AMSI VACATION RESEARCH SCHOLARSHIPS 2018-2019

A Grid-based particle method for solving hyperbolic curvature flows

Solene Hegarty-Cremer
Supervised by Dr Pascal Buenzli
Queensland University of Technology

20/02/19

Vacation Research Scholarships are funded jointly by the Department of Education and Training and the Australian Mathematical Sciences Institute.

AMSI AUSTRALIAN MATHEMATICAL SCIENCES INSTITUTE

Australian Government
Department of Education and Training

Abstract

The aim of this project is to adapt the so-called grid-based particle method (GBPM) to simulate the motion of biological tissue boundaries subject to a certain type of geometric control. The grid-based particle method (GBPM) is a hybrid front-tracking method able to handle complex geometries and topological changes easily. The GBPM relies on marker particles on the moving interface as well as an underlying grid. The grid is used to redistribute the marker particles after their evolution, and to search for neighbouring marker particles. This method has been used successfully to evolve PDEs on evolving surfaces using standard finite difference schemes, and is therefore well-suited to be adapted for hyperbolic curvature flows. The method will be compared with previous results in simple tissue geometries, and extended to model situations of tissue fusion and fragmentation.

1 Introduction

Key to modelling complex physical and biological systems is the ability to track and evolve the system's boundaries and interfaces. This becomes a non-trivial problem when the interfaces evolve according to the dynamics of the system. Multiple methods have been developed to track these evolving interfaces and they fall within two broad categories: Lagrangian tracking methods and capturing methods. In Lagrangian tracking methods, the interface is explicitely represented by Lagrangian markers which are connected through a parameterisation of the surface. Front tracking methods are included in this category. The advantages of these methods are their efficiency, accuracy, and simplicity to implement. However, topological changes are difficult to track due to the linking between the Lagrangian markers. Furthermore, a quasi-uniform sampling of the surface is not easily maintained, which results in the surface having to be re-parameterised frequently as it is evolving. Capturing methods implicitly embed the interface in a scalar field function which is defined on an Eulerian mesh. Therefore evolving the surface involves evolving the scalar function. Under this category fall level set methods, the phase field method, and the volume of fluid method. These methods allow topological changes of the interface to be handled easily however their implicit nature results in less efficiency. Additionally, the representation of surface-bound quantities, such as surfactants, or biological cells, is also less convenient in these methods.

The method discussed in this report is a grid-based particle method (GBPM), which combines Lagrangian meshless particles over a fixed Eulerian mesh. This maintains quasi-uniform sampling and requires no information about the connection between points. This allows multiple surfaces to be tracked and in [2], it is shown that this method is able to resolve interactions between surfaces,

whether that be merging or crossing; this report explores the merging case. Furthermore, this method has a smaller increase in computational load with increase in grid resolution than capturing methods, as only the active grid-points surrounding the interface are surveyed when evolving the surface. This report discusses a MATLAB implementation of this method and replicating the results from [2]. Furthermore, as per [3], advecting a scalar quantity over the interface and solving a PDE on the interface is also explored.

2 Mathematical Algorithm

The interface is represented by a set of grid points $\{\mathbf{p}_0, \mathbf{p}_1, \ldots \mathbf{p}_i, \ldots \mathbf{p}_N\}$, which lie on the underlying Eulerian grid and have associated footpoints, $\mathbf{y}_i$, which lie on the interface. This is illustrated in Figure 1.

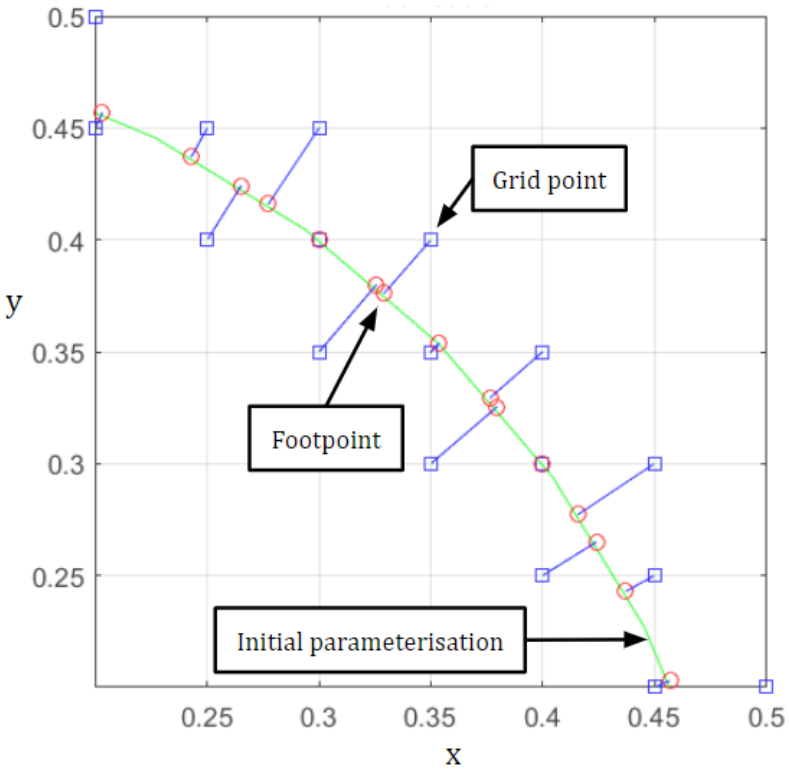

Figure 1: Illustrative initial interface

A grid point is only active if it is less than γ units from its footpoint, where γ is the radius of the computational tube. At each grid point, all required information about the surface at its footpoint,

including footpoint coordinates, the surface normal, and the curvature is stored. The interface is then evolved according to a velocity field which is applied to each footpoint. At each footpoint, the interface is interpolated locally and resampled to find the new closest footpoint to the active grid point. Once the interface has been resampled, all inactive gridpoints neighbouring active gridpoints are considered for activation. Neighbouring is taken to mean the 8 point stencil around a gridpoint. This involves a local surface interpolation around the proposed grid point, then determining if the proposed footpoint is within the computational tube. The following subsections define the algorithm in more detail.

Table 1: Table of Variables

Variable	**Notation**
i^{th} active grid point coordinates	$\mathbf{p}_i$
i^{th} footpoint coordinates	$\mathbf{y}_i$
Normal of the i^{th} footpoint before movement	$\mathbf{n}_i'$
Initial parameterisation of the interface	$\mathbf{F}(s)$
Local interpolation of the surface	$f(\tilde{x})$
Computational tube width	γ

2.1 Initialising the Interface

Given an initial parameterisation of the interface $\mathbf{F}(s)$, for each point in the underlying Eulerian grid, with spacing Δx, the closest footpoint on this interface is found. If the distance between the grid point and footpoint is less than γ, where distance is defined as

$$d(s, \mathbf{y}) = \frac{1}{2}||\mathbf{F}(s) - \mathbf{y}||^2 \tag{1}$$

the grid point is activated. The footpoint coordinates, as well as the inward-facing normal defined by the parameterisation and the curvature are stored. The interface is then resampled once according to the resampling technique described in the following section before movement is applied.

2.2 Movement and Resampling

In this study, movement is modelled using an imposed velocity field that can be dependent on position, normal direction, and/or curvature. This velocity field is applied to every footpoint after resampling in the first timestep then after activation every following timestep.

In order to keep the grid points referring to their closest point on the surface, after movement the surface must be resampled. Therefore a local representation of the surface is needed. For each grid point $\mathbf{p}_i$, at least m of the closest footpoints are found. These are then ordered according to their distance from $\mathbf{p}_i$ so that we have $\{\mathbf{y}_0, \mathbf{y}_1, ...\mathbf{y}_m, ...\}$, where $\mathbf{y}_0$ is the closest footpoint. Note that post-movement $\mathbf{y}_0$ may not be the footpoint associated with $\mathbf{p}_i$. Using only the closest m footpoints, the surface is then interpolated. We defined a local coordinate system, $\{\mathbf{n}_0'^{\perp}, \mathbf{n}_0'\}$ with $\mathbf{y}_0$ as the origin. All m footpoints are converted to this system such that we have m points $(\tilde{x}_j, \tilde{y}_j)$, with the $\tilde{x}_j$ lying between some $[\tilde{x}_{\min}, \tilde{x}_{\max}]$. A quadratic is then fitted through the footpoints in the local coordinate system using least-square fitting, that is the equation

$$\begin{bmatrix} m & \sum \tilde{x}_j & \sum \tilde{x}_j^2 \\ \sum \tilde{x}_j & \sum \tilde{x}_j^2 & \sum \tilde{x}_j^3 \\ \sum \tilde{x}_j^2 & \sum \tilde{x}_j^3 & \sum \tilde{x}_j^4 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} \sum \tilde{y}_j \\ \sum \tilde{x}_j \tilde{y}_j \\ \sum \tilde{x}_j^2 \tilde{y}_j \end{bmatrix} \tag{2}$$

is solved for $\boldsymbol{\alpha}$ such that our local interpolation is $f(\tilde{x}) = \alpha_0 + \alpha_1 \tilde{x} + \alpha_2 \tilde{x}^2$. The grid point $\mathbf{p}_i$ is converted to the local coordinates, and the value of $\tilde{x}$ which minimises the distance between $\tilde{\mathbf{p}}_i$ and $f(\tilde{x})$ is found and defined as the new footpoint, with local coordinates $\tilde{\mathbf{y}}^* = (\tilde{x}^*, f(\tilde{x}^*))$. This is achieved through the finding the root following minimiser,

$$\tilde{x}^3 + a_2 \tilde{x}^2 + a_1 \tilde{x} + a_0 = 0 \tag{3}$$

with

$$a_2 = \frac{3\alpha_1}{2\alpha_2} \qquad a_1 = \frac{\alpha_1^2 + 2\alpha_0\alpha_2 + 1}{2\alpha_2^2} \quad \text{and} \quad a_0 = \frac{\alpha_0\alpha_1}{2\alpha_2^2} \tag{4}$$

This is achieved by using the MATLAB function `fzero` with the local x-coordinate of the grid point as the starting point. The normal to the surface and the curvature are also updated according to the local interpolation. The normal is calculated using the following approximation

$$\tilde{\mathbf{n}}^* = \frac{\tilde{\mathbf{y}}^* - \tilde{\mathbf{p}}_i}{|\tilde{\mathbf{y}}^* - \tilde{\mathbf{p}}_i|}. \tag{5}$$

However, this approximation may result in switching the normal from inward-facing to outward facing, thus the following adjustment is made

$$\tilde{\mathbf{n}}^* = \text{sgn}[\tilde{\mathbf{n}}^* \cdot \tilde{\mathbf{n}}_0'] \tilde{\mathbf{n}}^* \tag{6}$$

where $\tilde{\mathbf{n}}^*$ is the new normal in local coordinates and $\tilde{\mathbf{n}}_0'$ is the old normal (pre-movement) in local coordinates. The curvature at the new footpoint is defined as

$$\kappa(\tilde{x}^*) = \frac{2\alpha_2}{[1 + (\alpha_1 + 2\alpha_2 \tilde{x}^*)^2]^{3/2}} \tag{7}$$

2.2.1 Differentiating between different sections of the interface

As the footpoints are not connected, a method of differentiating between different sections of the interface is needed to be able to detect when far portions of the interface are finding themselves close to each other in the Euclidian space, e.g. when these portions are about to collide. We use the difference in the normals associated with each of the m closest footpoints used for interpolation. The threshold on the difference between the normals $\mathbf{n}_0'$ and $\mathbf{n}_j'$ as

$$\mathbf{n}_0' \cdot \mathbf{n}_j' \leq \cos(\theta_{\max}) \tag{8}$$

If this condition is not satisfied, footpoint j is not used in the local interpolation. In this project, $\theta_{\max}$ was taken as $3\pi/4$. Note that since $\mathbf{y}_0$ is not necessarily the footpoint associated with the current grid point $\mathbf{p}_i$, $\mathbf{n}_0'$ is only defined after the search for the closest m footpoints.

2.3 Activating and Deactivating Points

2.3.1 Activating New Grid Points

Post movement and resampling, new grid points need to be activated to ensure the computational tube covers the interface. Under the assumption that the time step defined is suitably small, only inactive gridpoints neighbouring active grid points are considered. For each of these grid points, the resampling technique detailed in 2.2 is carried out, the only difference being that the current footpoint is not considered for interpolation as it is not yet defined. Once the point which minimises $d\left(f(\tilde{x}), \mathbf{y}_i^*\right)$ is found, the grid point is only activated if $d\left(f(\tilde{x}, \mathbf{y}_i^*\right) < \gamma$ and none of the deactivation rules as defined below are broken.

2.3.2 Deactivating Grid Points

There are four conditions under which an active grid point is deactivated:

1. The distance between the footpoint and grid point has exceeded γ
2. The minimising $\tilde{x}$ is outside interpolation domain $[x^*_{\min}, x^*_{\max}]$
3. The curvature at the footpoint exceeds the curvature threshold, taken as $1/\Delta x$
4. A collision is detected, that is, the scalar product of the normals is greater than $\cos(3\pi/4)$ and the distance between the footpoints is less than 2γ

In this project, interfaces which collide with each other are considered to merge. This is implemented in the grid-based particle method by simply deactivating points that satisfy condition 4 above.

2.4 Advection and curvature-induced changes in surface concentrations

To model the advection of a surface-bound scalar, we add a property μ to each footpoint. This scalar value is carried with the footpoint when the interface is evolved and resampled with a similar process to the resampling detailed in 2.2. We use the same local-coordinate system, however we now interpolate between the μ values, such that the least-squaresq equation becomes

$$\begin{bmatrix} m & \sum \tilde{x}_j & \sum \tilde{x}_j^2 \\ \sum \tilde{x}_j & \sum \tilde{x}_j^2 & \sum \tilde{x}_j^3 \\ \sum \tilde{x}_j^2 & \sum \tilde{x}_j^3 & \sum \tilde{x}_j^4 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \sum \tilde{c}_j \\ \sum \tilde{x}_j \tilde{\mu}_j \\ \sum \tilde{x}_j^2 \tilde{\mu}_j \end{bmatrix}. \tag{9}$$

Hence solving for $\boldsymbol{\beta}$, we obtain $g(\tilde{x}) = \beta_0 + \beta_1 \tilde{x} + \beta_2 \tilde{x}^2$. However, in this case we do not need to minimise the distance to a grid point, we simply evaluate this function at $\tilde{x}^*$, hence $\mu^* = g(\tilde{x}^*)$ is the resampled scalar value.

To model curvature-based diffusion, changes in the concentration of a surface-bound quantity, such as a surfactant, or biological cells, due to changes in the local surface area of the interface, we solve the following ODE for the curvature-bound scalar,

$$\frac{d\mu}{dt} = \kappa \mu v_n. \tag{10}$$

following the derivation in [1]. We simulate these equations numerically by assuming a mean curvature flow model in which the normal velocity of the interface v_n is given by $v_n = \alpha\kappa$ and we apply the forward Euler method in the movement stage of every time step of the GBPM to solve in time.

3 Results

The GBPM was applied with different initial conditions and vector fields. The first of these being the translation of a circle, in this case the vector field was defined as $V_x = -0.01, V_y = -0.01$. The results can be seen in the figure below.

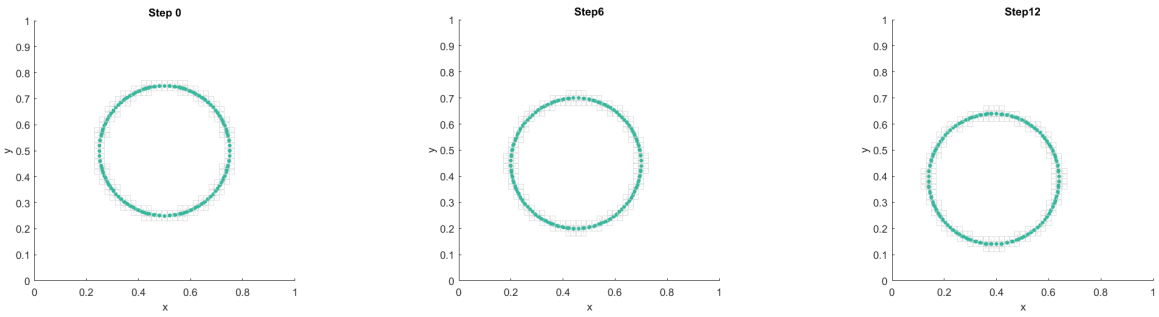

Figure 2: Circle under constant velocity field

The method was then applied to a single vortex flow.

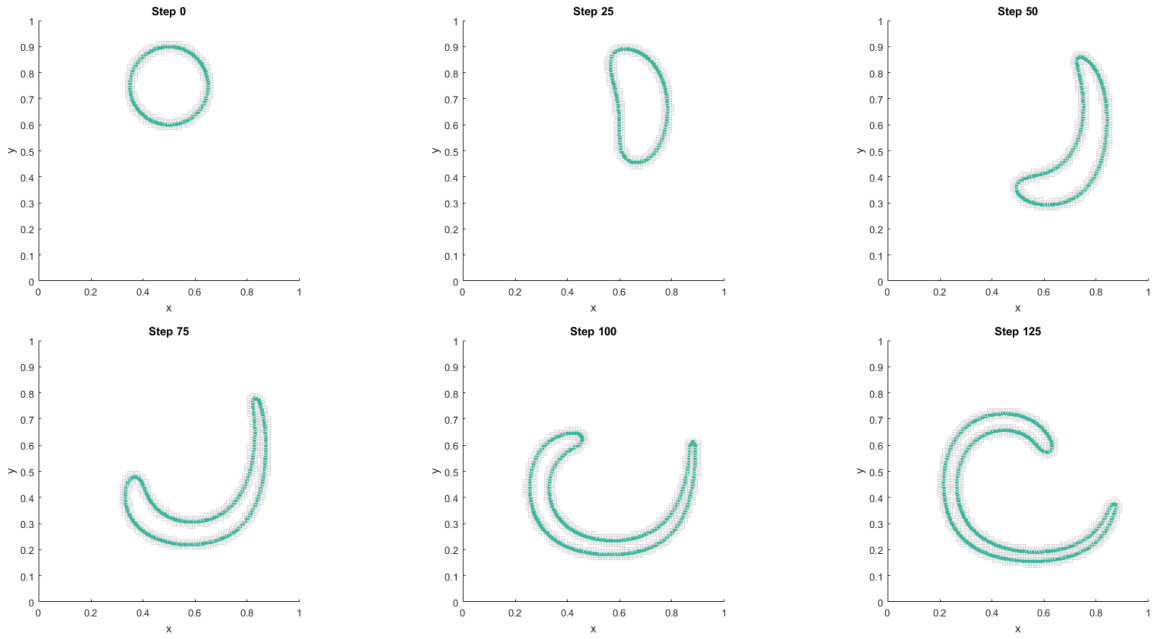

Figure 3: Single Vortex Flow

The initial interface was a circle centered at (0.5, 0.75) with a radius of 0.15. The velocity field is defined as

$$V_x = \sin\left(\pi x^2\right)\sin\left(\pi y\right)\cos\left(\pi y\right)$$
$$V_y = \sin\left(\pi y^2\right)\sin\left(\pi x\right)\cos\left(\pi x\right)$$

Figure 3 shows the method tracks the interface acurately despite the narrow tail.
Intrinsic vector fields were then tested. Mean curvature flow as defined by

$$\mathbf{V} = v_n\mathbf{n} \qquad v_n = \alpha\kappa$$

was applied to different initial interfaces, where v_n is the normal velocity of the interface.

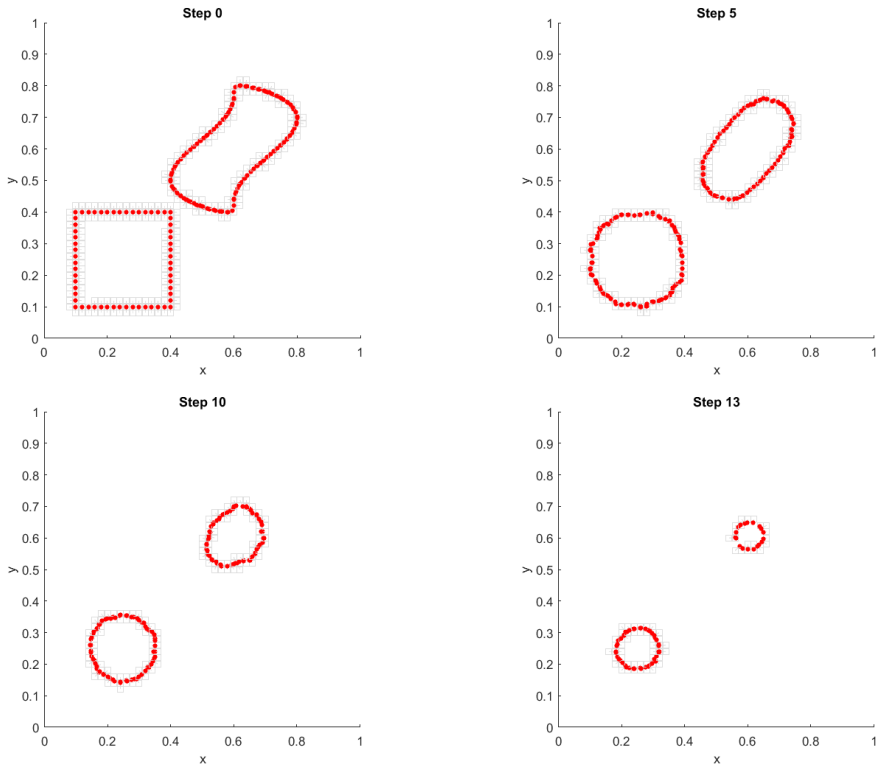

Figure 4: Mean Curvature Flow

As can be seen in Figure 4 the interfaces evolve into circles and shrink into points, as expected in mean-curvature flows.

The results for topological changes in the interface and advection of scalar values (without surface-

stretch-induced changes in these values) are shown below. The velocity field is defined as

$$\mathbf{V} = v_n \mathbf{n} \qquad v_n = c$$

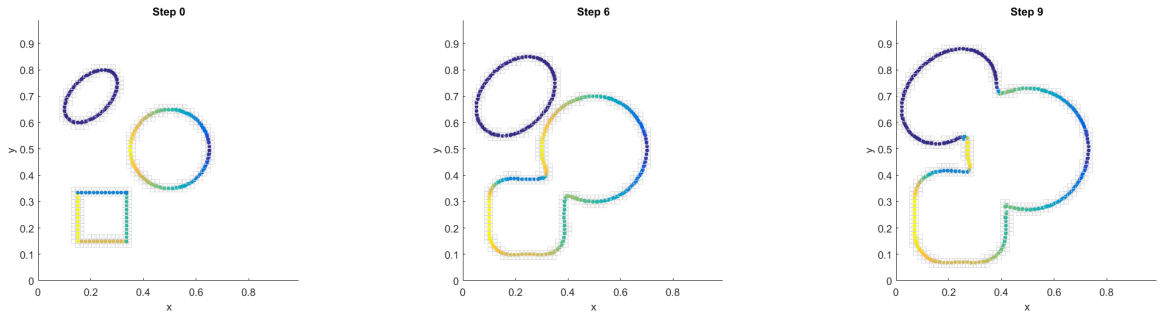

Figure 5: Topological Changes and Advection of Scalar Values

Figure 5 also shows the advection of scalar values. Scalars were arbitrarily assigned to each interface and advected. From figure **ref**, it is clear the scalar values are successfully advected, with some averaging between values at the intersection of the interfaces.

Finally, the ODE described in Equation 10 was solved on the interface. A forward Euler method is used to solve for density. As expected, the density increases at a rate proportional to the curvature of the surface.

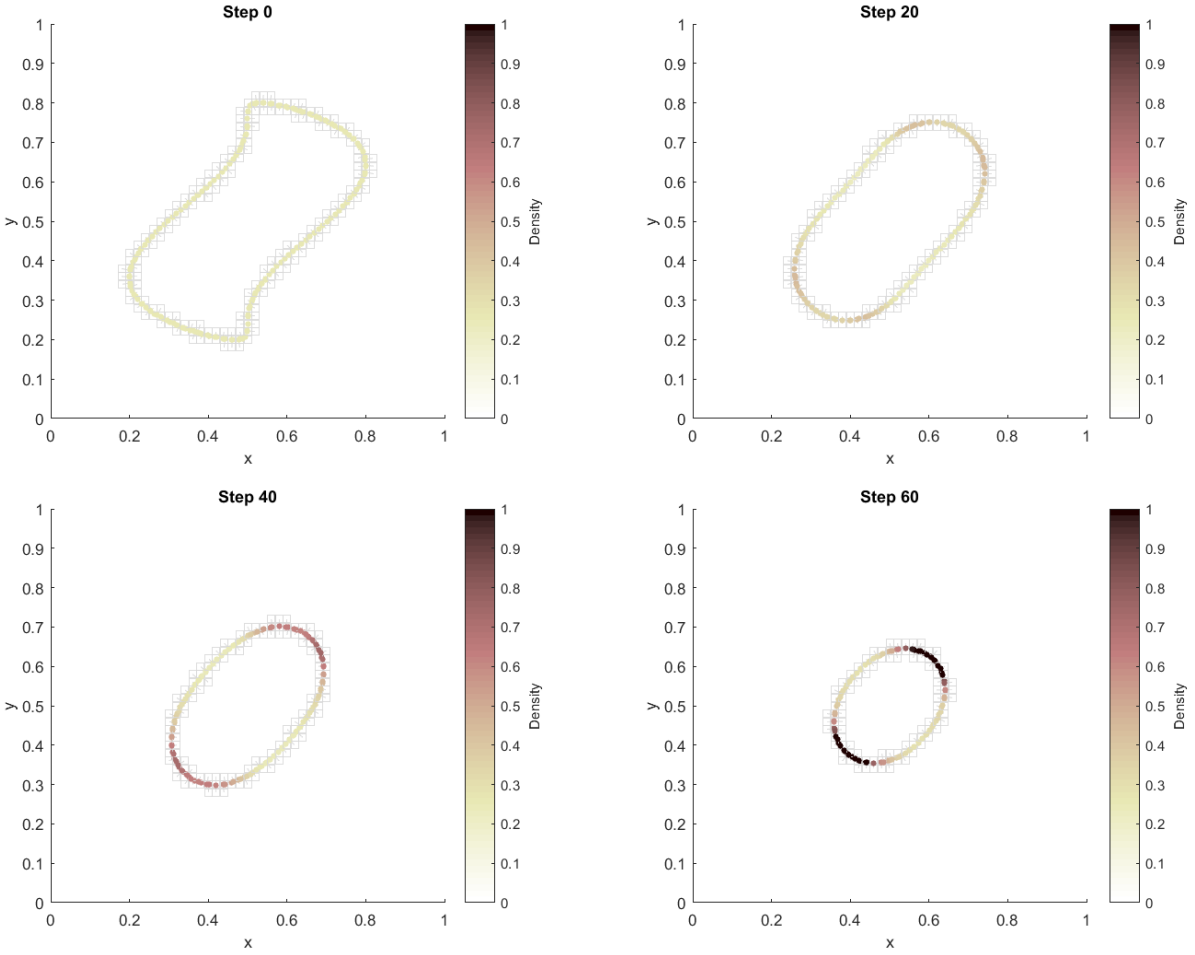

Figure 6: Curvature-Induced Density Changes

4 Discussion

The aim of this project was to implement the GBPM and apply it to simulate hyperbolic curvature flows. Although hyperbolic curvature flows were not simulated due to lack of time, it is clear that the GBPM is well suited to model these flows. It was shown that the method can track interfaces over external vector fields and intrinsic vector fields, including mean curvature flow. Topological changes in the interface(s) being tracked are also modelled effectively. Surface-bound scalar values can effectively be tracked and curvature-based density changes are modelled effectively.

The GBPM is seen to have multiple advantages over other interface tracking methods. Due to the underlying Eulerian grid, the method allows for efficient local (in the Euclidean space) search for colliding marker particles which provides a computationally efficient method of handling merging of interfaces.

Furthermore, increasing the number of independent interfaces tracked does not change or complicate the method. Despite the lack of information about the connection of the footpoints Figure **ref vortex** shows that differentiating between different sections of the interface using normals is effective. The curvature-induced density change modelling shows the advantage of the method in turning PDEs into ODEs to be solved on the surface as spatial derivatives can be modelled through evolving the interface.

The implementation of the grid-based particle method also revealed some issues with the method. Firstly, the method is sensitive to the definition of surface normals, and calculating the normal using the tangent to the local interpolation instead of the approximation (6) was necessary more frequently than detailed by [2]. This is straightfoward to implement for a quadratic local interpolation in 2D space, however needs to be considered if changing the order of interpolation and moving into 3D space. Furthermore, using MATLAB's inbuilt `fzero` on the minimizer defined by [2] did not always converge to the closest point on the interface to the grid point. This may be because two or more of the roots of the minimizer were closest to the initial starting guess (the local x-coordinate of the grid point). This resulted in having to use `fmindbd` on the distance function (1) and choosing the closest footpoint out of this result and the results from `fzero`. This is a component of the algorithm which needs to be explored further. Finally, in the local interpolation, because the closest m footpoints are found before we differentiate between the different sections of the interface, there may be less than m footpoints remaining which are on the 'correct' section of the interface. In some cases, this results in a close to singular matrix in (2). This could be corrected by undertaking a second search if there are less than m footpoints remaining after differentiating between different sections of the interface.

Although there are a few disadvantanges to the current implementation of the method, the numerous advantages of the GBPM encourage exploring further work with this method. Furthermore, the 3D variant of the method should be implemented. Finally hyperbolic curvature flows should be modelled as well as the advection **and** surface diffusion of scalar values along the interface.

5 Conclusion

This report has detailed an implmementation of the GBPM and has shown its capability to to track a variety of interfaces under different velocity fields. From this we can conclude that the method would be well suited to model hyperbolic curvature flows.

References

[0] [1] Alias, M. A. and Buenzli. P. R. 2017. Modeling the Effect of Curvature on the Collective Behavior of Cells Growing New Tissue. *Biophysical Journal* 112:193-204.

[2] Leung, S. and Zhao, H. 2009. A grid based particle method for moving interface problems. *Journal of Computational Physics*. 228:2993-3024.

[3] Leung, S. and Zhao, H. 2011. A grid based particle method for solving partial differential equations on evolving surfaces and modeling high order geometrical motion. Journal of Computational Physics. 230:2540-2561.