Mackenzie, J. A. and Nolan, M. and Rowlatt, C. F. and Insall, R. H. (2019) An adaptive moving mesh method for forced curve shortening flow. SIAM Journal on Scientific Computing, 41 (2). A1170–A1200. ISSN 1064-8275 , http://dx.doi.org/10.1137/18M1211969

This version is available at https://strathprints.strath.ac.uk/66528/

Strathprints is designed to allow users to access the research output of the University of Strathclyde. Unless otherwise explicitly stated on the manuscript, Copyright © and Moral Rights for the papers on this site are retained by the individual authors and/or other copyright owners. Please check the manuscript for details of any other licences that may have been applied. You may not engage in further distribution of the material for any profitmaking activities or any commercial gain. You may freely distribute both the url (https://strathprints.strath.ac.uk/) and the content of this paper for research or private study, educational, or not-for-profit purposes without prior permission or charge.

Any correspondence concerning this service should be sent to the Strathprints administrator: strathprints@strath.ac.uk

The Strathprints institutional repository (https://strathprints.strath.ac.uk) is a digital archive of University of Strathclyde research outputs. It has been developed to disseminate open access research outputs, expose data about those outputs, and enable the management and persistent access to Strathclyde's intellectual output.

SIAM J. Sci. Comput.
Vol. 41, No. 2, pp. A1170–A1200

AN ADAPTIVE MOVING MESH METHOD FOR FORCED CURVE SHORTENING FLOW*

J. A. MACKENZIE†, M. NOLAN‡, C. F. ROWLATT‡, AND R. H. INSALL§

Abstract. We propose a novel adaptive moving mesh method for the numerical solution of a forced curve shortening geometric evolution equation. Control of the mesh quality is obtained using a tangential mesh velocity derived from a mesh equidistribution principle, where a positive adaptivity measure or monitor function is approximately equidistributed along the evolving curve. Central finite differences are used to discretize in space the governing evolution equation for the position vector, and a second-order implicit scheme is used for the temporal integration. Simulations are presented indicating the generation of meshes which resolve areas of high curvature and are of second-order accuracy. Furthermore, the new method delivers improved solution accuracy compared to the use of uniform arc-length meshes.

Key words. geometric partial differential equations, forced curve shortening flow, moving mesh methods, tangential redistribution, monitor functions

AMS subject classifications. 35K65, 53C44, 53C80, 65M06, 65M50

DOI. 10.1137/18M1211969

1. Introduction.

Within the past 20 years there has been much interest in the numerical approximation of geometric flows (see, for example, [12, 14]). In this paper we consider an adaptive method for the solution of the curve evolution equation

$$\mathcal{V}(\boldsymbol{x},t) := \dot{\boldsymbol{x}} \cdot \boldsymbol{n} = \alpha(\boldsymbol{x},t)\kappa + \beta(\boldsymbol{x},t), \tag{1.1}$$

where $\boldsymbol{x}$ is the position vector of the evolving curve Γ, α and β are given functions with α being nonnegative, and κ is the curvature. In the special case when $\alpha(\boldsymbol{x},t) = 1$ and $\beta(\boldsymbol{x},t) = 0$ we get classical curve shortening flow. Geometric equations of the form (1.1) appear in many important application areas, such as material science [1], biological cell migration [29, 16], and image processing [20].

The numerical solution of geometric evolution laws poses many challenges, and a number of different techniques have been proposed which fall broadly into two categories: embedded methods and sharp interface methods. Examples of embedded techniques include phase-field methods [12] and the level set method [32, 30]. These methods identify the moving interface as the zero level set of an indicator function which is normally evolved through a fixed uniform background mesh. Grid generation

---

*Submitted to the journal's Methods and Algorithms for Scientific Computing section September 5, 2018; accepted for publication (in revised form) January 3, 2019; published electronically April 18, 2019.

http://www.siam.org/journals/sisc/41-2/M121196.html

Funding: This work was undertaken while the first author was a Leverhulme Trust Research Fellow under award RF-2014-522. The second author was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant EP/K032208/1, and by a Cancer Research UK postgraduate studentship award. The third author was supported by Cancer Research UK multidisciplinary project award C22713/A20017. The fourth author was supported by Cancer Research UK grant 15672.

†Corresponding author. Department of Mathematics and Statistics, University of Strathclyde, Glasgow G1 1XH, UK (j.a.mackenzie@strath.ac.uk).

‡Department of Mathematics and Statistics, University of Strathclyde, Glasgow G1 1XH, UK (michael.nolan@strath.ac.uk, christopher.rowlatt@strath.ac.uk).

§Cancer Research UK Beatson Institute, Glasgow G61 1BD, UK (r.insall@beatson.gla.ac.uk).

is therefore not an issue, although in reality efficient implementations of embedded methods may require some form of mesh adaptation.

Sharp interface or interface tracking methods represent the curve by the positions of a discrete set of nodal positions on the curve, and these points are evolved in such a way that their normal velocity satisfies (1.1). It is well appreciated, however, that methods which move mesh nodes purely in the normal direction quickly run into difficulty due to over concentration of grid points in areas with locally converging normals and to the opposite problem of dispersion of grid points in areas with diverging local normals. This can lead to lower accuracy, grid crossover, and instabilities, all of which can only be avoided by using an unreasonably small time step.

One way to maintain a good mesh quality is to introduce a tangential velocity $\mathcal{B}$ so that mesh nodes evolve according to the equation

$$\dot{\boldsymbol{x}} = \mathcal{V}\boldsymbol{n} + \mathcal{B}\boldsymbol{t}. \tag{1.2}$$

This approach is attractive because the presence of a tangential velocity has no effect on the shape of the evolving curve, as the shape is determined purely by its normal velocity $\mathcal{V}$. Many suggestions have been made of a suitable tangential velocity to improve solution accuracy and robustness. A nonlocal choice of $\mathcal{B}$ originally proposed by Hou, Lowengrub, and Shelley [17] maintains the relative local curve length between grid points. In [25] this method was generalized so that mesh points evolve to asymptotically equidistribute the arc-length between grid nodes. An alternative approach, giving rise to an intrinsic tangential velocity, was proposed by Barrett, Garcke, and Nürnberg (BGN) in a series of papers [3, 4]. Their method was shown to produce good quality meshes for a range of geometric evolution laws for curves in $\mathbb{R}^2$ and hypersurfaces in $\mathbb{R}^3$. The fully discrete original BGN schemes use a semi-implicit temporal integration method and hence are not guaranteed to exactly equidistribute arc-length. A fully implicit version of the BGN scheme was later proposed which exactly equidistributes arc-length [5]. However, exact equidistribution comes at the cost of having to solve a nonlinear system of equations at each time step. More recently, Elliott and Fritz proposed a finite element method using the DeTurk trick for curve shortening and mean curvature flow [15]. This method involves a parameter which interpolates between the methods of BGN and the scheme of Deckelnick [11]. The parameter also controls the rate at which grid nodes evolve to equidistribute arc-length.

All of the fully discrete BGN schemes and the Elliott and Fritz scheme are first-order accurate in time. Second-order temporal accuracy is achieved using a Crank–Nicolson scheme in [2], and the simulations presented there suggest that a considerable improvement in accuracy can be obtained using a higher-order time integration scheme. Solution accuracy can also be improved using some form of adaptive meshing technique because areas of high curvature require additional local resolution, and this cannot be achieved using a uniform arc-length mesh.

For time-dependent PDEs with localized solution features the use of adaptive moving mesh methods has proved popular [8, 19]. These methods generally use a fixed number of mesh nodes which are redistributed at each time step. Recently, we introduced an adaptive moving mesh method for the evolution of a curve which is driven in the normal direction by a function of curvature and a forcing function. In the tangential direction mesh points are moved according to a moving mesh PDE (MMPDE). The adaptive moving curve method forms part of a fitted bulk-surface formulation of a model of cell migration [21]. The aim of this paper is to improve and extend the method introduced in [21]. The equation for the tangential velocity is derived within the context of a gradient flow equation for the minimization of a functional related to the equidistribution of a mesh adaptivity criterion or monitor function. Within this class of methods, specific choices of the gradient flow direction are shown to reproduce some methods in the literature. To drive mesh adaptivity we develop a monitor function based on curvature. We present two temporal discretizations of the moving mesh equations and show that a newly proposed method is second-order accurate in time and space.

An outline of the rest of this paper is as follows. In the next section we present the geometric evolution law for the curve normal velocity as well as the tangential velocity arising from an adaptive moving mesh approach. In this section we also derive a monitor function to drive the tangential motion of mesh points to areas of high curvature to improve solution accuracy. The numerical discretization of the curve evolution equations is given in section 3. Numerical experiments are carried out in section 4 highlighting the improved performance of the moving mesh method compared to a uniform arc-length redistribution of mesh points. Finally, we make some conclusions and point out directions for future research in section 5.

2. Forced curve shortening flow.

A closed, embedded, regular plane curve $\Gamma(t)$ can be parameterized by the smooth function $\boldsymbol{x}(t) : \mathbb{R}/\mathbb{Z} \supset [0,1] \to \mathbb{R}^2$, such that $\Gamma(t) = \text{Image}(\boldsymbol{x}(t)) := \{\boldsymbol{x}(\xi,t),\ \xi \in [0,1]\}$. Let $F_\xi = \partial F/\partial \xi$ and $|\boldsymbol{a}| = \sqrt{\boldsymbol{a}\cdot\boldsymbol{a}}$, where $\boldsymbol{a}\cdot\boldsymbol{b}$ denotes the Euclidean inner product between the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. The unit tangent vector $\boldsymbol{t} = \boldsymbol{x}_\xi/|\boldsymbol{x}_\xi| = \boldsymbol{x}_s$, where s is the arc-length parameter and $\mathrm{d}s = |\boldsymbol{x}_\xi|\,\mathrm{d}\xi$. We define the unit normal vector $\boldsymbol{n}$ such that $\det(\boldsymbol{t},\boldsymbol{n}) = 1$ and define the signed curvature in the direction $\boldsymbol{n}$ by κ.

Using the Frenet–Serret formula, we have

$$\boldsymbol{x}_{ss} = \boldsymbol{t}_s = \kappa \boldsymbol{n}. \tag{2.1}$$

Applying the chain rule, we have

$$\boldsymbol{x}_\xi = \boldsymbol{x}_s \frac{ds}{d\xi} = \boldsymbol{x}_s |\boldsymbol{x}_\xi|, \tag{2.2}$$

and differentiation of (2.2) with respect to ξ and the use of (2.1) leads to the relation

$$\boldsymbol{x}_{\xi\xi} = \boldsymbol{x}_{ss}|\boldsymbol{x}_\xi|^2 + \boldsymbol{x}_s |\boldsymbol{x}_\xi|_s |\boldsymbol{x}_\xi| = \kappa |\boldsymbol{x}_\xi|^2 \boldsymbol{n} + |\boldsymbol{x}_\xi|_\xi \boldsymbol{t}. \tag{2.3}$$

If we multiply through (2.3) by $\boldsymbol{n}$, we can therefore express the curvature

$$\kappa = \frac{\boldsymbol{x}_{\xi\xi}\cdot\boldsymbol{n}}{|\boldsymbol{x}_\xi|^2}, \tag{2.4}$$

and hence in terms of the parameterization ξ, we can express the normal velocity as

$$\mathcal{V} = \dot{\boldsymbol{x}}\cdot\boldsymbol{n} = \alpha(\boldsymbol{x},t)\left(\frac{\boldsymbol{x}_{\xi\xi}\cdot\boldsymbol{n}}{|\boldsymbol{x}_\xi|^2}\right) + \beta(\boldsymbol{x},t). \tag{2.5}$$

2.1. Adaptive moving mesh approach for the tangential velocity.

The proposed tangential mesh velocity is based on the idea of mesh equidistribution. Let $M(\boldsymbol{x},t) > 0$ be a positive monitor function indicating areas of the curve which require additional resolution, such as regions of high curvature. The ξ-parameterization is said to equidistribute M over the curve $\Gamma(t)$ if

$$M\,\frac{ds}{d\xi} = \int_{\Gamma(t)} M\ \mathrm{d}s. \tag{2.6}$$

Differentiating (2.6) with respect to ξ we have the equivalent equidistribution condition,

$$\left(M\,\frac{ds}{d\xi}\right)_\xi = (M|\boldsymbol{x}_\xi|)_\xi = 0. \tag{2.7}$$

If the parametric domain is partitioned uniformly by grid points $\{\xi_i\}_{i=0}^N$, then the equidistribution condition (2.7) essentially ensures that the image mesh points on the curve are arranged so that the weighted arc-length $M\,\mathrm{d}s$ is constant. The derivation of a suitable curvature-based monitor function is given in section 2.2. If M is constant, then the satisfaction of the equidistribution condition leads to a uniform parameterization in terms of arc-length.

The equidistribution condition, in terms of the inverse mapping, is the Euler–Lagrange equation for the minimizer of the functional

$$I(\xi(s,t)) = \frac{1}{2}\int_{\Gamma(t)} \frac{1}{M}\left(\frac{\partial \xi}{\partial s}\right)^2\,\mathrm{d}s. \tag{2.8}$$

An evolution equation can be obtained from the gradient flow equation

$$\begin{aligned}\frac{\partial \xi}{\partial t} &= -\frac{P}{\tau}\frac{\delta I}{\delta s}\\ &= \frac{P}{\tau}\frac{\partial}{\partial s}\left(\frac{1}{M}\frac{\partial \xi}{\partial s}\right).\end{aligned} \tag{2.9}$$

Here, $\tau > 0$ is a mesh relaxation time determining the rate at which $\xi(s,t)$ evolves to minimize (2.8). The positive definite differential operator P allows a degree of flexibility in the method as we show below. Equation (2.9) is not in an ideal form as the independent variable is arc-length, s, so we need to change the role of the independent and dependent variables. Starting from the identity

$$\xi = \xi(s(\xi,t),t), \tag{2.10}$$

we can differentiate both sides with respect to ξ while keeping t fixed, and we find that

$$\frac{\partial s}{\partial \xi} = \left(\frac{\partial \xi}{\partial s}\right)^{-1}. \tag{2.11}$$

Differentiating (2.10) with respect to t while keeping ξ fixed gives

$$\frac{\partial s}{\partial t} = -\frac{\partial s}{\partial \xi}\frac{\partial \xi}{\partial t}. \tag{2.12}$$

Equation (2.9) can therefore be rewritten as

$$\frac{\partial s}{\partial t} = \frac{P}{\tau}\left(M\frac{\partial s}{\partial \xi}\right)^{-2}\frac{\partial}{\partial \xi}\left(M\frac{\partial s}{\partial \xi}\right). \tag{2.13}$$

Note that the time derivative of the position vector $\boldsymbol{x}$ in (1.2) is taken under the assumption that the value of the parameterization variable ξ is fixed. In terms of the arc-length parameterization, we have

$$\begin{aligned}\dot{\boldsymbol{x}} \equiv \left.\frac{\partial \boldsymbol{x}}{\partial t}\right|_{\xi} &= \left.\frac{\partial \boldsymbol{x}}{\partial t}\right|_{s} + \frac{\partial \boldsymbol{x}}{\partial s}\left.\frac{\partial s}{\partial t}\right|_{\xi} \\ &= \left.\frac{\partial \boldsymbol{x}}{\partial t}\right|_{s} + \left.\frac{\partial s}{\partial t}\right|_{\xi}\boldsymbol{t}. \qquad (2.14)\end{aligned}$$

Comparing (1.2) and (2.14) and using (2.13), we arrive at the tangential velocity equation

$$\mathcal{B} = \dot{\boldsymbol{x}} \cdot \boldsymbol{t} = \frac{P}{\tau}(M|\boldsymbol{x}_\xi|)^{-2}(M|\boldsymbol{x}_\xi|)_\xi. \qquad (2.15)$$

We can identify the equidistribution condition (2.7) as the driving force for tangential mesh movement, evolving the mesh nodes back towards the equidistribution of the monitor function M whenever it drifts away—the rate being controlled by the parameter τ. Particular choices for the operator P lead to distinct tangential velocities, some of which correlate with previously proposed methods. For example, attempting to minimize (2.9) by the steepest descent direction corresponds to the choice $P = 1$. In the special case when $M = 1$, $P = 1$, and $\tau = 1$, the tangential velocity $\mathcal{B} = -(|\boldsymbol{x}_\xi|^{-1})_\xi$, which was used in [11] for curve shortening flow. The choice $P = M|\boldsymbol{x}_\xi|^2$ results in the tangential velocity equation

$$\dot{\boldsymbol{x}} \cdot \boldsymbol{t} = \frac{1}{M\tau}(M|\boldsymbol{x}_\xi|)_\xi. \qquad (2.16)$$

This choice of P results in a tangential velocity equation which is more spatially balanced throughout the domain [18]. In the particular case where a uniform arc-length mesh is desired ($M = 1$), (2.16) is identical to that used in a recent method proposed by Elliott and Fritz [15] based on a harmonic map heat flow.

2.2. Choice of monitor function.

In the absence of a reliable error estimate for the approximation $\Gamma_h(t)$ of $\Gamma(t)$, we base our analysis of a suitable monitor function on a study of interpolation error. The aim is to find a monitor function which, when equidistributed, results in a distribution of mesh points that minimizes an appropriate measure of the difference between a smooth curve Γ and its linear polygonal interpolant Γ_h. Here we focus on the minimization of the maximal distance between Γ and Γ_h. In Figure 1 we show the segment Γ_i of Γ with end points $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$. Also shown is the linear approximation $\Gamma_{h,i}$ of Γ_i. For each $\boldsymbol{x} \in \Gamma_i$, we define the distance, $d(\boldsymbol{x})$, from $\boldsymbol{x}$ to $\Gamma_{h,i}$ as the distance between $\boldsymbol{x}$ and $\boldsymbol{x}_* \in \Gamma_{h,i}$, where the line through $\boldsymbol{x}$ and $\boldsymbol{x}_*$ is perpendicular to $\Gamma_{h,i}$. To simplify the analysis, we note that the distance between Γ_i and $\Gamma_{h,i}$ is invariant to a coordinate rotation and translation. We therefore translate coordinates so that $\boldsymbol{x}_i$ maps to the origin, and we rotate coordinates such that the line segment between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$ is parallel to the positive $\bar{x}$ axis, as shown in Figure 1. Finding the maximal distance between Γ_i and $\Gamma_{h,i}$ is therefore equivalent to finding the maximum absolute value of the transformed graph $\bar{\Gamma}(\bar{x})$ for $0 \le \bar{x} \le |\boldsymbol{x}_{i+1} - \boldsymbol{x}_i|$, and this can be estimated using a standard argument from linear interpolation theory.

Without loss of generality, let us assume that the maximum of $\bar{\Gamma}$ occurs at $\bar{x}_*$ and assume $\bar{x}_*$ is closer to $\bar{x} = 0$ than $\bar{x} = h_i \equiv |\boldsymbol{x}_{i+1} - \boldsymbol{x}_i|$. Using a Taylor series expansion of $\bar{\Gamma}$ about $\bar{x} = 0$, and noting that $\bar{\Gamma}(0) = 0$ and $\bar{\Gamma}'(\bar{x}_*) = 0$, we have

$$\bar{\Gamma}(\bar{x}_*) = \frac{\bar{x}_*^2}{2}\bar{\Gamma}''(\bar{x}_*) + O(\bar{x}_*)^3.$$

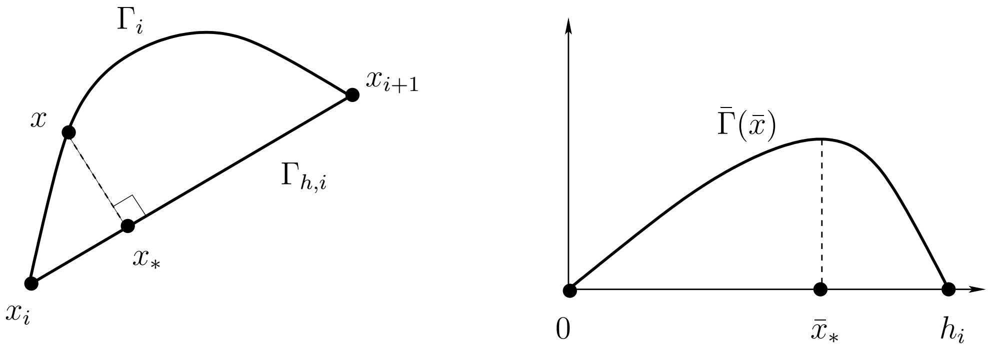


FIG. 1. *Left: Segment of a smooth curve Γ_i and interpolating linear approximation $\Gamma_{h,i}$ between mesh points $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$. The distance between the curves at point $\boldsymbol{x}$ is the distance from $\boldsymbol{x}$ to $\boldsymbol{x}_*$. Right: The translated and rotated segment is transformed into the graph $\bar\Gamma(\bar x)$. The maximal distance between Γ_i and $\Gamma_{h,i}$ is equal to the absolute maximum value $\bar\Gamma(\bar x_*)$.*

The absolute value of the curvature $|\bar\kappa|$ of $\bar\Gamma$ is

$$|\bar\kappa| = \frac{|\bar\Gamma''|}{(1+(\bar\Gamma')^2)^{3/2}},$$

and since $\bar\Gamma'(\bar x_*) = 0$, it follows that $|\bar\Gamma''(\bar x_*)| = |\bar\kappa(\bar x_*)|$. Therefore, we find that

$$\begin{aligned}\max_{\bar x\in(0,h_i)} |\bar\Gamma(\bar x)| &= \frac{\bar x_*^2}{2}|\bar\Gamma''(\bar x_*)| + O(\bar x_*)^3\\ &\le \frac{h_i^2}{8}|\bar\kappa(\bar x_*)| + O(h_i)^3. \end{aligned} \tag{2.17}$$

The curvature of Γ_i is clearly invariant to the translation and rotation mapping above, and hence an approximately optimal distribution of mesh points $\{\boldsymbol{x}_i\}_{i=0}^N$, which minimizes the maximal error over all segments, is obtained when

$$h_i^2|\kappa_i| = h_{i+1}^2|\kappa_{i+1}|, \qquad i = 1,\dots,N-1, \tag{2.18}$$

where $\kappa_i = \max_{\boldsymbol{x}\in\Gamma_i}|\kappa(\boldsymbol{x})|$. It therefore follows that the quantity $h_i|\kappa_i|^{1/2}$ is constant in each segment, and this suggests that a suitable monitor function for curve approximation should be based on equidistribution of $|\kappa|^{1/2}$.

Since κ can potentially be zero at flat sections of a curve, it is important to include a positive floor on the monitor function to ensure that no area of the curve becomes starved of mesh points. A simple curvature-based monitor function therefore takes the form

$$M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2}). \tag{2.19}$$

Motivated by the design of suitable monitor functions for singular perturbation problems [6], we consider the floor

$$M_{\text{floor}}(t) = \frac{1}{|\Gamma(t)|}\int_{\Gamma(t)} |\kappa|^{1/2}\,\mathrm{d}s. \tag{2.20}$$

A similar floor was used in the adaptive solution of evolutionary PDEs in one dimension [7]. Major advantages of the floor (2.20) are that it does not require any a priori choice of parameters and that it adapts to the length of the evolving curve.

Note that although we have focused on the derivation of a suitable monitor function based on the maximal distance between $\Gamma_h(t)$ and $\Gamma(t)$, alternative monitor functions can be derived to minimize different error measures. For example, it has been shown that equidistribution of $|\kappa|^{1/3}$ leads to an interpolatory linear polygonal curve which minimizes the discrepancy in the enclosed area between Γ and Γ_h, and equidistribution of $|\kappa|^{2/3}$ minimizes the total length discrepancy [33].

3. Numerical discretization.

The time integration interval $(0, T]$ is partitioned using N_T time steps of size $\Delta t = T/N_T$. We represent the evolving curve at time $t^n = n\Delta t$ by the closed linear polygonal curve joining the discrete plane points $\boldsymbol{x}_i^n$, $i = 0, \dots, N$. To enforce periodicity we set $\boldsymbol{x}_0^n = \boldsymbol{x}_N^n$, and we also use the ghost nodes $\boldsymbol{x}_{-1}^n = \boldsymbol{x}_{N-1}^n$ and $\boldsymbol{x}_{N+1}^n = \boldsymbol{x}_1^n$. The parameterization interval $\xi \in [0, 1]$ is discretized using a uniform step size $\Delta\xi = 1/N$. Using central differences, we approximate the unit tangent vector at $\boldsymbol{x}_i^n$ by

$$\boldsymbol{t}_i^n = \frac{\boldsymbol{x}_{i+1}^n - \boldsymbol{x}_{i-1}^n}{|\boldsymbol{x}_{i+1}^n - \boldsymbol{x}_{i-1}^n|} = (t_1^n, t_2^n),$$

and we set $\boldsymbol{n}_i^n = (t_2^n, -t_1^n)$. We use central finite differences to approximate the spatial derivatives in (2.5), and we consider two temporal integration schemes. The first approach, which was introduced in [21], is based on a first-order fully implicit backward Euler scheme. This leads to a nonlinear algebraic system which is solved using Picard iteration. If $\boldsymbol{x}^{[n,m]}$ denotes the approximation of $\boldsymbol{x}^n$ at iteration level m, then the discretized normal velocity equation takes the form

$$\begin{aligned}&[-\mu_i^{[n+1,m]}\boldsymbol{x}_{i-1}^{[n+1,m+1]} + (1+2\mu_i^{[n+1,m]})\boldsymbol{x}_i^{[n+1,m+1]} - \mu_i^{[n+1,m]}\boldsymbol{x}_{i+1}^{[n+1,m+1]}] \cdot \boldsymbol{n}_i^{[n+1,m]} \\ &\qquad = \boldsymbol{x}_i^n \cdot \boldsymbol{n}_i^{[n+1,m]} + \Delta t \beta_i^{[n+1,m]}\end{aligned} \tag{3.1}$$

for $i = 1, \dots, N$, where

$$\begin{aligned}\mu_i^{[n,m]} &= \frac{4\Delta t \alpha_i^{[n,m]}}{|\boldsymbol{x}_{i+1}^{[n,m]} - \boldsymbol{x}_{i-1}^{[n,m]}|^2}, \\ \alpha_i^{[n,m]} &= \alpha(\boldsymbol{x}_i^{[n,m]}, t^n) \quad \text{and} \quad \beta_i^{[n,m]} = \beta(\boldsymbol{x}_i^{[n,m]}, t^n).\end{aligned}$$

The second scheme uses a second-order Crank–Nicolson fully implicit temporal discretization of the normal velocity equation (2.5), and the solution is found using the iteration

$$\begin{aligned}&\left[-\frac{\mu_i^{[n+1,m]}}{2}\boldsymbol{x}_{i-1}^{[n+1,m+1]} + (1+\mu_i^{[n+1,m]})\boldsymbol{x}_i^{[n+1,m+1]} - \frac{\mu_i^{[n+1,m]}}{2}\boldsymbol{x}_{i+1}^{[n+1,m+1]}\right] \cdot \boldsymbol{n}_i^{[n+\frac{1}{2},m]} \\ &\quad = \frac{\mu_i^n}{2}[\boldsymbol{x}_{i-1}^n - 2\boldsymbol{x}_i^n + \boldsymbol{x}_{i+1}^n] \cdot \boldsymbol{n}_i^{[n+\frac{1}{2},m]} + \boldsymbol{x}_i^n \cdot \boldsymbol{n}_i^{[n+\frac{1}{2},m]} + \frac{\Delta t}{2}(\beta_i^{[n+1,m]} + \beta_i^n)\end{aligned} \tag{3.2}$$

for $i = 1, \dots, N$, where

$$\boldsymbol{n}_i^{[n+\frac{1}{2},m]} = \frac{(\boldsymbol{n}_i^{[n+1,m]} + \boldsymbol{n}_i^n)}{2}.$$

The spatial discretization is applied to a reformulation of the tangential velocity equation (2.15). Using the identity

$$|\boldsymbol{x}_\xi|_\xi = \frac{\boldsymbol{x}_\xi \cdot \boldsymbol{x}_{\xi\xi}}{|\boldsymbol{x}_\xi|}$$

we can write (2.15) as

$$\dot{\boldsymbol{x}} \cdot \boldsymbol{t} = \frac{P(M_\xi \boldsymbol{x}_\xi + M\boldsymbol{x}_{\xi\xi}) \cdot \boldsymbol{t}}{\tau(M|\boldsymbol{x}_\xi|)^2}. \tag{3.3}$$

Since $\boldsymbol{t} = \boldsymbol{x}_\xi/|\boldsymbol{x}_\xi|$, we can rewrite (3.3) as

$$\left(\dot{\boldsymbol{x}} - \frac{PM}{\tau(M|\boldsymbol{x}_\xi|)^2}\boldsymbol{x}_{\xi\xi}\right) \cdot \boldsymbol{t} = \frac{PM_\xi|\boldsymbol{x}_\xi|}{\tau(M|\boldsymbol{x}_\xi|)^2}. \tag{3.4}$$

To discretize (3.4) we use central differences to approximate the spatial terms and a first-order backward Euler time integration scheme. This results in the set of equations

$$\begin{aligned} &[-\nu_i^{[n+1,m]}\boldsymbol{x}_{i-1}^{[n+1,m+1]} + (1+2\nu_i^{[n+1,m]})\boldsymbol{x}_i^{[n+1,m+1]} - \nu_i^{[n+1,m]}\boldsymbol{x}_{i+1}^{[n+1,m+1]}] \cdot \boldsymbol{t}_i^{[n+1,m]} \\ &\quad = \boldsymbol{x}_i^n \cdot \boldsymbol{t}_i^{[n+1,m]} + \frac{\Delta t P_i^n (M_{i+1}^n - M_{i-1}^n)}{\tau(M_i^n|\boldsymbol{x}_{i+1}^{[n+1,m]} - \boldsymbol{x}_{i-1}^{[n+1,m]}|)^2}(|\boldsymbol{x}_{i+1}^{[n+1,m]} - \boldsymbol{x}_{i-1}^{[n+1,m]}|) \end{aligned} \tag{3.5}$$

for $i = 1, \ldots, N$, where

$$\nu_i^{[n,m]} = \frac{4\Delta t M_i^n P_i^n}{\tau(M_i^n|\boldsymbol{x}_{i+1}^{[n+1,m]} - \boldsymbol{x}_{i-1}^{[n+1,m]}|)^2}.$$

Note that the monitor function M and spatial balancing operator P are always treated explicitly. This is justified because, in general, one may wish to adapt the mesh to solution features (such as a travelling wave front), which will only be known at time level t^n. The coupled set of $2N$ equations, comprised of (3.1) or (3.2) for the normal velocity and (3.5) for the tangential velocity, is solved for $\boldsymbol{x}^{[n+1,m+1]}$, and the Picard iteration is stopped when

$$|\boldsymbol{x}^{[n+1,m+1]} - \boldsymbol{x}^{[n+1,m]}| < 10^{-6}.$$

The solution after the final iteration is then used as the approximation $\boldsymbol{x}^{n+1}$. For the initial guess at the start of each iteration we set $\boldsymbol{x}^{[n+1,0]} = \boldsymbol{x}^n$. The maximum number of iterations allowed is fixed at 200, and if the Picard solver cannot converge within this limit, then the simulation stops.

The first integration scheme is therefore a backward Euler method for both the normal and tangential velocity equations. The second scheme, which we denote by CNBE, uses a Crank-Nicolson scheme for the normal equations and a backward Euler method for the tangential equations. It may appear odd that we have exclusively used a first-order scheme for the tangential equations. However, as mentioned earlier, the tangential position of the mesh points should not have a major effect on the overall shape of the curve because this is determined by its normal velocity. The backward Euler scheme has therefore been chosen because it has better stability properties compared to the Crank–Nicolson scheme. Numerical experiments in the next section indicate that the CNBE scheme is second-order accurate in terms of the enclosed area error measure.

The curvature-based monitor function (2.19) requires an approximation of curvature. Based on a central difference approximation of (2.4) we set

$$\kappa_i^n = \frac{4(\boldsymbol{x}_{i-1}^n - 2\boldsymbol{x}_i^n + \boldsymbol{x}_{i+1}^n)\cdot \boldsymbol{n}_i^n}{|\boldsymbol{x}_{i+1}^n - \boldsymbol{x}_{i-1}^n|^2}. \tag{3.6}$$

For the time-dependent floor (2.20) we use a simple quadrature approximation, and hence the discrete approximation of the monitor function (2.19) takes the form

$$M_i^n = \frac{1}{2|\Gamma_h(t^n)|}\sum_{j=1}^{N}\left(\frac{|\kappa_{j+1}^n|^{1/2} + |\kappa_j^n|^{1/2}}{2}\right) h_j^n + \frac{1}{2}|\kappa_i^n|^{1/2}, \tag{3.7}$$

where

$$|\Gamma_h(t^n)| = \sum_{j=1}^{N} h_j^n. \tag{3.8}$$

To enhance the robustness of the adaptive grid procedure, we smooth the monitor function by using a spatial averaging technique [7], so that

$$\tilde{M}_i = \frac{\sum_{k=i-p}^{i+p} M_k (q/(q+1))^{|k-i|}}{\sum_{k=i-p}^{i+p} (q/(q+1))^{|k-i|}}, \tag{3.9}$$

where q is a positive real number and p is a nonnegative integer. For the simulations presented later we set $p = 2$ and $q = 3$.

3.1. Initial grid generation.

To initiate the moving mesh method, it is important to be able to generate a starting mesh which equidistributes the monitor function. This ensures a smooth initial evolution of the mesh points and improves solution accuracy and stability. We will assume that the initial curve can be expressed in terms of the parameterized variable u. Of course a uniform partition of the u domain is unlikely to equidistribute the given monitor function. We therefore need to generate a partition $\{u_i\}_{i=0}^N$ such that

$$\int_0^{u_i} M(u)|\boldsymbol{x}_u|\,\mathrm{d}u = \frac{i}{N}\int_0^1 M(u)|\boldsymbol{x}_u|\,\mathrm{d}u, \qquad i = 0,\dots,N.$$

To generate an approximation of the equdistributing mesh we will use an adaptation of the so-called de Boor algorithm [10]. We assume that M and $|\boldsymbol{x}_u|$ can be evaluated on an arbitrary background partition $\{u_i^{old}\}_{i=0}^N$ and that the function $M(u)|\boldsymbol{x}_u|$ is approximated by the piecewise constant function

$$\rho(u) = \begin{cases} M(u_{1/2})|\boldsymbol{x}_u|_{1/2}, & u \in [u_0, u_1], \\ M(u_{3/2})|\boldsymbol{x}_u|_{3/2}, & u \in (u_1, u_2], \\ \vdots & \vdots \\ M(u_{N-1/2})|\boldsymbol{x}_u|_{N-1/2}, & u \in (u_{N-1}, u_N], \end{cases}$$

where $u_{i+1/2} = (u_i + u_{i+1})/2$, $i = 0,\dots,N-1$. A new partition $\{u_i^{new}\}_{i=0}^N$, which exactly equidistributes $\rho(u)$, can be found using inverse linear interpolation (algorithmic details can be found in [31, 19]). The new partition of course only equidistributes

ρ over the old partition, and hence iteration is used to update the partition further by simply setting the old partition to be the new partition and repeating the de Boor step. We can then generate a sequence of partitions that eventually converges to the final approximately equidistributed partition of the parameterized domain. The physical mesh point locations $\{x(u_i), y(u_i)\}_{i=0}^{N}$ are obtained from the parametric map of the final converged partition $\{u_i\}_{i=0}^{N}$.

4. Numerical experiments.

4.1. Curve shortening flow of a circle.

We first consider an initial unit circle shrinking according to curve shortening flow $\dot{\boldsymbol{x}} \cdot \boldsymbol{n} = \kappa$. The exact solution of this problem is a circle of radius $r(t) = \sqrt{1-2t}$, and simulations were performed up to time $T = 0.25$. We define the error in the approximation of the enclosed area at time $t \in (0, T]$ by $e_h(t) := A_h(t) - A(t)$, where $A(t) = A(0) - 2\pi t$ is the exact enclosed area for any closed curve evolved by classical curve shortening flow (and therefore will be used for other examples), and A_h is the enclosed area for a polygon

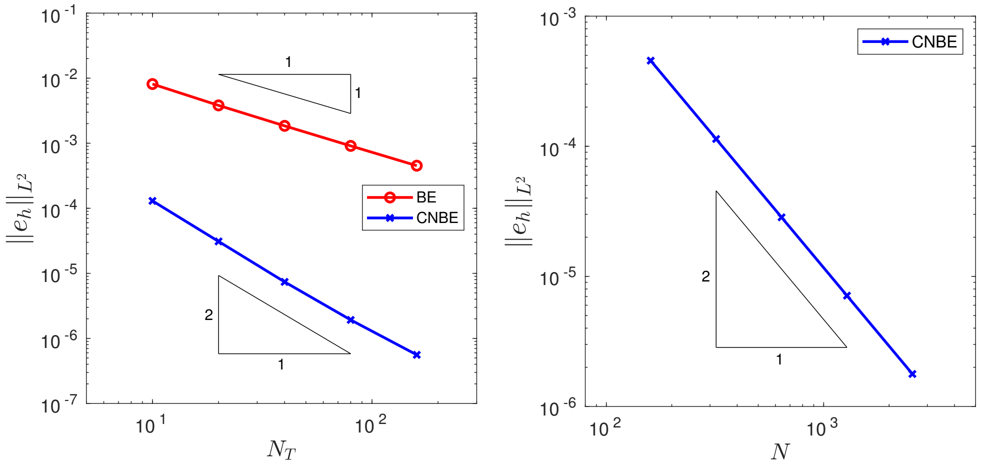
$$A_h(t) = \frac{1}{2} \sum_{i=0}^{N-1} \left(x_i(t) y_{i+1}(t) - x_{i+1}(t) y_i(t)\right).$$

Convergence will be studied using the $L^2([0, T])$ norm

$$\|e_h\|_{L^2} \approx \sqrt{\sum_{n=1}^{N_T} (A_h(t^n) - A(t^n))^2 \Delta t}.$$

To test the temporal rate of convergence of the two time integration schemes, backward Euler (BE) and Crank–Nicolson backward Euler (CNBE), simulations were performed using a fine spatial mesh with $N = 10^4$ points. The de Boor algorithm was used to construct the initial mesh, and we found no difference between the convergence properties of the two time integration schemes for any choice of spatial balancing operator P nor any choice of mesh relaxation time τ. Additionally, we found no difference in the convergence for each choice of monitor function M. Therefore, for the convergence studies we restrict ourselves (for brevity) to $M = 1$, $P = 1$, and $\tau = 0.1$. Figure 2(a) shows the decrease in the error as the number of time steps is increased. As expected, the BE scheme is first-order convergent, while the CNBE scheme is second-order convergent. Furthermore, the CNBE scheme is considerably more accurate than the BE scheme using the equivalent number of time steps. These results highlight the improvement in accuracy achievable using a higher-order time integration scheme for the normal velocity equations. Spatial convergence was tested for the CNBE scheme using a large number of time steps $N_T = 10^4$ (i.e., $\Delta t = 2.5 \times 10^{-5}$). As shown in Figure 2(b), the rate of spatial convergence is second order.

Each of the temporal integration schemes proposed requires the solution of a nonlinear system to evolve the solution forward in time. Table 1 displays the maximum and minimum number of Picard iterations required for each scheme, with $N = 10^4$, for each temporal resolution considered. Clearly, the computational cost of solving the nonlinear system each time step is small. The maximum number of Picard iterations for the BE scheme is always greater than (or equal to) the maximum for the CNBE scheme. Note that for both the BE scheme and the CNBE scheme, the maximum number of Picard iterations decreases as N_T is increased. Table 2 displays the maximum and minimum number of Picard iterations required for each scheme, with



(a) *Temporal convergence.* (b) *Spatial convergence.*

FIG. 2. (a) *Temporal and* (b) *spatial convergence in the L^2 norm of the approximation of the enclosed area when an initial circle is evolved by curve shortening flow.*

TABLE 1
(Circle.) Maximum and minimum number of Picard steps required for each scheme, with $N = 10^4$, for each temporal resolution.

	$N_T = 10$		$N_T = 20$		$N_T = 40$		$N_T = 80$		$N_T = 160$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	6	5	5	4	4	3	3	3	3	3
CNBE	5	5	4	4	4	3	3	3	3	3

TABLE 2
(Circle.) Maximum and minimum number of Picard steps required for each scheme, with $N_T = 10^4$, for each spatial resolution.

	$N = 160$		$N = 320$		$N = 640$		$N = 1280$		$N = 2560$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	2	2	2	2	2	2	2	2	2	2
CNBE	2	2	2	2	2	2	2	2	2	2

$N_T = 10^4$, for each spatial resolution. Once again, it is clear that the computational cost of solving the nonlinear system is minimal, requiring only two iterations per time step.

For this problem, there is no tangential movement of grid nodes because they move entirely in the normal direction to maintain a uniform arc-length distribution between mesh points. Additionally, due to the constant curvature in this example, there is no tangential movement of the grid nodes for the curvature-based monitor function. Therefore, we next consider an example with nonconstant curvature to assess the impact of the curvature-based monitor function on the accuracy of the two time integration schemes.

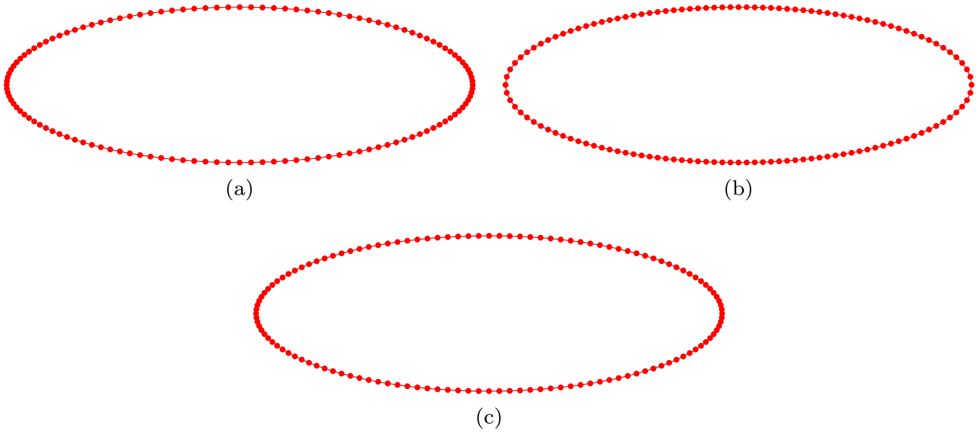


Fig. 3. *Initial mesh partitioning of the ellipse* (4.1)*, using* $N = 128$ *points, according to* (a) *a uniform* u*-parameterization,* (b) *an equidistributed uniform arc-length approximation* $M = 1$*, and* (c) *an equidistributed curvature-based monitor function* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

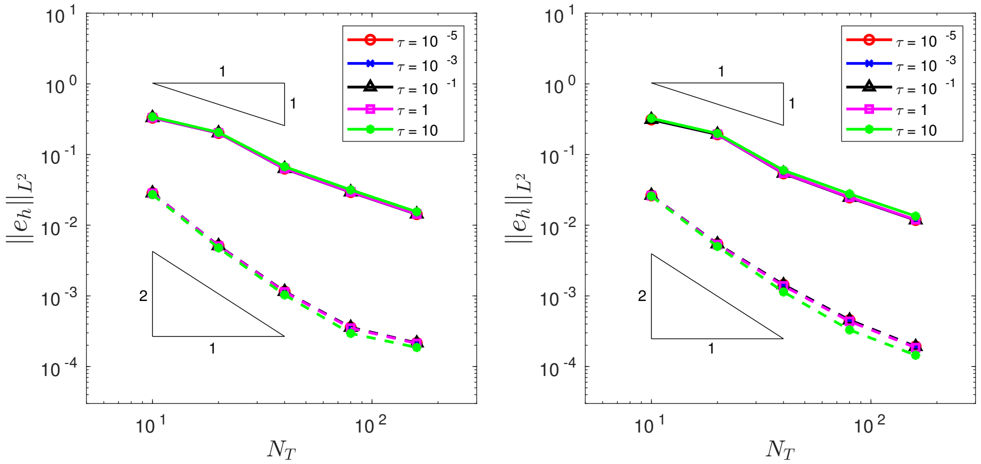
4.2. Curve shortening flow from an ellipse. We next consider curve shortening flow of an ellipse described parametrically by

$$\begin{aligned} x(u,0) &= 3\cos(2\pi u), \qquad 0 \le u \le 1, \\ y(u,0) &= \sin(2\pi u). \end{aligned} \tag{4.1}$$

Figure 3 illustrates the initial mesh partitioning of the ellipse (4.1), using $N = 128$ points, according to a uniform u-parameterization (Figure 3(a)), an equidistributed uniform arc-length approximation (Figure 3(b)), and an equidistributed curvature-based monitor function $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ (Figure 3(c)). Although grid points for the uniform u-parameterization and the equidistributed curvature-based monitor function may seem similar, the use of an initial mesh that equidistributes the given monitor function turns out to be important, which we will see later.

The initial position of the grid nodes is determined by the de Boor algorithm (section 3.1). In all simulations, we run to a final time of $T = 1.4$. Throughout this section, we assume a spatial balancing operator of the form $P = M|\boldsymbol{x}_\xi|^2$. The temporal convergence was tested for both the BE and CNBE schemes, using a fine mesh resolution of $N = 10^3$. Figure 4(a) illustrates the temporal convergence when $M = 1$. It is clear that the BE scheme demonstrates first-order convergence and the CNBE scheme demonstrates second-order convergence. (The slight flattening out of the error decrease for large values of N_T is due to the pollution of the global error by spatial error components.) Figure 4(b) illustrates the temporal convergence when $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$. Once again, it is clear that the BE scheme demonstrates first-order convergence and the CNBE scheme demonstrates second-order convergence. Additionally, Figures 4(a) and 4(b) demonstrate that, for our choice of spatial balancing operator $P = M|\boldsymbol{x}_\xi|^2$, the errors are robust to the choice of τ.

Following the circle example (section 4.1), we wish to illustrate the computational efficiency of the nonlinear Picard solver. Thus, for both monitor functions, Table 3 displays the maximum and minimum number of Picard iterations required for each scheme and for each temporal resolution considered when $\tau = 10$. Apart from when $N_T = 10$ (which corresponds to the largest time step size), the computational cost of



(a) $M = 1$. (b) $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

FIG. 4. *Temporal convergence in the* $L^2([0, T])$ *norm of the approximation of the enclosed area when an initial ellipse is evolved by curve shortening flow using the BE (solid line) and the CNBE (dashed line) scheme with* $P = M|\boldsymbol{x}_\xi|^2$ *and* (a) $M = 1$ *or* (b) $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

TABLE 3

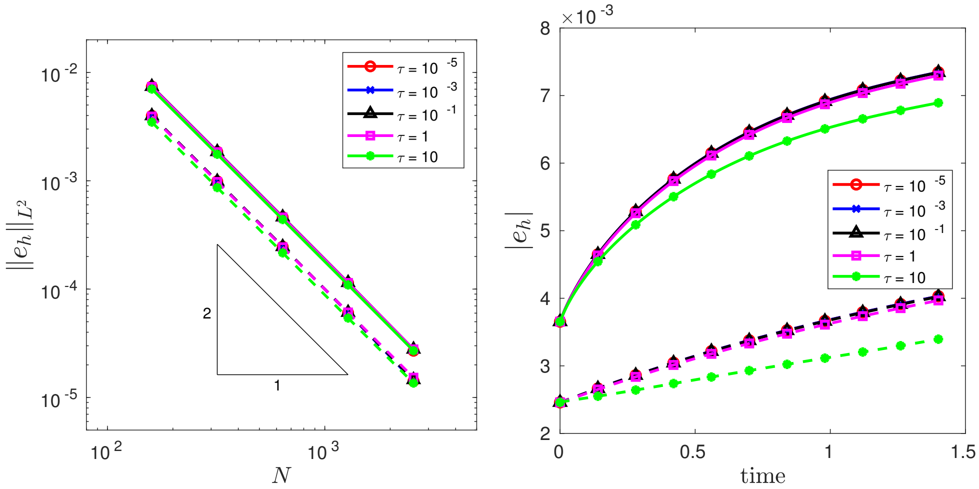
(Ellipse.) Maximum and minimum number of Picard steps required for each scheme, with $N = 10^3$, *for each temporal resolution when* $\tau = 10$ *and* $P = M|\boldsymbol{x}_\xi|^2$, *for both* $M = 1$ *and* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

	$M = 1$									
	$N_T = 10$		$N_T = 20$		$N_T = 40$		$N_T = 80$		$N_T = 160$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	23	10	23	8	13	7	8	5	7	4
CNBE	20	9	11	7	9	6	8	5	7	4

	$M = \frac{1}{2}(M_{\text{floor}} + \|\kappa\|^{1/2})$									
	$N_T = 10$		$N_T = 20$		$N_T = 40$		$N_T = 80$		$N_T = 160$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	22	9	33	7	12	6	7	4	6	4
CNBE	20	9	11	7	7	5	6	4	6	4

the CNBE scheme is reasonably low. Additionally, we note that using a curvature-based monitor function does not substantially increase the computational cost of the Picard solver. For $N_T = 10, 20$ we note that the BE scheme struggles compared with the CNBE scheme. Indeed, for the BE scheme, when $N_T = 20$ and the curvature-based monitor function is used, there is an increase in the maximum number of Picard iterations. This spike is reflected in the convergence plots (Figure 4), where there is a slight bump for $N_T = 20$. However, as the number of time steps increases, the maximum number of Picard iterations decreases for both schemes.

The spatial convergence was tested using a large number of time steps $N_T = 10^4$. Figure 5(a) illustrates the spatial convergence of the CNBE scheme when $M = 1$ (solid line) and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ (dashed line). The convergence is clearly second order for all values of τ for both $M = 1$ and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$, but



(a) *Spatial convergence.* (b) *Absolute value of the area error in time.*

FIG. 5. (a) *Spatial convergence in the $L^2([0,T])$ norm of the approximation of the enclosed area when an initial ellipse is evolved by curve shortening flow using the CNBE scheme for all τ and for both $M = 1$ (solid line) and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ (dashed line).* (b) *Absolute value of the area error in time for the CNBE scheme with $N = 160$ for both $M = 1$ (solid line) and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ (dashed line).*

crucially, the curvature-based monitor function produces an improved error compared with the uniform arc-length monitor function. This is due to the curve being more accurately approximated using a curvature-based monitor function compared with uniform arc-length. To this end, Figure 5(b) illustrates the absolute value error in the computed area for the same spatial balancing operator for both $M = 1$ and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$. Once again, it is clear that the curvature-based monitor function produces a much better mesh compared to a uniform arc-length.

As was demonstrated in the temporal convergence study, Figure 5(a) shows that, for a given monitor function, all values of τ perform similarly. Thus, for both monitor functions, Table 4 displays the maximum and minimum number of Picard iterations required for each scheme and for each spatial resolution considered when $\tau = 10$. As was seen for the circle example (section 4.1), the computational cost of the nonlinear Picard solver is minimal, requiring at most three iterations, and increasing the spatial resolution does not affect the maximum number of iterations.

4.2.1. Uniform *u*-parameterization. In the previous section, we used the de Boor algorithm to construct the initial approximation to a given curve. An obvious question is, Why is this necessary? Therefore, in this section we provide evidence for the importance of using an equidistributed initial grid. Indeed, here the initial position of the grid nodes is given by a uniform u-parameterization described by (4.1) and illustrated in Figure 3(a). It is clear from Figure 3(a) that the initial distribution of the grid nodes resulting from a uniform u-parameterization is similar to the distribution obtained from an equidistributed curvature-based monitor function (Figure 3(c)). In general, this will not be the case (as will be shown later). Therefore, to emphasize the importance of using an equidistributed initial grid we will restrict ourselves to the uniform arc-length monitor function $M = 1$ (Figure 3(b)).

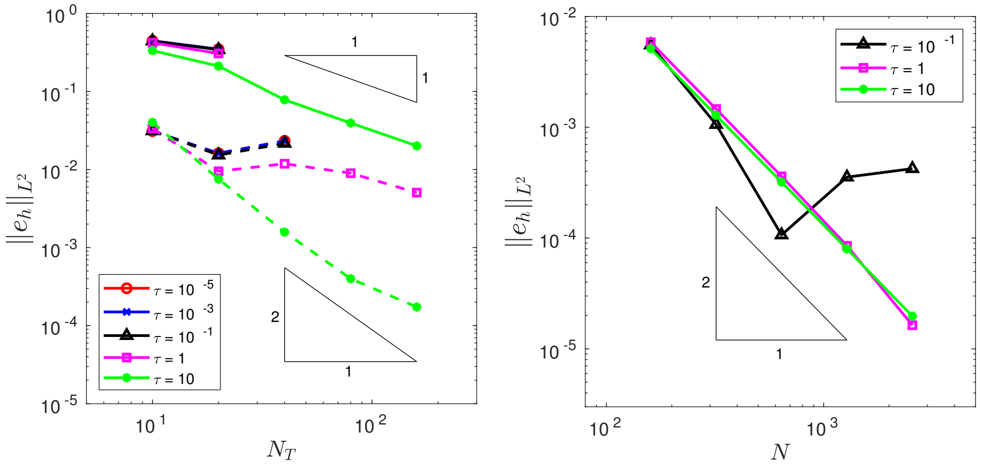
Figure 6(a) illustrates the temporal convergence when $M = 1$ and $P = M|\boldsymbol{x}_\xi|^2$

TABLE 4

(Ellipse.) Maximum and minimum number of Picard steps required for each scheme, with $N_T = 10^4$*, for each spatial resolution when* $\tau = 10$ *and* $P = M|\boldsymbol{x}_\xi|^2$*, for both* $M = 1$ *and* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

	$M = 1$									
	$N = 160$		$N = 320$		$N = 640$		$N = 1280$		$N = 2560$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	3	2	3	2	3	2	3	2	3	2
CNBE	3	2	3	2	3	2	3	2	3	2

	$M = \frac{1}{2}(M_{\text{floor}} + \|\kappa\|^{1/2})$									
	$N = 160$		$N = 320$		$N = 640$		$N = 1280$		$N = 2560$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	3	2	3	2	3	2	3	2	3	2
CNBE	2	2	2	2	2	2	2	2	2	2

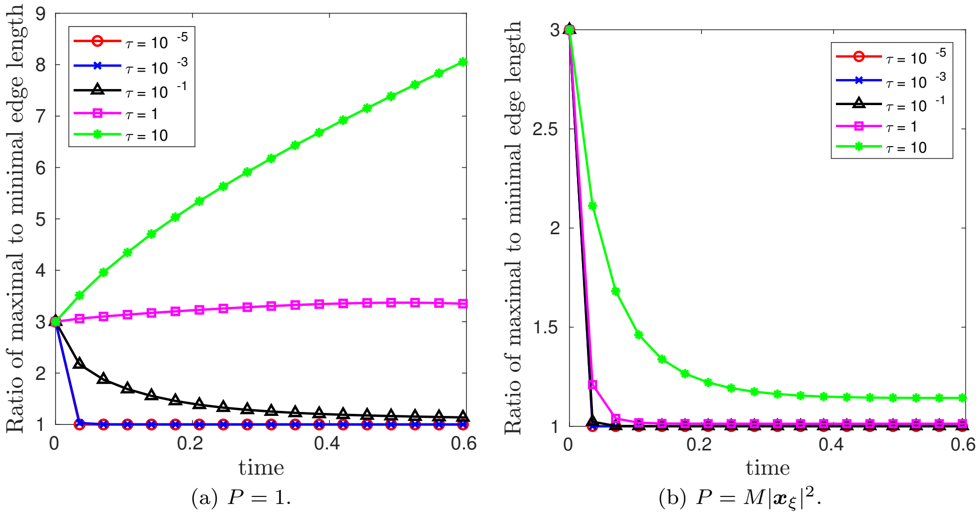


(a) *Temporal convergence.* (b) *Spatial convergence.*

FIG. 6. (a) *Temporal and* (b) *spatial convergence in the* $L^2([0,T])$ *norm of the approximation of the enclosed area when an initial ellipse is evolved by curve shortening flow with* $M = 1$ *and* $P = M|\boldsymbol{x}_\xi|^2$.

for both the BE (solid line) and CNBE (dashed line) schemes. From Figure 6(a), it is clear that the only value of τ to obtain the expected convergence for either the BE or CNBE scheme is $\tau = 10$. For the CNBE scheme, when $\tau = 1$ we see pre-asymptotic convergence, while no convergence is possible for the other values of τ because the nonlinear Picard solver is unable to converge within the maximum number of iterations. The BE scheme cannot obtain convergence for any value of τ except $\tau = 10$. Evidently the choice of τ is important for the time scales considered. Taking too small a value of τ in relation to Δt pollutes the convergence rate significantly and also increases the computational cost of the Picard solver, to the point where the solver may not converge at all. Although the precise relationship between τ and Δt is not currently known, we hypothesize that the ratio between τ and Δt is of greater importance than the individual values, and generally one should avoid having $\tau \ll \Delta t$.

Figure 6(b) illustrates the spatial convergence when $M = 1$ and $P = M|\boldsymbol{x}_\xi|^2$ for



(a) $P = 1$. (b) $P = M|\boldsymbol{x}_\xi|^2$.

FIG. 7. *(Ellipse.) Ratio of maximal to minimal edge length in time when* $M = 1$, $N = 10^3$, $N_T = 40$ *with* (a) $P = 1$ *and* (b) $P = M|\boldsymbol{x}_\xi|^2$.

the CNBE scheme. It is clear that $\tau = 1$ and $\tau = 10$ achieve the expected second-order convergence. However, no convergence was possible for $\tau = 10^{-5}$ and $\tau = 10^{-3}$, and pre-asymptotic convergence was seen for $\tau = 0.1$. To illustrate the role of the mesh relaxation time τ, Figure 7 plots the ratio of the maximal to minimal edge length of the evolving mesh in time. First, when $P = 1$, Figure 7(a) demonstrates that when $\tau = 10^{-5}$ or $\tau = 10^{-3}$ the MMPDE equidistributes arc-length within a few time steps. At first glance, a quick convergence to a uniform arc-length mesh appears desirable, but this rapid equidistribution results in a loss of smoothness in the nodal trajectories. Therefore, having an initially well-defined grid, constructed by equidistributing a given monitor function, will help prevent rapid equidistribution of the nodal points in time and thus increase the robustness of the algorithm by reducing the dependency on the mesh relaxation time τ. Also illustrated in Figure 7(a) is the potential negative effect of having too large a value of τ. Too large a value could prevent the MMPDE from equidistributing a given monitor function in time and hence produce undesirable nodal clustering.

From Figure 6, when $P = M|\boldsymbol{x}_\xi|^2$ we expect to see a rapid equidistribution within a few time steps for $\tau \leq 1$, and this is precisely what is seen in Figure 7(b). These results suggest that the spatial balancing operator $P = M|\boldsymbol{x}_\xi|^2$ alters the time scale of mesh relaxation, allowing larger values of τ to be used. Incidentally, this explains why the spatial balancing operator $P = M|\boldsymbol{x}_\xi|^2$ was chosen here. As just discussed, having an initially equidistributed grid prevents rapid equidistribution of the nodal points in time and thus increases the robustness of the algorithm by allowing smaller values of τ, while the spatial balancing operator $P = M|\boldsymbol{x}_\xi|^2$ allows larger values of τ as illustrated in Figures 6 and 7(b). These two observations combined enable us to minimize the dependency on the mesh relaxation time τ. This is illustrated in the temporal (Figure 4) and spatial (Figure 5(a)) convergence plots given previously, where each value of τ performed similarly.

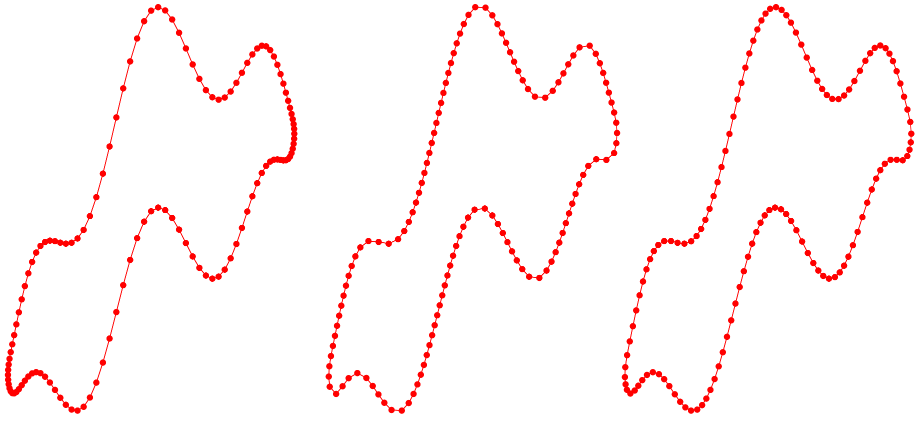

(a) (b) (c)

FIG. 8. *Initial mesh partitioning of the nonconvex curve* (4.2)*, using* $N = 128$ *points, according to* (a) *a uniform* u*-parameterization,* (b) *an equidistributed uniform arc-length approximation* $M = 1$*, and* (c) *an equidistributed curvature-based monitor function* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

4.3. Curve shortening flow of a nonconvex initial curve.

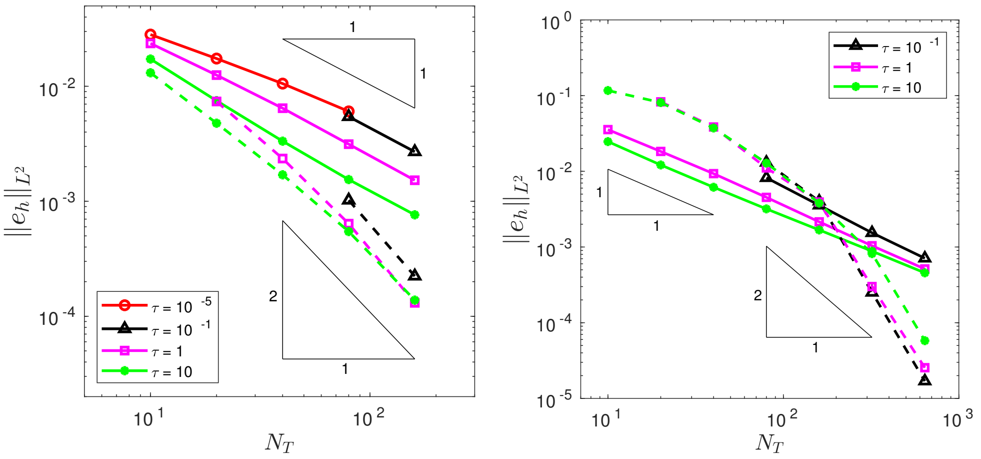
We next consider curve shortening flow of the nonconvex initial curve

$$
\begin{aligned}
(4.2)\quad & x(u,0) = \cos(2\pi u), \qquad 0 \le u \le 1,\\
& y(u,0) = 0.5\sin(2\pi u) + \sin(\cos(2\pi u)) + \sin(2\pi u)\left(0.2 + \sin(2\pi u)\sin^2(6\pi u)\right).
\end{aligned}
$$

This example was used previously to test tangentially stabilized curve evolution algorithms [2, 3, 24, 33].

Figure 8 illustrates the initial mesh partitioning of the nonconvex curve (4.2), using $N = 128$ points. From Figure 8(a), we can see that when the initial mesh is obtained using a uniform u-parameterization, the distribution of points is far from ideal, with some areas of high curvature having poor resolution while others have severe nodal clustering. Similarly, Figure 8(b) illustrates that an equidistributed uniform arc-length mesh is also poor at resolving areas of high curvature. The best initial mesh is obtained from an equidistributed curvature-based monitor function $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ (Figure 8(c)), where we can observe a good balance of the distribution of mesh points towards high-curvature regions and areas of low curvature.

Following the ellipse example (section 4.2), the initial position of the grid nodes is determined by the de Boor algorithm (section 3.1). In all simulations, we run to a final time of $T = 0.25$. Once again, we assume a spatial balancing operator of the form $P = M|\boldsymbol{x}_\xi|^2$. The temporal convergence was tested for both the BE and CNBE schemes, using a fine mesh resolution of $N = 10^3$. Figure 9(a) illustrates the temporal convergence when $M = 1$. It is clear that the BE scheme demonstrates first-order convergence for $\tau = 1$ and $\tau = 10$ while giving only partial results for $\tau = 10^{-5}$ and $\tau = 0.1$. No convergence results were obtained for $\tau = 10^{-3}$. For $\tau = 1$, the CNBE scheme demonstrates approximate second-order convergence and slightly less than second-order convergence for $\tau = 10$. Only partial results were obtained for $\tau = 0.1$, and no convergence is seen for $\tau = 10^{-5}$ and $\tau = 10^{-3}$. Figure 9(b) illustrates the temporal convergence when $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$. Once again, it is clear that the BE scheme demonstrates first-order convergence for $\tau = 1$ and $\tau = 10$



(a) $M = 1$.

(b) $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

FIG. 9. *Temporal convergence in the* $L^2([0, T])$ *norm of the approximation of the enclosed area when the nonconvex initial curve is evolved by curve shortening flow using the BE (solid line) and the CNBE (dashed line) scheme with* $P = M|\boldsymbol{x}_\xi|^2$ *and* (a) $M = 1$ *or* (b) $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

but only partial results for $\tau = 0.1$. Unlike when $M = 1$, convergence results could not be obtained for either $\tau = 10^{-5}$ or $\tau = 10^{-3}$. However, the CNBE scheme does not behave as we expect; the error for the CNBE is larger than that for the BE scheme for the smallest values of N_T. This error then plummets, achieving greater than second-order accuracy. In this example, there is substantial tangential motion as the regions of very high curvature flatten out, and this may be a source of additional error compared to other examples.

Following the previous examples (sections 4.1 and 4.2), we wish to illustrate the computational efficiency of the nonlinear Picard solver. As can be seen from Figure 9, for each monitor function, all values of τ do not perform similarly enough, with respect to the L^2 error measure, unlike the ellipse example (this is particularly true for the BE scheme). Therefore, for both monitor functions, we choose $\tau = 10$ as this seems to perform more robustly. Tables 5 and 6 display the maximum and minimum number of Picard iterations required for each scheme and for each temporal resolution considered when $\tau = 10$. For the smallest values of N_T (Table 5), we see that both the BE and CNBE schemes slightly struggle for both the uniform arc-length and curvature-based monitor functions. However, we note that as N_T is increased, the maximum number of Picard iterations decreases. It is therefore not surprising that the computational cost for the nonlinear Picard solver becomes far more reasonable for the largest values of N_T (Table 6).

The spatial convergence was once again tested using a large number of time steps $N_T = 10^4$. Figure 10(a) illustrates the spatial convergence of the CNBE scheme when $M = 1$ (solid line) and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ (dashed line). The convergence is clearly second order for all values of τ for $M = 1$, while for $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ second-order convergence is seen for all τ except $\tau = 10$, where greater than second-order convergence is seen. Crucially, however, the curvature-based monitor function produces an improved error compared with the uniform arc-length monitor function. This is due to the curve being more accurately approximated using a curvature-

TABLE 5

(Nonconvex.) Maximum and minimum number of Picard steps required for each scheme, with $N = 10^3$*, for each temporal resolution* $N_T = 10, 20, 40, 80, 160$ *when* $\tau = 10$ *and* $P = M|\boldsymbol{x}_\xi|^2$*, for both* $M = 1$ *and* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$*.*

	$M = 1$									
	$N_T = 10$		$N_T = 20$		$N_T = 40$		$N_T = 80$		$N_T = 160$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	21	8	18	7	17	5	18	5	16	4
CNBE	29	7	16	6	14	5	14	4	13	4

	$M = \frac{1}{2}(M_{\text{floor}} + \|\kappa\|^{1/2})$									
	$N_T = 10$		$N_T = 20$		$N_T = 40$		$N_T = 80$		$N_T = 160$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	14	9	13	7	11	6	11	5	11	4
CNBE	26	10	17	8	12	7	12	5	10	4

TABLE 6

(Nonconvex.) Maximum and minimum number of Picard steps required for each scheme, with $N = 10^3$*, for each temporal resolution* $N_T = 320, 640, 1280, 2560$ *when* $\tau = 10$ *and* $P = M|\boldsymbol{x}_\xi|^2$*, for both* $M = 1$ *and* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$*.*

	$M = 1$							
	$N_T = 320$		$N_T = 640$		$N_T = 1280$		$N_T = 2560$	
	Max	Min	Max	Min	Max	Min	Max	Min
BE	14	3	11	3	8	3	7	2
CNBE	11	3	9	3	8	3	6	2

	$M = \frac{1}{2}(M_{\text{floor}} + \|\kappa\|^{1/2})$							
	$N_T = 320$		$N_T = 640$		$N_T = 1280$		$N_T = 2560$	
	Max	Min	Max	Min	Max	Min	Max	Min
BE	11	3	9	3	7	3	6	2
CNBE	10	3	9	3	8	3	6	2

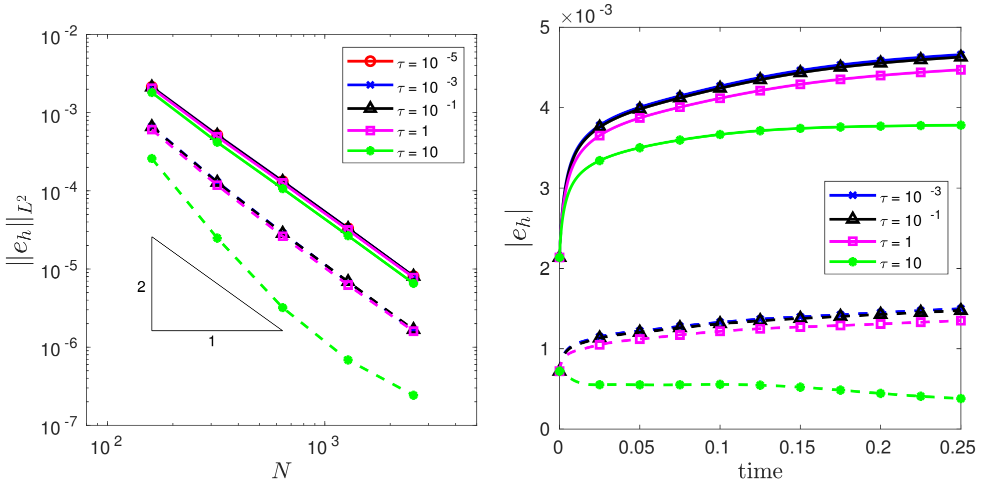
based monitor function compared with uniform arc-length. To this end, Figure 10(b) illustrates the absolute value error in the computed area for the same spatial balancing operator for both $M = 1$ and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$. Once again, it is clear that the curvature-based monitor function produces a much better mesh compared to a uniform arc-length.

Figure 10(a) shows that, for a given monitor function, all values of τ perform similarly except for the curvature-based monitor function when $\tau = 10$, where greater than second-order convergence was seen. Thus, for both monitor functions, Table 7 displays the maximum and minimum number of Picard iterations required for each scheme and for each spatial resolution considered when $\tau = 1$. As was seen for the previous examples (sections 4.1 and 4.2), the computational cost of the nonlinear Picard solver is low, requiring at most seven iterations.

4.4. Curve shortening flow with a singularity in finite time.

We now consider curve shortening flow for the initial curve given by the parameterization

$$x(u, 0) = \cos(4\pi u)\cos(2\pi u), \qquad 0 \le u \le 1, \tag{4.3}$$
$$y(u, 0) = \cos(4\pi u)\sin(2\pi u).$$



(a) *Spatial convergence.* (b) *Absolute value of the area error in time.*

FIG. 10. (a) *Spatial convergence in the* $L^2([0,T])$ *norm of the approximation of the enclosed area when the initial nonconvex curve is evolved by curve shortening flow using the CNBE scheme for all* τ *and for both* $M = 1$ *(solid line) and* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ *(dashed line).* (b) *Absolute value of the area error in time for the CNBE scheme with* $N = 160$ *for both* $M = 1$ *(solid line) and* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ *(dashed line).*

TABLE 7

(Nonconvex.) Maximum and minimum number of Picard steps required for each scheme, with $N_T = 10^4$*, for each spatial resolution when* $\tau = 1$ *and* $P = M|\boldsymbol{x}_\xi|^2$*, for both* $M = 1$ *and* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$*.*

	$M = 1$									
	$N = 160$		$N = 320$		$N = 640$		$N = 1280$		$N = 2560$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	5	2	6	2	6	2	6	2	7	2
CNBE	5	2	6	2	6	2	6	2	6	2

	$M = \frac{1}{2}(M_{\text{floor}} + \lvert\kappa\rvert^{1/2})$									
	$N = 160$		$N = 320$		$N = 640$		$N = 1280$		$N = 2560$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	4	2	5	2	5	2	5	2	5	2
CNBE	4	2	5	2	5	2	5	2	5	2

The initial curve is self-intersecting and develops a geometric singularity in a finite time. Once again, the initial position of the grid nodes is determined by the de Boor algorithm, and we assume a spatial balancing operator of the form $P = M|\boldsymbol{x}_\xi|^2$. The simulation was run until $T = 0.086$.

Figure 11 illustrates a comparison of a *gold standard* approximation, with a fine spatial resolution $N = 10^3$ (solid line), against a uniform arc-length approximation, where $M = 1$ (dotted line), and a curvature-based approximation, where $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ (line with markers). For the gold standard simulation we use $M = 1$ and note that, for a time step size of $\Delta t = 10^{-5}$, the simulation evolved smoothly for a choice of $\tau = 10^{-3}$. To allow comparison with the results presented in [15], the uniform arc-length and curvature-based simulations were carried out with $N = 64$. The time step size used in [15] is of the order $\Delta t = 10^{-4}$. To highlight the improvement from using a curvature-based grid over a uniform arc-length grid we use the same time step size as in the gold standard solution. The solutions are plotted at the same times as those in [15]. We can see that the coarse uniform arc-length grid is a reasonable approximation at $t = 0.02$, but due to the lack of resolution of high-curvature regions the accuracy deteriorates considerably to the extent that the singularity in the curve occurs between $t = 0.08$ (Figure 11(c)) and $t = 0.0828$ (Figure 11(d)). However, for the gold standard approximation, the singularity occurs between $t = 0.0829$ (Figure 11(e)) and $t = 0.086$ (Figure 11(f)). The curvature-based grid has an improved approximation when compared with the uniform arc-length grid for times $t \geq 0.08$ due to the improved resolution of high-curvature regions afforded by a curvature-based grid. The singularity occurs between $t = 0.0829$ and $t = 0.086$. As demonstrated by Elliott and Fritz [15], it is not possible for an iterative, fully implicit scheme to converge past the singularity. In [15], a semi-implicit approach was used, which enabled the numerics to jump the geometric singularity. Here, the semi-implicit nature is achieved by restricting the number of iterations used in the nonlinear Picard solver. For the BE scheme, a semi-implicit approach can be obtained by allowing only a single Picard iteration, while for the CNBE scheme (depicted in Figure 11) this limit is two.

4.5. Forced curve shortening flow.

Thus far we have only considered classical curve shortening flow $\dot{\boldsymbol{x}} \cdot \boldsymbol{n} = \kappa$. In this section, we add a constant forcing so that

$$\dot{\boldsymbol{x}} \cdot \boldsymbol{n} = \alpha\kappa + \beta,$$

where $\alpha = \alpha(\boldsymbol{x}, t) = 1$ and $\beta = \beta(\boldsymbol{x}, t) = 10$. Unfortunately, for forced curve shortening flow, we do not possess an exact solution for the evolving area. Therefore, we define the error in the approximation of the enclosed area at time $t \in (0, T]$ by $e_h(t) := A_h(t) - A_{gs}(t)$, where $A_{gs}(t)$ is the enclosed area for the *gold standard* approximation and A_h is as defined previously (see section 4.1). Convergence will be studied using the absolute value error

$$|e_h| = |A_h - A_{gs}|\,.$$

4.5.1. Forced curve shortening flow of a unit l_p-ball.

In this section, our initial curve is defined as a unit l_p-ball,

$$\Gamma_p = \left\{\boldsymbol{x} \in \mathbb{R}^2 \colon \|\boldsymbol{x}\|_p = 1\right\},$$

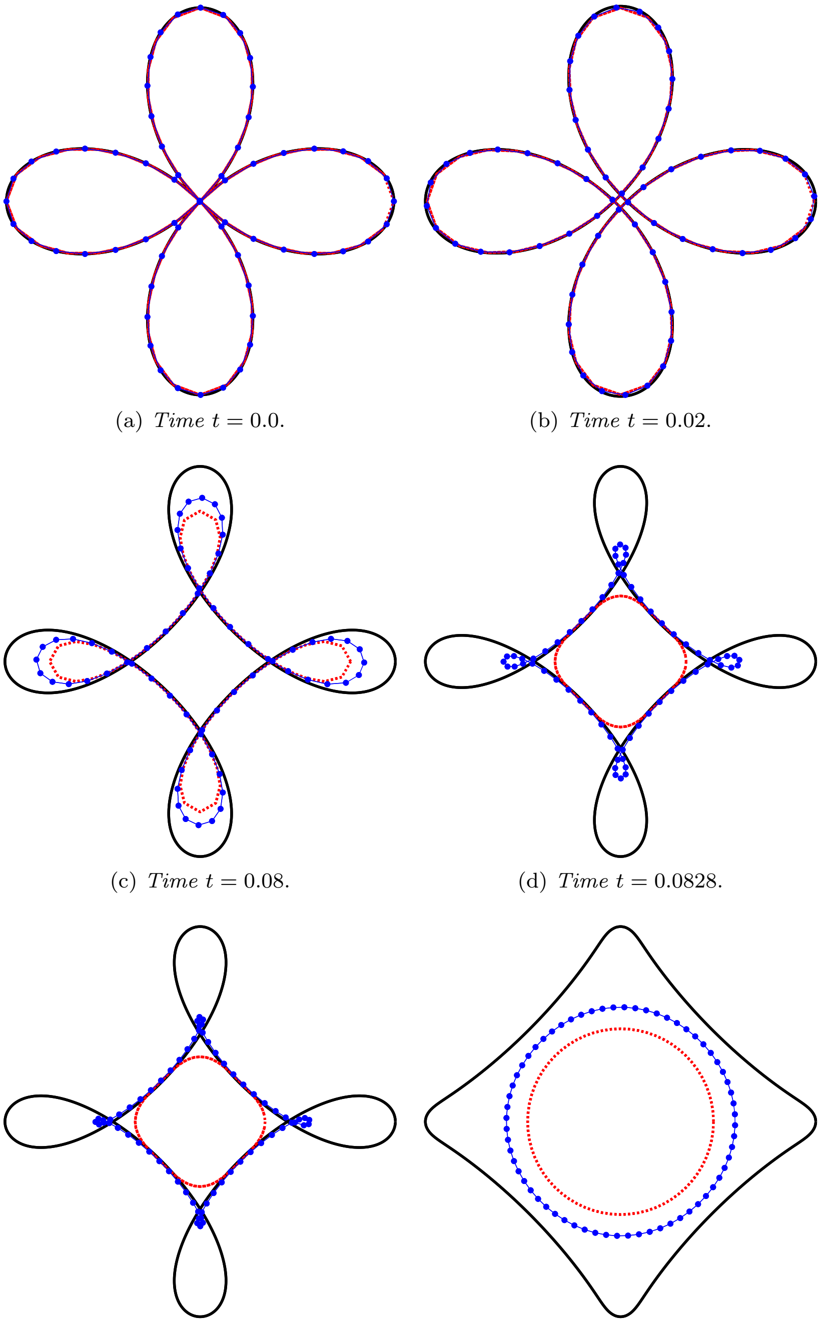
where for any $\boldsymbol{x} = (x, y) \in \mathbb{R}^2$, the l_p norm is defined as $\|\boldsymbol{x}\|_p = (|x|^p + |y|^p)^{1/p}$. Following previous sections, we define the curve parametrically as

$$\begin{aligned} x(u,0) &= r(u)\cos(2\pi u), \qquad 0 \leq u \leq 1, \\ y(u,0) &= r(u)\sin(2\pi u), \end{aligned} \tag{4.4}$$

where we assume that the radius depends on the parameter u. Substituting this into the definition of Γ_p yields the following expression for the radius:

$$r(u) = \frac{1}{(|\cos(2\pi u)|^p + |\sin(2\pi u)|^p)^{1/p}}. \tag{4.5}$$

Throughout this section, we assume that $p = 10$.


(a) *Time* $t = 0.0$. (b) *Time* $t = 0.02$.

(c) *Time* $t = 0.08$. (d) *Time* $t = 0.0828$.

(e) *Time* $t = 0.0829$. (f) *Time* $t = 0.086$.

FIG. 11. *Curve shortening flow of the self-intersecting curve* (4.3) *using the CNBE scheme, for* $\tau = 10^{-3}$ *and* $\Delta t = 10^{-5}$, *comparing a* gold standard *approximation with* $N = 10^3$ *(solid line) against a uniform arc-length approximation* $M = 1$ *(dotted line) and a curvature-based approximation* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ *(line with markers). As in* [15], *images are rescaled. (See online version for color.)*

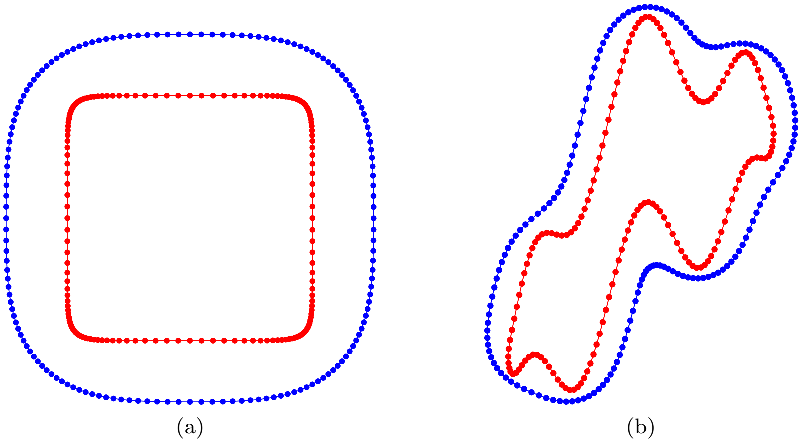


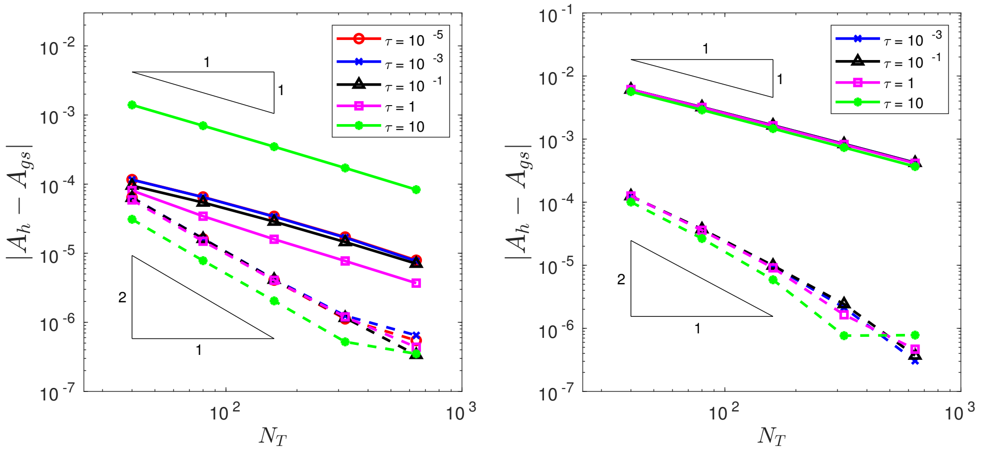
Fig. 12. *Initial (inner curve) and final (outer curve) mesh partitioning of* (a) *the* l_p*-ball* (4.4) *and* (b) *the nonconvex initial curve* (4.2) *using* $N = 160$ *points. The initial mesh is obtained by equidistributing the curvature-based monitor function* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$*.) (See online version for color.)*

Figure 12(a) illustrates the initial (inner curve) and final (outer curve) mesh partitioning of the curve defined as an l_p-ball of order $p = 10$ using $N = 160$ points. The initial mesh is obtained by equidistributing the curvature-based monitor function. Using the given values of the parameters α and β, it is clear that the addition of a constant forcing term causes the length of the curve to increase in time rather than decrease. Also, it is clear that the curvature-based monitor function obtains a high resolution around the high-curvature regions.

In all simulations, we run to a final time of $T = 0.05$. Once again, we assume a spatial balancing operator of the form $P = M|\boldsymbol{x}_\xi|^2$. The temporal convergence was tested for both the BE and CNBE schemes using a fine mesh resolution of $N = 10^4$. Figure 13(a) illustrates the temporal convergence when $M = 1$. It is clear that the BE scheme demonstrates first-order convergence for all values of τ. For $\tau = 10$, the error is significantly larger than for the other values of τ. The CNBE scheme demonstrates second-order convergence for all τ. Figure 13(b) illustrates the temporal convergence when $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$. Once again, it is clear that the BE scheme demonstrates first-order convergence for all τ, and, unlike when $M = 1$, the error value is similar for all τ. The CNBE scheme also once again demonstrates second-order convergence for all τ (except $\tau = 10^{-5}$, where the nonlinear Picard solver could not converge) and that the error values are similar.

Following the classical curve shortening examples (sections 4.1, 4.2, and 4.3), we wish to illustrate the computational efficiency of the nonlinear Picard solver. Table 8 displays the maximum and minimum number of Picard iterations required for each scheme and for each temporal resolution considered when $\tau = 1$ for both monitor functions. It is clear that the computational cost of both the BE and CNBE schemes is small and that using a curvature-based monitor function does not substantially increase the computational cost of the Picard solver.

The spatial convergence was tested using a large number of time steps, $N_T = 10^4$. Figure 14(a) illustrates the spatial convergence of the CNBE scheme when $M = 1$ (solid line) and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ (dashed line). The convergence is clearly



(a) $M = 1$.

(b) $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

FIG. 13. *Temporal convergence in the absolute value error of the approximation of the enclosed area when the initial l_p-ball is evolved by forced curve shortening flow using the BE (solid line) and CNBE (dashed line) schemes with $P = M|\boldsymbol{x}_\xi|^2$ and* (a) $M = 1$ *or* (b) $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

TABLE 8

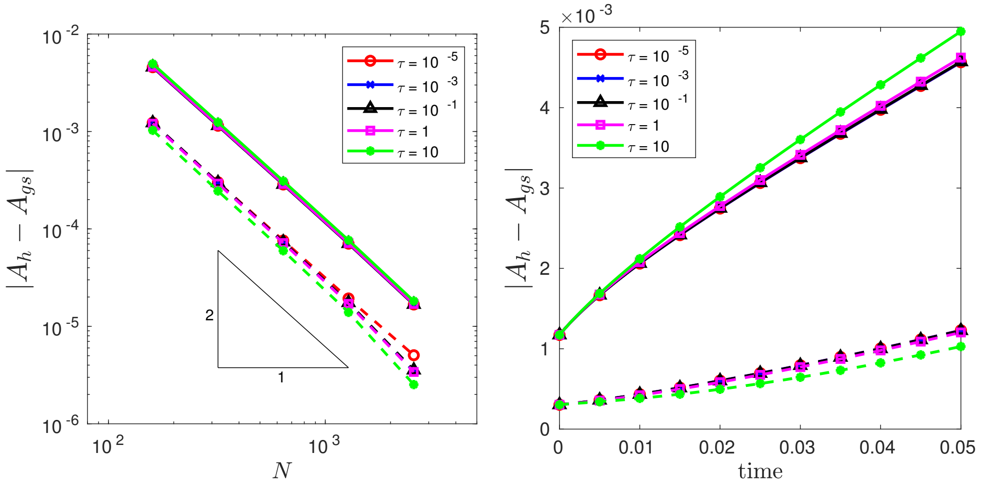
(Forced l_p-ball.) Maximum and minimum number of Picard steps required for each scheme, with $N = 10^4$, for each temporal resolution $N_T = 40, 80, 160, 320, 640$ when $\tau = 1$ and $P = M|\boldsymbol{x}_\xi|^2$, for both $M = 1$ and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

	$M = 1$									
	$N_T = 40$		$N_T = 80$		$N_T = 160$		$N_T = 320$		$N_T = 640$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	6	4	5	4	5	3	4	3	3	3
CNBE	6	4	5	4	5	3	4	3	3	3

	$M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$									
	$N_T = 40$		$N_T = 80$		$N_T = 160$		$N_T = 320$		$N_T = 640$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	7	5	6	4	5	4	4	3	3	3
CNBE	7	5	6	4	5	4	4	3	3	3

second-order for all values of τ for both $M = 1$ and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$. Crucially, once again the curvature-based monitor function produces an improved error compared with the uniform arc-length monitor function. As before, this is due to the curve being more accurately approximated using a curvature-based monitor function compared to uniform arc-length. To this end, Figure 14(b) illustrates the evolution of the absolute value error in time for the same spatial balancing operator for both $M = 1$ (solid line) and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ (dashed line). Once again, it is clear that the curvature-based monitor function produces a much better mesh compared to uniform arc-length.

Figure 14(a) shows that, for a given monitor function, all values of τ perform similarly. Thus, for both monitor functions, Table 9 displays the maximum and minimum number of Picard iterations required for each scheme and for each spatial resolution considered when $\tau = 1$. Once again, we observe that the computational



(a) *Spatial convergence.* (b) *Absolute value of the area error in time.*

Fig. 14. (a) *Spatial convergence in the absolute value error of the approximation of the enclosed area when the initial l_p-ball is evolved by forced curve shortening flow using the CNBE scheme for all τ and for both $M = 1$ (solid line) and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ (dashed line).* (b) *Absolute value error in time for the CNBE scheme with $N = 160$ for both $M = 1$ (solid line) and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ (dashed line).*

Table 9

(Forced l_p-ball.) Maximum and minimum number of Picard steps required for each scheme, with $N_T = 10^4$, for each spatial resolution when $\tau = 1$ and $P = M|\boldsymbol{x}_\xi|^2$, for both $M = 1$ and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

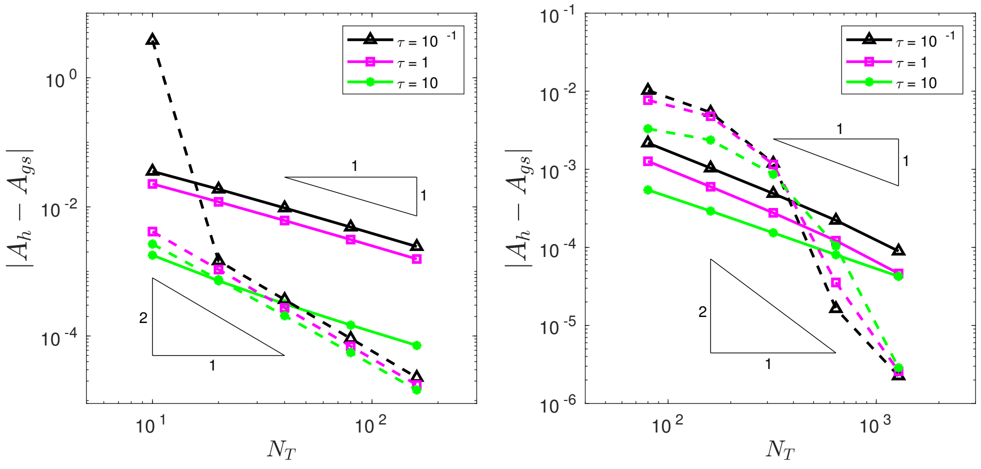
	$M = 1$									
	$N = 160$		$N = 320$		$N = 640$		$N = 1280$		$N = 2560$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	2	2	2	2	2	2	2	2	2	2
CNBE	2	2	2	2	2	2	2	2	2	2

	$M = \frac{1}{2}(M_{\text{floor}} + \lvert\kappa\rvert^{1/2})$									
	$N = 160$		$N = 320$		$N = 640$		$N = 1280$		$N = 2560$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	2	2	2	2	2	2	2	2	2	2
CNBE	2	2	2	2	2	2	2	2	2	2

cost of the nonlinear Picard solver is minimal, requiring at most two iterations per time step.

4.5.2. Forced curve shortening flow of a nonconvex initial curve. In this section, the initial nonconvex curve is described parametrically as in (4.2). Figure 12(b) illustrates the initial (inner curve) and final (outer curve) mesh partitioning of the curve defined by (4.2) using $N = 160$ points. Once again, the initial mesh is obtained by equidistributing the curvature-based monitor function. This example was previously considered by Balažovjech and Mikula [2]. The final mesh position depicted in Figure 12(b) demonstrates good agreement with the final mesh position presented in [2] (their Figure 4.2). Note that the forcing constant used here ($\beta = 10$) is the same as that in [2].

Following [2], all simulations ran to a final time of $T = 0.02$. Once again, we



(a) $M = 1$. (b) $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

FIG. 15. *Temporal convergence in the absolute value error of the approximation of the enclosed area when the nonconvex initial curve is evolved by forced curve shortening flow using the BE (solid line) and CNBE (dashed line) schemes with* $P = M|\boldsymbol{x}_\xi|^2$ *and* (a) $M = 1$ *or* (b) $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

assume a spatial balancing operator of the form $P = M|\boldsymbol{x}_\xi|^2$. The temporal convergence was tested for both the BE and CNBE schemes using a fine mesh resolution of $N = 10^4$. Figure 15(a) illustrates the temporal convergence when $M = 1$. No convergence results were obtained for either the BE or the CNBE scheme as the nonlinear Picard solver could not converge for $\tau = 10^{-5}$ and $\tau = 10^{-3}$. It is clear that the BE scheme demonstrates first-order convergence for all other values of τ. For $\tau = 10$, the error is significantly lower than for the other values of τ. The CNBE scheme demonstrates second-order convergence for $\tau = 0.1$, $\tau = 1$, and $\tau = 10$. Figure 13(b) illustrates the temporal convergence when $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$. Once again, it is clear that for $\tau = 0.1$, $\tau = 1$, and $\tau = 10$ the BE scheme demonstrates first-order convergence, while for the CNBE scheme, where the convergence rate is greater than second-order, we see the same accelerated convergence that was previously demonstrated (Figure 9(b)).

To illustrate the computational efficiency of the nonlinear Picard solver, we choose $\tau = 1$. Tables 10 and 11 display the maximum and minimum number of Picard iterations required for each scheme and for each temporal resolution considered when $\tau = 1$ for both monitor functions. For the smallest values of N_T (Table 10) we see that when $M = 1$, both the BE and CNBE schemes struggle and require a large number of iterations per time step, but as N_T is increased this number decreases. Table 10 also demonstrates an improvement in the maximum number of Picard iterations when the curvature-based monitor function is used. Indeed, this improvement is fairly substantial where a drop from 58 to 31 iterations can be observed for the CNBE scheme when $N_T = 10$. Once again, as N_T is increased, the maximum number of iterations decreases. For $N_T \geq 320$ (Table 11) the computational cost of the nonlinear Picard solver is once again low and continues to decrease as N_T is increased for both schemes and for both monitor functions.

The spatial convergence was tested using a large number of time steps $N_T = 10^4$.

Table 10

(Forced nonconvex.) Maximum and minimum number of Picard steps required for each scheme, with $N = 10^4$*, for each temporal resolution* $N_T = 10, 20, 40, 80, 160$ *when* $\tau = 1$ *and* $P = M|\boldsymbol{x}_\xi|^2$*, for both* $M = 1$ *and* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$*.*

	$M = 1$									
	$N_T = 10$		$N_T = 20$		$N_T = 40$		$N_T = 80$		$N_T = 160$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	51	18	45	13	35	9	25	7	17	5
CNBE	58	20	45	14	34	10	24	7	17	5

	$M = \frac{1}{2}(M_{\text{floor}} + \|\kappa\|^{1/2})$									
	$N_T = 10$		$N_T = 20$		$N_T = 40$		$N_T = 80$		$N_T = 160$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	30	14	23	10	16	8	12	6	9	6
CNBE	31	16	23	9	17	8	12	7	9	6

Table 11

(Forced nonconvex.) Maximum and minimum number of Picard steps required for each scheme, with $N = 10^4$*, for each temporal resolution* $N_T = 320, 640, 1280, 2560$ *when* $\tau = 1$ *and* $P = M|\boldsymbol{x}_\xi|^2$*, for both* $M = 1$ *and* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$*.*

	$M = 1$							
	$N_T = 320$		$N_T = 640$		$N_T = 1280$		$N_T = 2560$	
	Max	Min	Max	Min	Max	Min	Max	Min
BE	12	4	8	3	6	3	5	2
CNBE	12	4	8	3	6	3	4	2

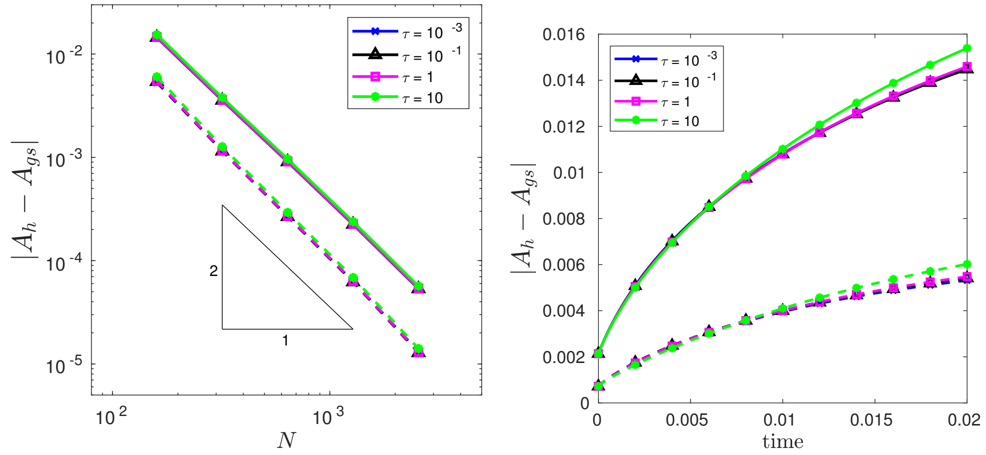
	$M = \frac{1}{2}(M_{\text{floor}} + \|\kappa\|^{1/2})$							
	$N_T = 320$		$N_T = 640$		$N_T = 1280$		$N_T = 2560$	
	Max	Min	Max	Min	Max	Min	Max	Min
BE	8	5	6	5	6	4	5	4
CNBE	7	6	7	5	6	4	5	4

Figure 16(a) illustrates the spatial convergence of the CNBE scheme when $M = 1$ (solid line) and $M = \frac{1}{2}(M_{\text{floor}}+|\kappa|^{1/2})$ (dashed line). The convergence is clearly second order for all values of τ for both $M = 1$ and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$. Crucially, once again the curvature-based monitor function produces an improved error compared to the uniform arc-length monitor function. Figure 16(b) illustrates the evolution of the absolute value error in time for the same spatial balancing operator for both $M = 1$ (solid line) and $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ (dashed line). Once again, it is clear that the curvature-based monitor function produces a much better mesh compared to uniform arc-length. Note that no convergence results were obtained for $\tau = 10^{-5}$.

Figure 16(a) shows that, for a given monitor function, all values of τ perform similarly. Thus, for both monitor functions, Table 12 displays the maximum and minimum number of Picard iterations required for each scheme and for each spatial resolution considered when $\tau = 1$. Once again, we observe that the computational cost of the nonlinear Picard solver is minimal, requiring at most three iterations per time step.

5. Conclusions.

We have presented an adaptive moving mesh method to simulate forced curve shortening flow. The method features a new strategy to control



(a) *Spatial convergence.* (b) *Absolute value of the area error in time.*

FIG. 16. (a) *Spatial convergence in the absolute value error of the approximation of the enclosed area when the nonconvex initial curve is evolved by forced curve shortening flow using the CNBE scheme for all* τ *and for both* $M = 1$ *(solid line) and* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ *(dashed line).* (b) *Absolute value error in time for the CNBE scheme with* $N = 160$ *for both* $M = 1$ *(solid line) and* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$ *(dashed line).*

TABLE 12

(Forced nonconvex.) Maximum and minimum number of Picard steps required for each scheme, with $N_T = 10^4$*, for each spatial resolution when* $\tau = 1$ *and* $P = M|\boldsymbol{x}_\xi|^2$*, for both* $M = 1$ *and* $M = \frac{1}{2}(M_{\text{floor}} + |\kappa|^{1/2})$.

	$M = 1$									
	$N = 160$		$N = 320$		$N = 640$		$N = 1280$		$N = 2560$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	3	2	3	2	3	2	3	2	3	2
CNBE	3	2	3	2	3	2	3	2	3	2

	$M = \frac{1}{2}(M_{\text{floor}} + \lvert\kappa\rvert^{1/2})$									
	$N = 160$		$N = 320$		$N = 640$		$N = 1280$		$N = 2560$	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
BE	2	2	3	2	3	2	3	2	3	2
CNBE	2	2	3	2	3	2	3	2	3	2

mesh movement in the tangential direction using a curvature-based monitor function. A novel hybrid time integration scheme has also been proposed. For classical and forced curve shortening flow of convex curves, the numerical experiments indicate that the method is spatially and temporally second-order accurate. We demonstrated the importance of the initial mesh for producing consistent convergence results (section 4.2.1) and presented a generalization of the de Boor algorithm that can be used to generate initially equidistributed grids (section 3.1).

For nonconvex curves evolved by classical and forced curve shortening flow, we found second-order temporal convergence when the uniform arc-length monitor function was used (Figures 9(a) and 15(a)) and at least second-order convergence when the curvature-based monitor function was used (Figures 9(b) and 15(b)). Analysis of this interesting observation is beyond the scope of this article and therefore is left as future work. Spatial second-order convergence was seen for classical and forced curve shortening of a nonconvex initial curve. Use of a curvature-based monitor function, has been shown to improve solution accuracy compared to the use of uniform arc-length meshes.

Our approach requires the solution of a nonlinear system, which we chose to obtain using Picard iterations. It was demonstrated that the computational cost of the nonlinear Picard solver is reasonable and that using a curvature-based monitor function did not significantly impact this cost. Here we enforced the nonlinear Picard solver to iterate to convergence (as was done in [3, 4]) and stopped the simulation when the solver was unable to converge. The lack of convergence in the Picard solver could be used as an indication of required temporal refinement. This interesting study of temporal adaptivity is beyond the scope of this article and therefore is also left as future work. For curve shortening flow with a singularity in finite time (section 4.4), we demonstrated that the lack of convergence of the nonlinear solver can be circumvented by fixing the maximum number of Picard iterations at a low value, thus allowing the numerics to *skip* the singularity, which is analogous to a semi-implicit approach [15].

Some immediate extensions include the application of the method to image segmentation problems and anisotropic curve evolution problems. The method can also be applied to physical or biological problems where the driving force of interface motion depends on field variables located on the interface. An important situation where this occurs is in the modeling of eukaryotic cell migration and chemotaxis, where the cell membrane is the interface between the extracellular and intracellular regions. Recent computational models assume that membrane motion is driven by mechanical and biochemical forces which depend on the cell receptor and protein densities on the membrane as well as the membrane curvature [28, 27, 29, 21, 22]. For all of these problems it is possible to use the adaptive moving mesh approach proposed here to resolve solution features along the interface. It remains to be seen how to devise a suitable monitor function to redistribute mesh points to simultaneously resolve interface geometry and interface-bound solution fields.

In the future we also plan to extend the adaptive moving mesh technique to the evolution of surfaces in three dimensions. For this class of problems it is especially important to devise a tangential velocity field to avoid problems with degenerating grid quality. Several methods have recently been proposed based on the control of volume, angle, and length metrics [23, 26]; discrete conformal mappings [4]; and harmonic mappings [15]. Some work has also been done on the use of moving mesh methods for stationary surfaces, including a sphere [13, 34, 35], and parametric surfaces [9]. However, none of these methods has been specifically devised to include solution adaptivity on evolving surfaces. It is hoped that the experience of applying moving mesh methods for two-dimensional planar problems [7] can be used to develop robust and adaptive surface evolution techniques to be applied to the solutions of PDEs on evolving surfaces.

Acknowledgment. The first author thanks the Isaac Newton Institute for Mathematical Sciences for its hospitality during the programme “Coupling Geometric PDEs with Physics for Cell Morphology, Motility and Pattern Formation.”

REFERENCES

[1] S.B. Angenent and M.E. Gurtin, *Multiphase thermodynamics with an interfacial structure* 2. *Evolution of an isothermal interface*, Arch. Ration. Mech. Anal., 108 (1989), pp. 323–391.

[2] M. Balažovjech and K. Mikula, *A higher order scheme for a tangentially stabilized plane curve shortening flow with a driving force*, SIAM J. Sci. Comput., 33 (2011), pp. 2277–2294, https://doi.org/10.1137/100795309.

[3] J.W. Barrett, H. Garcke, and R. Nürnberg, *On the variational approximation of combined second and fourth order geometric evolution equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1006–1041, https://doi.org/10.1137/060653974.

[4] J.W. Barrett, H. Garcke, and R. Nürnberg, *On the parametric finite element approximation of evolving hypersurfaces in* $\mathbb{R}^3$, J. Comput. Phys., 227 (2008), pp. 4281–4307.

[5] J.W. Barrett, H. Garcke, and R. Nürnberg, *The approximation of planar evolution by stable fully implicit finite element schemes that equidistribute*, Numer. Methods Partial Differential Equations, 27 (2011), pp. 1–30.

[6] 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, https://doi.org/10.1016/S0168-9274(99)00065-3.

[7] G. Beckett, J.A. Mackenzie, A. Ramage, and D.M. Sloan, *On the numerical solution of one-dimensional PDEs using adaptive methods based on equidistribution*, J. Comput. Phys., 167 (2001), pp. 372–392, https://doi.org/10.1006/jcph.2000.6679.

[8] C.J. Budd, W. Huang, and R.D. Russell, *Adaptivity with moving grids*, Acta Numer., 18 (2009), pp. 111–241.

[9] B. Crestel, R.D. Russell, and S.J. Ruuth, *Moving mesh methods on parametric surfaces*, Procedia Engineering, 124 (2015), pp. 148–160.

[10] C. de Boor, *Good approximation by splines with variables knots* II, in Proceedings of the Conference on the Numerical Solution of Differential Equations (Univ. Dundee, Dundee, 1973), Lecture Notes in Math. 363, Springer, 1974, pp. 12–20.

[11] K. Deckelnick, *Weak solutions of the curve shortening flow*, Calc. Var. Partial Differential Equations, 5 (1997), pp. 489–510.

[12] K. Deckelnick, G. Dzuik, and C.M. Elliott, *Computation of geometric partial differential equations and mean curvature flow*, Acta Numer., 14 (2005), pp. 139–232.

[13] Y. Di, R. Li, T. Tang, and P. Zhang, *Moving mesh methods for singular problems on a sphere using perturbed harmonic mappings*, SIAM J. Sci. Comput., 28 (2006), pp. 1490–1508, https://doi.org/10.1137/050642514.

[14] G. Dzuik and C.M. Elliott, *Finite element methods for surface PDEs*, Acta Numer., 22 (2013), pp. 289–396.

[15] C.M. Elliott and H. Fritz, *On approximations of the curve shortening flow and of the mean curvature flow based on the DeTurk trick*, IMA. J. Numer. Anal., 37 (2017), pp. 543–603.

[16] C.M. Elliott, B. Stinner, and C. Venkataraman, *Modelling cell motility and chemotaxis with evolving surface finite elements*, J. Roy. Soc. Interface, 9 (2012), pp. 3027–3044.

[17] T.Y. Hou, J.S. Lowengrub, and M.J. Shelley, *Removing the stiffness from interfacial flows with surface tension*, J. Comput. Phys., 114 (1994), pp. 312–338.

[18] W. Huang, *Practical aspects of formulation and solution of moving mesh partial differential equations*, J. Comput. Phys., 171 (2001), pp. 753–775.

[19] W. Huang and R.D. Russell, *Adaptive Moving Mesh Methods*, Appl. Math. Sci. 174, Springer, 2011.

[20] M. Kass, A. Witkin, and D. Terzopulos, *Snakes: Active contour models*, Int. J. Comput. Vis., 1 (1987), pp. 321–331.

[21] G. MacDonald, J.A. Mackenzie, M. Nolan, and R.H. Insall, *A computational method for the coupled solution of reaction-diffusion equations on evolving domains and manifolds: Application to a model of cell migration and chemotaxis*, J. Comput. Phys., 309 (2016), pp. 207–226, https://doi.org/10.1016/j.jcp.2015.12.038.

[22] J.A. Mackenzie, M. Nolan, and R.H. Insall, *Local modulation of chemoattractant concentrations by single cells: Disection using a bulk-surface computational model*, Interface Focus, 6 (2016), 20160036, https://doi.org/10.1098/rsfs.2016.0036.

[23] K. Mikula, M. Remešíková, P. Sarkoci, and D. Ševčovič, *Manifold evolution with tangential redistribution of points*, SIAM J. Sci. Comput., 36 (2014), pp. A1384–A1414, https://doi.org/10.1137/130927668.

[24] K. Mikula and D. Ševčovič, *Evolution of plane curves driven by a nonlinear function of curvature and anisotropy*, SIAM J. Appl. Math., 61 (2001), pp. 1473–1501, https://doi.org/10.1137/S0036139999359288.

[25] K. Mikula and D. Ševčovič, *A direct method for solving an anisotropic mean curvature flow of plane curves with an external force*, Math. Methods Appl. Sci., 27 (2004), pp. 1545–1565.

[26] S. Morigi, *Geometric surface evolution with tangential contribution*, J. Comput. Appl. Math., 233 (2010), pp. 1277–1287.

[27] M.P. Neilson, D. Veltman, P.J.M. van Haastert, S.D. Webb, J.A. Mackenzie, and R.H. Insall, *Chemotaxis: A feedback-based computational model robustly predicts multiple aspects of real cell behaviour*, PLoS Biol., 9 (2011), e1000618, https://doi.org/10.1371/journal.pbio.1000618.

[28] M.P. Neilson, J.A. Mackenzie, S.D. Webb, and R.H. Insall, *Use of the parameterised finite element method to robustly and efficiently evolve the edge of a moving cell*, Integr. Biol., 2 (2010), pp. 687–695, https://doi.org/10.1039/c0ib00047g.

[29] M.P. Neilson, J.A. Mackenzie, S.D. Webb, and R.H. Insall, *Modelling cell movement and chemotaxis using pseudopod-based feedback*, SIAM J. Sci. Comput., 33 (2011), pp. 1035–1057, https://doi.org/10.1137/100788938.

[30] S.J. Osher and R.P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer-Verlag, 2002.

[31] J.M. Sanz-Serna and I. Christie, *A simple adaptive technique for nonlinear wave problems*, J. Comput. Phys., 67 (1986), pp. 348–360.

[32] J.A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Materials Science*, Cambridge University Press, 1999.

[33] D. Ševčovič and S. Yazaki, *Evolution of plane curves with a curvature adjusted tangential velocity*, Jpn. J. Ind. Appl. Math., 28 (2011), pp. 413–442.

[34] H. Weller, P. Browne, C. Budd, and M. Cullen, *Mesh adaptation on the sphere using optimal transport and the numerical solution of a Monge–Ampère type equation*, J. Comput. Phys., 308 (2016), pp. 102–123.

[35] A.T.T. McRae, C.J. Cotter, and C.J. Budd, *Optimal-transport–based mesh adaptivity on the plane and sphere using finite elements*, SIAM J. Sci. Comput., 40 (2018), pp. A1121–A1148, https://doi.org/10.1137/16M1109515.