, rather than solving (5) for a time $t \in [0, T]$, it is generally only possible to use this stochastic solution to advance the solution of (4) over one single time step from $t^n$ to $t^{n+1}$. In this case, $\xi_0$ and $\eta_0$ should be interpreted as the values of $\xi$ and $\eta$ at time $t^n$ and the monitor function, $w$, is given at either $t^n$ or $t^{n+1}$ and remains constant over the time step.

## 4 The numerical method

**Stochastic solver and domain decomposition.** The use of the stochastic solution (5) for the time-relaxed Winslow mesh generator with parameters (6) is straightforward. We solve (7a) using the classical Euler–Maruyama scheme, i.e. we employ linear time-stepping. An alternative would be to use exponential time-stepping as advocated e.g. in [1, 3, 6]. In our tests, linear time-stepping gives sufficient accuracy. The components of the Brownian motion $\mathrm{d}\mathbf{W}(t)$ are computed as $\sqrt{\Delta t}\,\mathcal{N}(0,1)$, where $\mathcal{N}(0,1)$ is a normally distributed random number with mean zero and variance one [7].

The time dependent weight only becomes available at each time step (due to a possible coupling with a physical PDE). Hence we are only able to employ formula (7b) to integrate over a single time step, i.e. from $t^n$ to $t^{n+1}$. Over this time step, the weight function is evaluated at $t^n$ and held constant, i.e. we have $w^n(x, y) = w(t^n, x, y)$ in (7a). Accordingly, $\xi_0$ in Eq. (7b) is to be interpreted as $\xi_0^n = \xi(t^n, x, y)$, i.e. the values of the computational coordinates at the current time $t^n$. Moreoever, the boundary functions $f$ and $g$ are updated at each time to reflect changes in the physical solution.

We then numerically integrate the SDE (7a) from $t^n$ to $t^{n+1}$. The drift vector $\mathbf{b} = \nabla w$ is required everywhere along the path of the stochastic process $\mathbf{X}(t)$ but is only available directly at the grid points of the domain. Bilinear interpolation is used to obtain the values of $\mathbf{b}$ in between these grid points. The quantity $\nabla w$ is approximated using finite differences.

In the DD context, the stochastic solution is only required at a selection of points, $(x_k^i, y_k^i)$, which live on the interfaces between sub-domains. One time step $\Delta t$ is split into several smaller sub-time steps in order to numerically integrate the SDE (7a) from $t^n$ to $t^{n+1}$. We found this splitting of $\Delta t$ into sub-time steps necessary to determine, with sufficient accuracy, whether the stochastic processes has left the domain $\Omega_\mathrm{p}$ during $\Delta t$. This is not unlike the $M^k$ approach for mesh generation discussed in [5]. At each sub-time step, a boundary test is performed to determine whether the stochastic process has left the domain $\varOmega_{\rm p}$. If this is the case, the process contributes via the second term in Eq. (7b) to the approximation of $\xi(t^{n+1}, x_k^i, y_k^i)$. If the stochastic process did not leave the domain until $t^{n+1}$ is reached, it contributes to the first term in the approximation of $\xi(t^{n+1}, x_k^i, y_k^i)$ in Eq. (7b). The computation of $\eta(t^{n+1}, x_k^i, y_k^i)$ is handled analogously. The expected values are then replaced by arithmetic means. Note, it is not desirable to make $\varDelta t$ itself smaller, as this would degrade the efficiency of the (deterministic) implicit sub-domain solver, which is described below.

**Deterministic sub-domain solver.** The values of $\xi$ and $\eta$ along the subdomain interfaces serve as boundary conditions for the sub-domain solver. The sub-domain solver we employ is an implicit finite-difference discretization of Eq. (3). The matrix system is solved using an LU-factorization.

**Parallelization and further speed-up.** It is well-known that Monte-Carlo techniques converge rather slowly [8] and are usually most competitive for problems in high dimensions. The use of the stochastic solution to obtain the interface values of a DD problem only, however, is considerably more efficient and provides a fully parallel grid generation algorithm. Moreover, the DD method requires no iteration. The stochastic solutions on the interfaces can be determined at each point separately and each Monte-Carlo simulation is independent. Additionally, each sub-domain solution could potentially be assigned to a single processor once the interface solutions are obtained, yielding excellent scalability. Due to the fully parallel nature of the algorithm, the method is also fault tolerant. This renders the method suitable for an implementation on massively parallel computing architectures, cf. [1, 2, 3].

A further source of improvement stems from the fact that $\xi$ and $\eta$ do not have to be computed at all grid points along the interfaces. As proposed in [1] it may be sufficient to use the stochastic solution only at few points on the interface and recover the solution at the remaining interface points using interpolation. In [3] we have used a relatively simple optimal placement strategy to determine the most important locations on the interface where the stochastic solution should be computed. We use the same strategy in the present algorithm, i.e. the stochastic solution is computed near the maxima and minima of $w_x$ and $w_{xx}$ along the horizontal interfaces and $w_y$ and $w_{yy}$ along the vertical interfaces.

## 5 Numerical Results

We present an example for our SDD method to generate an adaptive (moving) mesh for the weight function $w = 1/\rho$, where

$$\rho = 1+\alpha\exp\left(\beta\left|\left(x-\frac{1}{2}-\frac{1}{4}\cos(2\pi t)\right)^2+\left(y-\frac{1}{2}-\frac{1}{4}\sin(2\pi t)\right)^2-\frac{1}{100}\right|\right).$$

We choose the parameters $\alpha = 10$ and $\beta = -50$ used in [5]. Both the physical and computational domain are the unit square. The grid we generate has $41 \times 41$ nodes and is divided into four sub-domains. On the interfaces we determine the stochastic solution at the key points using the optimal placement strategy mentioned in the previous section. Piecewise cubic Hermite interpolation is used to determine the remaining interface points. We integrate (3) up to $t = 0.75$ using $\Delta t = 0.001$. Each time step is split into 20 sub-time steps while solving the SDE (7a) and $N = 10000$ Monte-Carlo simulations are used to estimate the expected value in (7b). The resulting meshes at $t = 0.25$, $t = 0.5$, and $t = 0.75$ are depicted in Fig. 5.

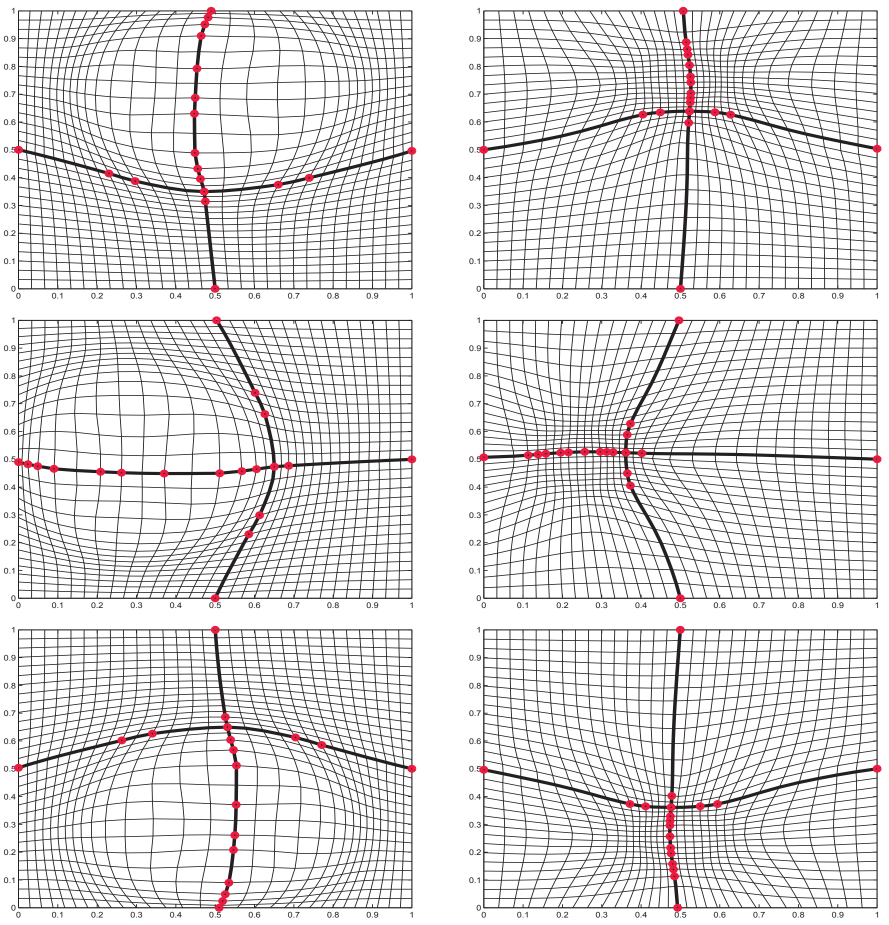

**Fig. 1** Top to bottom: Meshes obtained from the parabolic mesh generator (3) using the SDD method at $t = 0.25$, $t = 0.5$, and $t = 0.75$. Left: Meshes over the physical domain. Right: Meshes over the computational domain obtained from the former using natural neighbor interpolation. Thick line: Sub-domain interfaces. Circles: Points where the mesh is obtained using the stochastic solution (7).

The method is able to produce smooth meshes over the physical domain that adapt well to the time-dependent monitor function. No explicit smoothing was applied to the final meshes in this example. In general we have found sub-domain smoothing to be a way to further reduce the number of Monte-Carlo simulations needed in the probabilistic expression (7b), see [3].

## 6 Conclusion

In this paper we have proposed a new stochastic domain decomposition method for the construction of adaptive moving meshes suitable for time-dependent problems. The method is fully parallelizable as the values of the computational coordinates $\xi$ and $\eta$ on the single sub-domains can be determined without information exchange from neighboring sub-domains and all the interface values can be computed independently.

Future refinements include the use of exponential time-stepping to solve the SDE (7a). More generally, more sophisticated boundary tests could better determine the first exit time of a stochastic process. This will allow using larger time steps in the solution of (7a) thus making the method more efficient. An alternate approach to generate time dependent meshes is to apply the stochastic–DD method from [3] to the sequence of elliptic problems which result from discretizing (2) in time.

**Acknowledgements.** This research was supported by NSERC (Canada). AB is a recipient of an APART Fellowship of the Austrian Academy of Sciences. The authors thank Professor Weizhang Huang (Kansas) and the two anonymous referees for helpful remarks.

## References

1. Acebrón, J.A., Busico, M.P., Lanucara, P., Spigler, R.: Domain decomposition solution of elliptic boundary-value problems via Monte Carlo and quasi-Monte Carlo methods. SIAM J. Sci. Comput. **27**(2), 440–457 (2005)
2. Acebrón, J.A., Rodríguez-Rozas, Á., Spigler, R.: Efficient parallel solution of nonlinear parabolic partial differential equations by a probabilistic domain decomposition. J. Sci. Comput. **43**(2), 135–157 (2010)
3. Bihlo, A., Haynes, R.D.: Parallel stochastic methods for PDE based grid generation. Comput. Math. Appl. **68**(8), 804–820 (2014)
4. Crowley, W.P.: An 'equipotential' zoner on a quadrilateral mesh. Tech. rep. (1962)
5. Huang, W., Russell, R.D.: Adaptive Moving Mesh Methods. Springer, New York (2010)
6. Jansons, K.M., Lythe, G.D.: Exponential timestepping with boundary test for stochastic differential equations. SIAM J. Sci. Comput. **24**(5), 1809–1822 (2003)
7. Karatzas, I., Shreve, S.E.: Brownian motion and stochastic calculus, *Graduate Texts in Mathematics*, vol. 113. Springer, New York (1991)
8. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes 3rd edition: The art of scientific computing. Cambridge University Press, UK (2007)
9. Winslow, A.M.: Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh. J. Comput. Phys. **1**(2), 149–172 (1966)
10. Winslow, A.M.: Adaptive-mesh zoning by the equipotential method. Tech. Rep. UCID-19062, Lawrence Livermore National Lab., CA (USA) (1981)