

MATHEMATICAL AND NUMERICAL METHODS FOR UNDERSTANDING IMMUNE CELL MOTION DURING WOUND HEALING

GIULIA LUPI¹, SEOL AH PARK¹, MARTIN AMBROZ¹, RESUL OZBILGIC²,
MAI NGUYEN CHI², GEORGES LUTFALLA² AND KAROL MIKULA^{1,*}

Abstract. In this paper, we propose a new workflow to analyze macrophage motion during wound healing. These immune cells are attracted to the wound after an injury and they move showing both directional and random motion. Thus, first, we smooth the trajectories and we separate the random from the directional parts of the motion. The smoothing model is based on curve evolution where the curve motion is influenced by the smoothing term and the attracting term. Once we obtain the random sub-trajectories, we analyze them using the mean squared displacement to characterize the type of diffusion. Finally, we compute the velocities on the smoothed trajectories and use them as sparse samples to reconstruct the wound attractant field. To do that, we consider a minimization problem for the vector components and lengths, which leads to solving the Laplace equation with Dirichlet conditions for the sparse samples and zero Neumann boundary conditions on the domain boundary.

Mathematics Subject Classification. 35Q68, 92C55, 65N08, 65D10, 92C37, 60J60.

Received February 4, 2025. Accepted November 15, 2025.

1. INTRODUCTION

The movement of immune cells, particularly macrophages, and neutrophils, has been intensively studied [1–4], and characterized in different tissues [2, 4] and structures [3]. These leukocytes are key players in the innate immune response; they are among the first cells to respond to tissue damage and infection, participating in the killing of invading pathogens, clearing of cellular debris, and tissue repair. After an injury, danger signals are released from the wound and attract immune cells. Cues that can influence the navigation of macrophages are numerous and include gradients of chemoattractants that cause macrophage chemotaxis, inflammatory mediators, or changes in mechanical force [5]. Finding the wound attractant field is an important biological question, that holds significant clinical importance. Indeed, chronic inflammatory diseases and other pathologies are associated with an inappropriate immune response, and finding how wound chemoattractants propagate and control macrophage movement may help to develop new strategies to prevent an excessive or unbalanced immune response. Danger signals are thought to be released from the wound margin and include the so-called DAMPs

Keywords and phrases: Partial differential equations, evolving curves, trajectory smoothing, flowing finite volume method, anomalous diffusion, mean squared displacement, vector field reconstruction, Laplace operator, sparse samples.

¹ Department of Mathematics, Slovak University of Technology, Radlinského 11, 81005 Bratislava, Slovakia.

² Laboratory of Pathogens and Host Immunity (LPHI), CNRS/Université de Montpellier, Place Eugène Bataillon, 34095 Montpellier cedex 5, France.

* Corresponding author: karol.mikula@stuba.sk

(damage-associated molecular patterns). These were initially identified *in vitro* and the actual contribution of some of them (such as Reactive oxygen species (ROS) and calcium) has also been studied *in vivo* [6]. However, the understanding of the main contributors is still unclear. To address this problem for neutrophils migration, the authors in [7] reconstructed the chemoattractant field that drives chemotaxis of this specific type of immune cells during wound healing using the zebrafish. While macrophages behave differently than neutrophils at the wound [8], their chemoattractant field has not been determined.

In this paper, we propose a new workflow to analyze macrophage trajectories and use the information extracted from the trajectories to reconstruct the wound attractant field. We exploit the optical clarity and genetic amenability of the zebrafish larva and, thanks to the macrophage reporter line, we follow the *in vivo* behavior of macrophages in response to tail fin injury. The segmentation-based tracking approach developed in [9] allowed us to obtain 2D+ time macrophage trajectories from microscopy videos. Moving toward a wound, macrophages exhibit both directional and random motion. We assume that the wound attractant field causes the directional part, while the random parts are parts of trajectories where the attractant field does not have a direct influence. Therefore, we first smooth the trajectories and remove the random parts. The mathematical model we propose to smooth the trajectories is based on evolving curve approach in the Lagrangian formulation. The smoothing process is influenced by three terms: trajectory *smoothing* term depending on its curvature, term *attracting* trajectories to the original one, and *tangential velocity*. The tangential velocity does not change the shape of the evolving curve, but it redistributes the points on the curve, improving the stability of the numerical computations. It can be defined in different ways, for example by imposing a uniform redistribution of points, by conserving the initial distribution of points, or a redistribution that depends on the curvature [10–14]. In this paper, we use the asymptotically uniform redistribution of points; see [13, 15] for more details.

In the proposed trajectory smoothing model, the curve evolution is driven by the following partial differential equation

$$\frac{\partial \mathbf{x}}{\partial t} = -\delta k \mathbf{N} + \lambda [(\mathbf{x}_0 - \mathbf{x}) \cdot \mathbf{N}] \mathbf{N} + \alpha \mathbf{T}, \quad (1.1)$$

where \mathbf{x} is a point on the evolving curve, $-\delta k \mathbf{N}$ is the smoothing term depending on the local curvature k , $\lambda [(\mathbf{x}_0 - \mathbf{x}) \cdot \mathbf{N}] \mathbf{N}$ is the attracting term, α is the tangential velocity, \mathbf{T} the unit tangent vector, \mathbf{N} the unit normal vector, and λ and δ are the model parameters. We can rewrite equation (1.1) into the form

$$\frac{\partial \mathbf{x}}{\partial t} = \beta \mathbf{N} + \alpha \mathbf{T}, \quad (1.2)$$

where β is the velocity in the normal direction.

Equation (1.2) is a general equation for curve evolution models: the normal velocity β can be defined in different ways and drives the evolution of the curve, depending on the purpose of the application. In [16], the authors developed a segmentation model based on curve evolution in the Lagrangian formulation. They defined the normal velocity taking into account the attracting vector field given by the image gradient. In [17], the authors developed a wildland fire propagation model based on surface curve evolution and empirical principles of fire propagation. In particular, they considered the evolution of a 3D surface curve and defined the normal velocity of the projected plane curve β taking into account parameters such as wind speed and direction and topographic slope. In [18], we proposed a model for the smoothing of macrophage trajectories in the Lagrangian formulation. The normal velocity is composed of the smoothing term and the attracting term so that the result of the smoothing keeps the trajectory close to the original one. To define the attracting term we identified, for each point on the evolving curve, the closest point on the original trajectory. The attracting term was then defined as the normal component of the vector connecting these two points.

In this paper, we propose a new way of defining the attracting term by considering the evolution of the lengths of the original segments. For each point on the evolving curve, we find to which of the original segments it belongs and compute its corresponding point on the original curve. This can not be done directly because the tangential velocity moves the points along the curve to redistribute them asymptotically uniformly. So we

follow the evolution of the original points and find where they would have been if there had been no tangential redistribution of points. By this new approach, the evolving curve is kept close to the original in a consistent and more accurate way compared to [18].

In [18], we used the Hausdorff distance between two evolving curves as a stopping criterion. We stopped the curve evolution steps (also referred to as iterations within the text) when this distance was less than a given tolerance; thus, the stopping criterion considered the numerical steady state of the evolution. In this paper, we present a new way of defining the stopping criterion. Since we want to smooth the curve and keep only the directional parts of the motion, the new approach considers the self-intersections of the curves. In fact, where the curves self-intersect, the cells do not have a clear directionality but are moving back and forth. Therefore, we use the condition of having no further self-intersections as the stopping criterion. To detect the self-intersections of the curves, we developed a self-intersection detection algorithm based on topological changes detection algorithm from [17]: the main idea is to construct a background (pixel) grid and check whether two non-consecutive grid points belong to the same pixel. If so, the self-intersection is detected; the details of the algorithm will be explained in Section 2.3.1.

Since the main idea is to smooth the curve while keeping the directional parts close to the original, we considered an adaptive choice of the δ and λ parameters based on the curve self-intersections. Equation (1.1) then becomes

$$\frac{\partial \mathbf{x}}{\partial t} = -\delta(\mathbf{x}, t)k\mathbf{N} + \lambda(\mathbf{x}, t)[(\mathbf{x}_0 - \mathbf{x}) \cdot \mathbf{N}]\mathbf{N} + \alpha\mathbf{T}, \quad (1.3)$$

where $\delta(\mathbf{x}, t)$ and $\lambda(\mathbf{x}, t)$ now depend on the spatial and temporal coordinates.

The ability to distinguish between directional and random motion is crucial to accurately describe the different modes of motion within a trajectory. A particular method for separating a trajectory into subregions is described, *e.g.*, in [19]. The authors developed a rolling window algorithm to separate active and passive parts of cell motion, taking into account the local mean squared displacement (MSD) and directional persistence in a local window. The mean squared displacement is a common tool for analyzing random motion; it distinguishes the type of diffusive motion followed by the walker. The motion can be diffusive, such as Brownian motion, superdiffusive, such as Levy-flights (random walks where small steps are interspersed with long jumps), and subdiffusive. Subdiffusive motion is usually observed in crowded environments where the walker is not free to move in any direction but is constrained by the environment [20–22]. Several studies have analyzed cell movement by examining the MSD [23–29]. In [27], the authors analyzed the behavior of metastatic and non-metastatic cancer cells migrating on linear microtracks. The large-scale statistical analysis using the averaged MSD showed that metastatic cancer cells follow a different movement strategy than non-metastatic ones. In [26], a similar analysis showed that a generalized Levy walk describes T-cell motility in the brains of infected mice.

In our approach, the smoothing model allows us to separate the directional parts from the random parts. We define two different ways of finding the random parts. The first takes into account the evolution of the lengths of the original segments. During the evolution, a segment may disappear. This usually happens in regions where the curvature is high and the segments are short, which appears in random regions. If the segment disappears, we consider it to be a random segment; consecutive disappeared segments form a random sub-trajectory. The other way to define the random parts is to consider the self-intersections. In fact, the random parts of the trajectories are characterized by back-and-forth motion without a clear directionality, which causes the curve self-intersection. A random sub-trajectory is then defined as the segments belonging to a self-intersecting part. Once we obtained the random parts, to characterize the type of diffusion followed by the cells in regions where the vector field does not have a direct influence on the motion, we used the MSD.

Finally, we computed the velocities on the smoothed curves and used them as sparse samples to reconstruct the wound attractant field. Many different computational contexts require the reconstruction of vector fields from sparse samples: the applications of such a process include fluid dynamics visualization, texture synthesis, non-photorealistic rendering, optical flow fields, and map registration [30–32]. In [31], the authors proposed local piecewise polynomial approximations using least squares methods. In [33], a different approach was taken, where

linearly independent vector fields, termed 'basis fields,' were combined to approximate the vector field, maintaining invariance under coordinate transformations. In [30], the authors introduced a model for reconstructing smooth tangent vector fields. Using sparse samples as user-specified constraints, they applied a coordinate-free approach and employed the discrete Laplace operator to obtain a smooth field. Laplace interpolation has also been applied to image processing tasks like image inpainting and compression [34, 35]. In [32], the Locally Affine Globally Laplace (LAGL) transformation model was developed, involving the design of affine transformations between maps, followed by solving the Laplace equation to smoothly reconstruct intermediate points.

In this paper, we solve a minimization problem for the two vector components and the vector length considering Dirichlet conditions for the sparse samples and zero Neumann boundary conditions at the domain boundary. Solving the minimization problem is equivalent to solving the Laplace equation with prescribed boundary conditions. To prove the existence and uniqueness of a weak solution for the considered problem, we use the theorem proved in [36]. For the discretization, we place the unknowns at the vertices of the pixels and use them as grid nodes for the finite difference scheme. This results in a smooth vector field where the information from the sparse samples is interpolated in regions where the vectors influence each other and extrapolated in regions where they do not. Considering only the minimization problem for the two vector components leads to a smooth vector field in which the lengths of the vectors decrease with distance from the Dirichlet conditions. This is due to the property of the Laplace equation of averaging neighboring values. Since we did not want to add any new information, but to reconstruct the vector field from the information given by the sparse samples, we also considered the minimization problem for the vector lengths. This leads to a smooth vector field, where the direction of the vectors changes smoothly from one Dirichlet condition to the other, and where the vector lengths are interpolated/extrapolated from the Dirichlet conditions.

To test the model on real macrophage trajectories, we considered 3 different datasets, which we call Dataset_1, Dataset_2, and Dataset_3. In all of them, zebrafish larvae were utilized to study macrophage motion during wound healing. At 3 days post-fertilization, transgenic reporter larvae, Tg(mfap4:mCherry-F), were employed. These larvae express a farnesylated mCherry fluorescent protein under the control of the macrophage-specific promoter mfap4, enabling macrophages to display mCherry at their membranes. Macrophage cells are followed in the red channel. All larvae were wounded at the caudal fin fold region and subsequently imaged using a Spinning Disk Confocal microscope.

Migrating macrophages were imaged starting 30 minutes to 1-hour post-amputation, over a duration of 13 hours, with a time interval of 2.5 minutes between frames and a z-step of 1 μm . For all datasets, the pixel size was 0.319489 μm . For numerical experiments, we used 2D+time projection images, where the three-dimensional (3D) microscopy data were flattened into a 2D plane by applying a maximum intensity projection. In this method, the maximum intensity value along the z-axis was selected for each pixel.

The paper is organized as follows. In Section 2, we will introduce the mathematical model for smoothing trajectories, define the attracting term, and consider the discretization of the model. Then we will present the self-intersections detection algorithm and illustrate how to choose the parameters adaptively. We will conclude Section 2 by presenting results on real macrophage trajectories. In Section 3, we will illustrate how to detect the random parts and analyze them using the mean squared displacement. Finally, in Section 4, we will define the model we propose for vector field reconstruction from sparse samples and illustrate its discretization. The results will be shown for 3 different datasets of macrophages moving during wound healing.

2. TRAJECTORIES SMOOTHING

Let Γ be an open plane curve with fixed endpoints

$$\begin{aligned} \Gamma : [0, 1] &\rightarrow \mathbb{R}^2, \\ u &\mapsto \mathbf{x}(u), \end{aligned} \tag{2.1}$$

where $\mathbf{x}(u) = (x_1(u), x_2(u))$ is a point of the curve Γ . We consider the time evolution of a point $\mathbf{x}(u, t) \in \Gamma_t$ of the curve Γ driven by the equation (1.3), where t is time, k is the curvature, \mathbf{x}_0 is a point on the original curve,

\mathbf{T} is the unit tangent vector, \mathbf{N} is the unit normal vector, α is the tangential velocity and $\lambda(\mathbf{x}, t)$ and $\delta(\mathbf{x}, t)$ are positive parameters.

From now on we will indicate by

$$w = (\mathbf{x}_0 - \mathbf{x}) \cdot \mathbf{N} \quad (2.2)$$

the attracting term, and by

$$\beta = -\delta k + \lambda w, \quad (2.3)$$

the velocity in the normal direction.

The tangential velocity is defined in such a way that the points are asymptotically uniformly distributed: the formula is obtained by considering the ratio between the local and the global length and letting it tend to 1; see [13, 15] for details on the derivation and discretization. On one hand, the redistribution of points along the curve improves the stability of the numerical computations; on the other hand, it makes the definition of attracting term more complicated. In fact, the points also move in the tangential direction. If we then define the *corresponding point* of the point $\mathbf{x}(u)$ simply as $\mathbf{x}_0(u)$, we can obtain that $\mathbf{x}(u)$ is attracted to a completely different part of the trajectory. Therefore, to define the attracting term independently of the tangential velocity, we follow the evolution of the original segments as if there was no tangential redistribution. Two points are defined as corresponding if they belong to the same evolving segment.

2.1. Attracting term

In order to understand the main idea of the model described below, we must first note how trajectories are found in videos. The original trajectories are piecewise linear segments where the time between two linear segment endpoints is 1 time unit, and a point of the trajectory corresponds to the position of a cell in a time frame. First, for the numerical discretization, we add grid points to the linear segments: refining the grid is a necessary step for the numerical computations to improve the accuracy. To do this, we set a value \bar{h} and add grid points inside the linear segments in a distance of \bar{h} , creating an initial, nearly uniform distribution of grid points. Then we distinguish between a *segment*, which connects two of the original points of the trajectory, and an *element*, which connects two grid points of the evolving curve. Our interest is in the *segments*: the model we propose computes the new length of the segments on the evolving curve as if there was no tangential velocity, and it finds the endpoints of the segment on the evolving curve. The model to find the new lengths of the segments has already been proposed in [18], but was not used in the definition of attracting term; we recall here the derivation and later its discretization for completeness. Once we have the segments on the evolving curve and the original ones, we define the attracting term as the normal component of the vector connecting the points in the corresponding segments.

Let us consider the unit arc length parameterization s of the curve Γ so that it holds $ds = |\mathbf{x}_u| du$. The local length g is defined as

$$g = |\mathbf{x}_u|, \quad (2.4)$$

while the global length is

$$L = \int_{\Gamma} ds = \int_0^1 g du = \int_0^1 |\mathbf{x}_u| du. \quad (2.5)$$

We consider the time derivative L_t of the global length; using the Frenet-Serret formulas we obtain

$$g_t = |\mathbf{x}_u|_t = g k \beta + g \alpha_s = g k \beta + \alpha_u. \quad (2.6)$$

Then

$$L_t = \int_0^1 g_t \, du = \int_0^1 (gk\beta + \alpha_u) \, du = \int_0^1 gk\beta \, du + \alpha(1) - \alpha(0) = \int_{\Gamma} k\beta \, ds, \quad (2.7)$$

where $\alpha(1) = \alpha(0) = 0$ because the endpoints of the curve are fixed.

We can consider the same equation (2.7) for the evolution of the lengths of the segments of the original trajectory. For the j -th segment, we obtain

$$(L_t)_j = \int_{u_{j-1}}^{u_j} (gk\beta + \alpha_u) \, du = \int_{\mathbf{x}(u_{j-1})}^{\mathbf{x}(u_j)} k\beta \, ds + \alpha(u_j) - \alpha(u_{j-1}), \quad (2.8)$$

In this case, in general, it holds $\alpha(u_j) \neq 0$, $\alpha(u_{j-1}) \neq 0$. But, for our purpose, it is reasonable to consider the model for the evolution of the length of the segments with $\alpha = 0$. We obtain

$$(L_t)_j = \int_{\mathbf{x}(u_{j-1})}^{\mathbf{x}(u_j)} k\beta \, ds, \quad (2.9)$$

Note that the tangential velocity does not influence the shape of the evolving curve, it only redistributes the points on the evolving curve. It follows that, in a continuous setting, the shape of the evolved curve will be the same with or without the tangential velocity; therefore, considering $\alpha = 0$ allows us to find what is the real length of the segments after smoothing. The method consists of the following steps.

1. Find the new length of the segment using equation (2.9).
2. Find the endpoints of the segments on the smoothed curve.
3. Define N the number of grid points inside the new segment and consider a uniform distribution of N points inside the original segment.
4. Define the attracting term as

$$(\mathbf{x}^0 - \mathbf{x})(u) = \mathbf{x}^0(u) - \mathbf{x}(u), \quad (2.10)$$

where $\mathbf{x}^0(u)$ and $\mathbf{x}(u)$ are grid points in the same segment on the original and smoothed curve, respectively.

A segment can disappear during the evolution; if this happens, we consider that the segment degenerates to a point, *i.e.* the two endpoints merge. In this case, the point is attracted to the center of the original segment. If the same happens for consecutive segments, we generalize the formula and consider the center of mass of the segments and the vector

$$(\mathbf{x}^0 - \mathbf{x})(u) = \mathbf{x}^{0,M} - \mathbf{x}(u), \quad (2.11)$$

where $\mathbf{x}^{0,M}$ is the center of mass. Since the tangential velocity does not affect the shape of the evolving curve, we consider only the normal component of the attracting vector. The details of the numerical discretization are explained in the next section.

2.2. Numerical algorithm

The discretization of equation (1.3) is done as in [18]. We recall here the main steps for completeness.

Let $g = |\mathbf{x}_u| = \sqrt{(\frac{\partial x_1}{\partial u})^2 + (\frac{\partial x_2}{\partial u})^2}$. If we denote by s the unit arc length parametrization of the curve Γ , then $ds = |\mathbf{x}_u| du = g du$ and $du = \frac{1}{g} ds$. The unit tangent vector \mathbf{T} is defined as $\mathbf{T} = \frac{\partial \mathbf{x}}{\partial s} = \mathbf{x}_s$ while the unit normal

vector \mathbf{N} is $\mathbf{N} = \mathbf{x}_s^\perp$ such that $\mathbf{T} \wedge \mathbf{N} = -1$. If $\mathbf{T} = (\frac{\partial x_1}{\partial s}, \frac{\partial x_2}{\partial s})$ then $\mathbf{N} = (\frac{\partial x_2}{\partial s}, -\frac{\partial x_1}{\partial s})$. From the Frenet-Serret formulas, we have $\mathbf{T}_s = -k\mathbf{N}$ and $\mathbf{N}_s = k\mathbf{T}$, where k is the curvature. It follows that $-k\mathbf{N} = \mathbf{T}_s = (\mathbf{x}_s)_s = \mathbf{x}_{ss}$.

We can rewrite (1.3) into the form of the so-called intrinsic partial differential equation

$$\mathbf{x}_t = \delta \mathbf{x}_{ss} + \alpha \mathbf{x}_s + \lambda w \mathbf{x}_s^\perp \quad (2.12)$$

which is suitable for numerical discretization. Since $\mathbf{x} = (x_1, x_2)$, (2.12) represents a system of two partial differential equations for the two components of the position vector \mathbf{x} .

Let us consider the following form of the intrinsic PDE (2.12)

$$\mathbf{x}_t - \alpha \mathbf{x}_s = \delta \mathbf{x}_{ss} + \lambda w \mathbf{x}_s^\perp, \quad (2.13)$$

where w is given by (2.2) and α is defined in [18]. For equation (2.13), we consider Dirichlet boundary conditions, which means fixed endpoints, and as initial condition, the original trajectory.

Consider an open curve discretized into $n + 2$ grid points, $i = 0, \dots, n + 1$. We fixed the two points \mathbf{x}_0 and \mathbf{x}_{n+1} at the endpoints of the curve. Consequently, for these two points, the velocity is zero, *i.e.* $\alpha_0 = \beta_0 = \alpha_{n+1} = \beta_{n+1} = 0$. We define the i -th element $\mathbf{h}_i = \mathbf{x}_i - \mathbf{x}_{i-1}$ with length $h_i = |\mathbf{x}_i - \mathbf{x}_{i-1}|$.

For the spatial discretization, we use the flowing finite volume method [15]. Consider the finite volume $\mathbf{p}_i = [\mathbf{x}_{i-\frac{1}{2}}, \mathbf{x}_{i+\frac{1}{2}}]$ where $\mathbf{x}_{i+\frac{1}{2}}$ represents the middle point between \mathbf{x}_i and \mathbf{x}_{i+1} . Integrating over the finite volume \mathbf{p}_i , using the Newton-Leibniz formula and considering α_i , w_i , δ_i , and λ_i constant over the finite volume we obtain

$$\begin{aligned} & \frac{h_i + h_{i+1}}{2} (\mathbf{x}_i)_t + \frac{\alpha_i}{2} (\mathbf{x}_i - \mathbf{x}_{i+1}) - \frac{\alpha_i}{2} (\mathbf{x}_i - \mathbf{x}_{i-1}) \\ & = \delta_i \left(\frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{h_{i+1}} - \frac{\mathbf{x}_i - \mathbf{x}_{i-1}}{h_i} \right) + \lambda_i w_i \left(\frac{\mathbf{x}_{i+1} - \mathbf{x}_{i-1}}{2} \right)^\perp. \end{aligned} \quad (2.14)$$

For the time discretization, we use the semi-implicit time scheme, which is implicit in the intrinsic diffusion term and uses the inflow-implicit/outflow-explicit strategy for the intrinsic advection term [37]. Let m be the time step index and τ the length of the discrete time step. The basic idea of the inflow-implicit/outflow-explicit strategy is that we treat outflow from a cell explicitly while inflow implicitly [37]. Therefore, we define

$$\begin{aligned} b_{i-\frac{1}{2}}^{in} &= \max(-\alpha_i^m, 0), b_{i-\frac{1}{2}}^{out} = \min(-\alpha_i^m, 0), \\ b_{i+\frac{1}{2}}^{in} &= \max(\alpha_i^m, 0), b_{i+\frac{1}{2}}^{out} = \min(\alpha_i^m, 0). \end{aligned} \quad (2.15)$$

Approximating the time derivative by finite difference, we obtain the fully discrete scheme

$$\begin{aligned} & \mathbf{x}_{i-1}^{m+1} \left(-\frac{\delta_i}{h_i^m} - \frac{b_{i-\frac{1}{2}}^{in}}{2} \right) + \mathbf{x}_{i+1}^{m+1} \left(-\frac{\delta_i}{h_{i+1}^m} - \frac{b_{i+\frac{1}{2}}^{in}}{2} \right) + \\ & \mathbf{x}_i^{m+1} \left(\frac{h_{i+1}^m + h_i^m}{2\tau} + \frac{\delta_i}{h_i^m} + \frac{\delta_i}{h_{i+1}^m} + \frac{b_{i-\frac{1}{2}}^{in}}{2} + \frac{b_{i+\frac{1}{2}}^{in}}{2} \right) = \mathbf{x}_i^m \frac{h_{i+1}^m + h_i^m}{2\tau} \\ & - \frac{b_{i+\frac{1}{2}}^{out}}{2} (\mathbf{x}_i^m - \mathbf{x}_{i+1}^m) - \frac{b_{i-\frac{1}{2}}^{out}}{2} (\mathbf{x}_i^m - \mathbf{x}_{i-1}^m) + \lambda_i w_i^m \left(\frac{\mathbf{x}_{i+1}^m - \mathbf{x}_{i-1}^m}{2} \right)^\perp, \end{aligned} \quad (2.16)$$

for $i = 1, \dots, n$ where n is the number of unknown grid points.

The system (2.16) is represented by a strictly diagonally dominant matrix, then it is always solvable by the classical Thomas algorithm without any restriction on the time step τ .

For complex trajectories, the evolving curve may have singularities. For those parts of the trajectories, we then replace the second-order inflow-implicit/ outflow-explicit strategy to discretize the advection term with the first-order implicit upwind scheme [17]. We consider the singularity when the angle between two consecutive elements is less than 120° , *i.e.* $\angle \mathbf{h}_i \mathbf{h}_{i+1} < 120^\circ$. In the upwind scheme we consider only $b_{i-\frac{1}{2}}^{in}$ and $b_{i+\frac{1}{2}}^{in}$, therefore we obtain the fully discrete scheme

$$\begin{aligned} & \mathbf{x}_{i-1}^{m+1} \left(-\frac{\delta_i}{h_i^m} - b_{i-\frac{1}{2}}^{in} \right) + \mathbf{x}_{i+1}^{m+1} \left(-\frac{\delta_i}{h_{i+1}^m} - b_{i+\frac{1}{2}}^{in} \right) \\ & + \mathbf{x}_i^{m+1} \left(\frac{h_{i+1}^m + h_i^m}{2\tau} + \frac{\delta_i}{h_i^m} + \frac{\delta_i}{h_{i+1}^m} + b_{i-\frac{1}{2}}^{in} + b_{i+\frac{1}{2}}^{in} \right) = \mathbf{x}_i^m \frac{h_{i+1}^m + h_i^m}{2\tau} \\ & + \lambda_i w_i^m \left(\frac{\mathbf{x}_{i+1}^m - \mathbf{x}_{i-1}^m}{2} \right)^\perp, \end{aligned} \quad (2.17)$$

for $i = 1, \dots, n$, which is also represented by a strictly diagonally dominant matrix, and therefore solvable by the Thomas algorithm. Recall that the endpoints \mathbf{x}_0 and \mathbf{x}_{n+1} of the curve are fixed. The use of (2.17) improves the robustness of the scheme with respect to singularities. In the next Section, we will describe how to discretize the attracting term w_i .

2.2.1. Discretization of attracting term

From (2.4) we have

$$g = |\mathbf{x}_u| = \sqrt{\left(\frac{\partial x_1}{\partial u} \right)^2 + \left(\frac{\partial x_2}{\partial u} \right)^2} \quad (2.18)$$

To discretize (2.9), we approximate the time derivative by the finite difference and consider the formulas for k , β given by [15, 16, 38]

$$\begin{aligned} k_i^m &= \text{sgn}(\mathbf{h}_{i-1}^m \wedge \mathbf{h}_{i+1}^m) \frac{1}{2h_i^m} \arccos\left(\frac{\mathbf{h}_{i-1}^m \cdot \mathbf{h}_{i+1}^m}{h_{i-1}^m h_{i+1}^m} \right), \\ \beta_i^m &= -\delta k_i^m + \lambda w_i^m, \\ L^m &= \sum_{l=1}^{n+1} h_l^m, \end{aligned} \quad (2.19)$$

where $\mathbf{h}_i^m = \mathbf{x}_i^m - \mathbf{x}_{i-1}^m$, $|\mathbf{h}_i^m| = h_i^m$ and $\mathbf{h}_{i-1}^m \wedge \mathbf{h}_{i+1}^m$ is the wedge product, *i.e.* the determinant of the matrix with columns \mathbf{h}_{i-1}^m and \mathbf{h}_{i+1}^m . The expression for k_i^m comes from the definition of curvature as $k = \frac{\partial \varphi}{\partial s}$, where φ is the tangent angle and s is the arc-length, and it was used in [15, 16, 38] as well. For the first and last elements, since we don't have the values of h_{i-1}^m , respectively h_{i+1}^m , we set $k_1^m = k_2^m$ and $k_{n+1}^m = k_n^m$. Note that, for $m = 0$, $w_i^0 = 0$ for every $i = 1, \dots, n$, therefore $\beta_i^0 = -\delta k_i^0$. This is natural since in the first iteration the evolving curve and the original curve are the same: it follows that the attracting vector is $\mathbf{0}$.

To relate the index j of the segments to the index i of the grid points we will indicate by $\mathcal{I}(u_{j-1})$, $\mathcal{I}(u_j)$ the corresponding i index of the endpoint $\mathbf{x}(u_{j-1})$, respectively $\mathbf{x}(u_j)$, see Figure 1. We obtain the fully discrete approximation of equation (2.9) for the new length of the j -th segment

$$L_j^{m+1} = L_j^m + \tau \sum_{i=\mathcal{I}(u_{j-1})+1}^{\mathcal{I}(u_j)} h_i^m k_i^m \beta_i^m. \quad (2.20)$$

Experimentally, it holds $L_j^{m+1} \leq L_j^m$. From (2.20) it is clear that if the segment is in a region with high curvature (like the random parts), its length will decrease faster and it may also disappear, $L_j^{m+1} = 0$.

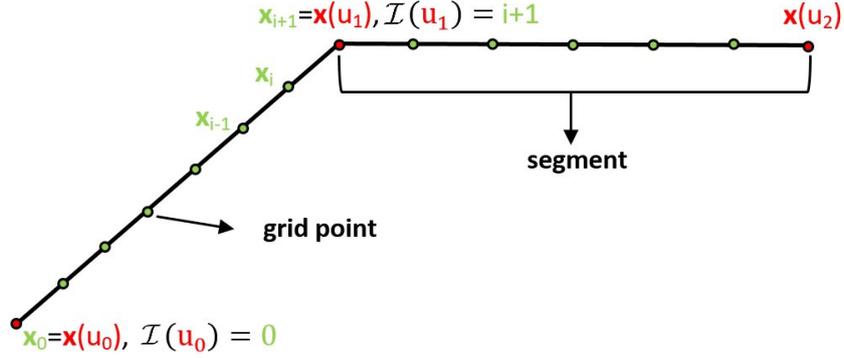


FIGURE 1. Visualization of curve discretization. The green dots represent the grid points and the red dots represent the endpoints of the segments.

After obtaining the new length L_j^{m+1} of the segment applying the formula in (2.20), we need to compute the new i index for $\mathcal{I}(u_{j-1})$ and $\mathcal{I}(u_j)$. The straightforward approach would be, starting from the first segment L_1^{m+1} , to sum up the lengths of the elements h_i^{m+1} until the sum is equal to L_j^{m+1} and consider $\mathcal{I}(u_j)$ as the last endpoint of the corresponding element h_i^{m+1} .

Consider $\sum_{j=1}^M L_j^{m+1}$, where M is the number of original segments. A problem arises since in the numerical computations it holds

$$\sum_{i=1}^{n+1} h_i^{m+1} \neq \sum_{j=1}^M L_j^{m+1}. \quad (2.21)$$

It follows that we need to define a way to relate the two global lengths. Consider the ratio

$$r_j^{m+1} = \frac{L_j^{m+1}}{\sum_{i=1}^M L_i^{m+1}}, \quad (2.22)$$

and define the discrete length of the segment as

$$L_j^d = r_j^{m+1} \sum_{i=1}^{n+1} h_i^{m+1}, \quad (2.23)$$

so that

$$\sum_{j=1}^M L_j^d = \sum_{i=1}^{n+1} h_i^{m+1}. \quad (2.24)$$

Having obtained the discrete lengths L_j^d , we can apply the straightforward approach. Let us illustrate the approach for L_1^d ; the process is then repeated for all others L_j^d , $j = 2, \dots, M$. We sum the lengths of the elements h_i until $\sum_{i=1}^k h_i \geq L_j^d$, for some $k \in \{1, \dots, n+1\}$. If

$$\sum_{i=1}^k h_i = L_j^d, \quad (2.25)$$

then $\mathcal{I}(u_1) = k$. Otherwise, we find the point \mathbf{x}^* on the element \mathbf{h}_i such that equality (2.25) holds. Then, we find which point between \mathbf{x}_k and \mathbf{x}_{k-1} is the closest point to \mathbf{x}^* and move it to \mathbf{x}^* . Finally, we set either $\mathcal{I}(u_1) = k$ or $\mathcal{I}(u_1) = k - 1$ depending on which point was moved to \mathbf{x}^* and $L_j^{m+1} = L_j^d$.

Notice that we focused only on the index of the last endpoint of each segment $\mathcal{I}(u_j)$: that is because the last endpoint of the segment j is the first endpoint of the segment $j + 1$.

Let us now focus on the disappearing segments. We add the following conditions: if $L_j^{m+1} < h_{mean}$, then $L_j^{m+1} = 0$, where

$$h_{mean} = \frac{1}{n+1} \sum_{i=1}^{n+1} h_i.$$

If we obtain $L_j^{m+1} = 0$, it means the first and the last endpoints of that segment at time step index $m + 1$ became the same point, then $\mathcal{I}(u_j) = \mathcal{I}(u_{j-1})$. So we modify the algorithm for finding the new indexes i for $\mathcal{I}(u_j)$ adding the following condition

- if $r_j^{m+1} = 0$ then $\mathcal{I}(u_j) = \mathcal{I}(u_{j-1})$.

This condition will cause that when we apply formula (2.20) for every time step index $m + k$, with $k > 1$, we will obtain

$$L_j^{m+k} = L_j^{m+1} = 0.$$

Once we have obtained the new length of the segments, from our algorithm we are also able to calculate how many grid points are inside a segment: since the algorithm gives us the first and the last end points of a segment, it is sufficient to consider $N_j = \mathcal{I}(u_j) - \mathcal{I}(u_{j-1}) + 1$. Let us consider for every segment on the original curve, indexed by j , a uniform distribution of N_j grid points on the segment. Therefore, we obtain a new parametrization of the original curve. Let us indicate the grid points in the new parametrization on the original curve with the same index i , *i.e.* \mathbf{x}_i^0 , $i = 0, \dots, n + 1$. Then, the attracting vector for the grid points is defined as

$$(\mathbf{x}^0 - \mathbf{x})_i^m = \mathbf{x}_i^0 - \mathbf{x}_i^m. \quad (2.26)$$

If a segment disappears, let's say $L_j^m = 0$, we consider it degenerated to a point, so $\mathcal{I}(u_{j-1}) = \mathcal{I}(u_j)$. In this case, we consider the middle point of the segment

$$\mathbf{x}^{0,M} = \frac{\mathbf{x}_{\mathcal{I}(u_{j-1})}^0 + \mathbf{x}_{\mathcal{I}(u_j)}^0}{2}. \quad (2.27)$$

If more subsequent segments disappear, we generalize the formula calculating the center of mass for those segments. For the point where one or more segments degenerated to, we define the attracting vector as

$$(\mathbf{x}^0 - \mathbf{x})_i^m = \mathbf{x}^{0,M} - \mathbf{x}_i^m \quad (2.28)$$

and w_i^m is defined as

$$w_i^m = (\mathbf{x}^0 - \mathbf{x})_i^m \cdot \mathbf{N}_i^m, \quad (2.29)$$

where $\mathbf{N}_i^m = \left(\frac{\mathbf{x}_{i+1}^m - \mathbf{x}_{i-1}^m}{h_i^m + h_{i+1}^m} \right)^\perp$.

2.3. Numerical experiments

First, to see how the different parts of the normal velocity work, we studied the results for different values of the parameters λ and δ . In [18] we studied the results for simple curves to see the influence of the attracting term. The steady state keeps the curve closer to the original one than in the case where there is only the curvature term. Moreover, varying the values of λ also changes the steady state; the higher the value of λ , the more the curve is kept closer to the original. Here we consider the new definition of the attracting term defined in Sections 2.1 and 2.2.1 and different values of λ , namely $\lambda = 10$ and $\lambda = 50$. The other parameters were chosen as follows: $\delta = 0.003$, $\tau = 0.00001$, and $\omega = 0.8$. Here, ω is the parameter that controls the speed of convergence to uniform redistribution of points in the tangential velocity α ; see [13, 15] for details. Figure 2 shows the results for different values of λ . The stopping criterion in this case considers the Hausdorff distance between two evolving curves and stops the iterations when the distance is less than the tolerance $\epsilon = 0.00065$ as in [18]. As one can see, in both cases the curve is smoothed everywhere, both in the random parts and in the directional parts of the motion, but the loops are not completely removed. We noticed that where the curve intersects itself, the cell doesn't have a clear directionality, but moves back and forth. Since we only want to keep the directional parts of the motion, we developed a new stopping criterion based on the curve self-intersections. The algorithm from [17] for fast ($O(n)$) detection of topological changes was adjusted to detect self-intersections; we will recall it in the next section.

2.3.1. Detection of self-intersections

The main idea of self-intersection detection is to construct a background grid and check which points of the trajectory belong to the same pixel. If the points are not consecutive, then they are considered to be points where the curve is self-intersecting. To do this, we consider the initial distance \bar{h} of the grid points and construct 2 background grids with pixel size $2\bar{h}$ shifted by \bar{h} . Recall that in the beginning, we add grid points to the original segments at intervals of \bar{h} , obtaining an almost uniform distribution of points (see Sect. 2.1). In the first part of the algorithm, we go through the points of the curve, marking the pixels in the background grids that correspond to these points as -1 . Then we go through each grid point and check whether its pixel has been marked as -1 . If so, we assign that pixel the index i of the current grid point. If we are in a self-intersecting region, a grid point with index j will be positioned in a pixel already marked with index i , where $j > i$. We define a self-intersection if $j - i > 4$, *i.e.* there are at least five other grid points between the two intersecting points.

Figure 3 shows the process for a curve; only one of the background grids is shown for clarity. The light blue curve is the original trajectory and the rectangles show the regions of self-intersection. Figures 3a, 3b, 3c show the zooms of such rectangles. In fact, they are not consecutive grid points detected in the same pixel. The blue asterisks are the grid points of the trajectory, the red ones are the grid points where the self-intersection is detected and the green ones are the grid points belonging to the selected self-intersecting part. Note that, as it happens in Figure 3a, for a self-intersecting part there may be more pairs of grid points detected as self-intersections. In such a case, we select the start and end points so that the selected self-intersecting part contains all the others. The background grid takes into account the initial distance between the points, which is fixed at the beginning. During the evolution, the points move in the normal direction and also along the curve due to tangential redistribution. As a result, the average distance between the points changes. For this reason, before detecting the self-intersections, we add another step to the algorithm and remove every second point if the mean distance is less than $\bar{h}/2$. In the next iteration, the tangential velocity redistributes the points and the mean distance is kept closer to \bar{h} . The self-intersection detection is done at each time step and requires $2n$ operations, so the computational cost at each time step is $O(n)$, where n is the number of grid points, and thus it increases the necessary computational cost optimally.

We denoted the 3 datasets as Dataset_1, Dataset_2, Dataset_3, and used this notation in Figures 4, 5, 6, 7 and in Figures 17, 18, 19.

Figures 4, 5, 6 show the result of the smoothing using the curve self-intersections as a stopping criterion. When no more self-intersections are detected, the algorithm runs for 50 more curve evolution steps and stops. We added this step to avoid having sharp angles in the regions where the self-intersections are resolved. In

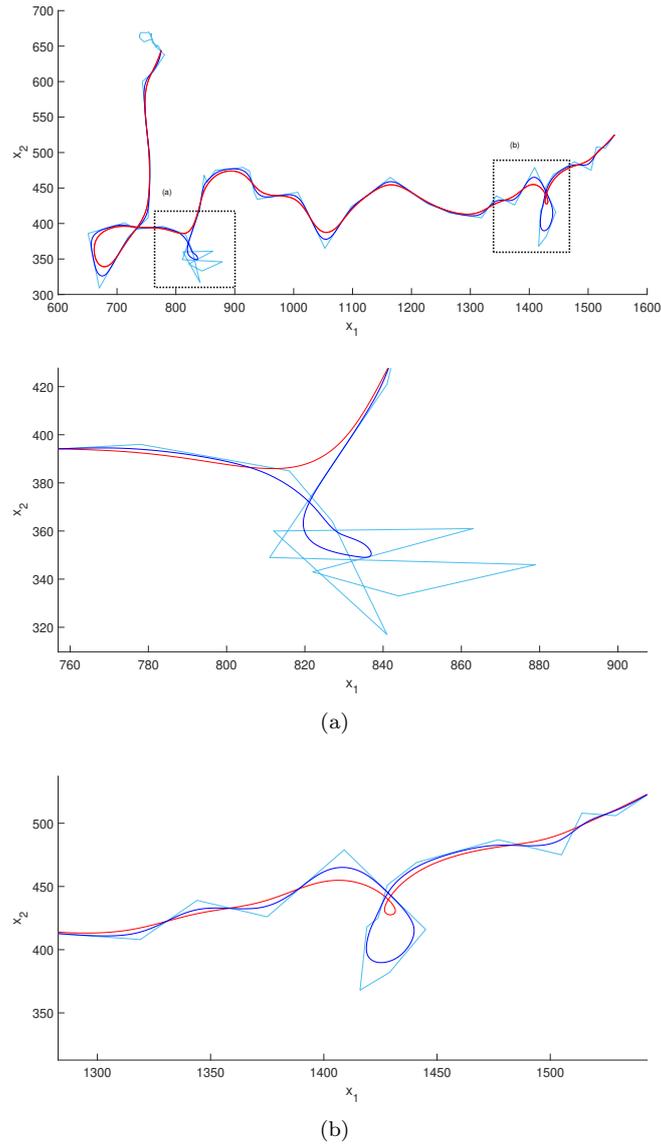


FIGURE 2. Original trajectory (light blue line) and results of smoothing for $\lambda = 10$ (red line) and $\lambda = 50$ (blue line). The black boxes indicate the regions chosen for the zooms. Middle and bottom: zooms of the smoothing process.

Figures 4, 5, 6 we have compared the results for 2 different values of δ , namely $\delta = 0.003$ and $\delta = 0.01$. The other parameters were chosen as follows: $\lambda = 20$, $\tau = 0.000001$, and $\omega = 50$. The light blue lines are the original trajectories, while the green curves are the smoothed curves; the result of the curve evolution with $\delta = 0.003$ is shown at the top, while that with $\delta = 0.01$ is shown at the bottom. As one can see, a smaller value of δ keeps the trajectories closer to the original ones. However, this is not always the case. In fact, the resulting evolved curve may be smoother when considering a smaller value of δ than when considering a larger value of δ . This is the case, for example, for the trajectory shown in Figure 7. Also in this figure, the result of the curve evolution with $\delta = 0.003$ is shown at the top, while the one with $\delta = 0.01$ is shown at the bottom. As one can notice, this

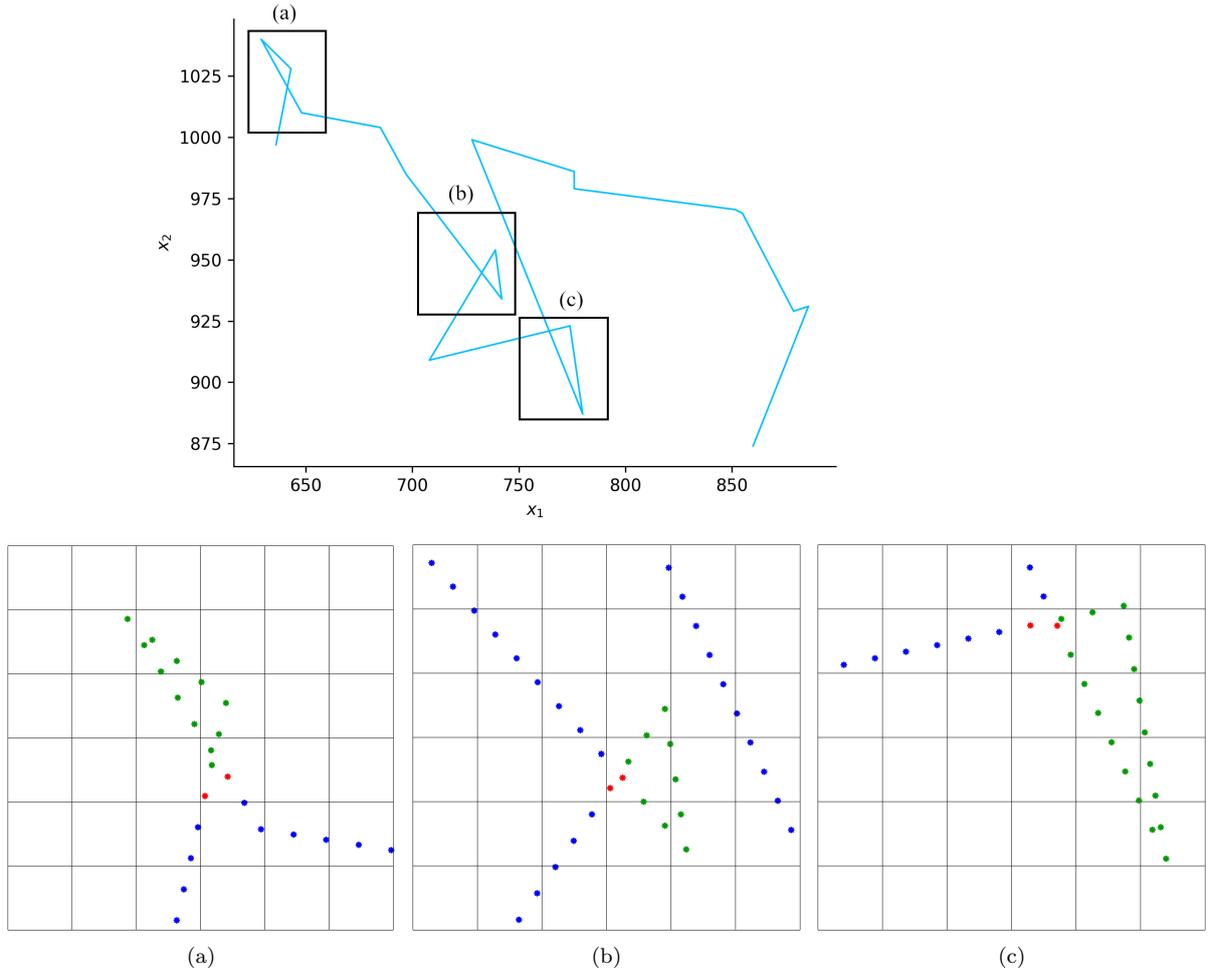


FIGURE 3. Top: original trajectory (light blue curve) and regions of self-intersections (black rectangles). Bottom: zooms of the rectangles. Grid points of the trajectory (blue asterisks), grid points where the self-intersection is detected (red asterisks), and grid points belonging to a self-intersecting part (green asterisks).

time the curve obtained with a smaller value of δ is smoother. This is due to the number of curve evolution steps needed to solve the self-intersections. For $\delta = 0.003$ the algorithm ran 24638 curve evolution steps, while for $\delta = 0.01$ the number of curve evolution steps was 6728. Since the algorithm takes longer to remove the self-intersections for small values of δ , the resulting curve is too smooth and we lose details of the directional parts of the motion. For this reason, we considered an adaptive choice of parameters. The details of this choice are explained in the next section.

2.3.2. Adaptive choice of parameters

Once we have detected the self-intersections, we can use the results not only as a stopping criterion but also to adaptively select the smoothing parameters. In fact, our goal is to smooth the trajectories in the self-intersecting parts and have almost no smoothing for the directional parts. Therefore we choose δ and λ in the following way: consider $0 \leq \delta_{min} < \delta_{max}$, and $\lambda_{max} > 0$. Let us indicate by i_1 and i_2 the indexes of a detected self-intersection,

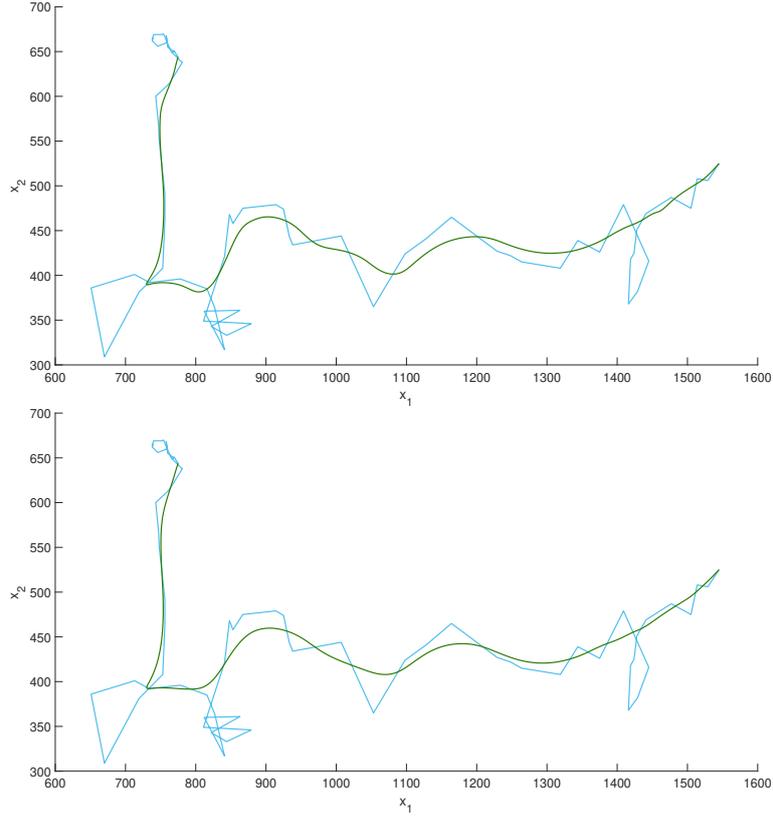


FIGURE 4. Original trajectory (light blue line) and results of smoothing (green lines) for $\delta = 0.003$ (top) and $\delta = 0.01$ (bottom). When no further self-intersections are detected, the algorithm performs 50 additional curve evolution steps before stopping (trajectory is from Dataset_1).

with $i_1 < i_2$. Then

$$\begin{cases} \delta_i = \frac{(6-i_1+i)\delta_{max}}{6}, \lambda_i = \frac{(6-i_1+i)\lambda_{max}}{6} & \text{for } i = i_1 - 5, \dots, i_1 \\ \delta_i = \delta_{max}, \lambda_i = \lambda_{max} & \text{for } i = i_1, \dots, i_2 \\ \delta_i = \frac{(6+i_2-i)\delta_{max}}{6}, \lambda_i = \frac{(6+i_2-i)\lambda_{max}}{6} & \text{for } i = i_2, \dots, i_2 + 5 \\ \delta_i = 0, \lambda_i = 0 & \text{otherwise.} \end{cases} \quad (2.30)$$

Notice that, for the grid points belonging to directional parts of motion, we will have $\beta_i = 0$ but $\alpha_i \neq 0$, so that the points will move in the tangential direction and redistribute along the curve. Once no more self-intersections are detected, the algorithm runs for 50 more curve evolution steps with $\delta_i = \delta_{min}$ and $\lambda_i = \lambda_{max}$ for $i = 1, \dots, n$. The reason for this is that we want to remove the sharp angles in the original trajectories and get a smooth curve.

For the smoothing of macrophage trajectories, the parameters were chosen as follows: $\delta_{min} = 0.003$, $\delta_{max} = 0.01$, $\lambda_{max} = 20$, $\tau = 0.000001$, and $\omega = 50$.

Figures 8, 9, 10, 11 show the results of the trajectories considered in Figures 4, 5, 6 and 7 respectively. The light blue curve is the original trajectory, while the green one is the smoothed curve. The adaptive choice of parameters allows us to keep the curve close to the original trajectory in the directional parts and to remove the self-intersections.

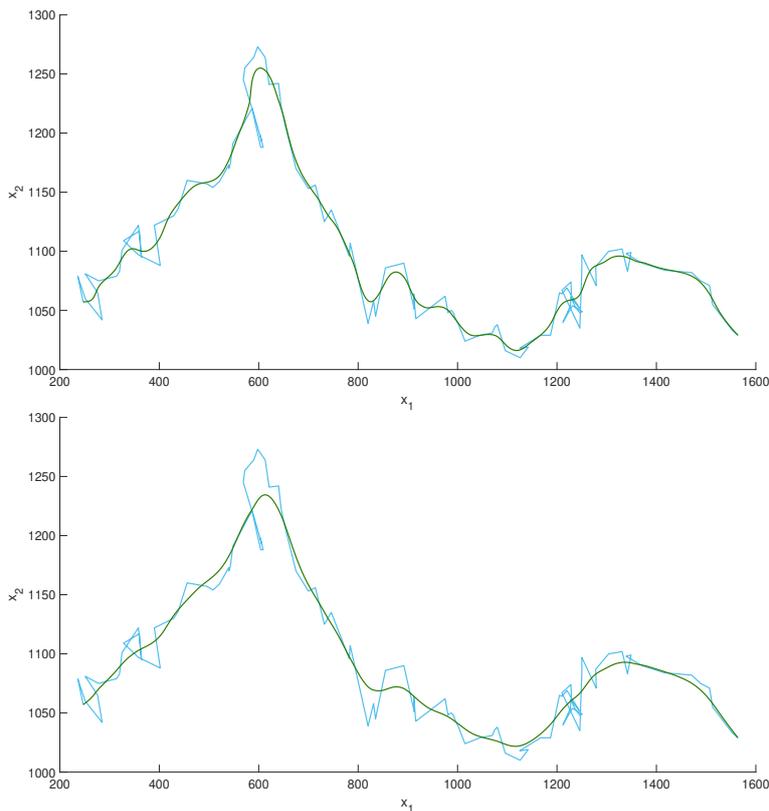


FIGURE 5. Original trajectory (light blue line) and results of smoothing (green lines) for $\delta = 0.003$ (top) and $\delta = 0.01$ (bottom). When no further self-intersections are detected, the algorithm performs 50 additional curve evolution steps before stopping (trajectory is from Dataset_1).

3. ANALYSIS OF RANDOM PARTS

Once the trajectories have been smoothed, we want to analyze the random parts. A common tool to analyze random walks is the so-called mean squared displacement (MSD); indeed, it allows us to distinguish the type of diffusion the random walker shows. The motion can be either diffusive, subdiffusive, or superdiffusive. For example, Brownian motion is a type of diffusive process; examples of anomalous diffusion, superdiffusion, and subdiffusion are the Levy flights or the random walks in crowded environments, respectively [21, 22].

The mean squared displacement is a measure of the deviation of a particle concerning a reference point. It is defined as

$$\rho(t) = \langle |\mathbf{x}(t) - \mathbf{x}^0|^2 \rangle, \quad (3.1)$$

where \mathbf{x}^0 is the reference point that can be either the initial point of the trajectory or just a point of the trajectory as it will be discussed in the next section.

Consider first normal diffusion; the probability density function of the positions of particles is Gaussian and the variance grows linearly with time. It can be proved [21] that the MSD grows as

$$\rho(t) \sim t, \quad (3.2)$$

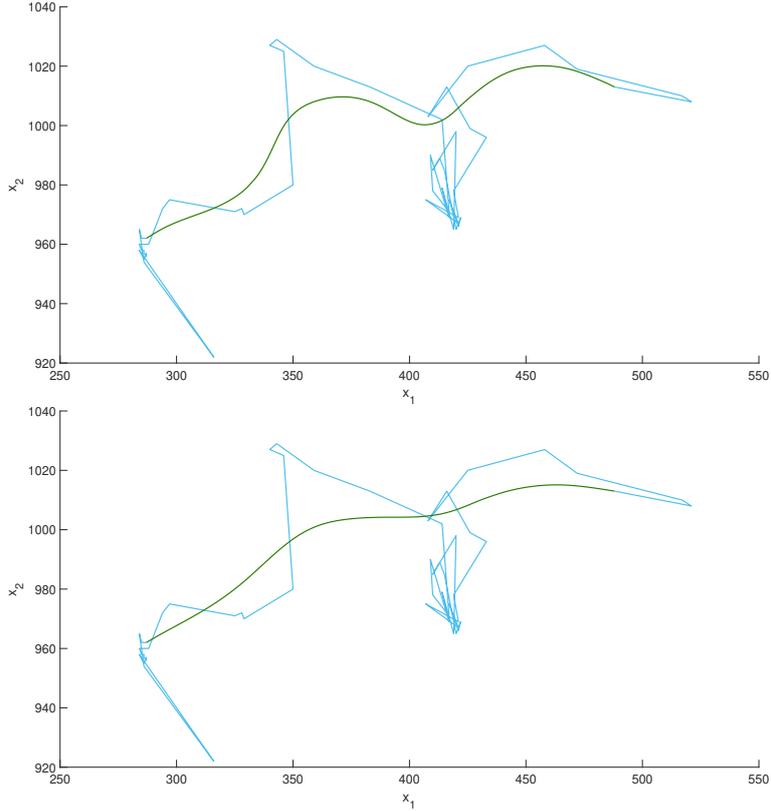


FIGURE 6. Original trajectory (light blue line) and results of smoothing (green lines) for $\delta = 0.003$ (top) and $\delta = 0.01$ (bottom). When no further self-intersections are detected, the algorithm performs 50 additional curve evolution steps before stopping (trajectory is from Dataset_3).

in particular

$$\rho(t) = 2dDt, \quad (3.3)$$

where d is the dimension of the space, D is the diffusion coefficient and t is time.

To characterize anomalous diffusion one usually defines the *Hurst exponent* H

$$\rho(t) \sim t^{2H} \quad (3.4)$$

to see how the MSD grows with time. Then

- For $0 < H < \frac{1}{2}$ the random motion is subdiffusive.
- For $H = \frac{1}{2}$ one obtains normal diffusion.
- For $H > \frac{1}{2}$ the random motion is superdiffusive.
- For $H = 1$ one obtains ballistic motion.
- For $H > 1$ the random motion is superballistic.

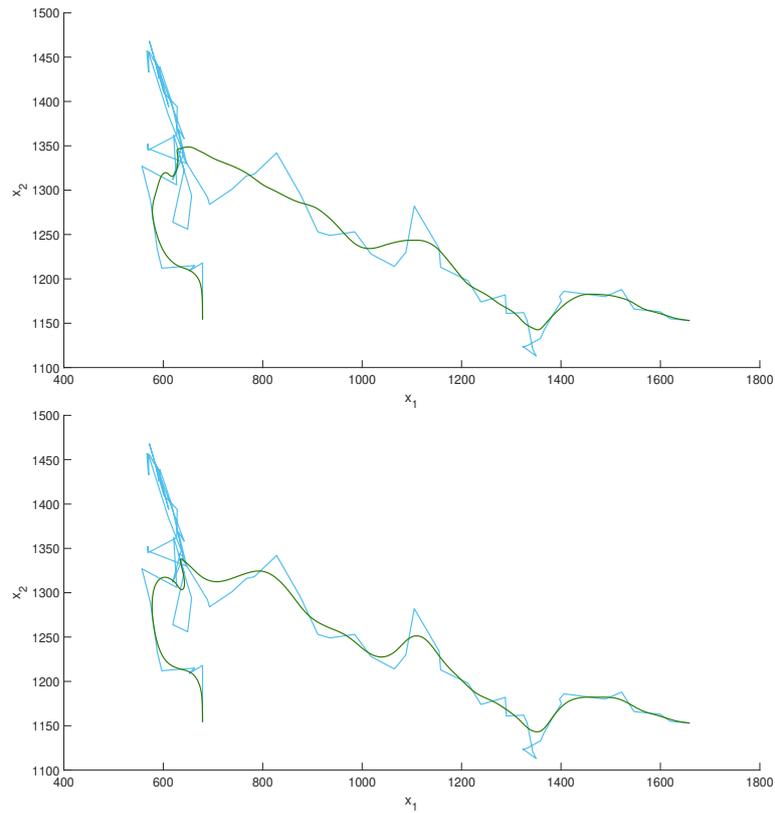


FIGURE 7. Original trajectory (light blue line) and results of smoothing (green lines) for $\delta = 0.003$ (top) and $\delta = 0.01$ (bottom). When no further self-intersections are detected, the algorithm performs 50 additional curve evolution steps before stopping (trajectory is from Dataset_3).

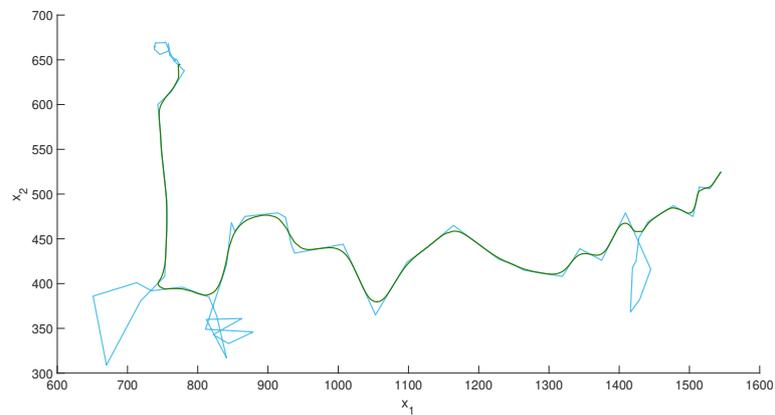


FIGURE 8. Original trajectory (light blue line) and results of smoothing for adaptive choice of parameters (green line). When no further self-intersections are detected, the algorithm performs 50 additional curve evolution steps before stopping.

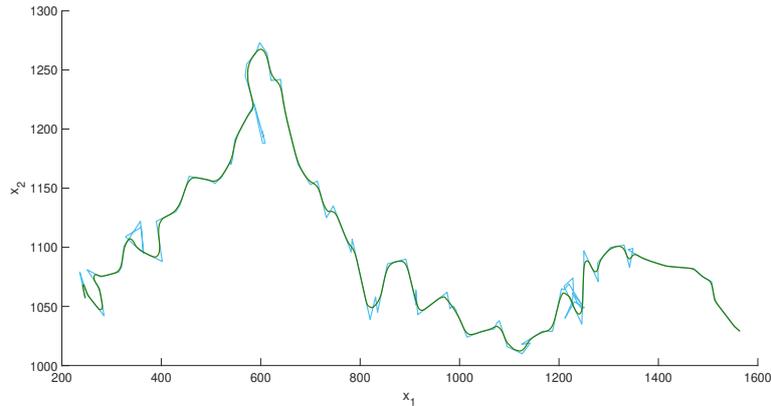


FIGURE 9. Original trajectory (light blue line) and results of smoothing for adaptive choice of parameters (green line). When no further self-intersections are detected, the algorithm performs 50 additional curve evolution steps before stopping.

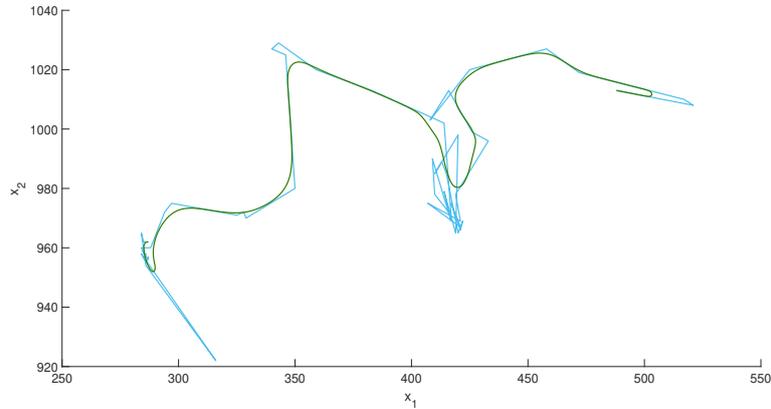


FIGURE 10. Original trajectory (light blue line) and results of smoothing for adaptive choice of parameters (green line). When no further self-intersections are detected, the algorithm performs 50 additional curve evolution steps before stopping.

Therefore, to distinguish the type of diffusion followed by the random walker one has to find H . Considering the logarithm

$$\log \rho(t) \sim 2H \log t, \quad (3.5)$$

and the dependence on time of the MSD in a log–log scale, results in $\alpha = 2H$ being the slope of the line obtained in the double-log scaling. Figure 12 shows the plot of MSD versus time in the log–log scale. In the next section we will give two different discrete definitions of $\rho(t)$ and discuss their differences.

3.1. Mean squared displacement

The first way to define the MSD is the so-called *ensemble-averaged* MSD (EAMSD). Consider a set of trajectories. Every trajectory has an initial point: for the i -th trajectory it will be indicated as \mathbf{x}_i^0 and will be

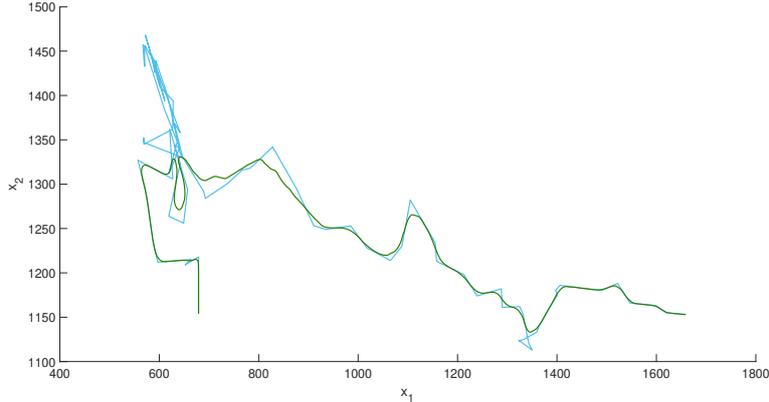


FIGURE 11. Original trajectory (light blue line) and results of smoothing for adaptive choice of parameters (green line). When no further self-intersections are detected, the algorithm performs 50 additional curve evolution steps before stopping.

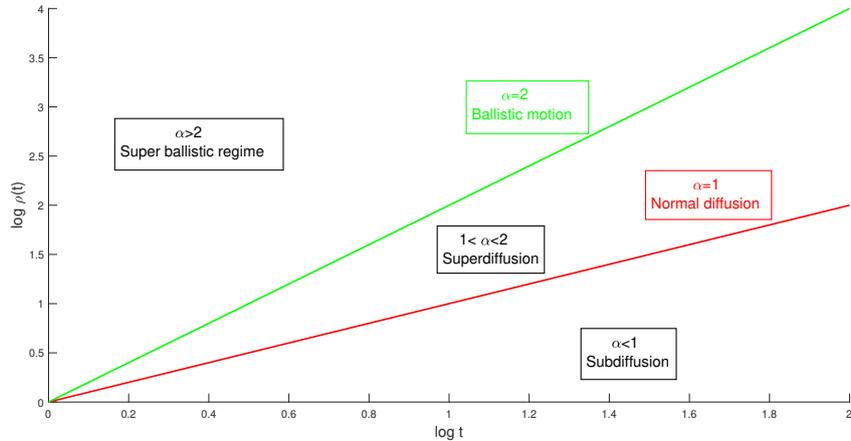


FIGURE 12. Plot of MSD versus time in log–log scale. Normal diffusion (red line) corresponds to slope $\alpha = 1$ ($H = \frac{1}{2}$), and ballistic motion (green line) corresponds to slope $\alpha = 2$ ($H = 1$). $\alpha > 2$ gives the superballistic regime, while $1 < \alpha < 2$ gives superdiffusion. Subdiffusion is obtained for $0 < \alpha < 1$.

considered the reference point for that trajectory. Then, the EAMSD at time t is defined as

$$\rho(t) = \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}_i(t) - \mathbf{x}_i^0\|^2, \quad (3.6)$$

where N is the total number of trajectories [39]. Notice that, in our case, trajectories do not have the same length therefore for different times t we get different values of N . We consider then the same formula but generalized as

$$\rho(t) = \frac{1}{N_t} \sum_{i=1}^{N_t} \|\mathbf{x}_i(t) - \mathbf{x}_i^0\|^2, \quad (3.7)$$

where N_t is the number of trajectories at time t and \mathbf{x}_i^0 is the initial point of the trajectory.

Getting statistically significant results would require many trajectories and long observation times: this is not usually the case in studying biological systems. For this reason, a different definition for the MSD is introduced, the so-called *time-averaged* MSD (TAMSD) [20, 39]. In this case, instead of time, we consider *time lags*. Consider the i -th trajectory with initial time t_i^0 and final time T_i . Consider the data acquisition time interval ΔT : for the i -th trajectory we have the cell position at times $(n\Delta T) + t_i^0$ for $n = 0, 1, 2, \dots, \frac{T_i - t_i^0}{\Delta T}$; here n is the time lag.

Let us denote by $K_i = \frac{T_i - t_i^0}{\Delta T}$ the final time lag for the i -th trajectory. The TAMSD is defined as

$$\bar{\rho}_i(n\Delta T) = \frac{1}{K_i - n + 1} \sum_{j=0}^{K_i - n} \|\mathbf{x}_i((j+n)\Delta T) - \mathbf{x}_i(j\Delta T)\|^2. \quad (3.8)$$

The intuitive idea is to consider many small trajectories inside one single trajectory and make an average of the displacement. In this way, we obtain more values for the MSD from a single trajectory. Notice that with increasing time lag n the number of sub-trajectories inside the original one decreases and this results in fewer data points and more variance in the value of the TAMSD. Therefore, to get meaningful results, the TAMSD for the i -th trajectory should always be calculated for time lags n in the following interval [40]

$$n \leq \frac{T_i - t_i^0 + 1}{4}. \quad (3.9)$$

One can then consider the *ensemble average of the time-averaged* MSD (EATAMSD) [39]. As in equation 3.7, we consider the generalized formula

$$\bar{\rho}(n\Delta T) = \frac{1}{N_n} \sum_{i=1}^{N_n} \bar{\rho}_i(n\Delta T), \quad (3.10)$$

where N_n is the number of trajectories for which $\bar{\rho}_i(n\Delta T)$ exists. We consider a similar restriction as the one for n for N_n . A too-small value of N_n would lead to high variance in the value of $\bar{\rho}(n\Delta T)$, therefore we consider only

$$N_n \geq \frac{N}{4}, \quad (3.11)$$

where N is the total number of trajectories.

Notice that, for $\bar{\rho}(n\Delta T)$ it holds similar asymptotic behaviors as in (3.2) for $\rho(t)$ and the classification of anomalous diffusion with the Hurst exponent as in (3.4) with the only difference that for $\bar{\rho}(n\Delta T)$, we consider the time lag n and not the time t [39].

We developed two different methods to detect the random parts, also referred to as random sub-trajectories throughout the text. One considers the evolution of the lengths of the original segments while the other considers the self-intersections in the trajectory.

3.2. Random parts extracted by evolving curves

In this approach, if a segment disappears during smoothing, we consider it to belong to a random part of the motion. Indeed, random segments are usually shorter and in areas of high curvature; this causes their lengths to vanish as they evolve.

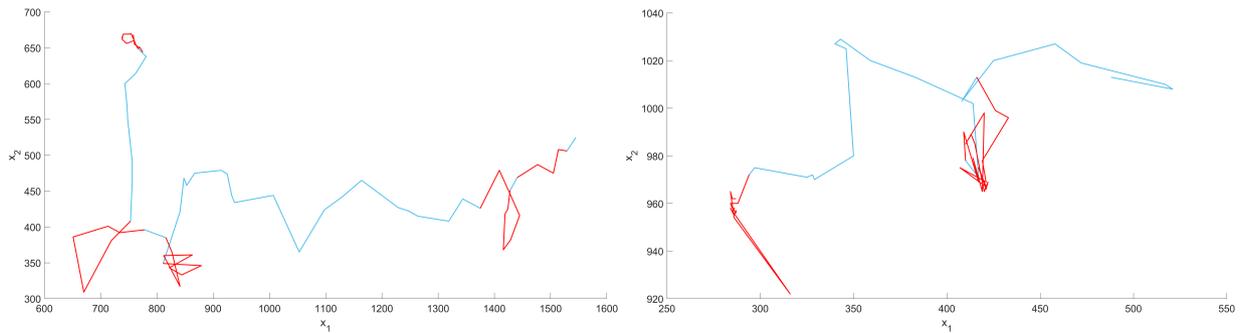


FIGURE 13. Extraction of random parts by evolving curves: original trajectory (light blue line), and extracted random parts (red lines) by evolving curves.

A random sub-trajectory is obtained when more consecutive segments disappear. To capture the random motion for all the points in the disappeared segments, we included also the point immediately preceding the start and the point immediately following the end of the random sub-trajectory. Since our goal is to analyze the diffusive behavior and do a statistical analysis of the extracted parts, we need a minimum number of points for each random sub-trajectory; therefore, we chose to consider only trajectories with at least 5 points.

We considered 3 different datasets as described in Section 1, extracted the random parts, and analyzed them separately. For all the datasets we chose the same parameters for the trajectory smoothing as follows: $\delta = 0.003$, $\lambda = 20$, $\omega = 50$, and $\tau = 0.000001$. The parameters δ and λ were constant and, when no more self-intersections were detected, the algorithm performed 50 more iterations and stopped. Figure 13 shows the result of the random parts detection for the trajectories shown in Figures 4, 6: the light blue lines are the original trajectories, and the red parts are the extracted random parts.

We then applied the formula (3.10) to the 3 different datasets, and calculated the values of α . We considered the log-log dependence on time of the EATAMSD and computed linear regression using the *polyfit* function in MATLAB. Figure 14 shows the results for the different datasets: the blue dots represent the values of the EATAMSD, while the red line is the result of the linear regression. The extracted random parts were 181 for the dataset in Figure 14a, 105 for the one in Figure 14b, and 77 for the one in Figure 14c. As one can see, the EATAMSD shows subdiffusive motion for the 3 datasets analyzed.

3.3. Random parts extracted from curve self-intersections

The second approach to finding the random parts considers the curve self-intersections. Indeed, the random segments are in areas of high curvature where the cell does a back-and-forth motion causing the random segments to intersect. Thanks to the algorithm described in Section 2.3.1, we can detect the two grid points where the intersection starts and ends. We define as random segments the segments to which the two points belong and all the segments in between. If the grid point where the start of the self-intersection is detected belongs to the 2 segment, *i.e.* is the endpoint of those segments, we consider as initial segment the one in the direction of parametrization. Similarly, if the grid point where the end of the self-intersection is detected belongs to the 2 segment, we consider as the last segment of the random sub-trajectory the segment in the direction opposite to parametrization. We then obtain random sub-trajectories; as in Section 3.2, we consider only trajectories with at least 5 points. The detection of the random parts, in this case, can be done directly from the original curves and does not depend on the smoothing parameters. Figure 15 shows the result for the trajectories shown in Figures 8, 10: the light blue line is the original trajectory, and the red parts are the extracted random parts.

We then applied the formula (3.10) to the 3 different datasets, and calculated the values of α as in Section 3.2. Figure 16 shows the results for the different datasets: the blue dots represent the values of the EATAMSD, while the red line is the result of the linear regression, and α is the slope of the line. In this case, the extracted random parts were 205 for the dataset in Figure 16a, 107 for the one in Figure 16b, and 83 for the one in Figure 16c.

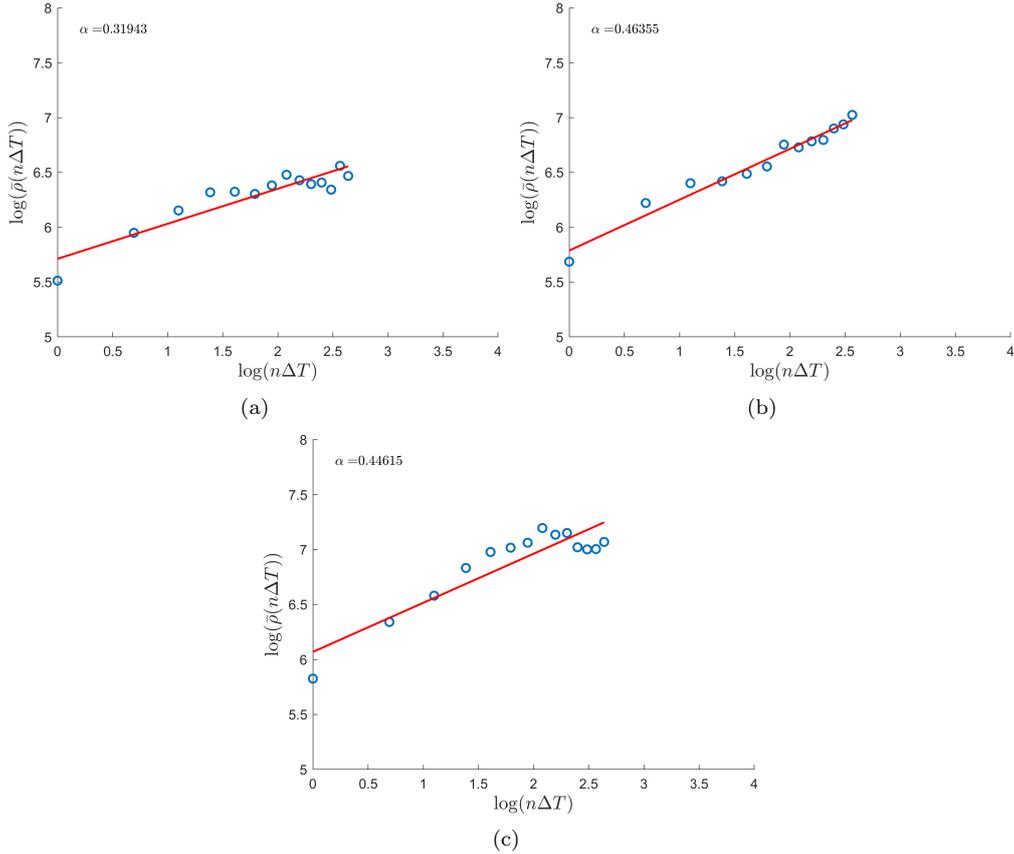


FIGURE 14. log–log plot of $\bar{\rho}(n\Delta T)$ versus $n\Delta T$. Results from the 3 datasets for random parts extracted by evolving curves. Blue dots represent the values of EATAMSD, the red line is the result of the linear regression; α is the slope of the red line.

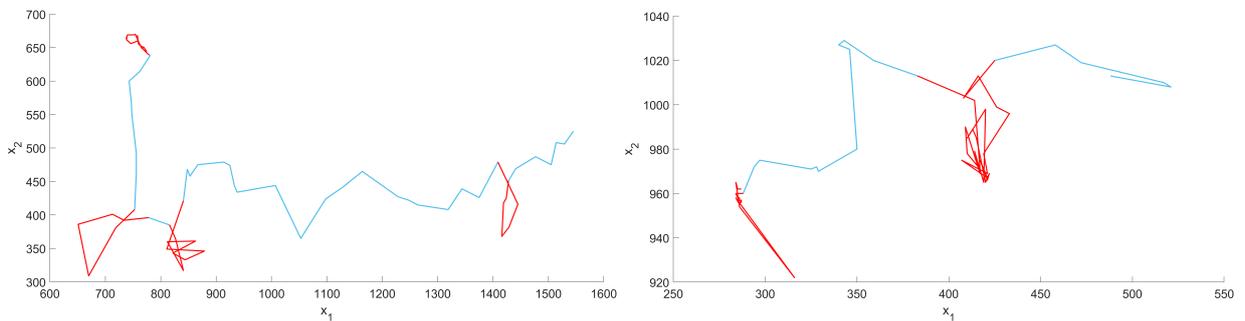


FIGURE 15. Extraction of random parts by evolving curves: original trajectory (light blue line), and extracted random parts (red lines) from curve self-intersections.

As one can see, also with this approach the EATAMSD shows subdiffusive motion for the 3 datasets analyzed. Finally, Table 1 illustrates the results of the slopes α for the 3 datasets and the 2 different approaches. As one can notice, the values for the different approaches are similar, with the maximum difference in the slope values between the two approaches being 0.16173 for the last dataset.

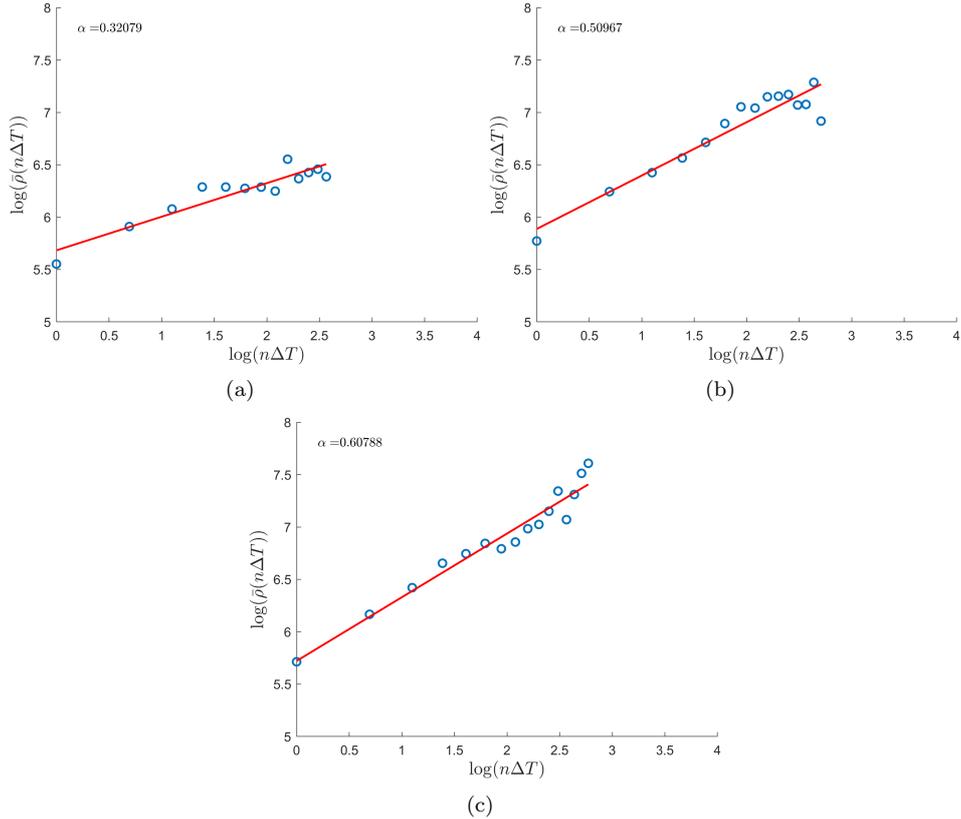


FIGURE 16. \log - \log plot of $\bar{\rho}(n\Delta T)$ versus $n\Delta T$. Results from the 3 datasets for random parts extracted from curve self-intersections. Blue dots represent the values of EATAMSD, the red line is the result of the linear regression; α is the slope of the red line.

TABLE 1. Slope α of the line in the \log - \log scale for 3 different datasets. The first column shows the results for the disappearing segments approach, while the second column shows the results for the self-intersection approach.

| Dataset | α for disappearing segments approach | α for self-intersections approach |
|-----------|---|--|
| Dataset_1 | 0.31943 | 0.32079 |
| Dataset_2 | 0.46355 | 0.50967 |
| Dataset_3 | 0.44615 | 0.60788 |

4. VECTOR FIELD RECONSTRUCTION

The mathematical model we propose to reconstruct the 2D vector field is based on interpolating/extrapolating the given information about the vector field. In particular, we use the smoothed velocities on the smoothed trajectories as sparse samples to reconstruct the wound attractant field. Initially, we consider the image domain of the 2D microscopy video of macrophages moving toward the wound. Then, we define the model domain by cutting off from the image domain some pixels, *i.e.* squares, where the vector field is 0 respectively where the sparse samples are located.

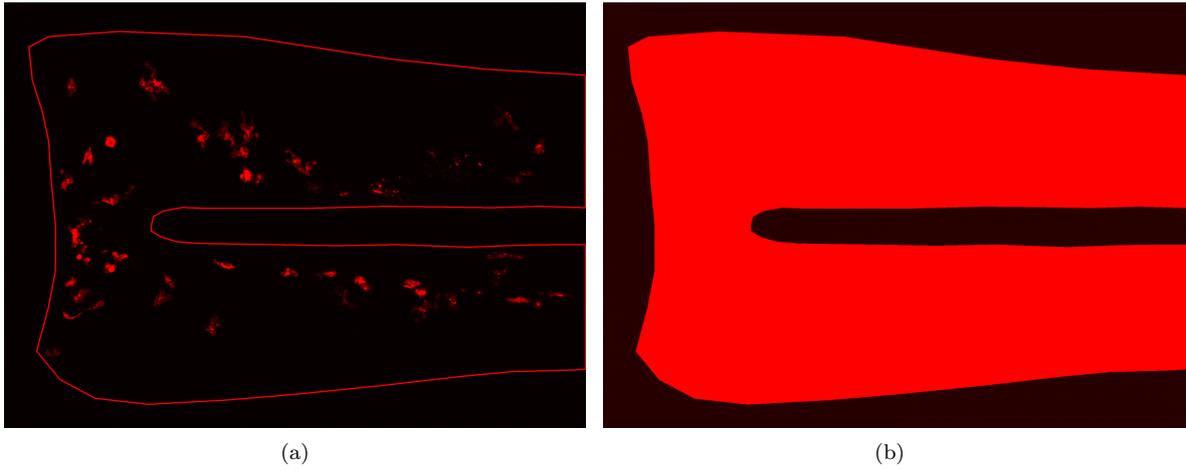


FIGURE 17. (a) Black area indicates the image domain (image is from Dataset_1). The red curve indicates the boundary of $\tilde{\Omega}_{cut}$. (b) $\tilde{\Omega}_{cut}$, region of the image domain where the wound attractant field is reconstructed.

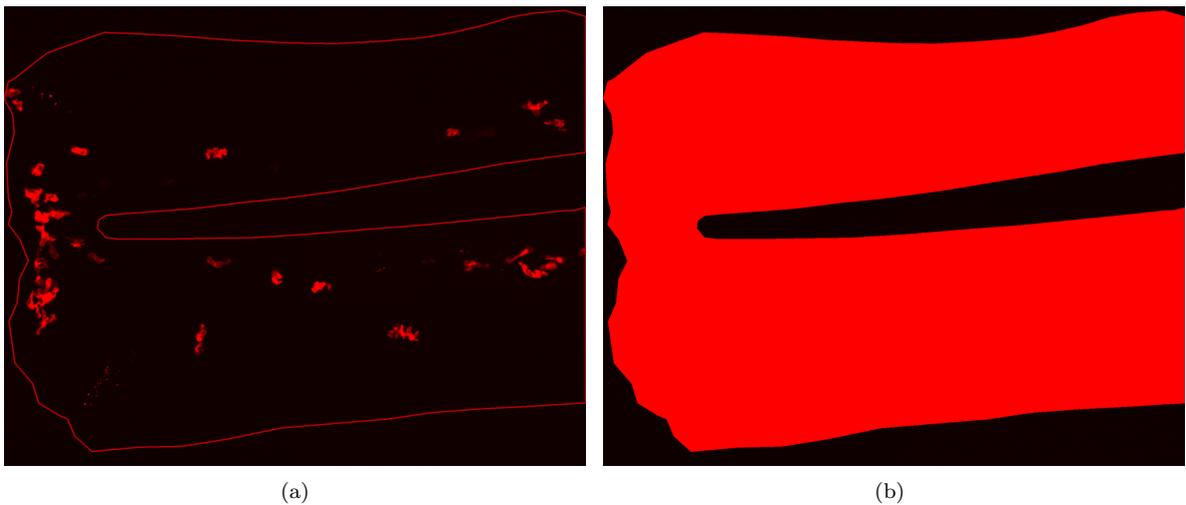


FIGURE 18. (a) Black area indicates the image domain (image is from Dataset_2). The red curve indicates the boundary of $\tilde{\Omega}_{cut}$. (b) $\tilde{\Omega}_{cut}$, region of the image domain where the wound attractant field is reconstructed.

Consider $\tilde{\Omega}$ the auxiliary domain, *i.e.* the image domain. In our specific application, we know that there is a part of the domain where the vector field is 0 because either is outside of the fish body or because the macrophages can not enter that part of the fish body. Therefore, we first consider the cut domain $\tilde{\Omega}_{cut}$ as indicated in Figures 17b, 18b, 19b. In the figures, the black area represents the image domain $\tilde{\Omega}$ while the red area represents $\tilde{\Omega}_{cut}$. Let us define

$$\Gamma^2 = \partial\tilde{\Omega}_{cut} \quad (4.1)$$

the domain boundary, cf. Figures 17a, 18a, 19a. By the construction, $\partial\tilde{\Omega}_{cut}$ is Lipschitz.

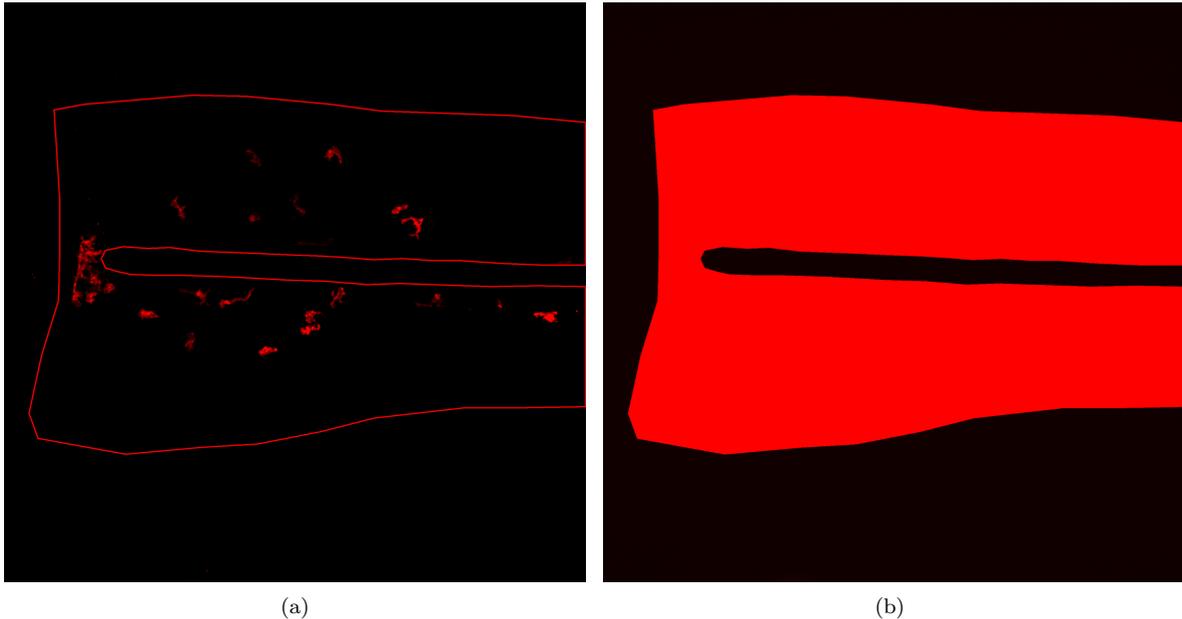


FIGURE 19. (a) Black area indicates the image domain (image is from Dataset_3). The red curve indicates the boundary of $\tilde{\Omega}_{cut}$. (b) $\tilde{\Omega}_{cut}$, region of the image domain where the wound attractant field is reconstructed.

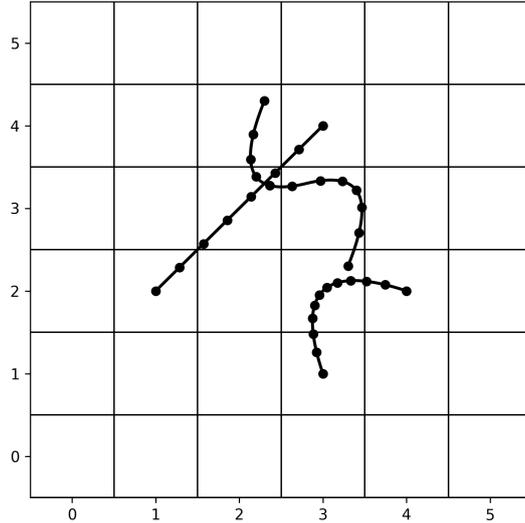


FIGURE 20. Example of trajectories (black lines) in a pixel grid. The black dots represent the trajectories' grid points.

Let us now consider the smoothed trajectories: each grid point on the curve is associated with a smoothed velocity vector. We consider a pixel grid of dimension $n_1 \times n_2$ with grid size h and check the pixels to which the grid points of the trajectories belong as shown in Figure 20. We then use those squares in the definition of Dirichlet conditions inside $\tilde{\Omega}_{cut}$. We will consider the 2 components of the vectors and the vector lengths separately; throughout the text, the term sparse sample will indicate the position and value of one of those. Let

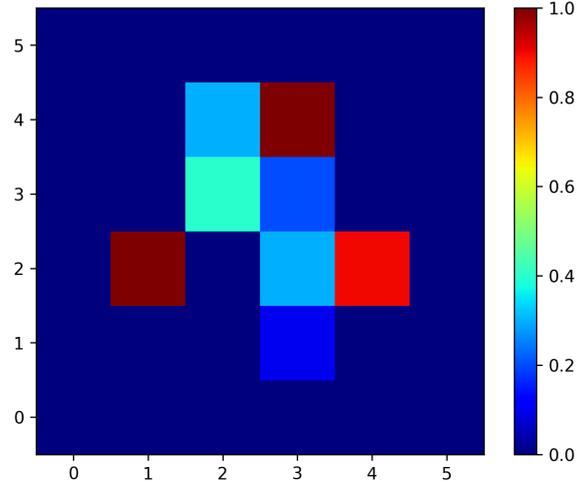


FIGURE 21. Visualization of the values of the squares in Figure 20: the value is the mean of the values of the grid points belonging to that square. The other squares have value 0 for visualization.

us denote by S_p , $p = 1, \dots, s$ the squares of size h where the sparse samples are located. We also assume that none of S_p touches the boundary of $\tilde{\Omega}_{cut}$. Let us define

$$\tau_p = \{v_{i,j} : v_{i,j} \text{ is value in center of } S_p \text{ for some } p \in \{1, \dots, s\}\}, \quad (4.2)$$

and

$$\tau_h = \{u_{i,j} : u_{i,j} \text{ is value in vertex of } S_p \text{ for some } p \in \{1, \dots, s\}\}. \quad (4.3)$$

The values of the Dirichlet conditions will be constructed by using the values of the sparse samples; if more grid points of the curve(s) belong to the same pixel, we consider as $v_{i,j}$ the mean value of the sparse samples belonging to that pixel. Therefore, if two or more trajectories cross in one pixel, we consider as $v_{i,j}$ the mean value of all the sparse samples belonging to that pixel. The visualization of the described process for the domain in Figure 20 is shown in Figure 21; the values of the other pixels are set to 0 for better visualization. To prove the existence and uniqueness of a weak solution for the minimization problem we will consider, we need to have a Lipschitz domain; this is not the case for the domain shown in Figure 21 if we would cut away the 2 squares that have in common only one vertex. Therefore, we add another step to the construction of the model domain. If the squares that we would cut away would have in common only one vertex, we add another two squares and set their values as the mean value of the two squares having in common that vertex. In Figure 22 is visualized the result for the domain shown in Figure 21. Let us define

$$S = \bigcup_{p=1}^s \bar{S}_p \quad (4.4)$$

and

$$\Gamma^1 = \partial S, \quad (4.5)$$

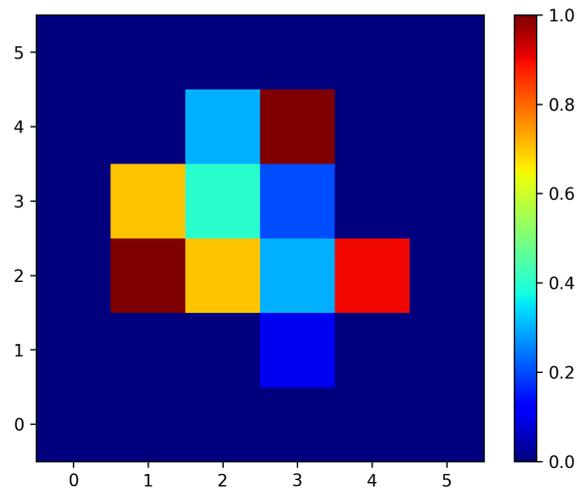


FIGURE 22. Lipschitz domain associated to the domain in Figure 21. The squares that have in common the vertex are considered and their value is set as the mean of the other two squares.

Indicate by

$$\Omega = \tilde{\Omega}_{cut}/S, \quad (4.6)$$

so that

$$\partial\Omega = \Gamma^2 \cup \Gamma^1. \quad (4.7)$$

The last step leading to the definition of Dirichlet boundary conditions is the definition of the values $u_{i,j} \in \tau_h$. To do that, we consider the squares S_p of which $u_{i,j}$ is the value of one of the vertices; to each such S_p a value $v_{i,j} \in \tau_p$ is associated. Therefore, we set $u_{i,j}$ as the mean of those values $v_{i,j}$. In Figure 23 is shown the result for the example in Figure 22.

We finally define the continuous function

$$g : \Gamma_1 \rightarrow \mathbb{R}, \quad (4.8)$$

on the boundary Γ_1 in the following way. If two vertices $u_{i,j}$ of an edge have the same value, then the value of g is set as constant on the edge. If the values of the vertices $u_{i,j}$ differ, then we define g on the edge as linearly changing from one value to the other.

Once we have defined the model domain, we consider the following minimization problem for the two vector components and the vector lengths.

$$\min_u \frac{1}{2} \int_{\Omega} \|\nabla u\|^2, \quad (4.9)$$

where $u : \Omega \rightarrow \mathbb{R}$ is a function defined on $\Omega \subseteq \mathbb{R}^2$ and $E[u] = \frac{1}{2} \int_{\Omega} \|\nabla u\|^2$ is the Dirichlet energy. Solving (4.9) leads to

$$-\Delta u^i = 0 \quad i = 1, 2, 3 \quad (4.10)$$

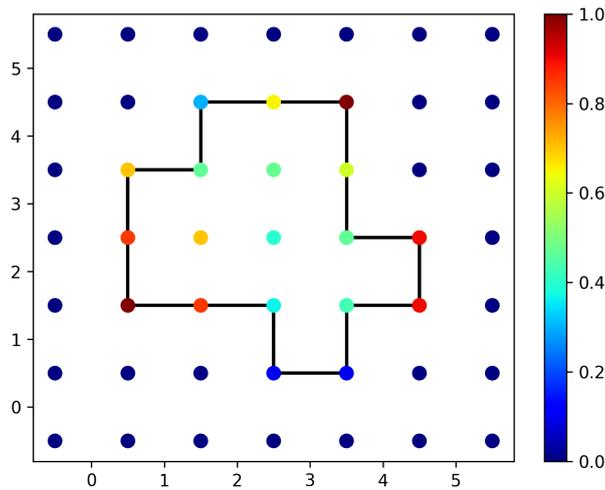


FIGURE 23. Visualization of the values of the vertices of the squares in Figure 22. The values of the vertices $u_{i,j}$ are set as the mean of the values of $v_{i,j}$.

with appropriate boundary conditions. Here, $u^1 = v_x$ and $u^2 = v_y$ are the two vector components, while $u^3 = L$ is the vector length. In particular, we set

- Zero Neumann boundary conditions on Γ^2

$$\frac{\partial u}{\partial \nu}(x) = 0 \quad x \in \Gamma^2, \quad (4.11)$$

where ν is the unit outer normal vector to the boundary Γ^2 .

- Dirichlet boundary conditions on Γ^1

$$u(x) = g(x) \quad x \in \Gamma^1, \quad (4.12)$$

where g is defined in (4.8).

The proof of the existence and uniqueness of a weak solution can be done in the same way as in [36]. Indeed, we constructed the Lipschitz domain Ω , therefore we can apply the theory from [41, 42]. When the double-Laplacian operator for the two vector components is used, the direction of the vectors changes smoothly from one to another, depending on the Dirichlet conditions. However, it is important to note that the Laplace equation averages the values of neighboring points. As previously discussed in [30, 36], while the double-Laplacian effectively maintains the desired directionality of the vectors, it tends to shorten their lengths. Since we want to find the vector field from the provided data, it is logical to consider reconstructing the vector lengths based on the given Dirichlet conditions. Consequently, we propose to apply the same minimization problem (4.9) for determining the vector lengths as well. The final triple-Laplacian model consists of solving the Laplace equation with appropriate boundary conditions three times, first for the x component, second for the y component, and last for the vector length. In particular, we prescribe zero Neumann boundary conditions on Γ^2 for every $i = 1, 2, 3$. On the other hand, the boundary conditions on Γ^1 for each subproblem are constructed considering the values of the sparse samples inside the domain. Therefore, for each $i = 1, 2, 3$, we define the function g_i as

described in (4.8), and solve

$$\begin{cases} -\Delta u^i(x) = 0, & x \in \Omega, \\ \frac{\partial u^i}{\partial \nu}(x) = 0, & x \in \Gamma^2, \\ u^i(x) = g^i(x), & x \in \Gamma^1. \end{cases} \quad (4.13)$$

The discretization of the boundary value problems (4.13) is based on [32]. In particular, for the vertices $u_{i,j}$ which are not vertices of any S_p , $p \in \{1, \dots, s\}$ the standard discretization of the Laplace equation is used. On the other hand, in the vertices of Γ^1 we prescribe Dirichlet values, and on the boundary Γ^2 we consider the discretization of zero Neumann boundary condition similar to [32]. In Section 4.1 we recall how to find the smoothed velocities, cf. [18], and results for the reconstruction of the wound attractant field are shown in Section 4.2.

4.1. Velocity vectors on smooth trajectories

To get the new velocity estimation consider $(\Delta_t)_j$ the time between the two endpoints $\mathbf{x}(u_{j-1})$ and $\mathbf{x}(u_j)$ of the segment j . In the beginning, every segment has the same value of $(\Delta_t)_j$; later, if a segment does not disappear, we consider the same $(\Delta_t)_j$ as in the beginning. On the other hand, if a segment j disappears, we add $\frac{(\Delta_t)_j}{2}$ to the first previous not disappeared segment $j_1 < j$,

$$(\Delta_t)_{j_1} = (\Delta_t)_{j_1} + \frac{(\Delta_t)_j}{2},$$

and $\frac{(\Delta_t)_j}{2}$ to the first following not disappeared segment $j_2 > j$

$$(\Delta_t)_{j_2} = (\Delta_t)_{j_2} + \frac{(\Delta_t)_j}{2}.$$

To find the new velocity estimation for the grid points on the evolved curve, we apply the following algorithm. Set $j = 1$, $i = 1$ and repeat until $j = M - 1$:

- If $r_j^{m+1} > 0$

$$|\mathbf{v}| = \frac{L_j^d}{(\Delta_t)_j}, \quad (4.14)$$

where L_j^d was defined in (2.23).

- for $i = \mathcal{I}(u_{j-1}), \dots, \mathcal{I}(u_j)$

$$\mathbf{v}_i^{m+1} = |\mathbf{v}| \frac{(\mathbf{x}_i^{m+1} - \mathbf{x}_{i-1}^{m+1})}{|\mathbf{x}_i^{m+1} - \mathbf{x}_{i-1}^{m+1}|}. \quad (4.15)$$

If the segment does not disappear, we consider the grid points inside that segment to have constant velocity equal to the new length of the segment divided by the time interval $(\Delta_t)_j$. On the other hand, if the j -th segment disappears, the time $(\Delta_t)_j$ is distributed between the previous and following segments that have not disappeared.

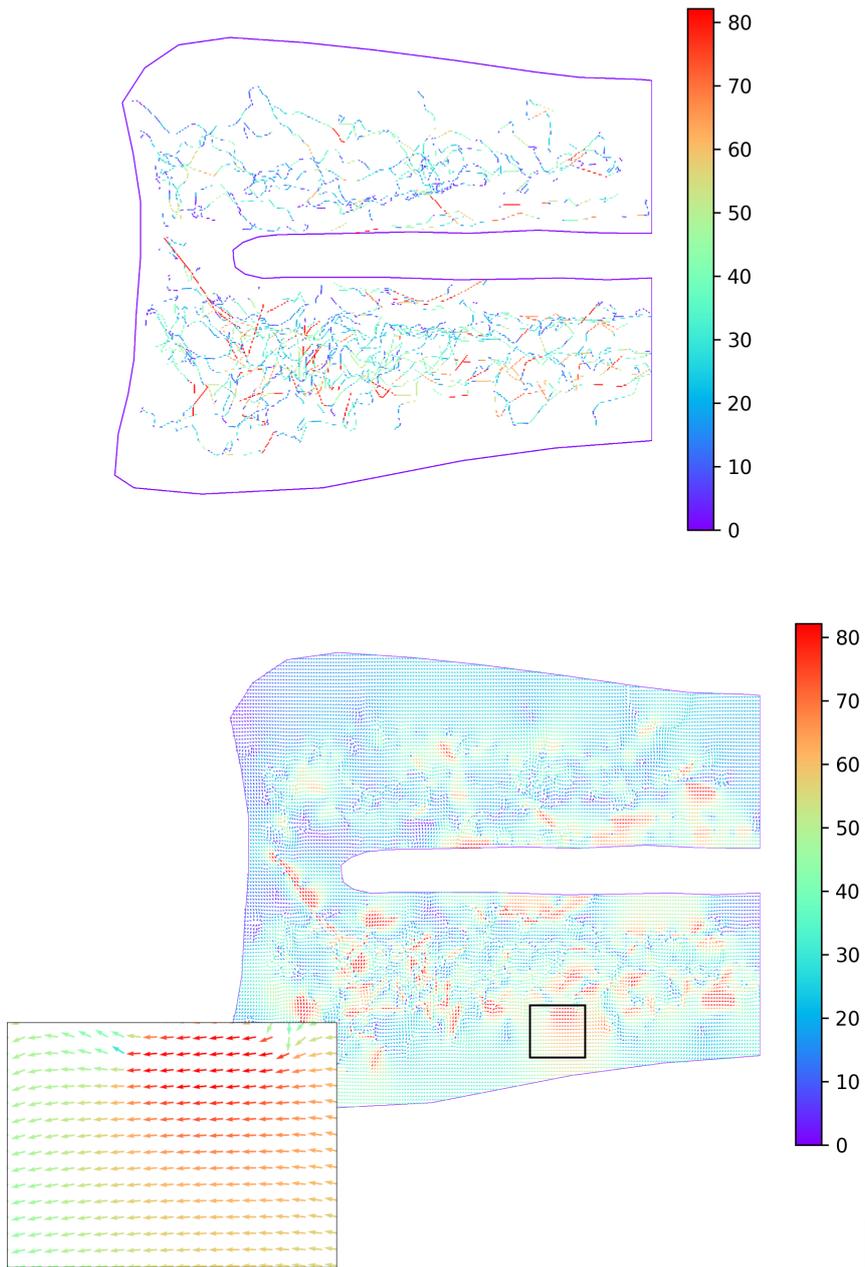


FIGURE 24. Results for the dataset shown in Figure 17. Top: Prescribed velocities on sparse samples obtained from smoothed macrophage trajectories. Bottom: Reconstructed vector field by triple-Laplacian. The zoomed region is indicated by a black box in the reconstructed vector field.

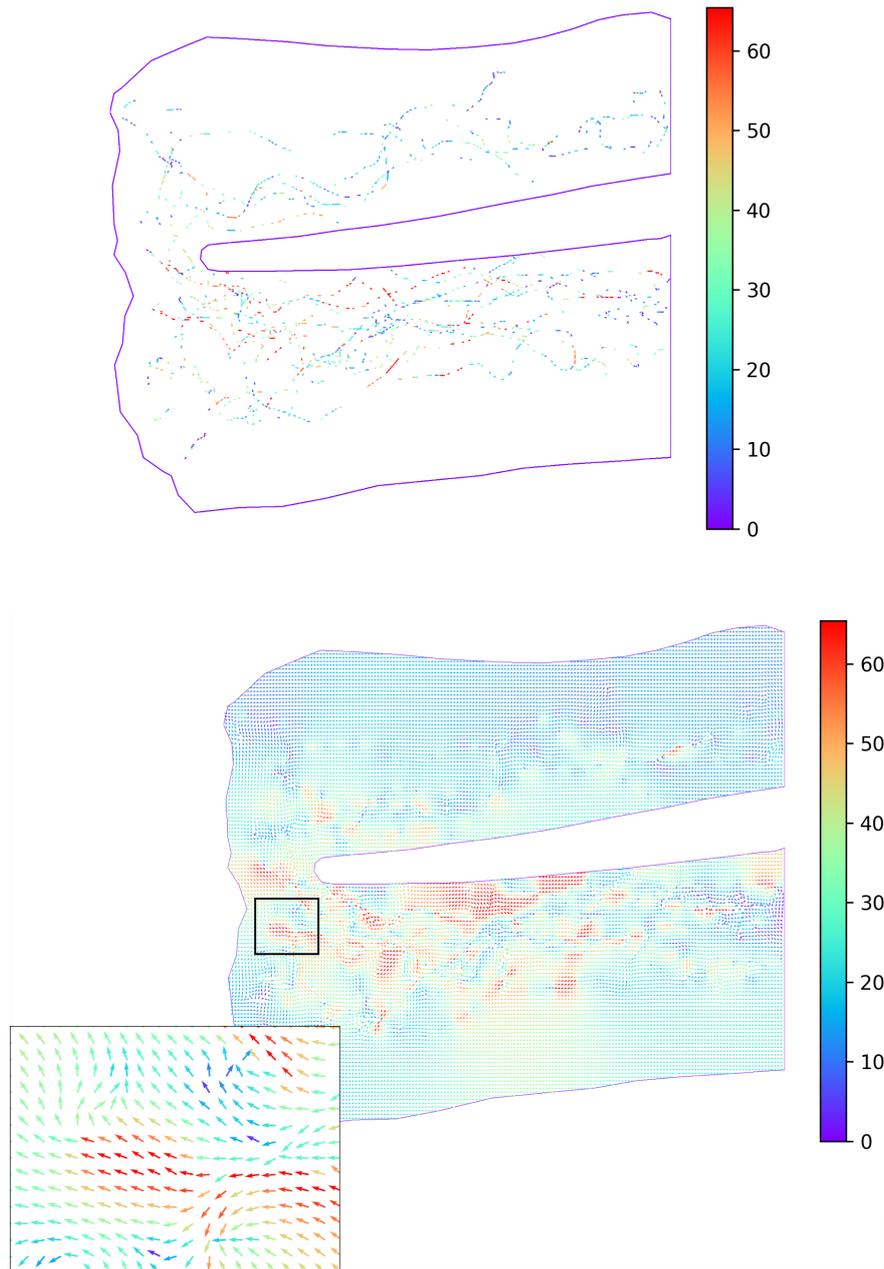


FIGURE 25. Results for the dataset shown in Figure 18. Top: Prescribed velocities on sparse samples obtained from smoothed macrophage trajectories. Bottom: Reconstructed vector field by triple-Laplacian. The zoomed region is indicated by a black box in the reconstructed vector field.

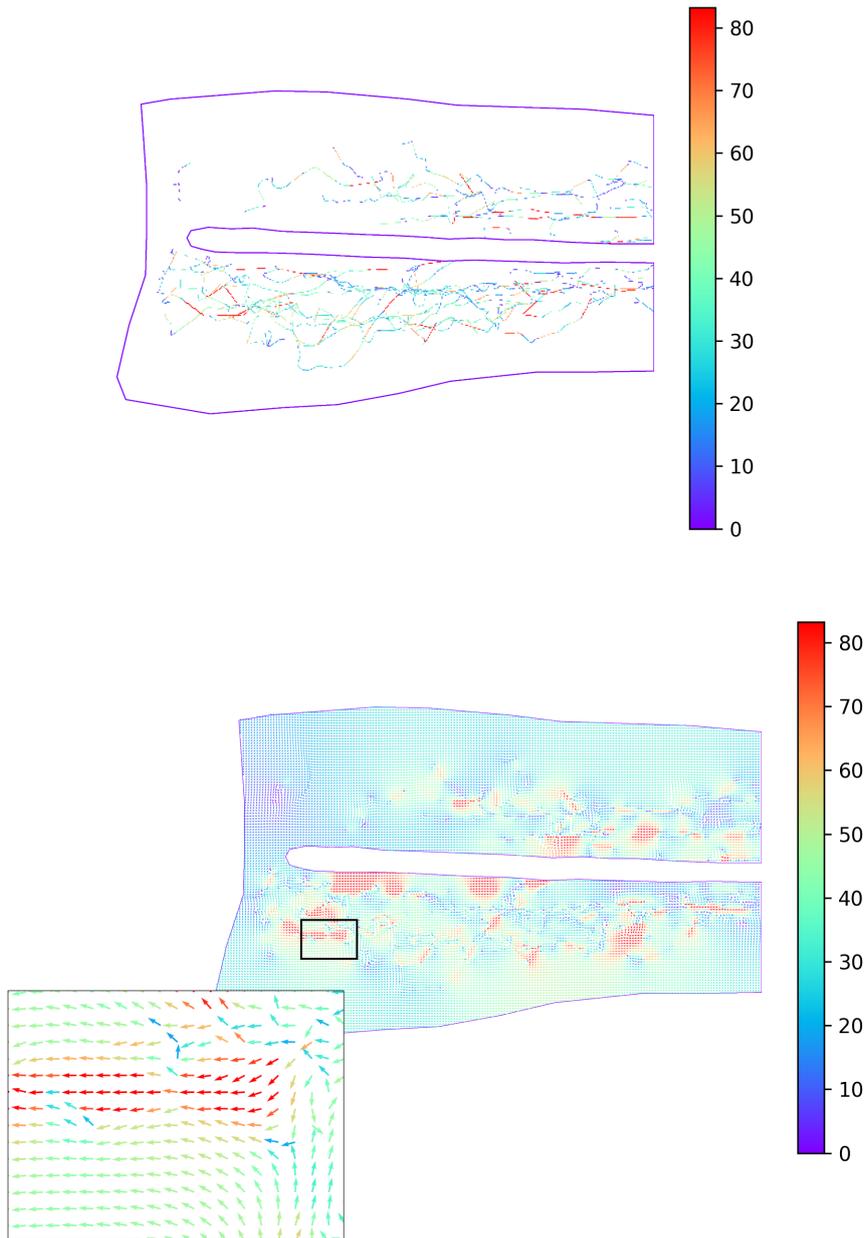


FIGURE 26. Results for the dataset shown in Figure 19. Top: Prescribed velocities on sparse samples obtained from smoothed macrophage trajectories. Bottom: Reconstructed vector field by triple-Laplacian. The zoomed region is indicated by a black box in the reconstructed vector field.

4.2. Results

For all the considered datasets we smoothed the trajectories with the model described in Section 2. Then, we computed the smoothed velocities with the algorithm described in Section 4.1 and used them as sparse samples to reconstruct the wound attractant field driving macrophages to the site of the wound.

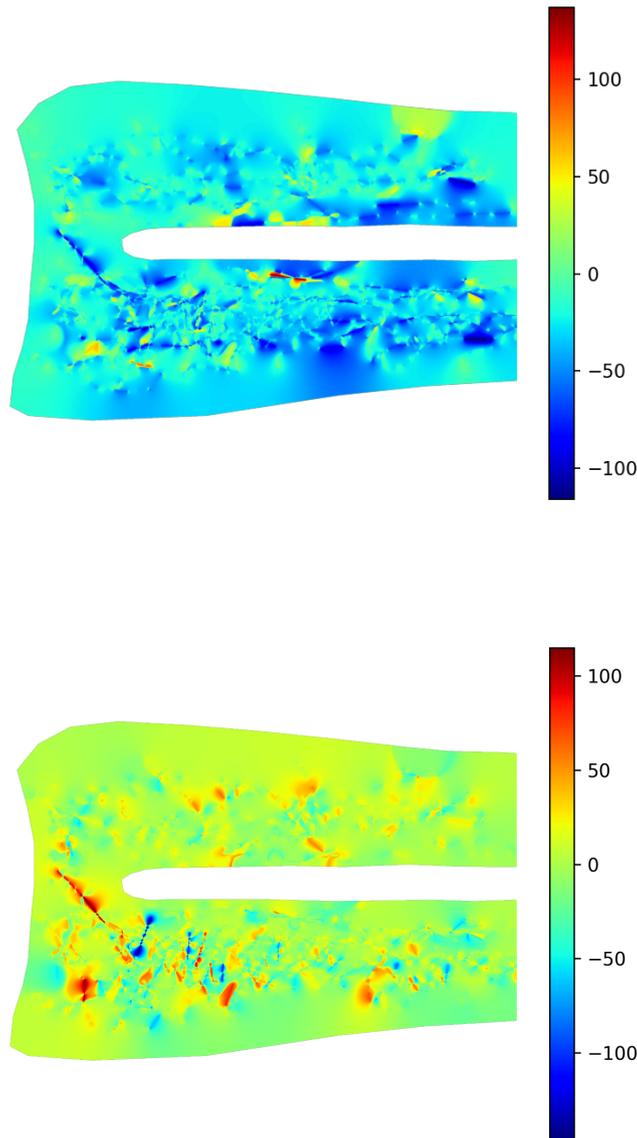


FIGURE 27. Projection in x and y direction of the reconstructed vector field shown in Figure 24. Top: x direction. Bottom: y direction.

Figures 24, 25, 26 show the results for the 3 different datasets whose model domain is visualized in Figures 17, 18, 19, respectively. For all of them, the wound is on the left side. On top, the velocity vectors used as sparse samples are visualized, and on the bottom, one can see the reconstructed vector field. The direction of the vectors is given by the solution of (4.13) for the two vector components, while the color represents the lengths of the vectors, *i.e.*, the solution of (4.13) for the vector lengths. For better visualization, the vectors are normalized.

Figures 27, 28, 29 show the projection of vectors in Figures 24, 25, 26 in x (horizontal) and y (vertical) direction. The images on top show the result for x direction. The blue color represents the regions of higher directionality; indeed, since the wound is on the left side, the more negative the value, the faster the cell is moving toward the wound. The images on the bottom show the results for the y direction. The blue color

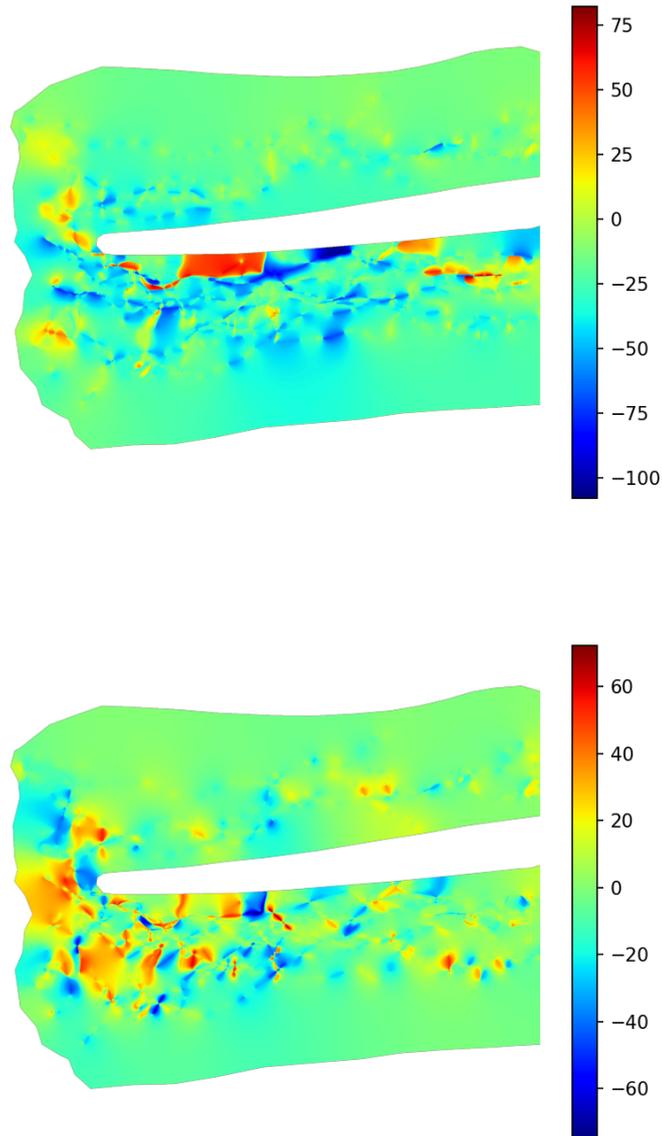


FIGURE 28. Projection in x and y direction of the reconstructed vector field shown in Figure 25.
Top: x direction. Bottom: y direction.

represents fast motion from top to bottom. As one can notice, the fast motion toward the wound (x direction) is in similar parts of the fish body for the 3 datasets, on the bottom and close to the central part of the fish body, where the macrophages can not enter. On the other side, the motion in the y direction seems to be random.

5. CONCLUSIONS

In this paper, we proposed a new workflow to analyze macrophage trajectories and reconstruct the velocity vector field driving macrophages toward the wound during wound healing. First, we smooth the trajectories; the smoothing algorithm allows us to separate the random parts from the directional parts of the motion. We considered two ways of defining the random parts. The first considers the evolution of the lengths of the original segments and defines a segment as random if it disappears during the smoothing process. This usually happens

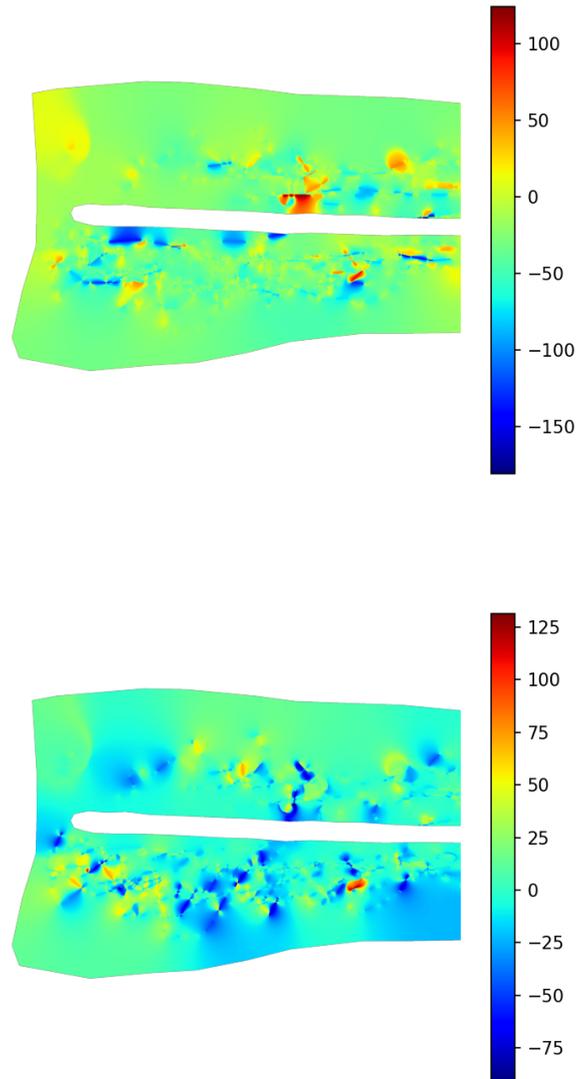


FIGURE 29. Projection in x and y direction of the reconstructed vector field shown in Figure 26. Top: x direction. Bottom: y direction.

in regions of high curvature where the cell is moving back and forth without a clear directionality. The second approach to defining the random parts takes into account the self-intersections of the trajectory. We detect the points where the curve intersects itself and define as random the segments that belong to a self-intersecting part of the trajectory. Once we had the random sub-trajectories, we examined them using the mean squared displacement (MSD). This is a common tool used to characterize the type of diffusion motion followed by a random walker. The analysis showed that the random motion is subdiffusive; subdiffusion usually occurs in crowded environments where the random walk is not free to move in any direction but is constrained by the environment. Finally, we used the velocities found on the smoothed trajectories as sparse samples to reconstruct the wound attractant field that drives the macrophages during wound healing. The model we proposed minimizes the Dirichlet energy for the two vector components and the vector length with appropriate boundary conditions. In solving the minimization problem, we ended up solving the Laplace equation with Dirichlet conditions on the

sparse samples and zero Neumann boundary conditions on the domain boundary. The result of the reconstruction is an interpolation/extrapolation process in which the information is extrapolated from the Dirichlet conditions where the vectors do not influence each other and interpolated in the regions where the vectors do influence each other. We developed such a model to extract the information about the velocity vector field only from the Dirichlet conditions, without the need to make any assumptions about the shape or properties of the wound attractant field. In this paper, for the vector field reconstruction, we considered trajectories (that is, the sparse samples) for the entire duration of the experiments (around 6 hours). However, it is possible to consider the trajectories only in a given time interval, and in that way, reconstruct the time dependent vector field. Such an approach was suggested in [43].

Modeling a phenomenon such as motion during wound healing *in vivo* is challenging due to the large number of possible parameters that can influence the motion of the cells. Whilst the motion is influenced by the wound attractant field, other causes may be responsible for the shape of the trajectories of these cells. Firstly, the motion of cells can vary significantly depending on the environment in which they are moving. Indeed, in [20] the authors studied the effects of a crowded environment on the motion of random walkers subject to a constant velocity vector field. They showed that the effect of constant drift increases the chances of trapping and that the trajectories show a high degree of variability. Therefore, the interplay between drift and environment can be responsible for the emergence of both superdiffusive and subdiffusive behavior. For this reason, it's important to separate the parts of the movement that are not influenced by the wound attractant field and study them separately. A possible future direction that could contribute to the understanding of macrophage movement is the segmentation of the surrounding environment. The information about the environment in which the cells move could be used both to determine a model for the random migration of macrophages and to reconstruct the velocity vector field taking into account more precise boundary conditions within the domain. Moreover, extend the overall workflow to 3D data, including all steps from segmentation and tracking to vector field reconstruction, could provide a more complete description of the process.

Another interesting aspect is the study of macrophage movement under normal conditions, *i.e.* without the presence of a wound. In this case, the overall trajectories of the macrophages could be considered as random walks, and, with a sufficiently large data set, a model of migration under normal conditions could perhaps be determined. In general, however, the workflow is not limited to this setting and can be applied to different cell types as well as to different physiological conditions. For example, in the case of wound healing, one could analyze how the reconstructed vector field varies when the shape of the wound changes, in the presence of multiple wounds, or with subsequent wounds. Finally, the smoothed trajectories and extracted random parts can be used in the classification of different states of macrophages. While many studies have reported differences in the morphology and dynamics of macrophage subtypes, such as M1, non-activated M1, and M2 macrophages, comprehensive quantitative analyses remain limited. By combining the segmented shapes of macrophages with properties of their smoothed trajectories and random motion components, a high-dimensional morpho-kinetic feature space can be defined and used for the graph-PDE supervised classification to differentiate between M1 and M2 macrophages [44].

FUNDING

This work was supported by grants H2020-955576, APVV-23-0186, VV-MVP-24-0116, VEGA 1/0249/24. This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 955576 and by the grants APVV-23-0186, VV-MVP-24-0116 and VEGA 1/0249/24.

DATA AVAILABILITY STATEMENT

All macrophages' video data are freely available at Zenodo <https://doi.org/10.5281/zenodo.17833359> [45].

REFERENCES

- [1] F. Barros-Becker, P.Y. Lam, R. Fisher and A. Huttenlocher, Live imaging reveals distinct modes of neutrophil and macrophage migration within interstitial tissues. *J. Cell Sci.* **130** (2017) 3801–3808.
- [2] P. Friedl and B. Weigelin, Interstitial leukocyte migration and immune function. *Nat. Immunol.* **9** (2008) 960–969.
- [3] E. Van Goethem, R. Poincloux, F. Gauffre, I. Maridonneau-Parini and V. Le Cabec, Matrix architecture dictates three-dimensional migration modes of human macrophages: differential involvement of proteases and podosome-like structures. *J. Immunol.* **184** (2010) 1049–1061.
- [4] L. Li, B. Yan, Y.Q. Shi, W.Q. Zhang and Z.L. Wen, Live imaging reveals differing roles of macrophages and neutrophils during zebrafish tail fin regeneration. *J. Biol. Chem.* **287** (2012) 25353–25360.
- [5] Z. Min, Electrical fields in wound healing—an overriding signal that directs cell migration. *Semin. Cell Dev. Biol.* **20** (2009) 674–682.
- [6] T. Sipka, R. Perocesi, M. Groß, F. Ellett, C. Pescia, C. Gonzalez, G. Lutfalla and M. Nguyen-Chi, Damage-induced calcium signaling and reactive oxygen species mediate macrophage activation in zebrafish. *Front. Immunol.* **12** (2021) ID 636585.
- [7] V. Kadirkamanathan, S.R. Anderson, S.A. Billings, X. Zhang, G.R. Holmes, *et al.*, The neutrophil’s eye-view: Inference and visualisation of the chemoattractant field driving cell chemotaxis in vivo. *PLoS One* **7** (2012) e35182.
- [8] F. Ellett, L. Pase, J. Hayman, A. Andrianopoulos and G. Lieschke, Mpeg1 promoter transgenes direct macrophage-lineage expression in zebrafish. *Blood* **117** (2011) e49–e56.
- [9] S.A. Park, T. Sipka, Z. Krivá, M. Nguyen-Chi, G. Lutfalla and K. Mikula, Segmentation-based tracking of macrophages in 2d+time microscopy movies inside a living animal. *Comput. Biol. Med.* **153** (2023) 106499.
- [10] T.Y. Hou, J.S. Lowengrub and M.J. Shelley, Removing the stiffness from interfacial flows with surface tension. *J. Computat. Phys.* **114** (1994) 312–338.
- [11] M. Kimura, Numerical analysis of moving boundary problems using the boundary tracking method. *Japan J. Ind. Appl. Math.* **14** (1997) 373–398.
- [12] K. Mikula and D. Ševčovič, Evolution of plane curves driven by a nonlinear function of curvature and anisotropy. *SIAM J. Appl. Math.* **61** (2001) 1473–1501.
- [13] 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) 1545–1565.
- [14] D. Ševčovič and S. Yazaki, Evolution of plane curves with a curvature adjusted tangential velocity. *Japan J. Ind. Appl. Math.* **28** (2011) 413–442.
- [15] K. Mikula, D. Ševčovič and M. Balazovjeh, A simple, fast and stabilized flowing finite volume method for solving general curve evolution equations. *Commun. Computat. Phys.* **7** (2010) 195–211.
- [16] K. Mikula, J. Urbán, M. Kollár, M. Ambroz, I. Jarolínek, J. Šibík and M. Šibíková, An automated segmentation of natura 2000 habitats from sentinel-2 optical data. *Discrete Continuous Dyn. Syst.* **14** (2021) 1017–1032.
- [17] M. Ambroz, M. Balažovjeh, M. Medl’a and K. Mikula, Numerical modeling of wildland surface fire propagation by evolving surface curves. *Adv. Computat. Math.* **45** (2019) 1067–1103.
- [18] G. Lupi, K. Mikula and S.A. Park, Macrophages trajectories smoothing by evolving curves. *Tatra Mountains Math. Publ.* **86** (2023) pp. 145–170.
- [19] D. Arcizet, B. Meier, E. Sackmann, J.O. Rädler and D. Heinrich, Temporal analysis of active and passive transport in living cells. *Phys. Rev. Lett.* **101** (2008) 248103.
- [20] P. Kubala, M. Cieřla and B. Dybiec, Diffusion in crowded environments: Trapped by the drift. *Phys. Rev. E* **104** (2021) 044127.
- [21] I.M. Sokolov, Models of anomalous diffusion in crowded environments. *Soft Matter* **8** (2012) 9043–9052.
- [22] G.M. Viswanathan, M.G.E. Da Luz, E.P. Raposo and H.E. Stanley, The Physics of Foraging: An Introduction to Random Searches and Biological Encounters. Cambridge University Press (2011).
- [23] K. De Bruin, N. Ruthardt, K. Von Gersdorff, R. Bausinger, E. Wagner, M. Ogris and C. Bräuchle, Cellular dynamics of egf receptor-targeted synthetic viruses. *Mol. Ther.* **15** (2007) 1297–1305.
- [24] F. Dumas, N. Destainville, C. Millot, A. Lopez, D. Dean and L. Salomé, Confined diffusion without fences of a g-protein-coupled receptor as revealed by single particle tracking. *Biophys. J.* **84** (2003) 356–366.
- [25] P. Dieterich, R. Klages, R. Preuss and A. Schwab, Anomalous dynamics of cell migration. *Proc. Natl. Acad. Sci.* **105** (2008) 459–463.

- [26] T.H. Harris, E.J. Banigan, D.A. Christian, *et al.*, Generalized lévy walks and the role of chemokines in migration of effector cd8+ t cells. *Nature* **486** (2012) 545–548.
- [27] S. Huda, B. Weigel, K. Wolf, *et al.*, Lévy-like movement patterns of metastatic cancer cells revealed in microfabricated systems and implicated in vivo. *Nat. Commun.* **9** (2018) 4539.
- [28] A. Kusumi, Y. Sako and M. Yamamoto, Confined lateral diffusion of membrane receptors as studied by single particle tracking (nanovid microscopy). effects of calcium-induced differentiation in cultured epithelial cells. *Biophys. J.* **65** (1993) 2021–2040.
- [29] G.M. Lee, A. Ishihara and K.A. Jacobson, Direct observation of Brownian motion of lipids in a membrane. *Proc. Natl. Acad. Sci. U.S.A.* **88** (1991) 6274–6278.
- [30] M. Fisher, P. Schröder, M. Desbrun and H. Hoppe, Design of tangent vector fields. *ACM Trans. Graph.* **26** (2007) 56–es.
- [31] M. Lage, F. Petronetto, A. Paiva, H. Lope, T. Lewiner and G. Tavares, Vector field reconstruction from sparse samples with applications, in *19th Brazilian Symposium on Computer Graphics and Image Processing* (2006) 297–306.
- [32] K. Mikula, M. Ambroz and R. Mokošová, What was the river Ister in the time of Strabo? A mathematical approach. *Tatra Mountains Math. Publ.* **80** (2021) 71–118.
- [33] F.A. Mussa-Ivaldi, From basis functions to basis fields: vector field approximation from sparse data. *Biol. Cybernet.* **67** (1992) 479–489.
- [34] Z. Belhachmi, D. Bucur, B. Burgeth and J. Weickert, How to choose interpolation data in images. *SIAM J. Appl. Math.* **70** (2009) 333–352.
- [35] C.-B. Schönlieb, *Partial Differential Equation Methods for Image Inpainting*. Cambridge University Press (2015).
- [36] G. Lupi and K. Mikula, Vector field reconstruction from sparse samples by triple-laplacian, in *Proceedings of the Conference Algoritmy* (2024) 85–98.
- [37] K. Mikula and M. Ohlberger, Inflow-implicit/outflow-explicit scheme for solving advection Equations, in *Finite Volumes in Complex Applications VI, Problems & Perspectives* (2011) 683–692.
- [38] M. Ambroz, M. Kollár and K. Mikula, Semi-implicit scheme for semi-automatic segmentation in naturasat Software, in *Proceedings of the Conference Algoritmy* (2020) 171–180.
- [39] B.M. Regner, D. Vučinić, C. Domnisoru, T.M. Bartol, M.W. Hetzer, D.M. Tartakovsky and T.J. Sejnowski, Anomalous diffusion of single particles in cytoplasm. *Biophys. J.* **104** (2013) 1652–1660.
- [40] N. Ruthardt, D.C. Lamb and C. Bräuchle, Single-particle tracking as a quantitative microscopy-based approach to unravel cell entry mechanisms of viruses and pharmaceutical nanoparticles. *Mol. Ther.* **19** (2011) 1199–1211.
- [41] J. Nečas, *Direct Methods in the Theory of Elliptic Equations*. Springer, Heidelberg (2012).
- [42] K. Rektorys, *Variational Methods in Mathematics, Science and Engineering*. Springer (2012).
- [43] G. Lupi and K. Mikula, Time dependent vector field reconstruction of moving macrophages- modeling, discretization, and results. *Proceedings of the Conference CISP-BMEI* (2025) pp. 1-8.
- [44] S.A. Park, G. Lupi, R. Ozbilgic, M.N. Chi and K. Mikula, Morpho-kinetic profiling of different modes of macrophages during wound healing in zebrafish. *Work Prog.* (2024). Manuscript in preparation. <https://doi.org/10.48550/arXiv.2503.10552>.
- [45] G. Lupi et al. Datasets for “Mathematical and numerical methods for understanding immune cell motion during wound healing”. Available on Zenodo <https://doi.org/10.5281/zenodo.17833359>

Please help to maintain this journal in open access!



This journal is currently published in open access under the Subscribe to Open model (S2O). We are thankful to our subscribers and supporters for making it possible to publish this journal in open access in the current year, free of charge for authors and readers.

Check with your library that it subscribes to the journal, or consider making a personal donation to the S2O programme by contacting subscribers@edpsciences.org.

More information, including a list of supporters and financial transparency reports, is available at <https://edpsciences.org/en/subscribe-to-open-s2o>.