SIAM J. SCI. COMPUT.
Vol. 41, No. 2, pp. A1331–A1356

ADAPTIVE SOLUTION TO TWO-DIMENSIONAL PARTIAL DIFFERENTIAL EQUATIONS IN CURVED DOMAINS USING THE MONGE–AMPÉRE EQUATION*

KELSEY L. DIPIETRO† AND ALAN E. LINDSAY†

Abstract. We present a finite difference method for mesh generation and adaptation with application to the solution of partial differential equations in curved domains. For mesh generation, we construct a mapping F between a fixed rectangular domain Ω_C and a curved physical domain Ω_P using a finite difference approximation of the fully nonlinear Monge–Ampére equation solved with a damped Newton's method. Paired with grid adaptation, this method gives a finite difference approximation for curved geometries that distributes mesh points toward localized interior features such as sharp interfaces using a solution dependent monitor function. The method dynamically resolves fine scale PDE behavior on stationary and evolving curved domains while retaining the simplicity of performing all computations on a static rectangular grid with a fixed number of mesh points and connectivity. We display the efficacy of the methods on a variety of linear and nonlinear PDE examples on convex and select nonconvex curved domains.

Key words. optimal transport, mesh generation, finite diffrences, curved domains, nonlinear partial differential equations

AMS subject classifications. 65N50, 65N06, 35R37, 35J96

DOI. 10.1137/18M123075X

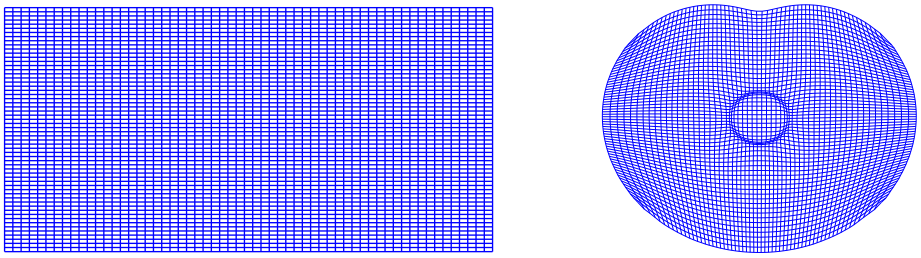
1. Introduction.

We present a finite difference method for mesh generation and adaptation with application to the solution of partial differential equations (PDEs) on curved domains. The method utilizes the Monge–Ampére equation to create a mapping from a static computational domain to a curved, dynamically adapted physical domain (cf. Figure 1). All computations for the PDE are performed on a static, uniform rectangular grid, allowing for simple finite difference discretization. The method produces grids that conform to convex and select nonconvex, static, and evolving curved domains that can dynamically adapt to resolve fine scale interior structures.

The numerical solution of PDEs on general domains is a common problem in a variety of physical and biological problems. In a large class of such problems, the solution may exhibit fine scale behavior to which the numerical method must adapt in order to ensure that accuracy is maintained. A vast range of approaches have been developed in the finite difference, finite volume, and finite element spaces. The methodologies developed herein focus on using finite difference methods capable of performing r-adaptive mesh refinement, so we briefly discuss previous finite difference approaches on curved domains.

Standard finite difference methods use rectangular grids with a fixed number of mesh nodes where neighboring points define the Taylor series approximations for the derivatives at each grid point. Because of the natural grid structure in finite differences, additional methods are required to handle curved boundaries. One of

*Submitted to the journal's Methods and Algorithms for Scientific Computing section December 5, 2018; accepted for publication March 14, 2019; published electronically April 30, 2019.
http://www.siam.org/journals/sisc/41-2/M123075.html
Funding: The work of the first author was supported by the National Science Foundation under grant DGE-1313583. The work of the second author was supported by the National Science Foundation under grant DMS-1516753.
†Applied Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46556 (kdipiet1@nd.edu, a.lindsay@nd.edu).



(a) Rectangular computational mesh Ω_C of dimension 60×60.

(b) Adapted physical grid on a non-convex domain Ω_P.

FIG. 1. *Example of a mesh generated from the methods in this paper: mapping a rectangular computational grid (left) to an adapted grid in a curved domain (right).*

the first methods to address curved domains was the finite difference time domain method [43], which uses a staircase approximation to interpolate the solution onto the boundary. The approach has been widely used but is limited in its application to a specific set of PDEs and loses accuracy for more complex problems and domains. These limitations led to a number of improvements for treating a wider class of PDEs on curved domains, most of which are classified into two methods: unstructured grids and body fitted curvilinear coordinate grids.

In the unstructured grid case, the domain is covered by an irregular distribution of points. For each central node, nearby nodes are selected as the supporting nodes to build a stencil approximation for the derivatives [5, 6, 35]. This method offers a meshless, unstructured approach to treating curved boundaries with Dirichlet, Neumann, or mixed boundary conditions. However, it comes at the cost of irregular grid spacing that requires unique Taylor series expansions for each derivative at each central node, making computational aspects more tedious.

The need for computational efficiency and simplicity has motivated the development and use of curvilinear coordinate systems and elliptic grid generation of meshes on curved domains [1, 20, 25, 39]. Each of the methods differs in the mechanisms that create the mapping, but the overall principle is the same: take a uniform computational mesh and use a coordinate transformation to generate a global curvilinear coordinate system conforming to the boundary. The Monge–Ampére equation is an example of an elliptic grid generation approach, where equidistribution and optimal transport constraints determine a unique two-dimensional coordinate transformation for convex domains.

The aforementioned approaches focus on generating appropriate static meshes given a domain. If the PDE system has shocks, singularities, or other fine scale behaviors, an adapted mesh may be necessary to avoid losing solution accuracy. A recent h-adaptive solution to this problem [37] used a quadtree approach to create adaptive grids on curved domains, though care must be taken to avoid instabilities from dangling nodes that arise from the addition and removal of mesh points.

There has been burgeoning recent interest in using r-adaptive methods in which a fixed number of mesh points are dynamically redistributed to increase mesh density in areas of high solution interest [10, 12, 13]. These methods, initially developed for one-dimensional problems with shocks or singularities [27, 28, 38], have since been developed to solve a variety of higher dimensional problems [14, 16, 28], with applications including the Burgers equation, semilinear blow-up [10], numerical weather prediction [8], porous media flow [36], and fourth order singularity and interface problems [19].

A common feature of the aforementioned examples, based primarily on optimally transported maps, is that they were limited to logically rectangular domains.

This work provides a bridge between mesh generation and moving mesh adaptation for PDEs on curved domains using optimally transported maps and finite difference discretization. We solve a Monge–Ampére equation with suitable transport boundary conditions to generate meshes on curved domains. The result of this paper is a simple numerical method using finite difference approximations on logically rectangular grids to solve a general class of PDE problems in a wide range of geometries. The method dynamically increases mesh density to maintain solution accuracy in regions where the PDE has fine scale solution features.

The paper is organized as follows. We begin by introducing the Monge–Ampére approach for mesh generation in convex and nonconvex domains. We detail the discretization for the Monge–Ampére equations in the fully nonlinear case and the parabolic regularization case. Next, we give the coordinate transformation and discretization of the underlying PDE in the computational coordinates and discuss the implementation of generalized boundary conditions. Finally, we demonstrate the effectiveness and flexibility of these methods on a variety of linear and nonlinear PDEs on curved and time dependent domains. In Appendix A, we include a minimal code for this method to demonstrate the ease of implementation.

2. The Monge–Ampére equation. For mesh generation and adaptation, we create a coordinate transformation F from a fixed, uniform computational domain Ω_C to the physical domain Ω_P (see Figure 1),

$$F : \Omega_C \to \Omega_P. \tag{1}$$

This map is accompanied by an optimal transport boundary condition that maps the boundary of the computational domain to the boundary of the physical domain such that $F : \partial\Omega_C \to \partial\Omega_P$. While there are a variety of ways to create the mapping F, such as variational methods [26, 28], we utilize equidistribution and optimal transport techniques to generate two-dimensional meshes for solving PDEs.

The equidistribution principle [17] takes a time dependent, scalar monitor function $M(\mathbf{x}, t)$ as a surrogate for the solution error at $\mathbf{x} \in \Omega_P$ and seeks to distribute it evenly over the domain. This implies that mesh density is increased where $M(\mathbf{x}, t)$ is large, which corresponds to areas of high interest in the PDE solution. The equidistribution principle for a reference set $D \subset \Omega_C$ and its image $F(D) \subset \Omega_P$ is

$$\frac{\int_D d\boldsymbol{\xi}}{\int_{\Omega_C} d\boldsymbol{\xi}} = \frac{\int_{F(D,t)} M(\mathbf{x}, t) d\mathbf{x}}{\int_{\Omega_P} M(\mathbf{x}, t) d\mathbf{x}}. \tag{2}$$

Since the set $D \subset \Omega_C$ is arbitrary, condition (2) implies that

$$M(\mathbf{x}, t)|J(\mathbf{x})| = \theta(t) := \frac{\int_{\Omega_P} M(\mathbf{x}, t) d\mathbf{x}}{\int_{\Omega_C} d\boldsymbol{\xi}}, \tag{3}$$

where $|J(\mathbf{x})|$ is the determinant of the Jacobian of the transformation F. The equidistribution principle generates a unique mesh in one dimension, but not in higher dimensions. To obtain a unique mesh, the cost function

$$I = \int_{\Omega_C} |F(\boldsymbol{\xi}, t) - \boldsymbol{\xi}|^2 d\boldsymbol{\xi} \tag{4}$$

can be imposed so that the mesh is as close to uniform as possible while satisfying the equidistribution principle (3).

The mapping $F : \Omega_C \to \Omega_P$ with the optimal transport condition (4) and equidistribution principle (3) is unique and well posed, assuming a strictly convex target domain (for nonconvex domains, regularity is not guaranteed) [7]. Furthermore, the solution may be represented as the gradient of a scalar, convex mesh potential $\mathbf{x} = \nabla P = (P_\xi, P_\eta)$. It follows from (3) that P will satisfy the Monge–Ampére equation with an associated optimal transport boundary condition given as

$$M(\mathbf{x}, t)|H(P)| = \theta(t), \qquad \boldsymbol{\xi} \in \Omega_C; \tag{5a}$$

$$\nabla P = \mathbf{x}, \quad \mathbf{x} \in \partial\Omega_P, \quad \boldsymbol{\xi} \in \partial\Omega_C, \tag{5b}$$

where $|H(P)| = P_{\xi\xi}P_{\eta\eta} - P_{\xi\eta}^2$ is the determinant of the Hessian of P. When applied to moving mesh adaptation, the monitor function $M(\mathbf{x}, t)$ is dependent on the PDE solution and chosen to highlight regions of fine scale behavior. This allows the mesh to change to rapidly varying components in the PDE, without specifying mesh densities.

The system (5) is closely related to the classic optimal transport problem of minimizing the cost (under (4)) of transferring two densities $f(\boldsymbol{\xi})$, $g(\nabla P)$ from Ω_C to Ω_P. This problem is well known to satisfy the nonlinear Monge–Ampére equation

$$|H(P)| = \frac{f(\boldsymbol{\xi})}{g(\nabla P)}, \quad \boldsymbol{\xi} \in \Omega_C. \tag{6}$$

In the remaining sections we detail the numerical treatment of systems (5) and (6) with particular application to mesh generation and the numerical solution of PDEs.

3. Optimal transport boundary conditions. One of the most challenging aspects of solving the Monge–Ampére equation is finding a suitable approximation for the transport boundary condition (5b). The system (5) is well posed for convex domains [41, 42]; however, direct numerical simulation can give rise to instabilities. In scenarios where the mapping is between logical rectangles [10, 12, 19], simple nonhomogeneous Neumann boundary conditions can be applied to resolve this issue. For example, when $\Omega_C = \Omega_P = [\xi_l, \xi_r] \times [\eta_b, \eta_t]$, the relevant conditions are

$$P_\xi \,|_{\xi=\xi_l} = \xi_l, \quad P_\xi \,|_{\xi=\xi_r} = \xi_r, \quad P_\eta \,|_{\eta=\eta_b} = \eta_b, \quad P_\eta \,|_{\eta=\eta_t} = \eta_t. \tag{7}$$

The boundary conditions (7) fix the mesh points to the boundary while allowing them to move freely along it.

In previous work extending optimally transported maps to curved domains, the Monge–Ampére equation was used to generate a grid by manually mapping each of the four sides of Ω_C to the parameterized physical domain Ω_P, as opposed to creating a single mapping from a set of discrete points [21].

To create an automatic boundary mapping from a discrete representation of the curved target domain, the transport boundary condition is approximated with the Hamilton–Jacobi type equation [4, 22, 23]

$$\Psi(\nabla P) = 0, \quad \boldsymbol{\xi} \in \partial\Omega_C. \tag{8}$$

The choice of the Hamiltonian function Ψ depends on whether the geometry of the target set Ω_P is convex or nonconvex.

3.1. Signed distance. One potential choice for the boundary condition is the signed distance function given by

$$\Psi(\mathbf{x}) = \begin{cases} +\text{dist}(\mathbf{x}, \partial\Omega_P), & \mathbf{x} \notin \Omega_P; \\ -\text{dist}(\mathbf{x}, \partial\Omega_P), & \mathbf{x} \in \Omega_P, \end{cases} \tag{9}$$

which satisfies $\Psi(\nabla P) = 0$ at the boundary points $\partial\Omega_P$, making it an approximate solution for the transport boundary condition. To implement this condition at the discrete level, the signed distance function is calculated in terms of the convex hyperplanes of the target domain. In terms of the unit normals $\mathbf{n}_{\boldsymbol{\xi}}$ of Ω_C, an equivalent form of the boundary condition (9) is (cf. [4, Lemma 2.1])

$$\Psi(\nabla P) = \sup_{\|\mathbf{n}\|=1} \{\nabla P \cdot \mathbf{n} - \Psi^*(\mathbf{n}) \mid \mathbf{n} \cdot \mathbf{n}_{\boldsymbol{\xi}} > 0\} = 0, \tag{10a}$$

$$\Psi^*(\mathbf{n}) = \sup_{\mathbf{x}_0 \in \partial\Omega_P} \{\mathbf{x}_0 \cdot \mathbf{n}\}, \tag{10b}$$

where $\Psi^*(\mathbf{n})$ is the Legendre–Fenchel transform. The Lengendre–Fenchel transform gives a simple representation for the target domain by a discrete set of points. The condition (10a) specifies the normal derivative of P, thereby fixing mesh points to lie on the boundary but allowing them to move freely along it. An important feature of the Legendre–Fenchel transform is that it always returns the convex hull for a given domain [40].

3.2. L_2 projection formulation. As discussed previously, the Legendre–Fenchel transform imposes convexity given any target domain. To extend the method to nonconvex domains, we use an L_2 distance projection for boundary mappings. The L_2 projection boundary condition replaces the signed distance function by an L_2 minimization function, which essentially "pushes" the mesh points onto the boundary $\partial\Omega_P$. To solve the minimization problem, we follow the work of Froese [22] and formulate a series of subproblems $P^{(k)}$ for $k = 0, 1, 2, \ldots$ from a point $\boldsymbol{\xi}$ on the boundary of the computational domain to a set of points $\mathbf{x}^{(k)}$ on the boundary of the physical domain such that $\nabla P^{(k)} : \boldsymbol{\xi} \to \mathbf{x}^{(k)}$. The iterations are initialized with $P^{(0)}$, the solution obtained by solving the Monge–Ampére equation using the signed distance formulation which corresponds to the convex hull of Ω_P. The L_2 projection of $\nabla P^{(k)}$ onto $\partial\Omega_P$ is

$$\underset{\partial\Omega_P}{\text{Proj}}(\nabla P^{(k)}) = \underset{\mathbf{x} \in \partial\Omega_P}{\text{argmin}} \|\mathbf{x} - \nabla P^{(k)}\|_2^2. \tag{11}$$

Letting $\mathbf{n}_{\boldsymbol{\xi}}$ be the unit normal vectors of the computational domain Ω_C, we define

$$\phi^{(k)} = \underset{\partial\Omega_P}{\text{Proj}}(\nabla P^{(k)}) \cdot \mathbf{n}_{\boldsymbol{\xi}}. \tag{12}$$

The problem solved for the next iterate $P^{(k+1)}$ is

$$|H(P^{(k+1)})| = \frac{f(\boldsymbol{\xi})}{g(\nabla P^{(k+1)})} + P^{(k+1)}(\boldsymbol{\xi}_0), \quad \boldsymbol{\xi} \in \Omega_C; \tag{13a}$$

$$\Psi(\nabla P^{(k+1)}) = \nabla P^{(k+1)} \cdot \mathbf{n}_{\boldsymbol{\xi}} - \phi^{(k)} = 0, \quad \boldsymbol{\xi} \in \partial\Omega_C. \tag{13b}$$

The subproblems (13) are each solved by Newton's method. After convergence, the boundary condition (13b) is updated and the process is repeated until the difference between iterates $\|P^{(k+1)} - P^{(k)}\|_{L_\infty(\Omega_C)}$ is below a defined threshold (in practice 10^{-8}).

4. Discretization. The main goal of generating and adapting meshes on curved domains is to use them to accurately solve an underlying PDE posed on a curved domain. This entails discretizing two PDEs, the mesh equation and the underlying PDE. First, we detail the discretization of the Monge–Ampére equation and then turn our attention to discretizing the underlying PDE through the optimally transported map.

4.1. Discretization: Monge–Ampére for grid generation. Recalling the fully nonlinear Monge–Ampére equation for grid generation

$$|H(P)| = \frac{f(\boldsymbol{\xi})}{g(\nabla P)} + P(\boldsymbol{\xi}_0), \quad \boldsymbol{\xi} \in \Omega_C; \tag{14a}$$

$$\Psi(\nabla P) = 0, \quad \boldsymbol{\xi} \in \partial\Omega_C, \tag{14b}$$

where Ω_C is a fixed rectangular computational mesh. The term $P(\boldsymbol{\xi}_0)$ in (14a) is required to obtain a unique solution and remove the translation invariance resulting from the Neumann boundary condition. The problem is formulated as a nonlinear equation discretized using finite differences, where the Monge–Ampére is solved on the interior of the domain and the Hamilton–Jacobi equation is solved on the boundary. A family of wide stencil monotone schemes have been proposed in [4, 22], which guarantee convergence to a weak viscosity solution. In the present work, we implemented both the standard and monotone finite difference schemes and found they gave similar results for the examples considered. For brevity, we only describe the implementation of the standard second order scheme which is used for all the examples. The $N_\xi \times N_\eta$ rectangular computational grid for $\Omega_C = [\xi_l, \xi_r] \times [\eta_b, \eta_t]$ is defined by

$$\xi_i = \xi_l + (i-1)\Delta\xi, \quad i = 1, \dots, N_\xi; \qquad \eta_j = \eta_b + (j-1)\Delta\eta, \quad j = 1, \dots, N_\eta;$$

$$\Delta\xi = \frac{\xi_r - \xi_l}{N_\xi - 1}, \qquad \Delta\eta = \frac{\eta_t - \eta_b}{N_\eta - 1}. \tag{15}$$

We define $P_{i,j} = P(\xi_i, \eta_j)$ and set

$$\mathrm{MA}[P] = |H(P)| - \frac{f(\boldsymbol{\xi})}{g(\nabla P)} - P(\boldsymbol{\xi}_0). \tag{16}$$

The gradient $\nabla P = (P_\xi, P_\eta)$ and the determinant of the Hessian term $|H(P)| = P_{\xi\xi}P_{\eta\eta} - P_{\xi\eta}^2$ are approximated to second order by the stencils

$$[P_\xi]_{i,j} = \frac{P_{i+1,j} - P_{i-1,j}}{2\Delta\xi}, \qquad [P_\eta]_{i,j} = \frac{P_{i,j+1} - P_{i,j-1}}{2\Delta\eta},$$

$$[P_{\xi\xi}]_{i,j} = \frac{P_{i+1,j} - 2P_{i,j} + P_{i-1,j}}{2\Delta\xi^2}, \qquad [P_{\eta\eta}]_{i,j} = \frac{P_{i,j+1} - 2P_{i,j} + P_{i,j-1}}{2\Delta\eta^2}, \tag{17}$$

$$[P_{\xi\eta}]_{i,j} = \frac{P_{i+1,j+1} - P_{i+1,j-1} - P_{i-1,j+1} + P_{i-1,j-1}}{4\Delta\xi\Delta\eta}.$$

For the boundary condition $\Psi(\nabla P) = 0$ in (14b), an upwinding scheme is adopted that requires only the interior points of the domain. The formulation varies depending on whether the signed distance (convex target set) or projection function (nonconvex target set) are used. To discretize the signed distance function (10), we first define a discrete set of N_X normal vectors of the disk

$$\mathbf{n}_j = \left(\cos\frac{2\pi j}{N_X}, \sin\frac{2\pi j}{N_X}\right), \qquad j = 1, \dots, N_X. \tag{18}$$

For convex domains, we observe that $N_X = (N_\xi + N_\eta)$ provides good representation, while for nonconvex domains more points may be required. At each boundary point $\boldsymbol{\xi}_i \in \partial\Omega_C$, we discretize (10a) as

$$\Psi_i = \max_{\mathbf{n}=(n_1,n_2)\in\Lambda_i} \{\max(n_1,0)\mathcal{D}_\xi^- P(\boldsymbol{\xi}_i) + \min(n_1,0)\mathcal{D}_\xi^+ P(\boldsymbol{\xi}_i) + \max(n_2,0)\mathcal{D}_\eta^- P(\boldsymbol{\xi}_i) + \min(n_2,0)\mathcal{D}_\eta^+ P(\boldsymbol{\xi}_i) - \Psi^*(\mathbf{n})\}, \tag{19a}$$

$$\Psi^*(\mathbf{n}) = \sup_{\mathbf{x}_0\in\partial\Omega_P} \{\mathbf{x}_0 \cdot \mathbf{n}\}, \tag{19b}$$

where from (10a), $\Lambda_i = \{\|\mathbf{n}\| = 1 \mid \mathbf{n}\cdot\mathbf{n}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_i) > 0\}$ is the set of admissible directions at $\boldsymbol{\xi}_i$. In (19a), $\mathcal{D}_k^\pm$ are the standard second order forward and backward finite difference matrices applied in the respective directions $k = \{\xi, \eta\}$. For the projection formulation (20a), we discretize at each point $\boldsymbol{\xi}_i \in \partial\Omega_C$ by

$$\Psi_i = \max(n_1,0)\mathcal{D}_\xi^- P(\boldsymbol{\xi}_i) + \min(n_1,0)\mathcal{D}_\xi^+ P(\boldsymbol{\xi}_i) + \max(n_2,0)\mathcal{D}_\eta^- P(\boldsymbol{\xi}_i) + \min(n_2,0)\mathcal{D}_\eta^+ P(\boldsymbol{\xi}_i) - \operatorname*{Proj}_{\partial\Omega_P}(\nabla P^{(k)})\cdot\mathbf{n}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_i), \qquad (n_1,n_2) = \mathbf{n}_{\boldsymbol{\xi}}(\boldsymbol{\xi}_i), \tag{20a}$$

$$\operatorname*{Proj}_{\partial\Omega_P}(\nabla P^{(k)}) = \operatorname*{argmin}_{\mathbf{x}\in\partial\Omega_P} \|\mathbf{x} - \nabla P^{(k)}\|_2^2. \tag{20b}$$

For nonconvex target domains, it is possible for two (or more) points to be mapped to the same point on the boundary $\partial\Omega_P$ resulting in mesh tangling. To avoid tangling and obtain better convergence of the projection function formulation, more target points N_X are added and the density of points in the target domain Ω_P may be clustered near high curvature areas of $\partial\Omega_P$.

After discretizing the system (14), we use a relaxed Newton's method [2] to solve the resulting system of nonlinear equations for P to obtain the final mesh $\mathbf{x} = \nabla P$. An essential component of the success and efficiency of this approach is specification of the Jacobian of the system (14) which is given by

$$\nabla MA[P] = (\mathcal{D}_{\eta\eta}P)\mathcal{D}_{\xi\xi} + (\mathcal{D}_{\xi\xi}P)\mathcal{D}_{\eta\eta} - 2(\mathcal{D}_{\xi\eta}P)\mathcal{D}_{\xi\eta} - \frac{\partial G}{\partial x}\mathcal{D}_\xi - \frac{\partial G}{\partial y}\mathcal{D}_\eta - \mathbb{1}_0, \tag{21a}$$

$$\nabla\Psi(P) = \max(n_1,0)\mathcal{D}_\xi^- + \min(n_1,0)\mathcal{D}_\xi^+ + \max(n_2,0)\mathcal{D}_\eta^- + \min(n_2,0)\mathcal{D}_\eta^+, \tag{21b}$$

where $\mathcal{D}_k$ for $k = \{\xi, \eta, \eta\eta, \xi\xi, \xi\eta\}$ are the standard second order central finite difference matrices associated to (17) and $G(\boldsymbol{\xi}, \mathbf{x}) = f(\boldsymbol{\xi})/g(\mathbf{x})$ is the right-hand-side density term from the Monge–Ampére equation (14). The term $\mathbb{1}_0$ is a zero matrix except for a column of ones in the index corresponding to the location of $\boldsymbol{\xi}_0$. When discretizing the signed distance formulation (19), the normals (n_1, n_2) taken in (21b) correspond to the direction in Λ_i at which the maximum is taken. For discretization of the projection formulation (20a), the normals correspond to those of the computational domain.

4.2. Discretization: Parabolic Monge–Ampére for mesh adaptation. Once an initial mesh $\mathbf{x} = \nabla P$ has been determined from the solution of (14), it can be dynamically adapted to the solution of an associated PDE system through a solution dependent monitor function $M(\mathbf{x}, t)$. For a slowly varying $M(\mathbf{x}, t)$, it was shown in [10] that the parabolic Monge–Ampére (PMA) will evolve to equi-distribute the given monitor function. The PMA equation is the solution $Q(\boldsymbol{\xi}, t)$ of

$$\tau(I-\gamma\Delta_{\boldsymbol{\xi}})Q_t = (|H(Q)|M(\mathbf{x},t))^{\frac{1}{2}}, \quad \boldsymbol{\xi}\in\Omega_C, \quad t>0; \tag{22a}$$

$$Q_\xi = P_\xi, \quad \xi = \xi_l, \xi_r, \qquad Q_\eta = P_\eta, \quad \eta = \eta_b, \eta_t. \tag{22b}$$

Here $|H(Q)| = Q_{\xi\xi}Q_{\eta\eta} - Q_{\xi\eta}^2$, $\Delta_{\boldsymbol{\xi}} \equiv \partial_{\xi\xi}^2 + \partial_{\eta\eta}^2$, and τ and γ are user defined parameters controlling the meshing timescale and smoothing, respectively—in practice the default values are $\tau = \gamma = 0.1$. In (22b), $P(\boldsymbol{\xi})$ is the solution of (14) and acts as fixed Neumann boundary conditions for the PMA. The right-hand side of (22a) is raised to the power of $1/2$ so that the system remains globally well posed [10].

The determinant of the Hessian $|H(Q)|$ is discretized by fourth order central differences with appropriate stencils for boundary and boundary adjacent points that invoke boundary conditions (22b). The full stencils are in Appendix A of [19].

The monitor function $M(\mathbf{x},t)$ is chosen on a case by case basis so that mesh density increases where the PDE solution has fine scale behavior. For problems with sharp interfaces or moving fronts, arc-length or curvature based monitor functions are commonly chosen [8, 10]. In the case of finite time blow-up, monitor functions are constructed to inherit the natural scaling of the problem [11]. In cases where the monitor function uses derivatives of the solution, they are discretized using standard central finite difference stencils.

Once the monitor function has been chosen, additional steps are taken to ensure optimal quality of the mesh. An integral average,

$$M(\mathbf{x},t) \to M(\mathbf{x},t) + \alpha \int_{\Omega_P} M(\mathbf{x}',t)d\mathbf{x}', \tag{23}$$

is applied which ensures that all of the mesh points do not rush to the high interest areas [3]. The scalar parameter α controls the proportion of mesh points where the monitor function is high, leaving the remaining points to the rest of the mesh. The case of $\alpha = 1$ corresponds to a $50:50$ mesh.

To minimize errors from rapidly varying components of the monitor function, we apply the following fourth order smoothing filter four times during each iteration:

$$\begin{aligned} M_{i,j} \to M_{i,j} &+ \frac{2}{16}(M_{i+1,j} + M_{i-1,j} + M_{i,j-1} + M_{i,j+1}) \\ &+ \frac{1}{16}(M_{i+1,j+1} + M_{i-1,j-1} + M_{i+1,j-1} + M_{i-1,j+1}). \end{aligned} \tag{24}$$

4.3. Discretization in physical coordinates. Here we describe how the physical quantities $\nabla_{\mathbf{x}}u = (u_x, u_y)$ and $\Delta_{\mathbf{x}}u = u_{xx} + u_{yy}$ are discretized in computational coordinates for a mesh $\mathbf{x} = \nabla P$ which is obtained from the solution of (14). Using the computational coordinates $\boldsymbol{\xi} = (\xi,\eta)$ and the physical coordinates $\mathbf{x} = (x,y)$, we have the relationship

$$\begin{pmatrix} u_\xi \\ u_\eta \end{pmatrix} = \begin{pmatrix} x_\xi & y_\xi \\ x_\eta & y_\eta \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix}. \tag{25}$$

From $(x,y) = (P_\xi, P_\eta)$, the derivatives in computational variables are

$$u_x = \overline{J}(P_{\eta\eta}u_\xi - P_{\xi\eta}u_\eta), \tag{26a}$$

$$u_y = \overline{J}(-P_{\xi\eta}u_\xi + P_{\xi\xi}u_\eta), \tag{26b}$$

$$u_{xx} = \overline{J}P_{\eta\eta}\left(\overline{J}(P_{\eta\eta}u_\xi - P_{\xi\eta}u_\eta)\right)_\xi - \overline{J}P_{\xi\eta}\left(\overline{J}(P_{\eta\eta}u_\xi - P_{\xi\eta}u_\eta)\right)_\eta, \tag{26c}$$

$$u_{yy} = \overline{J}P_{\xi\eta}\left(\overline{J}(-P_{\xi\eta}u_\xi + P_{\xi\xi}u_\eta)\right)_\xi + \overline{J}P_{\xi\xi}\left(\overline{J}(-P_{\xi\eta}u_\xi + P_{\xi\xi}u_\eta)\right)_\eta. \tag{26d}$$

The Laplacian $\Delta_{\mathbf{x}} u$ can be written in the compact form

$$\Delta_{\mathbf{x}} u = \overline{J} \nabla_{\boldsymbol{\xi}} \cdot [\overline{J} \mathcal{A} \nabla_{\boldsymbol{\xi}} u], \tag{27a}$$

where $\mathcal{A}$ is the symmetric 2×2 matrix with entries

$$\mathcal{A}_{11} = P_{\xi\eta}^2 + P_{\eta\eta}^2, \qquad \mathcal{A}_{21} = \mathcal{A}_{12} = -P_{\xi\eta}(P_{\xi\xi} + P_{\eta\eta}), \qquad \mathcal{A}_{22} = P_{\xi\eta}^2 + P_{\xi\xi}^2. \tag{27b}$$

In the above equations, $\overline{J} = |H(P)|^{-1}$ is the reciprocal of the determinant of the Hessian. The form of (27) reveals that this method allows for the solution of PDEs on curved regions with adapted meshes at the cost of solving a related nonconstant coefficient problem on a fixed rectangular grid. We now discuss some of the small additional implementation details involved in this trade-off. Two of the appropriate stencils for the discretization of (27) are

$$\begin{aligned}
[(Au_\xi)_\xi]_{i,j} &= \frac{A_{i+\frac{1}{2},j}[u_\xi]_{i+\frac{1}{2},j} - A_{i-\frac{1}{2},j}[u_\xi]_{i-\frac{1}{2},j}}{\Delta\xi} \\
&= \frac{(A_{i,j} + A_{i+1,j})(u_{i+1,j} - u_{i,j}) - (A_{i,j} + A_{i-1,j})(u_{i,j} - u_{i-1,j})}{2\Delta\xi^2}, \qquad (28a) \\
[(Au_\xi)_\eta]_{i,j} &= \frac{A_{i,j+1}[u_\xi]_{i,j+1} - A_{i,j-1}[u_\xi]_{i,j-1}}{2\Delta\eta} \\
&= \frac{A_{i,j+1}(u_{i+1,j+1} - u_{i-1,j+1}) - A_{i,j-1}(u_{i+1,j-1} - u_{i-1,j-1})}{4\Delta\xi\Delta\eta}. \qquad (28b)
\end{aligned}$$

The expressions for $(Au_\eta)_\eta$ and $(Au_\eta)_\xi$ are similar.

4.3.1. Boundary conditions on the PDE. An important capability for any method of this type is the ability to implement a variety of boundary conditions for the governing equations. The case of Dirichlet conditions is easily handled by applying a logical mask and padding the vector of unknowns with the appropriate known boundary values. It is similarly straightforward to implement periodic boundary conditions in computational space. The generalized Neumann boundary condition,

$$\nabla_{\mathbf{x}} u \cdot \mathbf{n}_{\mathbf{x}} + \kappa\, u = g, \qquad \mathbf{x} \in \Omega_P, \tag{29}$$

where κ, g are given, requires more careful treatment. We give details for the left side of the domain and the other three sides follow a similar pattern. The normal vectors $\mathbf{n}_{\mathbf{x}}$ of Ω_P mapped from the left boundary of Ω_C are

$$\mathbf{n}_{\mathbf{x}} = \left(\frac{-y_\eta}{(x_\eta^2 + y_\eta^2)^{\frac{1}{2}}}, \frac{x_\eta}{(x_\eta^2 + y_\eta^2)^{\frac{1}{2}}} \right) = \left(\frac{-P_{\eta\eta}}{(\mathcal{A}_{11})^{\frac{1}{2}}}, \frac{P_{\xi\eta}}{(\mathcal{A}_{11})^{\frac{1}{2}}} \right). \tag{30}$$

The boundary condition (29) is transformed from physical space to computational space by calculating the normal derivative $(u_x, u_y) \cdot \mathbf{n}_{\mathbf{x}}$ via (26) to obtain

$$-\frac{\overline{J}}{(\mathcal{A}_{11})^{\frac{1}{2}}} (\mathcal{A}_{11} u_\xi + \mathcal{A}_{12} u_\eta) + \kappa\, u = g, \tag{31}$$

where $\mathcal{A}_{11}, \mathcal{A}_{12}$ are defined in (27b). We recall from (26) that the Lapalcian is

$$\Delta_{\mathbf{x}} u = \overline{J} \frac{\partial}{\partial \xi} \Big(\overline{J} (\mathcal{A}_{11} u_\xi + \mathcal{A}_{12} u_\eta) \Big) + \overline{J} \frac{\partial}{\partial \eta} \Big(\overline{J} (\mathcal{A}_{12} u_\xi + \mathcal{A}_{22} u_\eta) \Big), \qquad \boldsymbol{\xi} \in \partial\Omega_C, \tag{32}$$

where the two terms represent the normal and tangential components on the boundary, respectively. For the second term (tangential to left boundary), we substitute u_ξ from the Neumman boundary condition (31) given by

$$u_\xi = \frac{1}{\overline{J}(\mathcal{A}_{11})^{\frac{1}{2}}}(\kappa\, u - g) - \frac{\mathcal{A}_{12}}{\mathcal{A}_{11}}\, u_\eta \tag{33}$$

into $(\overline{J}(\mathcal{A}_{12}u_\xi + \mathcal{A}_{22}u_\eta))_\eta$ and discretize at noncorner points using staggered central differences (e.g., (28a)). For the first term in (32), corresponding to the normal component of the Laplacian on the left boundary, we use a one-sided approximation that invokes only internal points. The condition (31) implies that given the flux $\mathcal{F} = \overline{J}(\mathcal{A}_{11}u_\xi + \mathcal{A}_{12}u_\eta)$ and $C = (\mathcal{A}_{11})^{\frac{1}{2}}$, we have

$$\mathcal{F}_{1,j} = C_{1,j}\,[\kappa\, u - g]_{1,j}.$$

The component of (32) corresponding to the normal flux is approximated by

$$[\bar{J}\mathcal{F}_\xi]_{1,j} = \bar{J}_{1,j}\frac{-3\mathcal{F}_{1,j} + 4\mathcal{F}_{2,j} - \mathcal{F}_{3,j}}{2\Delta\xi} + \mathcal{O}(\Delta\xi^2). \tag{34}$$

In the discretization of $\mathcal{F}_{2,j}$ and $\mathcal{F}_{3,j}$ in (34), a higher order approximation of the terms u_ξ and u_η must be used to retain second order accuracy near the boundary due to differentiation of discontinuous truncation errors at near boundary points. For the four corner points, the one-sided approximation (34) is used for both terms in (32).

4.4. Pairing of PDE to evolving mesh. From an initial mesh and initial condition for the PDE, there are two main approaches to pairing the subsequent mesh and PDE dynamics—evolving them in a simultaneous or an alternating fashion. Here we discuss the merits and some implementation details of each approach.

4.4.1. Alternating solution approach. In scenarios where the PDE is evolving on a slow timescale or is very stiff, it may not be necessary to update the mesh at each time step. In such a scenario, it is efficient to only update the mesh at discrete time points with interpolation of the current solution onto the updated mesh. The PDE is in effect solved on a sequence of static meshes. A minimal working example of solving the heat equation on a moving curved domain using the alternating method is given in Appendix A.

4.4.2. Simultaneous solution approach. An advantage of using the PMA equation is that it is straightforward to couple the dynamics of the mesh and the PDE into one monolithic system. For the example of a simple reaction diffusion equation $u_t = \Delta_{\mathbf{x}}u + f(u, \mathbf{x}, t)$, this gives rise to the coupled system

$$u_t - \nabla_{\mathbf{x}}u \cdot \nabla Q_t = \Delta_{\mathbf{x}}u + f(u, \mathbf{x}, t), \quad \boldsymbol{\xi} \in \Omega_C, \quad t > 0; \tag{35a}$$

$$\tau(I - \gamma\Delta_{\boldsymbol{\xi}})Q_t = (|H(Q)|M(\mathbf{x}, t))^{\frac{1}{2}}, \quad \boldsymbol{\xi} \in \Omega_C, \quad t > 0, \tag{35b}$$

that may be solved simultaneously in time. In practice this system is solved by back substitution by first inverting (35b) with an FFT and homogeneous Neumann conditions invoked on Q_t. To avoid numerical instabilities associated with the Lagragian term [15, 30, 31], $\nabla_{\mathbf{x}}u \cdot \nabla Q_t$ is discretized by a second order upwinding scheme. The full stencils are found in Appendix C of [19]. This simultaneous approach avoids the need for interpolation since the mesh and PDE continually evolve together and is well suited to explicit time integration.

5. Examples. In this section we give a variety of examples on static and evolving geometries to demonstrate the versatility of the method. In the first example, we verify the expected convergence properties of the method and confirm second order convergence at interior points and first order convergence at boundary adjacent points, in particular at corner adjacent points. The reduction in accuracy at near boundary points arises in the solution of the Monge–Ampére system (14), which is then reflected in the PDE solution through the transformations (26). Similar first order error structure has been observed in related studies [4, 22]. To clarify the implementation details for this method, we have included in Appendix A a minimal working example for the solution of the diffusion equation moving curved domain with forward Euler integration.

5.1. Benchmark problem: Modified Stefan problem on circle with prescribed boundary motion. To confirm the expected convergence patterns of the method for time dependent PDEs, we solve the following modified Stefan problem posed on an expanding disk domain with Dirichlet boundary conditions (cf. [24]),

$$\frac{\partial u}{\partial t} - \Delta_{\mathbf{x}} u = f(\mathbf{x}, t), \quad 0 \le \|\mathbf{x}\| < \rho(t), \quad t > 0; \tag{36a}$$

$$\frac{d\rho}{dt} = -\frac{\partial u}{\partial n}, \quad \|\mathbf{x}\| = \rho(t), \quad t > 0; \tag{36b}$$

$$u(\mathbf{x}, t) = 0, \quad \|\mathbf{x}\| = \rho(t), \quad u(\mathbf{x}, 0) = J_0(r_0\|\mathbf{x}\|), \quad \rho(0) = 1. \tag{36c}$$

Here $J_0(z)$ is the zeroth order Bessel function of the first kind and r_0 is its smallest positive root. The remaining functions in (36) are defined as

$$f(\mathbf{x}, t) = \frac{a r_0^3 \beta(t)^2 \|\mathbf{x}\|}{2\sigma(t)^3} J_0'\left(\frac{r_0\|\mathbf{x}\|}{\sigma(t)}\right), \qquad \sigma(t) = \exp\left(\frac{a(\beta(t)-1)}{2}\right),$$

$$\beta(t) = \frac{1}{a}\mathrm{Ei}^{-1}(\mathrm{Ei}(a) - r_0^2 t e^a), \qquad a = \frac{2J_0'(r_0)}{r_0},$$

where $\mathrm{Ei}(z)$ is the exponential integral. This problem has the exact solution

$$u(\mathbf{x}, t) = \beta(t) J_0\left(\frac{r_0\|\mathbf{x}\|}{\sigma(t)}\right), \qquad \rho(t) = \sigma(t). \tag{37}$$

from which we calculate the absolute error

$$\mathcal{E}(t) = \|u_N(\mathbf{x}, t) - u(\mathbf{x}, t)\|_{L^\infty}, \tag{38}$$

based on a numerical solution $u_N(\mathbf{x}, t)$ of (36) from an $N \times N$ computational mesh.

The initial mesh is generated by solving (14) with uniform density $f(\boldsymbol{\xi})/g(\mathbf{x}) = 1$ on the computational mesh $\Omega_C = [-1, 1]^2$ and the prescribed boundary motion (37). The discretized system is solved using a fourth order Runge–Kutta time stepping scheme with the alternating approach in section 4.4.1. In Table 1, we show the convergence rates for the absolute error (38) for interior points and corner adjacent points. We observe that in the neighborhood of points surrounding the corners, the accuracy of the method decreases to first order.

5.2. Example: Blow-up on an elliptical domain. To illustrate the effectiveness of our methods for mesh adaptation, we solve a classical semilinear blow-up

TABLE 1
Absolute interior and corner adjacent errors $\mathcal{E}(T)$ as defined in (38) for problem (36). Solution evaluated at time $T = 0.005$ with convergence rates given in parentheses.

Mesh size ($N \times N$)	Interior error		Corner adjacent error	
$N = 40$	3.79×10^{-5}	( )	3.66×10^{-3}	( )
$N = 60$	1.64×10^{-5}	(2.0)	2.66×10^{-3}	(0.8)
$N = 80$	9.14×10^{-6}	(2.0)	2.07×10^{-3}	(0.9)
$N = 100$	5.79×10^{-6}	(2.0)	1.69×10^{-3}	(0.9)
$N = 120$	3.98×10^{-6}	(2.0)	1.42×10^{-3}	(0.9)

problem on the ellipse $\Omega_P = \{(\cos\theta, 0.75\sin\theta) \mid 0 < \theta < 2\pi\}$. The full problem statement is

$$u_t = \Delta_{\mathbf{x}} u + u^3, \quad \mathbf{x} \in \Omega_P, \quad t > 0; \tag{39a}$$

$$u(\mathbf{x}, t) = 0, \quad \mathbf{x} \in \partial\Omega_P, \quad t > 0; \tag{39b}$$

$$u(\mathbf{x}, 0) = 15\, e^{-25|\mathbf{x}|^2}, \quad \mathbf{x} \in \Omega_P. \tag{39c}$$

For this example, the monitor function must be chosen such that the mesh inherits the dynamic length scale of (39a). Assuming a small length scale $L(t)$, near singularity the mesh has the scaling $Q \sim L(t)S(\boldsymbol{\xi}, t)$. The PMA (22a) will be scale independent near the singularity under the condition (cf. [11]) that

$$M \sim \left(\frac{1}{L}\frac{dL}{dt}\right)^2.$$

In the case of power law blow-up profiles, the length scale has the form $L(t) = (t_c - t)^\alpha$, where t_c is the singularity time. This dictates that M should scale like

$$M \sim \frac{1}{(t_c - t)^2}$$

to ensure that the PMA is independent of length scale near the singularity. The invariance of the governing equation (39a) under the scaling

$$t \to \lambda t, \qquad \mathbf{x} \to \lambda^{\frac{1}{2}}\mathbf{x}, \qquad u \to \lambda^{-\frac{1}{2}} u \tag{40}$$

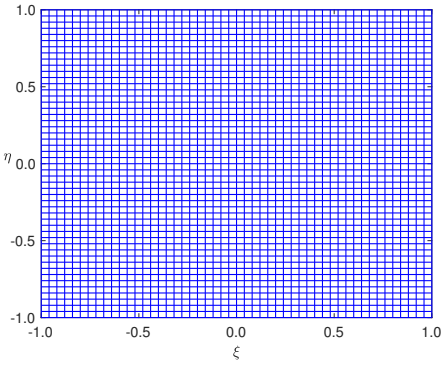
motivates the choice $M(\mathbf{x}, t) = u^4$ so that the mesh inherits the scale invariance of (39) near the singularity. This choice of $M(\mathbf{x}, t)$ combined with the integral averaging (23) clusters mesh points at the singularity while maintaining the known self-similar scaling structure associated with the blow-up profile of the PDE [11]. In addition, we use an adaptive time step determined by the Sundman transformation [9, 13],

$$\frac{dt}{dT} = g(u). \tag{41}$$

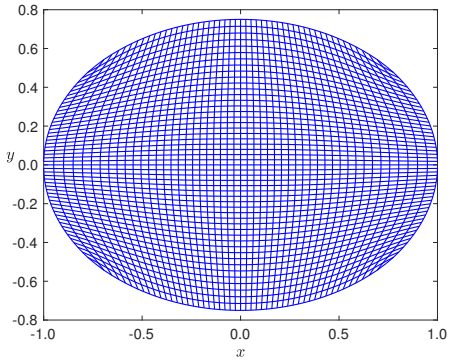
For a fixed dT, (41) dynamically adapts the time step dt so as not to overshoot the singularity time t_c. Incorporating the scalings of (40), we set $g(u) = (\|u\|_{L^\infty(\Omega_P)})^{-2}$.

The nonlinear Monge–Ampére equation (14) is solved with a uniform density and signed distance boundary condition to generate an initial uniform mesh on the ellipse. The PMA (22a) is paired to the PDE (39) using a Lagrangian coupling term to give a monolithic system, where the adaptive time step (41) is incorporated as an additional

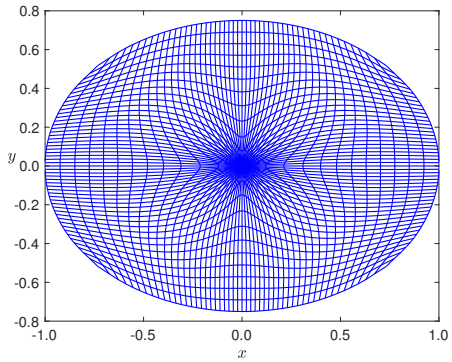
equation. The monolithic system of equations is solved explicitly using RK4 time stepping. As seen in Figure 2, we observe good regularity and retention of scaling behavior of the mesh near blow-up. Once again, we have the added benefit of being able to resolve the singularity to approximately $\|u\|_{L^\infty(\Omega_P)} = 10^6$ while using a fixed number of mesh points and standard finite difference stencils on a fixed, uniform, rectangular domain.



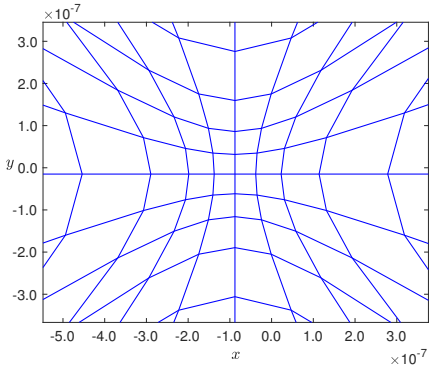
(a) Uniform 51×51 computational mesh.



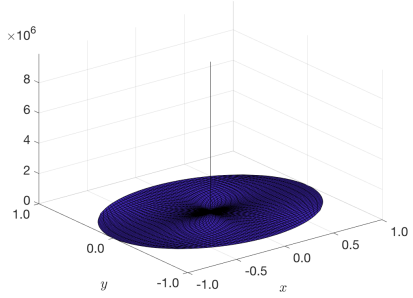
(b) Ellipse generated from solving (14) with boundary condition (19).



(c) Final mesh near blow-up.



(d) Zoomed mesh near blow-up.



(e) Final solution at $\|u\|_{L^\infty(\Omega_P)} = 10^6$.

FIG. 2. *Solution and corresponding meshes for solving the semilinear blow-up problem* (39) *on an ellipse. Mesh resolves well near the singularity point and shows good regularity when zoomed in near the singularity.*

5.3. Example: Sharp interfaces in a regularized model of microelectromechanical systems deflections. To show the ability of the adaptive method to handle propagating interfaces, we test it on a capacitor structure model from microelectromechanical systems (MEMS). The full model, including derivation, posttouchdown interface dynamics, and analysis of equilibrium configurations, can be found in [32, 33, 34]. The system considered is the parabolic, semilinear PDE

$$u_t = \Delta_{\mathbf{x}} u - \lambda \left[\frac{1}{(1+u)^2} - \frac{\varepsilon^2}{(1+u)^4} \right], \quad \mathbf{x} \in \Omega_P, \quad t > 0; \tag{42a}$$

$$u(\mathbf{x}, t) = 0, \quad \mathbf{x} \in \partial\Omega_P; \tag{42b}$$

$$u(\mathbf{x}, 0) = 0.5(1-\varepsilon)\big(\tanh[40(x^2 + (y+0.8)^2 - 0.1(0.6)^2)] - 1\big), \quad \mathbf{x} \in \Omega_P. \tag{42c}$$

In (42), the parameter $0 < \varepsilon \ll 1$ reflects the extent of an insulating layer placed between two elastic plates. The positive parameter λ quantifies the relative importance between the electric and electrostatic forces of the system. The behavior of the system (42) is characterized by a dynamic sharp interface which propagates outward from the center of the domain until eventually it is pinned at the boundary. In this example, we take Ω_P as the nonconvex limacon region whose boundary is given by

$$(x, y) = r(\theta)(\cos\theta, \sin\theta), \quad \theta \in (0, 2\pi), \qquad r(\theta) = 1.3 + 0.9\cos\theta.$$

For the parameters $\varepsilon = 0.05$ and $\lambda = 10$, we simulate (42) with the arc-length monitor function $M(\mathbf{x}, t) = \sqrt{1 + u_x^2 + u_y^2}$ and set $\alpha = 0.75$ in the Mackenzie regularization (23). The discretized mesh and PDE system is integrated simultaneously with the Lagrangian coupling term and using RK4 time integration until the steady state configuration is reached. Figure 3 gives snapshots of the solution and the mesh. In Figure 3 we see the mesh conforming to the nonconvex limacon shape and also dynamically tracking the movement of the interface as its perimeter increases over time. In addition, mesh density is decreased in the region behind the interface where less resolution is required.

5.4. Example: Heat equation on a splitting domain. As a more challenging example, we solve the heat equation with a constant source term and homogeneous Dirichlet boundary conditions on an evolving domain which undergoes a split. The PDE considered is

$$u_t = \Delta_{\mathbf{x}} u + 1, \quad \mathbf{x} \in \Omega_P, \quad t > 0; \tag{43a}$$

$$u(\mathbf{x}, t) = 0, \quad \mathbf{x} \in \partial\Omega_P, \quad t > 0; \tag{43b}$$

$$u(\mathbf{x}, 0) = 5\left(e^{-((\frac{x-0.5}{0.2})^2 + (\frac{y}{0.25})^2)} + e^{-((\frac{x+0.5}{0.2})^2 + (\frac{y}{0.25})^2)} \right), \quad \mathbf{x} \in \Omega_P. \tag{43c}$$

The domain starts as a single convex domain that evolves into a connected nonconvex domain given by $(x, y) = (r_c(\theta)\cos\theta, r_c(\theta)\sin\theta)$, until it pinches off into two convex dynamic domains $(x_1, y_1) = (-x_c(t) + r_{d_1}(\theta, t)\cos\theta, r_{d_1}(\theta, t)\sin\theta), (x_2, y_2) = (x_c(t) + r_{d_2}(\theta, t)\cos\theta, r_{d_2}(\theta, t)\sin\theta)$ moving in opposite directions with prescribed boundary motion.

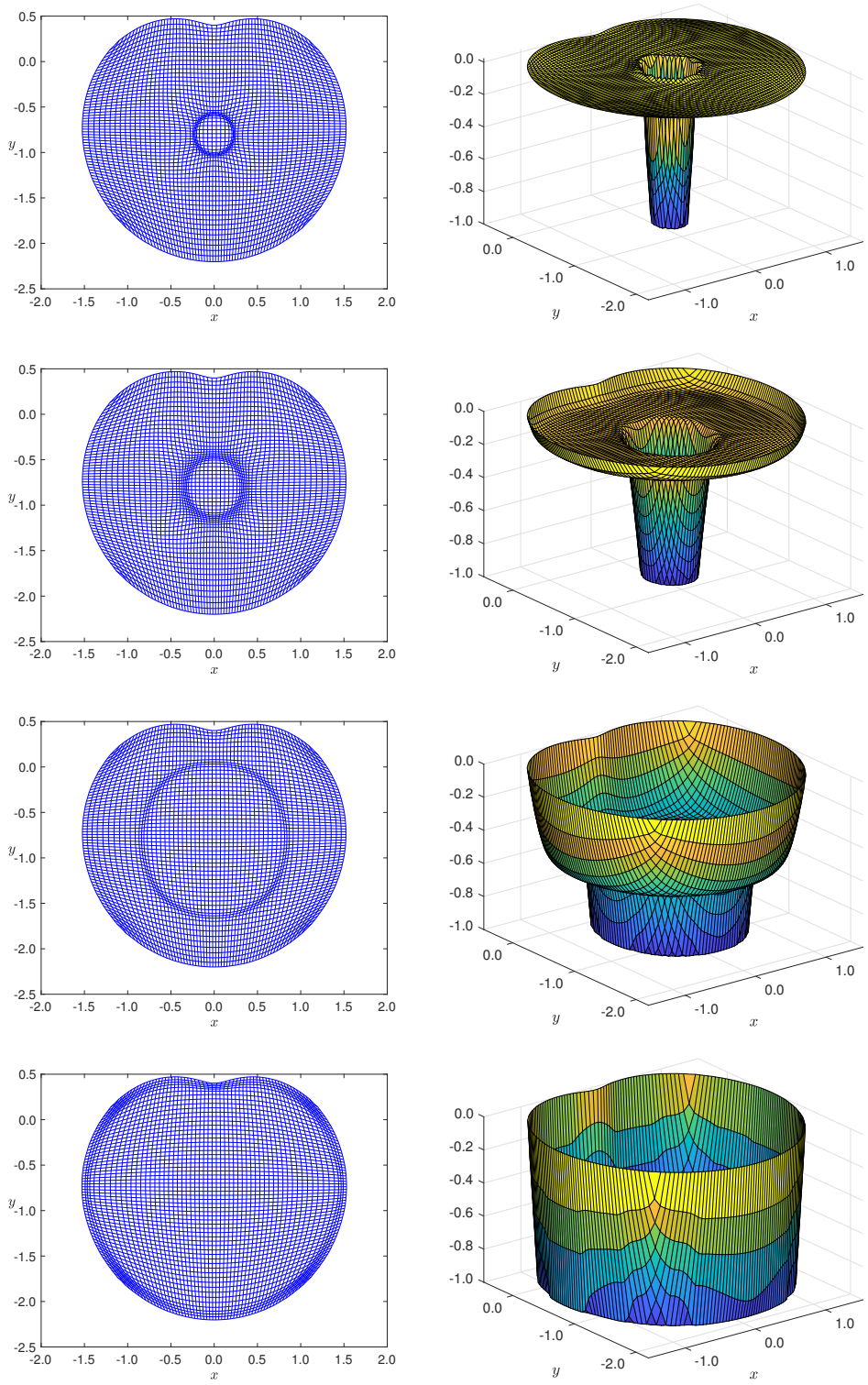


FIG. 3. *Mesh and PDE solution for the regularized MEMS equation* (42) *on a curved nonconvex domain.*

$$r_c(\theta) = \sqrt{1 - \nu \sin^2 \theta}, \quad \nu \in (0.5, 0.92), \quad \theta \in (0, 2\pi), \tag{43d}$$

$$r_{d_1}(\theta, t) = 0.5 + 0.05(0.15 \sin(4\theta) + 0.15 \cos(3\theta - 48\pi t)), \quad \theta \in (0, 2\pi), \tag{43e}$$

$$r_{d_2}(\theta, t) = 0.5 + 0.05(0.15 \sin(4\theta) + 0.1 \cos(2\theta - 36\pi t)), \quad \theta \in (0, 2\pi). \tag{43f}$$

The function $x_c(t)$ is the center of the convex shapes after the split and chosen to move the two domains slowly apart. The target domain is updated and remeshed every 100 time steps. In the numerical solution of this problem, we adopt the alternating approach in section 4.4.1 with an RK4 scheme. In the regime before the split, the fixed time step is $\mathrm{d}t = 5^{-7}$, which changes to $\mathrm{d}t = 5^{-5}$ after splitting. A smaller time step is required in the nonconvex regime as the mesh is highly refined near the splitting point.

The simulations are initialized with the single rectangular computational domain $\Omega_C = [-2, 2] \times [-1, 1]$. When the connected domain splits into two, each half of the computational domain becomes responsible for one of the target domains, i.e., $\Omega_{C_1} = [-2, 0] \times [-1, 1]$, $\Omega_{C_2} = [0, 2] \times [-1, 1]$. This example illustrates the ability of the method to handle a variety of domains, ranging from convex and simply connected to nonconvex to convex and disconnected, with all computations performed on a fixed rectangular grid. In Figures 4 and 5, we show the mesh and the PDE solution before and after the split, respectively.

For the mesh to properly resolve the nonconvex target boundary close to splitting, as seen in Figure 4, additional density is required near high curvature sections. This is accomplished by increasing the physical domain density function $g(\nabla P)$ in (14) which is described by a Gaussian centered around the high curvature section near the splitting point.

5.5. Example: Spot splitting in the singularly perturbed Schnakenburg reaction diffusion system. To illustrate the application of the Neumann boundary conditions derived in section 4.3.1, we consider the singularly perturbed Schnakenburg reaction diffusion system on the unit disk $\Omega_P = \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\| < 1\}$,

$$v_t = \varepsilon^2 \Delta_{\mathbf{x}} v - v + uv^2 \quad \text{in} \quad \Omega_P, \qquad \partial_n v = 0 \quad \text{on} \quad \partial\Omega_P; \tag{44a}$$

$$\varepsilon^2 u_t = \Delta_{\mathbf{x}} u + A - \frac{uv^2}{\varepsilon^2} \quad \text{in} \quad \Omega_P, \qquad \partial_n u = 0 \quad \text{on} \quad \partial\Omega_P. \tag{44b}$$

The singularly perturbed limit $\varepsilon \ll 1$ of the Schnakenburg system (44) gives rise to a wide range of complex localized spike dynamics [29]. The particular parameters $A = 8.8$, $\varepsilon = 0.03$, lie in the spot splitting regime where a localized spot undergoes slow motion before dividing into two smaller spots.

In this example, we initialize the simulation with initial conditions (v_0, u_0) corresponding to a single spot, determined from a radially symmetric quasi–steady state profile found by asymptotic reduction in the limit $\varepsilon \to 0$ [29]. For the chosen parameters values, the resulting problem is stiff and the alternating solution approach in section 4.4.1 is taken with an implicit Crank–Nicholson integration scheme and fixed time step $\mathrm{d}t = 0.05$. At each time step, the nonlinear system is solved using a damped Newton's method [2]. For the mesh equation, the modified arc-length monitor function $M(\mathbf{x}, t) = \sqrt{1 + 0.95(u_x^2 + u_y^2)}$ is chosen with $\tau = \gamma = 0.2$ taken as the speed and smoothing parameters in the PMA (22). The solution evolves on a slow timescale and we minimize interpolation costs by updating the mesh each second (every 20th time step).

In Figure 6 we see the method accurately resolving the mesh around the spot as it transits the domain and through the bifurcation point at which the spot splits in

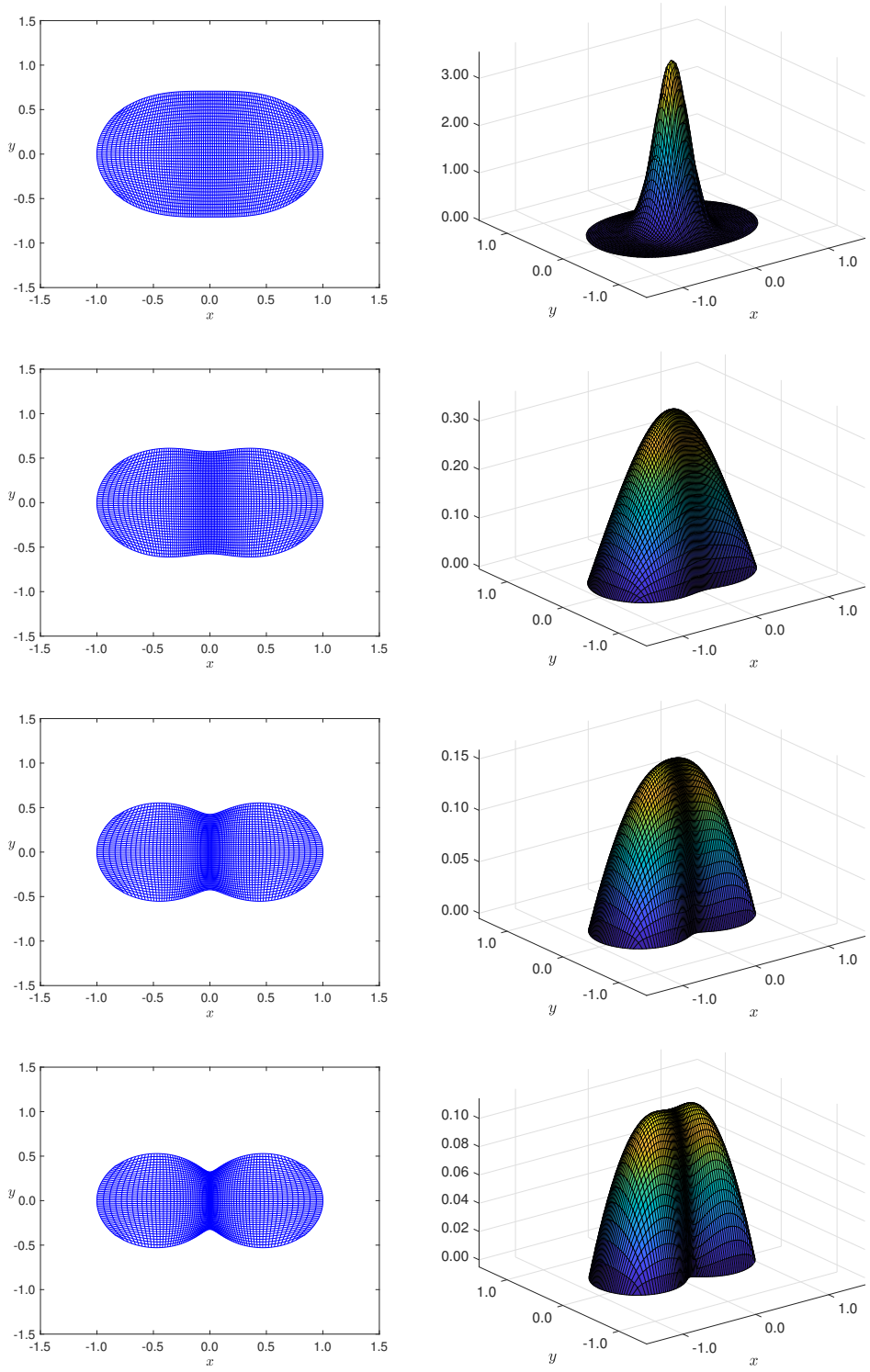


FIG. 4. *Heat equation with a constant source on the connected domain* (43d) *for set of discrete time points. The final figure corresponds to the parameter value* $\nu = 0.92$*, at which point the domains are split.*

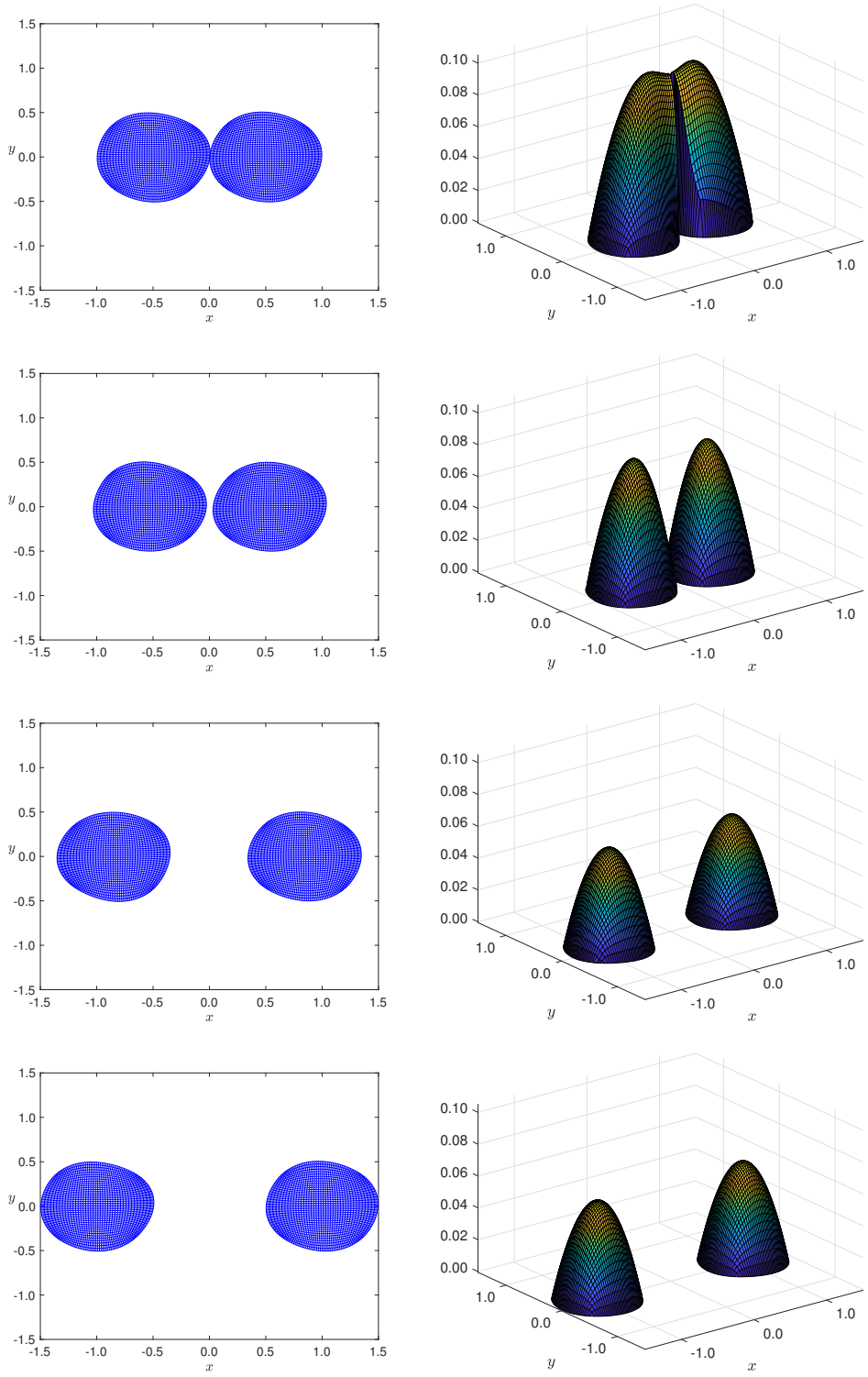


FIG. 5. *Heat equation with a constant source from the splitting point on the connected domain* (43d) *into the moving disconnected domains for a set of discrete time points.*

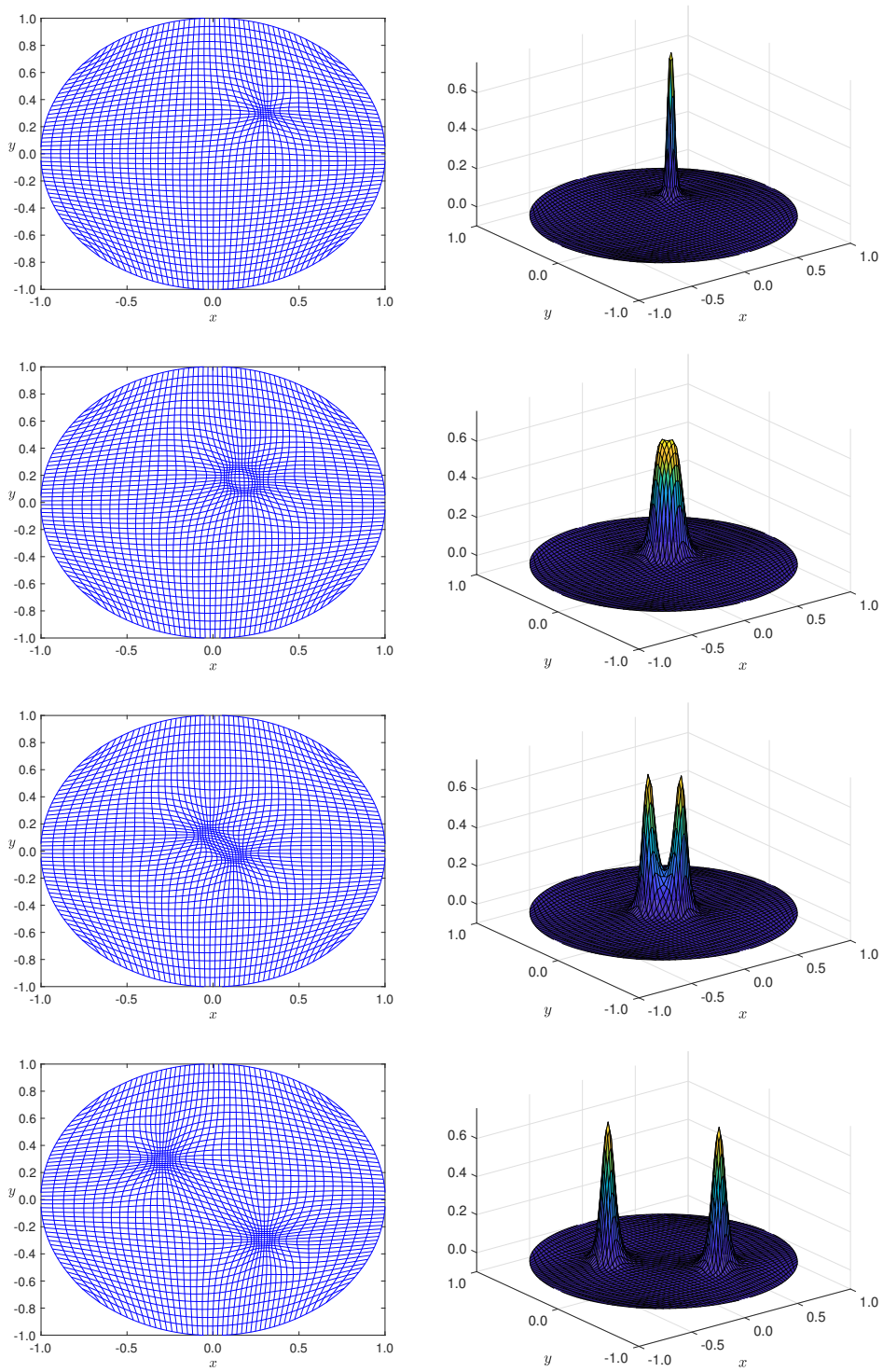


FIG. 6. *Solution to the Schnakenburg problem* (44) *on the unit disk with corresponding meshes. The mesh accurately resolves the spot as it moves toward the center, adapts to the splitting event, and resolves the separate spots as they move apart perpendicularly.*

two. This example illustrates the implementation of the method in a system PDE problem with Neumann boundary conditions.

6. Conclusion and future work. This paper has successfully developed a method for mesh generation and adaptation on curved domains using the Monge–Ampére equation and finite difference discretization. The methods presented in this paper provide a connection between elliptic grid generation, optimal transport theory, and r-adaptation methods applied to the solution of PDEs with fine scale solution behavior. The methods have been validated on a variety of PDEs, domains, and boundary conditions. We also provide example MATLAB code in Appendix A, which gives implementation details of some of the methods presented in this paper.

The advantages of this method are that it can accurately and efficiently resolve rapidly varying and fine scale PDE behavior while retaining the simplicity of performing all calculations on a fixed rectangular mesh. For convex target sets this method is robust and the user need only specify a cloud of points that lie on the boundary of the domain. In certain nonconvex regions we have shown that the method can be effective too. For challenging cases in which the curvature of the boundary is large, for example, the limiting case for the splitting domain problem in section 5.4, the mesh may experience tangling near the boundary. This can be mitigated by using a denser set of points to describe the boundary in that section. One avenue for future work would investigate the ways in which the numerical method breaks down at these points and develop more robust analysis of the limitations of the method on nonconvex domains.

The extension of optimal transport theory and the robust solution of the Monge–Ampére equation in nonconvex regions is a challenging open problem. For the adaptive solution of PDEs in complex domains with nonsimply connected features, MMPDE methods based on finite element discretization are effective for complex problems [18, 36]. Future work would also focus on creating a more robust numerical method for solving the nonlinear Monge–Ampére equation with dynamic PDE solution dependent densities in higher dimensional scenarios.

Appendix A. MATLAB example code. This is a minimal working MATLAB code of the method for the solution of the heat equation with a source term on a moving, curved domain. Mesh density in the target domain is increased in the region about the origin. For simplicity of exposition, a fixed step forward Euler time integration method is performed. For clarity of exposition, many robustness features (e.g., damped Newton method, mesh smoothing, adaptive time stepping) are omitted. Source code featuring some examples from this paper can be found at https://github.com/kdipiet10/ParabolicMongeAmpere_2DMovingMesh.

```
function HeatEquationMovingDisk
%% Solution of heat equation on a dynamic curved domain.
%% This is a minimal working example edited for brevity.

% Square Computaional Domain [-1,1]^2 with n x n mesh.
O.n = 40; O.h = 2/(O.n-1); [O.x1,O.x2] = ndgrid(-1:O.h:1);
%% Indices for the boundaries.
Ib.A=find(O.x1==-1 | O.x1==1 | O.x2==-1 | O.x2==1);
Ib.L = find(O.x1==-1); Ib.R = find(O.x1==1);
Ib.B = find(O.x2==-1); Ib.T = find(O.x2==1);

%% Finite Difference matrices for the computational.
M = make_D_matrices(O.n,O.h);
```

```
%% start and end time, timestep, PDE initial condidition.
t=0; tfinal=5; dt=1e-4; uinit=@(x,y) exp(-25*(x.^2+y.^2));
%% Represent the target set as a discrete number of points
NX=2*O.n; th=linspace(0,2*pi,NX)'; %discrete target domain
r1=@(s,t) 1.25+0.2*(0.15*sin(4*s)+0.2*cos(3*(s-30*pi*t)));
r2=@(s,t) 0.85*r1(s,t);
%% Vector representation of the target domain
Y = [r1(th,t).*cos(th) r2(th,t).*sin(th)];
%% Initial uniform mesh for computational domain
Q = 0.5*(O.x1(:).^2 + O.x2(:).^2);
%% Admissable directions for signed distance function.
int=2*pi*(1:NX)/NX; nj = [cos(int)' sin(int)']; %Eqn 18
%% Legendre fenchel transform for for signed distance
Hs = max(Y*nj')'; %Eqn 19b
%% Mesh density functions and their derivatives
k=10; O.g=@(x,y) 1+k*exp(-k*(x.^2+y.^2));
O.f=@(x,y)1+0*x; O.gx=@(x,y)-2*k^2*x.*exp(-k*(x.^2+y.^2));
O.gy=@(x,y) -2*k^2*y.*exp(-k*(x.^2+y.^2));
%% Get initial mesh by solving MA (eqn 14) with Newton
err = 10; newton_tol = 1e-6;
while ( err > newton_tol )
   [F,J]=rhs_MA(Q,Hs,nj,Ib,M,O); Q=Q-J\F; err=norm(F);
end
%%Intial condition for the PDE
x0=M.D1XC*Q; y0=M.D1YC*Q; u0=uinit(x0,y0); close all;
figure('units','normalized','position',[0 0 1 0.5]);
%% Alternating solution method Sec 4.4.1
while t < tfinal
    un = u0+dt*rhs_PDE(t,u0,Q,M,Ib); t=t+dt; % Euler Step
    %Update the domain boundary with at new time.
    Y = [r1(th,t).*cos(th) r2(th,t).*sin(th)];
    %Remap the mesh - solve MA system with Newton method.
    err = 10; Hs = max(Y*nj')';
    while (err > newton_tol)
       [F,J]=rhs_MA(Q,Hs,nj,Ib,M,O); Q=Q-J\F; err=norm(F);
    end
    xn = M.D1XC*Q; yn = M.D1YC*Q;
    %% Interpolate onto new mesh and apply Dirichlet BC.
    u0=griddata(x0,y0,un,xn,yn,'cubic'); u0(Ib.A)=0;
    %% Plot the mesh and the pde solution
    xx1 = reshape(xn,O.n,O.n); yy1 = reshape(yn,O.n,O.n);
    subplot(1,2,1), plot(xx1,yy1,'b',xx1',yy1','b');
    subplot(1,2,2), surf(xx1,yy1,reshape(u0,O.n,O.n));
    x0 = xn; y0 = yn;
    drawnow;
end

function dudt = rhs_PDE(t,u,Q,M,Ib)
%% The right hand side of the PDE.
% Derivatives of Q - appendix A of [19]
```

```
Qx = M.D1XC*Q; Qy = M.D1YC*Q;
f=zeros(size(Q)); f(Ib.L)=-Qx(Ib.L); f(Ib.R)=Qx(Ib.R);
g=zeros(size(Q)); g(Ib.B)=-Qy(Ib.B); g(Ib.T)=Qy(Ib.T);
Q2xi = M.D2XXN*Q + (25/(6*M.h))*f;
Q2eta = M.D2YYN*Q + (25/(6*M.h))*g; Q2xieta = M.D2XY4*Q;
%% Laplacian in computational coordinates, Sec 4.3.
J = Q2eta.*Q2xi-Q2xieta.^2; invJ = 1./J;
A11 = reshape(invJ.*(Q2xieta.^2 + Q2eta.^2),M.n,M.n);
A21 = reshape(-invJ.*(Q2xieta.* (Q2xi+ Q2eta)),M.n,M.n);
A22 = reshape(invJ.*(Q2xieta.^2 + Q2xi.^2),M.n,M.n);

u = reshape(u,M.n,M.n); v = zeros(M.n,M.n);
i = 2:M.n-1; ip = i + 1; im = i - 1;
% d/xi (A11 d/xi) Equation (28a)
v(i,:) = v(i,:) + ((A11(ip,:) + A11(i,:)).* u(ip,:) ...
    - (A11(ip,:) + 2*A11(i,:) + A11(im,:)).* u(i,:) ...
    + (A11(i,:) + A11(im,:)).*u(im,:))/2;
% d/xi (A21 d/eta) Equation (28b)
v(i,i) = v(i,i) + (A21(i,ip).*(u(ip,ip)-u(im,ip)) ...
       - A21(i,im).*(u(ip,im)-u(im,im)))/4;
% d/eta (A21 d/xi) Equation (28b)
v(i,i) = v(i,i) + (A21(ip,i).*(u(ip,ip)-u(ip,im)) ...
       - A21(im,i).*(u(im,ip)-u(im,im)))/4;
% d/eta (A22 d/eta) Equation (28a)
v(:,i) = v(:,i) + ((A22(:,ip) + A22(:,i)).* u(:,ip) ...
    - (A22(:,ip) + 2*A22(:,i) + A22(:,im)).* u(:,i) ...
    + (A22(:,i)+A22(:,im)).*u(:,im))/2;
Lapu = invJ.*v(:)/(M.h^2);
%% Heat equation with a constant source term.
dudt = Lapu + 1;

function [F,J] = rhs_MA(Q,Hs,nj,Ib,M,O)
%Monge-Ampere equation and its Jacobian - section 4.1.
n = M.n; ns = n^2; Q0 = Q(ceil(ns/2));
%% Interior Points - Derivatives in eq (17).
Qx=M.D1XC*Q; Qy=M.D1YC*Q; Qxx=M.D2XX*Q; Qyy=M.D2YY*Q;
Qxy = M.D2XY*Q;

f=O.f(O.x1(:),O.x2(:)); g=O.g(Qx,Qy); gx=O.gx(Qx,Qy);
gy=O.gy(Qx,Qy); Gx = -gx.*f./(g.^2);  Gy = -gy.*f./(g.^2);
Z = sparse(n^2,n^2);  Z(:,ceil((n^2)/2)) = 1;
%% Section 4.1, Eqns 14a, 21a
F = Qxx .* Qyy - Qxy.^2 - Q0 - f./g;
J = spdiags(Qxx,0,ns,ns)*M.D2YY...
  + spdiags(Qyy,0,ns,ns)*M.D2XX...
  - 2*spdiags(Qxy,0,ns,ns)*M.D2XY...
  - spdiags(Gx,0,ns,ns)*M.D1XC ...
  - spdiags(Gy,0,ns,ns)*M.D1YC - Z;
%% Hamilton Jacboi boundary condition H=0.
H = zeros(ns,1); n1 = zeros(ns,1); n2 = n1;
```

```
% Forward and backward first derivatives.
Qxm=M.D1XM*Q; Qxp=M.D1XP*Q; Qym=M.D1YM*Q; Qyp=M.D1YP*Q;
% Boundary Condition - Eqn 19a
[H(Ib.A),iC]=max(max(nj(:,1),0)*Qxm(Ib.A)'+...
 +min(nj(:,1),0)*Qxp(Ib.A)'+max(nj(:,2),0)*Qym(Ib.A)'...
 +min(nj(:,2),0)*Qyp(Ib.A)'-repmat(Hs,[1,length(Ib.A)]));
%% Jacobian - Eqn 21b
n1(Ib.A)=nj(iC,1); n2(Ib.A)=nj(iC,2);
Jb = spdiags(max(n1,0),0,ns,ns)*M.D1XM...
   + spdiags(min(n1,0),0,ns,ns)*M.D1XP...
   + spdiags(max(n2,0),0,ns,ns)*M.D1YM...
   + spdiags(min(n2,0),0,ns,ns)*M.D1YP;
F(Ib.A) = H(Ib.A); J(Ib.A,:) = Jb(Ib.A,:);

function M = make_D_matrices(n,h)
%% Finite difference derivative matrices
e = ones(n,1); I = speye(n); M.n = n; M.h = h;
%Forward, backward and central differences for first
%derivatives.
DP = spdiags([-3*e 4*e -e],[0 1 2],n,n);
DP(n-1,n-2) = 0; DP(n-1,n-1) = -2; DP(n-1,n) = 2;
DP(n,n) = 3; DP(n,n-1)=-4; DP(n,n-2) = 1;
M.D1XP = kron(I,DP/(2*h));  M.D1YP = kron(DP/(2*h),I);

DM = spdiags([e -4*e 3*e],[-2 -1 0],n,n);
DM(1,1) = -3; DM(1,2) = 4; DM(1,3) = -1;
DM(2,1) = -2; DM(2,2) = 2;
M.D1XM = kron(I,DM/(2*h));  M.D1YM = kron(DM/(2*h),I);

DC = spdiags([-e 0*e e],-1:1,n,n);
DC(1,1) = -3; DC(1,2) = 4; DC(1,3) = -1;
DC(n,n) = 3; DC(n,n-1) = -4; DC(n,n-2) = 1;
D2 = spdiags([e -2*e e],-1:1,n,n)/(h^2);
M.D2XX = kron(I,D2);  M.D2YY = kron(D2,I);
M.D2XY = kron(DC/(2*h),DC/(2*h));
%% Fourth Order derivatives for mesh equation.
%% See appendix A of [19]
ND24 = spdiags([-e 16*e -30*e 16*e -e],-2:2,n,n);
ND24(1,1)=-415/6;ND24(1,2)=96;ND24(1,3)=-36;
ND24(1,4)=32/3;ND24(1,5)=-3/2;
ND24(2,1)= 10;ND24(2,2)=-15;ND24(2,3)=-4;ND24(2,4)=14;
ND24(2,5)=-6; ND24(2,6) = 1;
ND24(n,n)=-415/6;ND24(n,n-1)=96;ND24(n,n-2)=-36;
ND24(n,n-3)=32/3;ND24(n,n-4)=-3/2;
ND24(n-1,n) = 10; ND24(n-1,n-1) = -15; ND24(n-1,n-2) = -4;
ND24(n-1,n-3)=14;ND24(n-1,n-4)=-6; ND24(n-1,n-5) = 1;
M.D2XXN = kron(I,ND24/(12*h^2));
M.D2YYN = kron(ND24/(12*h^2),I);
%% Fourth order differentiation matrices (for mesh).
DC4 = spdiags([e -8*e 0*e 8*e -e],-2:2,n,n);
```

```
DC4(1,1) = -25; DC4(1,2) = 48; DC4(1,3) = -36;
DC4(1,4) = 16; DC4(1,5)= -3;
DC4(2,1) = -3; DC4(2,2) = -10; DC4(2,3) = 18;
DC4(2,4) = -6; DC4(2,5) = 1;
DC4(n-1,n) = 3; DC4(n-1,n-1) = 10; DC4(n-1,n-2) = -18;
DC4(n-1,n-3) = 6; DC4(n-1,n-4) = -1;
DC4(n,n) = 25; DC4(n,n-1) = -48; DC4(n,n-2) = 36;
DC4(n,n-3)= -16; DC4(n,n-4)= 3;
DC4 = DC4/(12*h);
M.D2XY4=kron(DC4,DC4); M.D1XC=kron(I,DC4);
M.D1YC=kron(DC4,I);
```

REFERENCES

[1] S. Acosta and V. Villamizar, *Finite difference on grids with nearly uniform cell area and line spacing for the wave equation on complex domains*, J. Comput. Appl. Math., 234 (2010), pp. 1970–1979.

[2] U. M. Ascher, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Classics in Appl. Math. 13, SIAM, Philadelphia, 1995.

[3] G. Beckett and J. A. Mackenzie, *Convergence analysis of finite difference approximations on equidistributed grids to a singularly perturbed boundary value problem*, Appl. Numer. Math., 35 (2000), pp. 87–109, http://dx.doi.org/10.1016/S0168-9274(99)00065-3.

[4] J.-D. Benamou, B. D. Froese, and A. M. Oberman, *Numerical solution of the optimal transportation problem using the Monge–Ampére equation*, J. Comput. Phys., 260 (2014), pp. 107–126, https://doi.org/10.1016/j.jcp.2013.12.015.

[5] J. Benito, F. Urea, and L. Gavete, *Influence of several factors in the generalized finite difference method*, Appl. Math. Model., 25 (2001), pp. 1039–1053, https://doi.org/10.1016/S0307-904X(01)00029-4.

[6] J. Benito, F. Urea, and L. Gavete, *Solving parabolic and hyperbolic equations by the generalized finite difference method*, J. Comput. Appl. Math., 209 (2007), pp. 208–233, https://doi.org/10.1016/j.cam.2006.10.090.

[7] Y. Brenier, *Polar factorization and monotone rearrangement of vector-valued functions*, Commun. Pure Appl. Math., 44 (1991), pp. 375–417, https://dx.doi.org/10.1002/cpa.3160440402.

[8] 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 (2012).

[9] C. Budd, B. Leimkuhler, and M. Piggott, *Scaling invariance and adaptivity*, Appl. Numer. Math., 39 (2001), pp. 261–288, https://doi.org/10.1016/S0168-9274(00)00036-2.

[10] C. J. Budd, W. Huang, and R. D. Russell, *Adaptivity with moving grids*, Acta Numer., 18 (2009), pp. 111–241, https://doi.org/10.1017/S0962492906400015.

[11] C. J. Budd and J. F. Williams, *Parabolic monge–ampére methods for blow-up problems in several spatial dimensions*, J. Phys. A, 39 (2006), https://stacks.iop.org/0305-4470/39/i=19/a=S06.

[12] C. J. Budd and J. F. Williams, *Moving mesh generation using the parabolic monge–ampére equation*, SIAM J. Sci. Comput., 31 (2009), pp. 3438–3465, https://doi.org/10.1137/080716773.

[13] C. J. Budd and J. F. Williams, *How to adaptively resolve evolutionary singularities in differential equations with symmetry*, J. Engrg. Math., 66 (2010), pp. 217–236, https://doi.org/10.1007/s10665-009-9343-6.

[14] W. Cao, W. Huang, and R. D. Russell, *A study of monitor functions for two-dimensional adaptive mesh generation*, SIAM J. Sci. Comput., 20 (1999), pp. 1978–1994, https://doi.org/10.1137/S1064827597327656.

[15] H. Ceniceros, *A semi-implicit moving mesh method for the focusing nonlinear Schrödinger equation*, Commun. Pure Appl. Anal., 1 (2002), pp. 1–18.

[16] H. D. Ceniceros and T. Y. Hou, *An efficient dynamically adaptive mesh for potentially singular solutions*, J. Comput. Phys., 172 (2001), pp. 609–639, http://dx.doi.org/10.1006/jcph.2001.6844.

[17] C. de Boor, *Good approximation by splines with variable knots,* II, in Lecture Notes in Math. 363, Springer, Berlin, 1974, pp. 12–20, http://dx.doi.org/10.1007/BFb0069121.

[18] K. L. DiPietro, R. D. Haynes, W. Huang, A. E. Lindsay, and Y. Yu, *Moving mesh simulation of contact sets in two dimensional models of elasticelectrostatic deflection problems*, J. Comput. Phys., 375 (2018), pp. 763–782, https://doi.org/10.1016/j.jcp.2018.08.053.

[19] K. L. DiPietro and A. E. Lindsay, *Monge ampére simulation of fourth order pdes in two dimensions with application to elastic-electrostatic contact problems*, J. Comput. Phys., 349 (2017), pp. 328–350, https://doi.org/10.1016/j.jcp.2017.08.032.

[20] J. H. Ellison, C. A. Hall, and T. A. Porsching, *An unconditionally stable convergent finite difference method for Navier–Stokes problems on curved domains*, SIAM J. Numer. Anal., 24 (1987), pp. 1233–1248.

[21] J. M. Finn, G. L. Delzanno, and L. Chacón, *Grid generation and adaptation by monge-kantorovich optimization in two and three dimensions*, in Proceedings of the 17th International Meshing Roundtable, R. V. Garimella, ed., Springer, Berlin, 2008, pp. 551–568.

[22] B. D. Froese, *A numerical method for the elliptic Monge–Ampére equation with transport boundary conditions*, SIAM J. Sci. Comput., 34 (2012), pp. A1432–A1459, https://doi.org/10.1137/110822372.

[23] B. D. Froese and A. M. Oberman, *Convergent filtered schemes for the Monge–Ampére partial differential equation*, SIAM J. Numer. Anal., 51 (2013), pp. 423–444, https://doi.org/10.1137/120875065.

[24] E. S. Gawlik and A. J. Lew, *High-order finite element methods for moving boundary problems with prescribed boundary evolution*, Comput. Methods Appl. Mech. Engrg., 278 (2014), pp. 314–346, https://doi.org/10.1016/j.cma.2014.05.008.

[25] R. Holland, *Finite-difference solution of Maxwell's equations in generalized nonorthogonal coordinates*, IEEE Trans. Nuclear Sci., 30 (1983), pp. 4589–4591.

[26] W. Huang, *Measuring mesh qualities and application to variational mesh adaptation*, SIAM J. Sci. Comput., 26 (2005), pp. 1643–1666, https://doi.org/10.1137/S1064827503429405.

[27] W. Huang and R. D. Russell, *A moving collocation method for solving time dependent partial differential equations*, Appl. Numer. Math., 20 (1996), pp. 101–116, https://doi.org/10.1016/0168-9274(95)00119-0.

[28] W. Huang and R. D. Russell, *Adaptive mesh movement—the MMPDE approach and its applications*, J. Comput. Appl. Math., 128 (2001), pp. 383–398, http://dx.doi.org/10.1016/S0377-0427(00)00520-3.

[29] T. Kolokolnikov, M. Ward, and J. Wei, *Spot self-replication and dynamics for the Schnakenburg model in a two-dimensional domain*, J. Nonlinear Sci., 19 (2009), pp. 1–56.

[30] S. Li and L. Petzold, *Moving mesh methods with upwinding schemes for time-dependent pdes*, J. Comput. Phys., 131 (1997), pp. 368–377.

[31] S. Li, L. Petzold, and Y. Ren, *Stability of moving mesh systems of partial differential equations*, SIAM J. Sci. Comput., 20 (1998), pp. 719–738.

[32] A. Lindsay, J. Lega, and K. Glasner, *Regularized model of post-touchdown configurations in electrostatic MEMS: Equilibrium analysis*, Phys. D, 280–281 (2014), pp. 95–108, http://dx.doi.org/10.1016/j.physd.2014.04.007.

[33] A. E. Lindsay, *Regularized model of post-touchdown configurations in electrostatic MEMS: bistability analysis*, J. Engrg. Math., 99 (2016), pp. 65–77, https://dx.doi.org/10.1007/s10665-015-9820-z.

[34] A. E. Lindsay, J. Lega, and K. B. Glasner, *Regularized model of post-touchdown configurations in electrostatic MEMS: interface dynamics*, IMA J. Appl. Math., 80 (2015), pp. 1635–1663, https://doi.org/10.1093/imamat/hxv011.

[35] T. Liszka and J. Orkisz, *The finite difference method at arbitrary irregular grids and its application in applied mechanics*, Computers and Structures, 11 (1980), pp. 83–95.

[36] C. Ngo and W. Huang, *A study on moving mesh finite element solution of the porous medium equation*, J. Comput. Phys., 331 (2017), pp. 357–380, https://doi.org/10.1016/j.jcp.2016.11.045.

[37] A. Oberman and I. Zwiers, *Adaptive finite difference methods for nonlinear elliptic and parabolic partial differential equations with free boundaries*, J. Sci. Comput., 68 (2016), pp. 231–251.

[38] R. D. Russell, J. F. Williams, and X. Xu, *MOVCOL4: A moving mesh code for fourth-order time-dependent partial differential equations*, SIAM J. Sci. Comput., 29 (2007), pp. 197–220, http://dx.doi.org/10.1137/050643167.

[39] F. C. Thames, J. F. Thompson, C. Wayne Mastin, and R. L. Walker, *Numerical solutions for viscous and potential flow about arbitrary two-dimensional bodies using body-fitted coordinate systems*, J. Comput. Phys., 24 (1977), pp. 245–273.
[40] H. Touchette, *Legendre-Fenchel Transforms in a Nutshell*, unpublished report, 2005.
[41] N. S. Trudinger and X.-j. Wang, *On the second boundary value problem for Monge-Ampére type equations and optimal transportation*, Ann. Sc. Norm. Super. Pisa Cl. Sci. (5), 8 (2009), pp. 143–174.
[42] J. Urbas, *On the second boundary value problem for equations of monge-ampére type*, J. Reine Angew. Math., 1997 (1997), pp. 115–124.
[43] K. Yee, *Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media*, IEEE Trans., Antennas and Propagation, 14 (1966), pp. 302–307.