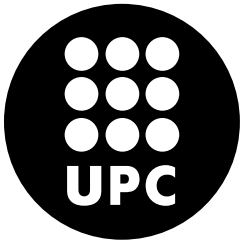
APPROXIMATION OF PHASE-FIELD MODELS WITH MESHFREE METHODS: EXPLORING BIOMEMBRANE DYNAMICS

Christian Peco Regales



Doctoral Thesis
Advisor: Marino Arroyo
Barcelona, September 2014

Departament de Matemàtica Aplicada III
Programa de Doctorat en Enginyeria Civil

To my parents

A good hockey player plays where the puck is. A great hockey player plays where the puck is going to be.

Wayne Gretzky

Abstract

Approximation of phase-field models with meshfree methods: exploring biomembrane dynamics

Christian Peco Regales

Biomembranes are the fundamental separation structure in animal cells, and are also used in engineered bioinspired systems. Their simulation is challenging, particularly when large shape changes and dynamics are involved, or micrometer systems are considered, ruling out atomistic or coarse-grained molecular modeling. The main goal of this thesis is to develop a computational framework to understand the dynamics of biomembranes embedded in a viscous fluid using phase-field models. Phase-field models introduce a scalar continuous field to define a diffuse moving interface, whose physics is encoded in partial differential equations governing it. These models can deal with dramatic shape and topological transformations and are amenable to multiphysics coupling. However, they present significant numerical challenges, such as the high-order character of the equations, the resolution of sharp and moving fronts, or the efficient time-integration. We address all these issues through a combination of meshfree spacial discretization using local maximum-entropy basis functions, and a Lagrangian variational formulation of the coupled elasticity-hydrodynamics. The smooth meshfree approach provides accurate approximations of the phase-field and can easily deal with local adaptivity, the Lagrangian approach naturally extend adaptivity to dynamics, and the variational formulation enables nonlinearly-stable robust variational time integration. The numerical implementation of these methods in a high-performance computing framework has motivated the development of a new computer code, which integrates state-of-the-art parallel libraries and incorporates

important technical contributions to overcome bottlenecks that arise in meshfree methods for large-scale problems. The resulting code is flexible and has been applied to other scientific problems in a number of collaborations dealing with flexoelectricity, metal forming, creeping flows, or fracture in materials with strongly anisotropic surface energy.

Acknowledgments

I would like to express my special appreciation and thanks to my advisor, Prof. Marino Arroyo. First, for his guidance, support, enthusiasm and fearless attitude towards research and work. I have deeply enjoyed the vast majority of the projects we have embraced and also felt his optimism, help and patience at the tough and uncertain moments that inevitably appear to give value and meaning to any significant task. Second, I would like to thank him for his priceless example not only within the professional field but also within the human side. His balanced and flexible way of dealing with delicate issues regarding the group and the individual has undoubtedly improved my perception of leadership, managing and education. I am truly indebted and thankful to the faculty of LaCàN and Departament de Matemàtica Aplicada III. In particular, Professors Antonio Huerta, Antonio Rodríguez, Sonia Fernández and Jose Muñoz. I met them at the Civil Engineering degree and all of them are responsible for the seeds that made me love and focus my attention in this new way of doing science that is Computational Mechanics. Without their motivation and support, this journey would not have been possible. I would also like to thank the reviewers of the thesis and the members of the committee for their useful comments and advice. I am very grateful to the people at LaCàN for providing such an enjoyable and stimulating working environment. A very special thanks to my colleagues and friends at Omega lab. I'll never forget the guidance, kindness and sense of humor of Adrián Rosolen, the neverending but wise and valuable advices of Daniel Millán, the deep conversations with Susanta Ghosh about the "true flavour of research", the rigorousness of Mohammad Rahimi inside the lab and his happy view of the life outside, that wonderful trip to Porto and the "Old Friends" bottle I shared with Amir Abdollahi, the passion for cleanness, order and beer of Behrooz Hashemian, the enjoyable conversations about Catalonia independence with Alejandro Torres and everything that

he uses to bring to the lab when he goes home, the example and kindness of Juan Vanegas, the wonderful coffee and beach time with Aditya, the James Bond smile of Bin Li when he does not understand something, the basketball matches of Kuan and last but not least, that crazy man known as Dimitri Kaurin. My special thanks also to David Modesto, Ana Tamayo and Cristina Diaz-Cereceda, among other colleagues of the "far, far side of the Campus". We spent perhaps too little but truly enjoyable and genuine time together. I'm going to miss my lunch time with my dear friend Gonzalo and our hot chocolates in the break of the french course.

I have so much to be grateful for my friends, specially Xavi and David, and my dear loved ones, my parents, sisters, little brother and brother in law ("Aut viam inveniam aut faciam!"), who have been a source of unconditional inspiration and encouragement. During this time I had the privilege of walking through heaven and hell, and it was in the darkness where I felt you brighter and closer. Life is like a collective sport, no real success is achieved just by oneself. Thank you for being my team now and always, everyone of you is a Ph.D. to me. Following with the sport, let me bewilder yet and again of how much happiness and inspiration could a pair of blades bring into my existence. Ice and hockey are already part of me, and so my dear friends Avi (gracias por encontrarme, profeta del hockey!), Judit and Cristina (hockey girls!), Ramón, Max, Artur, Octavi, Oriol, Ruth (la campeona), Sebastian (the doctor), Sebastián (el maestro), Anna, Xavi, Carles and Cristina (el trío maravilla del staff), Marc (I did it my waaaayyy!!!!!), Petit (locura sobre la pista) and Jarek (crack!).

And finally I thank to God. Or chaos, or chance or luck or fate or whatever word that is able to describe an astonishing chain of circumstances that ends up with you finding a treasure. In my case this treasure has a name and it is MaríaPaz. Thank you for such many moments that worth a lifetime. "Le vent se lève... Il faut tenter de vivre!".

Contents

Chapter 1

Introduction and overview

Phase-field modeling is a powerful methodology that introduces a scalar continuous field to describe and connect the different phases of a system. This field, usually called order parameter, takes constant values in the bulk and gradually changes between phases naturally identifying an interface. In contrast with classical sharp-interface methods (see Fig. 1.1), this interface is smeared over a diffuse region, whose thickness may model a physical phenomenon or result from mathematical regularization.

Phase-field models enable seamless calculations over the bulk and through the interface in a continuous way and thus presents several advantages within a range of applications that is vast and keeps growing quickly. It has become a corner stone in material sciences [1, 2], gaining popularity in a wide set of applications

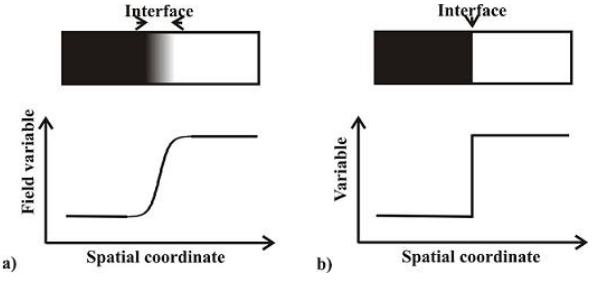


Figure 1.1: Diffuse and sharp-interface approaches. (nele.studentenweb.org)

in applied science and engineering such as fracture [3], microstructure formation and fracture evolution in ferroelectric materials [4], image segmentation [5], multi-phase flows [6], infiltration in porous medium [7], shape memory alloys [8] or tumor angiogenesis [9](see Fig. 1.2). The simulation of biomembrane systems has been a relatively recent addition in phase-field modeling [10, 11].

First steps of phase-field modeling date back to Van der Waals [12]. In his efforts to understand the density change between a liquid and its vapor, he inferred from a thermodynamical point of view that the gas-liquid density interface was more consistent if described as a diffuse transition rather than as a sharp one. These considerations gave birth to the idea of phase-field or diffuse interface modeling, which expanded quickly throughout the scientific community, but that was mainly formalized and developed over the last 50 years. The study of phase transitions of Ginzburg and Landau [13] in 1963 introduced the fundamental idea of the *order parameter*, which was interpreted as an independent state variable of the thermodynamical system. Hillert [14] applied similar concepts to build the first model for spinodal decomposition using a discrete phase-field, while Cahn and Hilliard [15] analyzed the same problem by using a continuous phase-field. Ginzburg-Landau and Cahn-Hilliard made critical contributions to the model, in particular adding to the mean field free energy (related to the bulk free energy) the contribution of phases and interfaces and thus giving rise to a consistent free energy functional and a formal structure to the modern understanding of phase-field modeling. Nevertheless, until that moment, the concept of a diffuse interface had been built on the purpose of approaching the physical reality and its properties (i.e. thickness and other related microscopic parameters) were understood as real and deducible from the energy potentials governing the system.

A second view of phase-field models came to light in 1987 when Langer [16] proposed a more phenomenological view of the interface. In the new framework, the

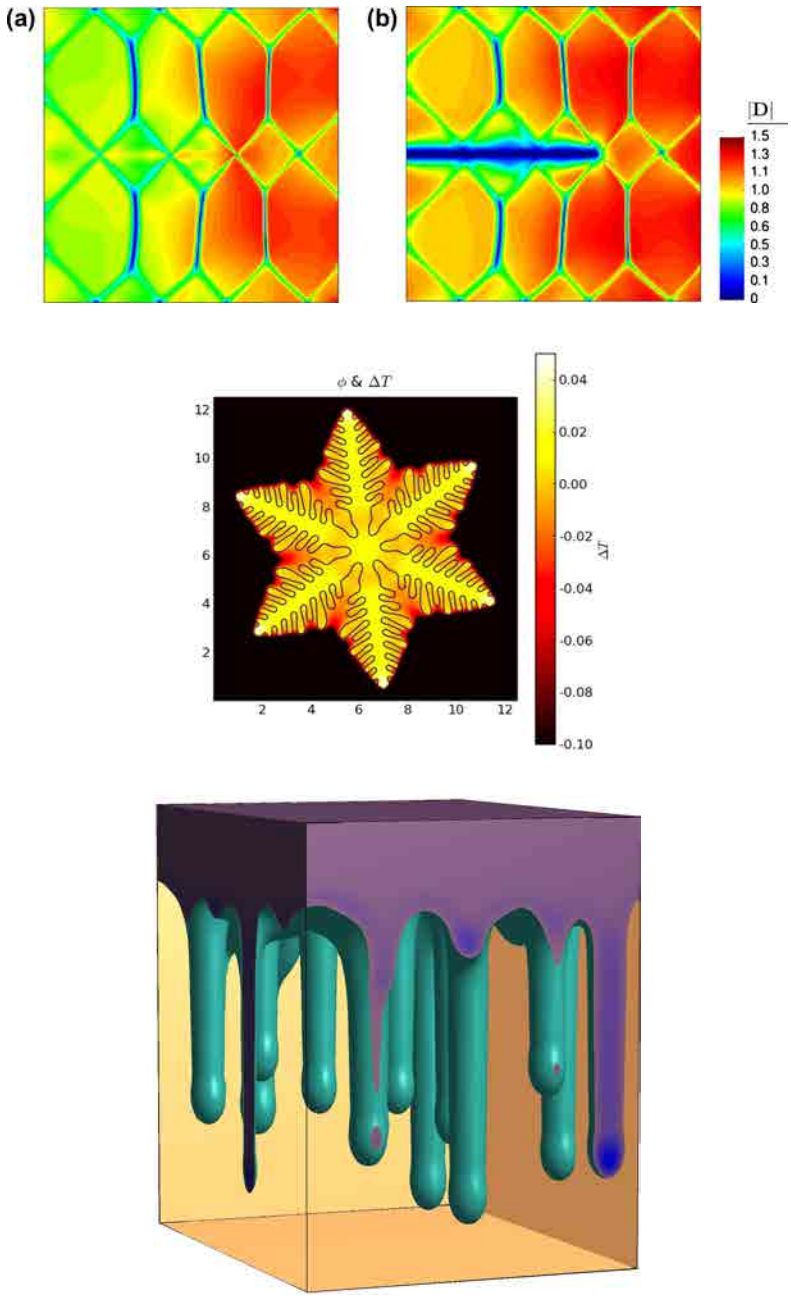


Figure 1.2: Phase-field applications. Top: evolution of fracture in ferroelectric single crystal [4]. Center: phase change. The non-uniform temperature results from the release of latent heat at the solidifying interface. The dendrite arms grow fastest where the temperature gradient is steepest (ctcms.nist.gov). Bottom: tridimensional phase-field simulation for gravity-driven infiltration in a porous media [7].

real diffusiveness of the interface as well as the related parameters are considered to be beyond the resolved scale and remain hidden at the microscopic level. Therefore, the interface thickness becomes a mathematical artifact that mimics the original sharp-interface model while keeping the phase-field formalism. This viewpoint also provides a natural route to phase-field model development as a regularization of the sharp-interface counterpart.

The modeling used in this thesis to simulate biomembrane systems follows this numerical concept of phase-field i.e. the ratio between the characteristic sizes of the vesicle radius and thickness is high enough to neglect the physical variations along the interface. The simulation of these systems with sharp-interface models poses several difficulties for classical parametrical approaches. From a numerical point of view, parametric sharp-interface approaches suffer when complex geometries and topology changes appear, e.g. merging and pinch-off phenomena, which can be extremely difficult to parameterize. They also run into the necessity of tracking the interface position at every time step. Many of these problems can be overcome with phase-field modeling (Table 1.1). Since the interface arises as a change between the phase-field values, it removes the surface tracking as well as other issues associated with topological and geometrical difficulties. Of course, phase-field models have their own drawbacks to be considered. The refinement of the interface, which has to be resolved, introduces numerical difficulties e.g. the gradually increasing sharp gradients located on the interface and the resulting stiffness of the system, both in time and space (we refer to Section 2.1). As a consequence, phase-field models tend to be computationally expensive.

Some geometrical and topological problems can be tackled with advanced techniques such as subdivision surfaces [17] and level set methods [18]. Level set methods provide a powerful technique to describe dynamical and complex sharp geometries using implicit functions. However, modeling interfacial physics and imposing com-

Table 1.1: Sharp-interface (parametrical) and phase-field modeling comparison.

Aspect/Method	Parametrical	Phase-Field
Physical field	vectorial	scalar
Domain	line/surface	2D/3D
Interface	explicit tracking	no tracking
Error sources	discretization	discretization
	interface tracking	model
Numerical challenges	mesh entanglement	local sharp-gradients
	topological changes	adaptivity

plex jump conditions may become difficult. Phase-field models, in contrast, eliminate these interface conditions and replace them by the order parameter field and a partial differential equation over the full domain which connects the interface with the rest of the physical system. Moreover, the connection of the phase-field with the physics is commonly modeled through a free energy that drives the kinetics of the system, which makes straight-forward to gradually improve the model by adding contributing energy terms.

We choose to discretize our Galerkin schemes with the local maximum entropy (LME) approximants, a meshfree method. These smooth approximants can deal with the second order derivatives present in many phase-field functionals and handle local refinement in a robust manner. LME present as well a number of features that make them suitable for phase-field models, such as their strict non-negativity, the straightforward imposition of boundary data (they present a weak Kronecker-delta property on the boundary) and the robustness of their evaluation. The variation diminishing property is particularly well-suited for phase-field models exhibiting step-like changes across the interphase.

The main objective of this thesis is to study biomembrane dynamics using phase-field models discretised with LME approximates. The simulation of biomembranes (both in space and time) is too expensive for atomistic and coarsed-grained methods (see Section 3.1), and continuum models appear as an alternative to study their be-

havior while keeping the computational cost below reasonable limits. In this thesis we propose methods that combine different numerical techniques to obtain a robust, scalable and computationally efficient code. We use a variational approach that accounts for a fourth-order phase-field model describing the vesicle and a dissipation term to consider the surrounding viscous fluid media. We minimize the action with respect to the phase field variable in the statics case to get equilibrium shape solutions. In the study of the dynamics we propose a Lagrangian particle method where we minimize with respect to the deformation mapping of the medium including both the interface and the background. Since the phase-field is convected by the motion, the refined regions of the mesh follow the interphase automatically. We extend the phase-field model to account for adhesion and investigate how confinement and kinetics could play a fundamental role in biological membrane shaping, motivated by recent experiments.

From a computational science perspective, we developed a $C++$ code that uses MPI for parallelization and some of the state-of-the-art libraries in scientific supercomputing i.e. PETSc, ParMetis, QHULL, TetGen. We found that meshfree routines in these packages are not as developed in the field as they are in mesh based techniques. In consequence, the implementation in supercomputing facilities has motivated a number of contributions to the implementation of meshfree methods. In particular, we present an optimization that coarse-grains the connectivity speeding-up critical algorithms in the system matrix assembly and also a compressed basis functions storage strategy to overcome memory bottlenecks. The structure of the resulting library is presented. We briefly report on collaborations in other problems beyond biomembranes, where the code has been successfully applied due to the flexible and user-friendly structure of its design.

This thesis is organized as follows. In Chapter 2, we justify the complexity arising from phase-field models in combination with meshfree methods and the need for

supercomputing. We also give an introduction to meshfree methods in general and to the LME approximants in particular. We devote Chapter 3 to the phase-field modeling of biomembranes. We motivate the problem, develop our contributions to the phase-field modeling of vesicles and briefly introduce our main results in statics, dynamics and ongoing research. Chapter 4 is a more technical document where we present a comprehensive description of the $C++$ code and point out the main contributions to the implementation of meshfree methods. In Chapter 5 we present different applications to which we have applied LME and the codes developed in this work. We finish with some concluding remarks, future research lines and an overview of the publication record in Chapter 6. The main papers have been referred in the text and can be consulted in Appendix A, Appendix B and Appendix C. Relevant parts of our code are detailed in Appendix D.

Chapter 2

Approximation of meshfree phase-field models

2.1 Model complexity

The main advantage of the phase-field model is the unified treatment of the interfacial tracking and mechanics, which potentially leads to simple, robust, scalable computer codes. Additionally, a meshfree method is well-suited to approximate high-order phase-field models, because the free energy functionals involve high order derivatives. This combination comes at the expense of a high computational cost, particularly if the phase-field modeling error with respect to the sharp-interface limit needs to be small. Furthermore, a meshfree method demands for a higher cost in terms of basis functions calculation, and suffers from the lack of a mesh-supported connectivity and heavier sparse matrices and assembly related routines.

In particular, biomembrane phase-field models resort to a 3D scalar PDE to describe an interphase, which would require a 2D vectorial description in a parametric representation i.e. while a vesicle surface could be parameterized and resolved in two dimensions, the phase-field model implies the use of a third dimension to describe the volume containing the interface. Moreover, the phase-field model accuracy de-

pends on the diffuse interface resolution, which is controlled by a width parameter ϵ. In biomembrane models, ϵ dictates the thickness of the membrane, which should be chosen as small as possible if the target is recovering the sharp-interface limit. Resolving a smaller ϵ implies an even smaller nodal spacing on the discretization grid. In our experience, the nodal spacing h should at least satisfy $\epsilon \geq 2h$. As a consequence, using uniform grids implies a spatial complexity of order $\mathcal{O}(\epsilon^{-d})$ in a phase-field model describing an interface of dimension $d-1$ embedded in a space of dimension d. This justifies the necessity for adaptive strategies, since the order of complexity can be then ideally lowered one dimension and match the interface dimension i.e. order $\mathcal{O}(\epsilon^{1-d})$.

Phase-field models introduce a field, which concentrates around a thin layer, introducing numerical stiffness into the system. This stiffness can be broadly explained as a property of the system by which large variations in the output arise from small changes in the input, hence limiting the time step in any explicit time integration. Furthermore, this type of stiffness arising from different scales is commonly reflected by a high condition number in the matrices, thus hampering iterative solvers, which are in general preferable to treat large sparse matrices. Indeed, it can be proved that the biomembrane phase-field model produces solutions with the profile $\phi(\boldsymbol{x}) = \tanh\left[\frac{dist(\boldsymbol{x})}{\sqrt{2}\epsilon}\right]$, where $dist(\boldsymbol{x})$ is the distance to the interface. Resolving this profile requires a very fine discretization for small values of ϵ, but this high resolution is only required in the vicinity of the interface. Away from it, the phase-field is nearly constant. These issues of phase-field models affect dramatically the performance and motivate two fundamental choices in the proposed strategy for solving the problems in this thesis: strong adaptivity and variational time integrators.

The other large source of complexity lies within the meshfree nature of the approximants used in the discretization. Their choice is motivated by a number of properties that make them desirable to solve these kind of problems, but comes with

a more complex structure that can damage the final performance. Approximation of high-order phase-field models has been previously tackled with other discretization techniques. Isogeometric analysis [19], a non-negative technology showing precise geometrical descriptions and the required smoothness to handle high-order operators, has been successful in approximating phase-field models for a variety of problems [20, 21]. However, the straight-forward h-refinement and robustness of meshfree methods in Lagrangian grids under large distortions has motivated the choice of LME approximants to discretize the biomembrane phase-field models presented in this thesis. Nevertheless, the non-negative character of these two technologies make them suitable to be combined to get advantage of their combined strengths [22]. Discretization of PDE's in a meshfree sparse framework involves a variety of routines including (i) neighbor search over domain, (ii) computation of the basis functions, (iii) creation of an sparse matrix structure and (iv) filling of matrix positions with a particular operation. As discussed in Section 4.1, the last two routines gain relevance as the size of the system increases, and are comparable to the final solver step, particularly in 3D. The not known *a priori* structure of the sparse matrices, the higher density of the connectivity matrix and the large number of integration points required, introduce a non-negligeable overhead in comparison with standard mesh-based methods e.g. FEM.

Thus, while mesh free methods provide smooth and flexible approximation for high-order PDE, these technologies can be computationally demanding, particularly in conjunction with phase-field models. For this reason, part of this thesis is devoted to their implementation in a HPC framework.

2.2 Meshfree methods and the Local Maximum Entropy Approximants

Meshfree methods provide an approximation to continuum field equations based on basis functions that do not rely on a mesh or its connectivity (see [23] for an introductory review). Therefore, many of the requirements associated with the quality of the elements in a traditional FEM are relaxed or disappear. This extra flexibility on the grid of nodes raises new challenges in the numerical implementation [24]. The most popular meshfree approximants are based on the moving least squares idea [25]. One of the first meshfree methods, the smoothed particle hydrodynamics [26], was originally designed in 1977 to solve astrophysical problems and applied later to fluid dynamics. From there on, a variety of methods have emerged, such as reproducing kernel particle method [27], partition of unity finite element method [28] and element free Galerkin [29] to mention a few. FEM strategies have succeed in countless computational mechanical and physical applications, but they have also run into obstacles when facing problems involving discontinuities, sharp fronts, large distortions and high-order derivatives. Meshfree methods offer several advantages such as shape functions with high-order continuity, robustness in dramatic grid deformations [30, 31] and easier local adaptivity [32, 33]. Most of meshfree Galerkin methods actually require a quadrature mesh in order to perform integration, and a higher number of quadrature points to accurately integrate the weak form due to their non-polynomial nature and non element-wise support. Other issues include the awkward treatment of essential boundary conditions due to non-satisfaction of the Kronecker delta property, the computational cost associated to neighborhood creation to determine the basis functions support and the computation of the basis functions themselves.

In recent years, the information theoretic concept of maximum-entropy has been

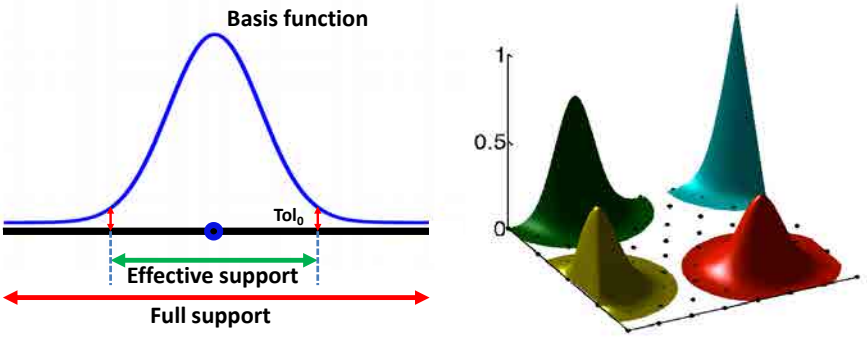


Figure 2.1: Left: Full support of local maximum entropy (LME) basis functions covers the convex hull of the computational domain. The effective numerical support size is given by the truncation error $\mathtt{T}ol_0$. Right: Representation of two-dimensional LME approximants. Notice the non-interpolant character and the smoothness of the basis functions, and the fulfillment of a weak Kronecker delta at the boundary of the convex hull.

put forth to develop polygonal approximants [34] and meshfree approximation schemes [35]. Local maximum-entropy approximants are non-negative and are endowed with features that are proper to convex approximants, such as monotonicity, smoothness (C^∞, and therefore handle without difficulties high-order derivatives) and variation diminishing property [35]. They satisfy *ab initio* a weak Kronecker-delta property at the boundary of the convex hull of the nodes [35] and therefore the imposition of essential boundary conditions in Galerkin methods is straightforward. Their convex geometry structure [35] enables the connection with other non-negative technologies like isogeometric analysis [19] or subdivision surface [36]. Other advantages include the robustness of their evaluation and a simpler quadrature [37]. Some of these concepts are illustrated in Fig. 2.1.

Traditional numerical methodologies like finite difference [10, 38] and spectral methods [39] have been used for phase-field models of biomembranes. Recently, isogeometric analysis [19], a Galerkin method based on tensor products of 1D NURBS

approximants, has shown an excellent performance for the Cahn-Hilliard equation, handling successfully the sharp transitions of the solutions without spurious overshoots [20, 40]. Although these structured methods can handle higher-order operators, they have difficulties in adapting to localized features. C^0 finite element approaches can deal with the high-order character of the functional by reformulating the model as a system of second order PDE [41] and are well suited for adaptivity [42], but suffer from poor accuracy for a given computational cost. A number of adaptive techniques have been developed for the Cahn-Hilliard model, including an adaptive multigrid finite-difference method [43, 44], a Fourier spectral moving-mesh method [45], an adaptive FEM with linear [46, 47, 48] and quadratic [49] shape functions after recasting the higher-order phase-field as a system of lower-order equations, and a finite volume approach for unstructured grids [50]. Adaptive methods based on finite differences [51, 52], Fourier spectral [53], or finite volumes [54, 55] have been proposed for other higher-order phase-field equations.

Maximum-entropy basis functions, denoted by $p_a(\boldsymbol{x}), a = 1, \ldots, N$ with $\boldsymbol{x} \in \mathbb{R}^d$, where d is the space dimension, are designed to be strictly non-negative and to fulfill the zeroth and first order consistency conditions

$$p_a(\boldsymbol{x}) \geq 0, \qquad \sum_{a=1}^{N} p_a(\boldsymbol{x}) = 1, \qquad \sum_{a=1}^{N} p_a(\boldsymbol{x})\ \boldsymbol{x}_a = \boldsymbol{x}, \tag{2.1}$$

where the last equation allows us to identify the vectorial weights $\boldsymbol{x}_a$ with the positions of the nodes associated with each basis function.

The idea behind local maximum-entropy basis functions is to construct local approximants as well as optimal from an information theory viewpoint. It means that these approximants have to exhibit a (Pareto) compromise between two competing objectives, minimum width (locality) and entropy maximization (information theory optimality criteria), subject to the consistency constraints (reproducibility condi-

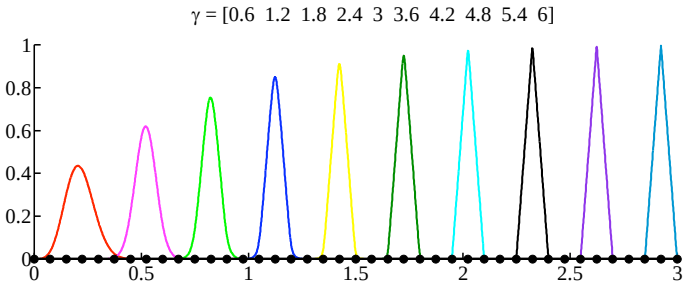


Figure 2.2: Seamless and smooth transition from meshfree to Delaunay affine basis functions. The transition is controlled by the non-dimensional nodal parameters γ_a, which here take linearly varying values from 0.6 (left) to 6 (right).

tions). These requirements enable us to write the following optimization program to select the approximants

$$\begin{aligned} \text{For fixed } \boldsymbol{x}, \text{ minimize } & \quad \sum_{a=1}^{N} \beta_a p_a |\boldsymbol{x} - \boldsymbol{x}_a|^2 + \sum_{a=1}^{N} p_a \ln p_a \\ \text{subject to } & \quad p_a \geq 0, \quad a = 1, \ldots, N \\ & \quad \sum_{a=1}^{N} p_a = 1, \qquad \sum_{a=1}^{N} p_a \boldsymbol{x}_a = \boldsymbol{x}, \end{aligned} \tag{2.2}$$

where the set of non-negative nodal parameters $\{\beta_a = \gamma_a / h_a^2\}_{a=1,\ldots,N}$ defines the locality of the approximants [35, 56]. The dimensionless parameter γ_a characterizes the degree of locality of the basis function associated to the node $\boldsymbol{x}_a$, while h_a represents the nodal spacing. The basis functions become sharper and more local as the value of the dimensionless parameter γ_a increases, and the Delaunay approximants arise as specialized limits ($\gamma_a \geq 4$ in the practice), as illustrated in Fig. 2.2 for a one-dimensional domain.

As fully detailed in [35], it can be mathematically proved that the optimization problem has a unique solution. The efficient solution follows from standard duality methods. Here, we just summarize the recipe for the final calculation of the basis

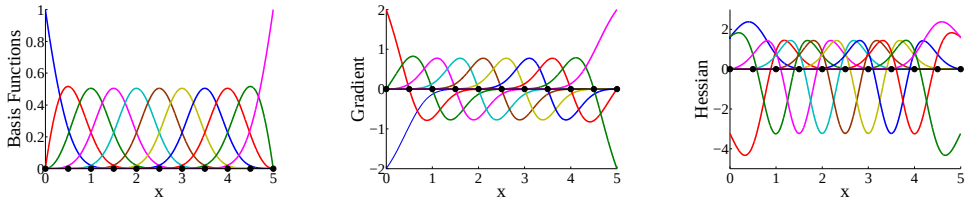


Figure 2.3: One-dimensional local maximum-entropy basis functions (left), and its first and second spatial derivatives (center-right) computed with a dimensionless parameter $\gamma = 0.8$.

functions. By analogy with statistical mechanics, we define the partition function

$$Z(\boldsymbol{x}, \boldsymbol{\lambda}) = \sum_{b=1}^{N} \exp\left[-\beta_b |\boldsymbol{x} - \boldsymbol{x}_b|^2 + \boldsymbol{\lambda} \cdot (\boldsymbol{x} - \boldsymbol{x}_b)\right]. \tag{2.3}$$

At each evaluation point $\boldsymbol{x}$, the Lagrange multiplier for the linear consistency condition is the unique solution to a solvable, convex, unconstrained optimization problem

$$\boldsymbol{\lambda}^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{\lambda} \in \mathbb{R}^d} \ln Z(\boldsymbol{x}, \boldsymbol{\lambda}). \tag{2.4}$$

This optimization problem with d unknowns is efficiently solved with Newton's method. Then, the basis functions adopt the form

$$p_a(\boldsymbol{x}) = \frac{1}{Z\left(\boldsymbol{x}, \boldsymbol{\lambda}^*(\boldsymbol{x})\right)} \exp\left[-\beta_a |\boldsymbol{x} - \boldsymbol{x}_a|^2 + \boldsymbol{\lambda}^*(\boldsymbol{x}) \cdot (\boldsymbol{x} - \boldsymbol{x}_a)\right]. \tag{2.5}$$

We refer to [57, 56] for the expressions to compute the gradient $\nabla p_a(\boldsymbol{x})$ and the Hessian matrix $Hp_a(\boldsymbol{x})$ of the local maximum-entropy basis functions, which are illustrated in Fig. 2.3 for a one-dimensional domain uniformly discretized and a dimensionless parameter $\gamma = 0.8$. We refer to [58] for a more detailed description of maximum-entropy approximants and their applications.

Chapter 3

Phase-field modeling of biomembranes

In this chapter the main contributions to the simulation of biomembranes are presented. First, in Section 3.1, we give a brief introduction of biomembranes from a biological-chemical point of view. In Section 3.2 we go through the theory underlying their modeling, and devote Sections 3.3 and 3.4 to modeling and simulation strategy for statics and dynamics, respectively. Further details can be found in papers Appendix A and Appendix B. We conclude with a review of the ongoing research regarding adhesion and confinement phenomena in Section 3.5.

3.1 An introduction to biomembranes

The study of living cells is a very important issue in different fields such as biological research, medicine and biotechnology. A general feature of all cells is an interface membrane between the machinery in the interior of the cell and the extracellular fluid, called the plasma membrane. Eukaryotic cells possess complex compartments made out of internal membranes, called organelles, instrumental in their function (see Fig. 3.1), which are also responsible for the transport of substances through cargo

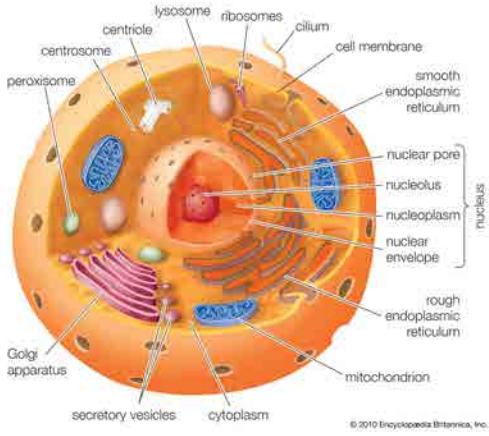


Figure 3.1: Human cell structure. Source: Encyclopedia Britannica Inc.

vesicles or tubes. They also play a key role in bio-mimetic engineered systems [59]. Their complex behavior, rich physical properties, formation and dynamics have been objects of experimental and theoretical investigation for biologists, chemists and physicists during many years [60, 61].

Cell membranes are mainly built from two mono-molecular layers of lipids (called lipid bilayers) held together by entirely non-covalent forces. They are around 4 nm in the thickness and from tens of nanometers up to millimeters in the lateral directions. Lipids in plasma membranes are chiefly phospholipids e.g. phosphatidyl ethanolamine. Phospholipids are amphiphilics with hydrocarbon tails and hydrophilic polar heads [59, 62, 63]. Therefore, they assemble naturally forming a hydrostable bilayer, which presents interesting mechanical properties (see Fig. 3.2).

Experimental observations as well as molecular simulations have revealed the in-plane fluidity of lipid membranes [64, 65] while they behave as flexible solids which store bending and extensional elastic energies. Upon mechanical loads, lipid bilayers may be curved, compressed, dilated, or sheared. In the equilibrium state and within the linear limit, the membrane response to these deformation modes is characterized

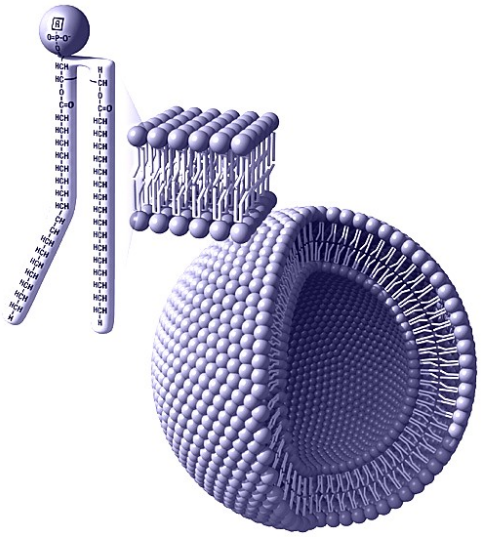


Figure 3.2: Phospholipid bilayer structure and self assembly. (Source: astrobiology.nasa.gov)

by the following constants: the bending rigidity, the Gaussian curvature modulus, the area compressibility modulus, and the shear elastic modulus. At physiological temperatures, most lipid membranes are fluid and therefore lipid membranes have no strength against shear forces i.e. shear elastic modulus is zero. Below the phase transition temperature of lipids, lipid membranes form a solid-like gel phase. Mechanically, the bilayer acquires a non-zero shear elasticity. In this so-called gel phase, the relative motion of membrane inclusions is hindered. The fluidity of membranes at higher temperatures is essential in the case of cellular membranes because it permits the displacement of membrane anchored macromolecules or inclusions, e.g. transmembrane proteins, and provides the necessary malleability for the membrane to form the intracellular organelles or mediate in tubular and vesicular transport of proteins (see Fig. 3.3).

The membrane itself, on the one hand, and the surrounding fluid, on the other, impose a hydrodynamic drag on the motion of an inclusion or in a membrane shape

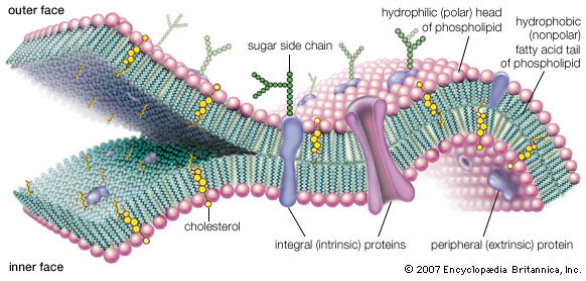


Figure 3.3: Phospholipid bilayer membrane composition. Source: Encyclopedia Britannica Inc.

change, as this involves a rearrangement (and shear) of the lipids and the surrounding fluid. The resistance or shear in the plane of the film is characterized by the interfacial shear viscosity of the membrane. One may equivalently define a viscosity related to the dilation and compression of the membrane. For a complete and more realistic description of the membranes, one has to consider the existence of two monolayers which may slip with respect to each other. The intermonolayer slippage is dragged by a friction force whose amplitude is proportional to the intermonolayer friction coefficient.

3.2 Vesicle modeling

Vesicles are closed biomembranes, which play an important role in biophysical processes such as the delivery of proteins, antibodies or drugs into cells, and separation of different types of biological macromolecules within cells. Vesicles serve as simplified models of more complex biological systems, and can be used to study the interaction between lipid bilayers and the surrounding medium, e.g. under osmotic stress [66], shear flow [67], or electrical fields [68]. Depending on the lipid composition, lipid bilayers can phase-separate forming multicomponent vesicles [69], which have also been the object of numerous studies as model systems for rafts.

Atomistic molecular dynamics (MD) simulations have been very useful in the prediction of the macroscopic characteristics of lipid membranes, but remain limited to small membrane patches due to the large number of atoms involved in closed vesicles, and more importantly due to the slow relaxation times of bending modes [70]. As a consequence, the computational cost scales as L^6 where L denotes the lateral dimension of the system. Coarse-grained simulation, where each particle represents a number of atoms, can reach larger systems, and there has been notable successful studies in recent years involving out-of-equilibrium phenomena at the scale of small vesicles, e.g. [71]. Yet, as in atomistic MD, the scaling of the computational cost poses a hard upper bound on the system sizes that can be reached with current computers. Continuum mechanics has been shown to be very efficient in explaining the statics of lipid bilayers, as well as their dynamics, particularly for large scale systems [72, 73, 74, 75, 76].

From a mechanical point of view, the fluid vesicle dynamics result from a balance of elastic forces and two main dissipative mechanisms : the bulk dissipation due to the drag force of the surrounding fluid and the internal membrane dissipation of lipid bilayer; likewise, this internal dissipation can be considered to arise from two main phenomena : the in-plane membrane viscosity and the intermonolayer slippage. We consider that the elastic behavior of the membrane is dominated by the bending energy, leaving the extensional energy as a constraint due to its nearly inextensible behaviour under common forces. Regarding the media, we note that the drag force of the ambient fluid is dominant at large scales, while at small scales it is negligible as compared to the surface viscosity. Here we focus on vesicle systems at large scales (few microns and above), where the mechanics are given by the interplay of bending elasticity and bulk viscosity (Fig. 3.4).

Therefore, the main effort is put on the continuum description of the coupled physics of the vesicle embedded in the fluid media, and the numerical approach

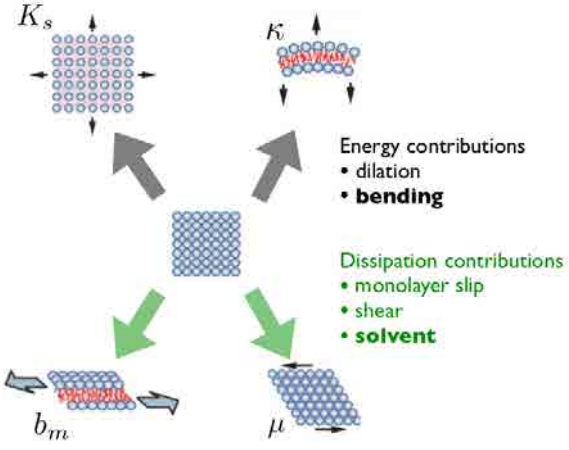


Figure 3.4: Elastic energies and dissipation mechanisms; the model considers the preponderance of bending and solvent in large scale dynamics

to tackle the problem. Nevertheless, other types of dissipation, such as the ones aforementioned, or the influence of chemical phenomena, such as the protein concentrations inducing additional curvature of the membrane, could be added to the model eventually.

In this study, we are looking for numerically tractable approach for the simulation of complex biological processes involving membranes embedded in a viscous fluid media, enabling us to examine fundamental questions about cell and organelle physiology. We approach the problem with an innovative continuum description based on a phase-field model, a meshfree approximation and a Lagrangian framework, resulting in a variational method that presents automatic adaptivity.

The development of the phase-field model for vesicles starts with the set up of an order parameter in the spirit of the Cahn-Hilliard [15] approach. In this framework, we define an order parameter $\phi(x)$ that takes values $+1$ and -1 to signal the exterior and the interior enclosed volumes of a membrane, respectively. We then find a connection between the field $\phi(x)$ and the relevant geometrical quantities involved

in the energy functionals governing the behavior of the system. We follow here the phase-field model for biomembranes proposed by [10] and developed in [**?**], which start approximating the area and enclosed volume of the vesicle. The volume description is almost trivial and serves as a primary example of this methodology. It is clear that the functional,

$$E_V = \int_\Omega \phi \, d\Omega, \tag{3.1}$$

approximates the difference between the exterior and the interior volumes, and the functional

$$E_A = \int_\Omega \left(\frac{\epsilon}{2} |\nabla \phi|^2 + \frac{1}{4\epsilon} (\phi^2 - 1)^2 \right) d\Omega, \tag{3.2}$$

is proportional to the surface area as $\epsilon \to 0$.

For fluid membranes in our scale range, the main term in the energy functional was introduced by Canham [77], Evans [78] and Helfrich [79] , which accounts for the bending elastic energy in a sharp-interface model,

$$E_H = \int_\Gamma \left(\frac{k_H}{2} (H - C_0)^2 + k_g K + \sigma \right) \, d\Gamma, \tag{3.3}$$

where Γ is the surface of the vesicle, H stands for the mean curvature, K for the Gaussian curvature, σ represents the surface tension, C_0 is the spontaneous curvature (may be modeled by area difference elasticity [80]) and k_H, k_g are the bending and Gaussian rigidities, respectively.

In our model, we let aside the Gaussian curvature term, whose integral remains constant for a uniform vesicle in the absence of topological changes by virtue of the Gauss-Bonnet theorem. The surface tension term is also omitted later, because it can be incorporated e.g. in the Lagrange multiplier enforcing constant area. Considering

for simplicity $C_0 = 0$, the mean curvature remains as the only contribution to the bending energy.

Under this assumptions, the Helfrich model admits a phase-field formulation [10, 11, 81], where the curvature energy and the associated constraints (area and volume) of the vesicle can be written as,

$$\begin{aligned} E(\phi) &= f_E \frac{k}{2\epsilon} \int_\Omega \left[\epsilon \Delta\phi + \left(\frac{1}{\epsilon}\phi + C_0\sqrt{2} \right) \left(1 - \phi^2\right) \right]^2 \, d\Omega \\ V(\phi) &= \frac{1}{2} \left(Vol(\Omega) + \int_\Omega \phi \, d\Omega \right) = V_0 \\ A(\phi) &= f_A \int_\Omega \left[\frac{\epsilon}{2} |\nabla\phi|^2 + \frac{1}{4\epsilon}(\phi^2 - 1)^2 \right] \, d\Omega = A_0 \end{aligned} \tag{3.4}$$

where ϵ is a small regularization parameter, $f_E = \frac{3}{8\sqrt{2}}$, $f_A = \frac{3}{2\sqrt{2}}$, Ω is the domain, and $\partial\Omega$ its boundary. The regions $\{\boldsymbol{x} : \phi(\boldsymbol{x}) > 0\}$ and $\{\boldsymbol{x} : \phi(\boldsymbol{x}) < 0\}$ represent the inside and outside of the membrane, while the level set $\{\boldsymbol{x} : \phi(\boldsymbol{x}) = 0\}$ can be used to realize the position of the membrane. Formal asymptotics [11], as well as rigorous mathematical analysis [82] (see also [83] for a review), provide the connection between the phase-field and the sharp-interface models when $\epsilon \to 0$. As this limit is never achieved in the numerical calculations, a modeling error is always present in practice. This model has been coupled with the Navier-Stokes equations in [39]. Similar ideas to couple phase-field models of biomembranes with fluid or other physical fields have been developed by other researchers as well [68, 84, 38, 46].

Vesicles are always surrounded by a solvent. In most situations of interest, vesicles evolve in low Reynolds number conditions. The assumption of creeping dynamics lets us pose the problem from an energetic standpoint and use the dissipation rate from the energy balance equation to state a variational principle. From the Rayleigh dissipation potential it is possible to derive the classical form of momentum balance

equations for creeping dynamics.

The Rayleigh dissipation potential for a compressible Newtonian fluid can be written as

$$Diss[\partial_t \boldsymbol{y}; \boldsymbol{y}] = \mu \int_\Omega \mathbf{d}' : \mathbf{d}' \, d\Omega + \frac{\lambda}{2} \int_\Omega (\text{div } \mathbf{v})^2 \, d\Omega, \tag{3.5}$$

$$\mathbf{d}' = \mathbf{d} - \bar{\mathbf{d}} = \frac{1}{2}(\nabla \mathbf{v} + \nabla \mathbf{v}^T) - \frac{1}{3}\text{div } \mathbf{v}\boldsymbol{I}. \tag{3.6}$$

For an incompressible Newtonian fluid, the second term above is replaced by the constraint

$$\text{div } \mathbf{v} = 0. \tag{3.7}$$

This expressions allow us to consider the action of the bulk fluid acting on the vesicle and drive accordingly the dynamics of the system. Considering a slightly compressible fluid simplifies the mathematical formulation and the numerical implementation. Nevertheless, a numerical treatment of the fully incompressible case using LME mesh free approximants is described in Chapter 5.

3.3 Vesicle statics: equilibrium shapes

We briefly introduce here the basics for biomembrane statics, which is fully developed in our paper in Appendix A. In the static approach we aim to minimize the elastic potential of the vesicle to get equilibrium shapes. These stable configurations have been widely studied in the biophysical literature, which provides a valuable source for comparison and validation. At this stage, the objective is to demonstrate that the numerical scheme proposed to solve the phase-field formulation is able to handle the different numerical challenges (sharp features, adaptivity, large deformations, topology changes) imposed by the phase-field model. We also note the relevance of using

adaptive grids with phase-field models by comparing different levels of refinement with regular uniform grids. We rely for refinement on Centroidal Voronoi Tessellations [85] to distribute appropriately the nodes based on the phase-field gradient.

In this framework, we simply minimize the elastic bending energy with respect to the phase-field over a domain Ω containing the vesicle, subject to volume and area constraints. This minimization leads to different local minima standing for the different stable shapes. Note that an additional constraint is added in statics, corresponding to the static moment in the vertical axis (in this work we develop the formulation in axisymmetric coordinates). This constraint is needed in statics to control the rigid solid movement in the axisymmetrical axis, and thus prevents the vesicle from escaping the simulation domain.

We consider the following expansion for the phase-field in terms of the basis functions

$$\phi(\boldsymbol{x}) \approx \phi_h(\boldsymbol{x}, \boldsymbol{\Phi}) = \sum_{a=1}^{N} p_a(\boldsymbol{x})\phi_a, \tag{3.8}$$

where $\boldsymbol{\Phi} = (\phi_1, \phi_2, ..., \phi_N)$ is an array containing the N nodal values of the phase-field, and we insert this ansatz into the variational problem describing the phase-field model to obtain the following algebraic optimization program:

$$\begin{aligned}
\text{Minimize} \quad & E_h(\boldsymbol{\Phi}) = E[\phi_h] = f_E \frac{k}{2\epsilon} \int_\Omega W_h^2 \, d\Omega \\
\text{subject to} \quad & V_h(\boldsymbol{\Phi}) = V[\phi_h] = \frac{1}{2}\left(Vol(\Omega) + \int_\Omega \phi_h \, d\Omega \right) = V_0 \\
& A_h(\boldsymbol{\Phi}) = A[\phi_h] = f_A \int_\Omega \left[\frac{\epsilon}{2} |\nabla \phi_h|^2 + \frac{1}{4\epsilon}(\phi_h^2 - 1)^2 \right] \, d\Omega = A_0 \\
& M_h(\boldsymbol{\Phi}) = M[\phi_h] = \int_\Omega \phi_h (z - z_c) \, d\Omega = 0 \\
& \phi_h|_{\partial\Omega} = -1,
\end{aligned} \tag{3.9}$$

where

$$W_h = \epsilon \Delta \phi_h + \left(\frac{1}{\epsilon} \phi_h + C_0 \sqrt{2} \right) \left(1 - \phi_h^2 \right). \tag{3.10}$$

The optimality conditions can be obtained from the Lagrangian function

$$\mathcal{L}(\boldsymbol{\Phi}, \boldsymbol{\nu}) = E_h(\boldsymbol{\Phi}) - \nu_A \left[A_h(\boldsymbol{\Phi}) - A_0 \right] - \nu_V \left[V_h(\boldsymbol{\Phi}) - V_0 \right] - \nu_M \left[M_h(\boldsymbol{\Phi}) - M_0 \right], \tag{3.11}$$

where the area, volume and static moment constraints are maintained by the Lagrange multipliers or physical reactions $\boldsymbol{\nu} = (\nu_A, \nu_V, \nu_M)$, where ν_A can be interpreted as a membrane tension and ν_V as a pressure difference across the membrane.

After defining a new set of variables $\tilde{\boldsymbol{x}} = (\boldsymbol{\Phi}, \boldsymbol{\nu}) = (\phi_1, \phi_2, ..., \phi_N, \nu_A, \nu_V, \nu_M)$, the optimal solution of this saddle-point problem can be sought with the Newton-Raphson method applied to the nonlinear set of equations given by $\partial_{\boldsymbol{\Phi}} \mathcal{L} = 0$, $\partial_{\boldsymbol{\nu}} \mathcal{L} = 0$. However, this approach may lead to mere stationary points, not minimizers of the elastic energy, physically unstable equilibria. Furthermore, given the difficulty in setting good initial guesses for the Lagrange multipliers, this solution strategy is not robust. A robust strategy that guarantees stable equilibria is based on the augmented Lagrangian method, which combines the standard Lagrangian with penalties. This method retains the exactness of the Lagrange multipliers method and the minimization principle of penalty methods. The minimization is performed iteratively on the phase-field variables for frozen Lagrange multipliers, which are updated explicitly (see [86, 87] for further details). The augmented Lagrangian is

$$\begin{aligned} \mathcal{L}_A(\boldsymbol{\Phi}, \boldsymbol{\nu}) = {} & E_h(\boldsymbol{\Phi}) - \nu_A \left[A_h(\boldsymbol{\Phi}) - A_0 \right] - \nu_V \left[V_h(\boldsymbol{\Phi}) - V_0 \right] - \nu_M \left[M_h(\boldsymbol{\Phi}) - M_0 \right] \\ & + \frac{1}{2\mu} \left| A_h(\boldsymbol{\Phi}) - A_0 \right|^2 + \frac{1}{2\mu} \left| V_h(\boldsymbol{\Phi}) - V_0 \right|^2 + \frac{1}{2\mu} \left| M_h(\boldsymbol{\Phi}) - M_0 \right|^2. \end{aligned} \tag{3.12}$$

We solve the problem in two stages. First, we follow the augmented Lagrangian method to find an approximate minimizer consistent with the constraints with a coarse tolerance. Then, this approximation is refined with the regular Newton-Raphson method on the extended set of variables $\tilde{\boldsymbol{x}}$. Since the initial guess for this second stage is very close to the actual minimizer, the algorithm never leads to unstable equilibria. The expressions to compute the gradients $\tilde{\boldsymbol{r}}(\boldsymbol{\Phi}, \boldsymbol{\nu})$ and $\tilde{\boldsymbol{r}}_A(\boldsymbol{\Phi}, \boldsymbol{\nu})$ of the Lagrangian and augmented Lagrangian are lengthy but straightforward, and can be found in Appendix A.

Numerical results recover stable equilibrium shapes that can be charted in a phase diagram that has been extensively studied (see [1, 2] and references therein). This diagram exhibits a number of equilibrium branches, including prolates, oblates, discocytes, or stomatocytes. The equilibrium shape for a given area, volume, and spontaneous curvature is not unique in general. For instance, upon deflation of an initially spherical vesicle without spontaneous curvature, the prolate-dumbbell and oblate-discocyte branches are possible. Mathematically, the transition shapes of the equilibrium branches can be tracked by changing the volume constraint and solving for constrained minimizers. A number of equilibrium shapes for the oblate equilibrium branch are plotted in Fig. 3.5.

We illustrate the accuracy of the proposed method by analyzing two specific aspects in axisymmetric examples: (i) the convergence of the phase-field model for a fixed regularization parameter ϵ using uniform grids, and (ii) the convergence of the phase-field model to the sharp-interface model when ($\epsilon \to 0$) and the points are adaptively distributed, which is essential to simulate thinner interfaces without significantly increasing the total number of degree of freedom.

To answer the first question we show in Table 3.1 the numerical energies for a discocyte equilibrium shape considering different values of ϵ and several grids of points in a computational domain $\Omega = [0, 1.5] \times [0, 2]$. The identification code (O: the

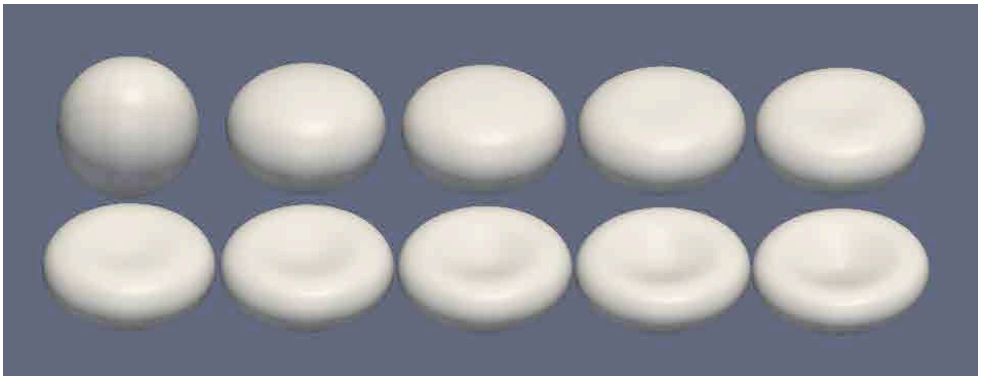


Figure 3.5: 3D views of the oblate equilibrium branch: each shape is computed by minimizing the energy and reducing by 5% the volume of the previous configuration.

Table 3.1: Energies of the discocyte equilibrium shape for different uniform grids of points and several values of ϵ. The size of the computational domain is $\Omega = [0, 1.5] \times [0, 2]$. Reference energy from a sharp interface simulation: $E_{discocyte} = 9.12657$.

ID	# nodes	$\bar{h}$	$\epsilon = 0.05$	$\epsilon = 0.04$	$\epsilon = 0.03$	$\epsilon = 0.02$	$\epsilon = 0.01$
O1	6124	0.024	9.71279	9.59056	–	–	–
O2	12271	0.017	9.72137	9.59446	9.43775	–	–
O3	24597	0.012	9.72671	9.59553	9.43483	9.29532	–
O4	49145	0.0084	9.73203	9.59786	9.43515	9.28938	–
O5	98388	0.0059	9.73536	9.59901	9.43481	9.28674	9.22082
O6	146545	0.0048	9.73716	9.59948	9.43422	9.28378	9.19139
O7	296344	0.0034	9.73989	9.60053	9.43437	9.28326	9.18627

oblate-discocyte branch) and the number of nodes for each grid are indicated in the first and the second column. As the CVT-generated grids are not perfectly uniform, the value of the average nodal spacing $\bar{h}$ is reported in the third column. The remaining columns show the energies computed for different values of the regularization parameter ϵ. We report the energies only when the transition profile is reasonably resolved, as decided by the relation $\epsilon > 2h$. Note the energy convergence from above as the number of points increases for each ϵ (columns). We can also observe how the value of the energy converges to the sharp interface value $E_{discocyte} = 9.12657$ as the parameter ϵ decreases.

Table 3.2: Energies of the discocyte equilibrium shape for several values of ϵ and uniform and adapted grids of 6,124 points. Reference energy from a sharp interface simulation: $E_{discocyte} = 9.12657$.

ID	# nodes	$\epsilon = 0.04$	$\epsilon = 0.03$	$\epsilon = 0.025$	$\epsilon = 0.02$	$\epsilon = 0.015$	$\epsilon = 0.01$
O1	6124	9.59056	–	–	–	–	–
O11	6124	9.59678	9.44002	–	–	–	–
O12	6124	–	9.43506	9.35810	9.28849	–	–
O13	6124	–	–	9.35970	9.28701	9.22588	9.18703
O7	296344	9.60053	9.43437	9.35488	9.28326	9.22399	9.18627

We address now the $\epsilon \to 0$ behavior in adapted grids. As argued in Chapter2, adaptivity is essential for numerical approaches based on phase-field models to be competitive. A possible strategy for adaptivity is to solve the optimization problem with a coarse grid of points (and thus a large value of ϵ), apply CVT to redistribute the nodes concentrating them around the interface, and compute the phase-field solution with a smaller ϵ for this new distribution of points. In practice, this strategy cannot be applied at once to get a strong refinement. Indeed, the initial coarse grid provides an inaccurate phase-field solution, which in turn produces an inadequate relocation of the points. This ultimately constraints unphysically the phase-field solutions. A better strategy is to adapt the grid and reduce ϵ progressively.

Table 3.2 reports the bending energies of the discocyte equilibrium shape for uniform and adapted grids and several regularization parameters. The first and the last rows correspond to uniform meshes with 6,124 and 296,344 nodes, and are the same as those reported in Table 3.1. The other rows correspond to adapted grids with 6,124 nodes, obtained in each step of the progressive adaption of the grid and reduction of ϵ. The first column of the table gives an identification code for the grids of points. A description of the features of each grid is given in Appendix A. The smooth transition between the successive grids is apparent in the figure, as the value of ϵ is slowly decreased in each step, while the refinement factor is increased to maintain the relative effect of the phase-field gradient. The minimum allowable

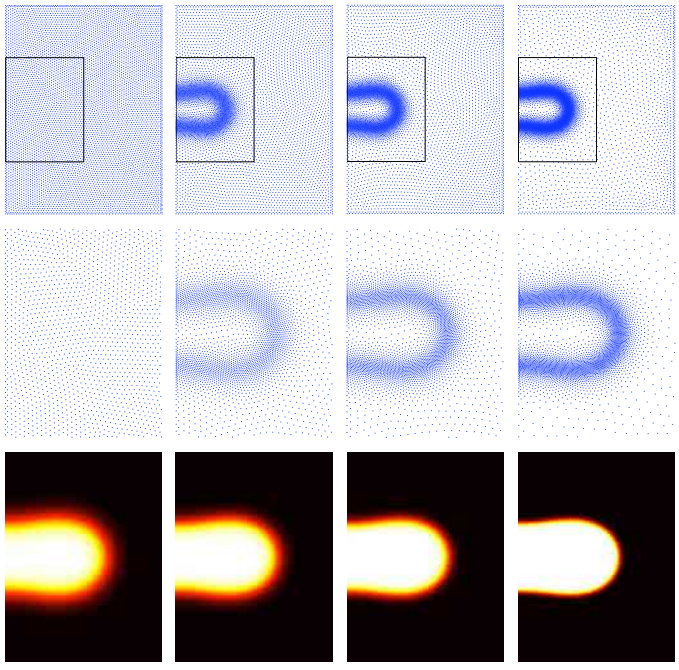


Figure 3.6: Discocyte equilibrium shape. Uniform and adapted grids of 6,124 points (top). From left to right: O1, O11, O12 and O13. Zoom of the areas indicated with black boxes (center). Phase-field (bottom). From left to right, the solutions correspond to $\epsilon = 0.04$, $\epsilon = 0.03$, $\epsilon = 0.02$, and $\epsilon = 0.01$.

value for the regularization parameter ϵ_{min} for a given grid is determined by the nodal spacing distribution. As expected, the ability of adapted grids to accurately support sharp phase-field solutions at an affordable cost is noteworthy. Adapted grids grant the same accuracy (measured by the optimal energy) as uniform grids with a 50-fold reduction in the number of degrees of freedom for $\epsilon = 0.01$. Figure 3.6 (bottom) shows the equilibrium phase-field for the grids referred to in Table 3.2 and shown in Figure 3.6 (top, center). It can be noticed that as the value of ϵ decreases, the thickness of the diffuse interface shrinks considerably.

Further details and numerical examples can be found in the corresponding paper Appendix A.

3.4 Vesicle dynamics : an adaptive Lagrangian approach

We introduce here the meshfree Lagrangian method proposed in our paper Appendix B to study vesicle dynamics. Having shown that the adaptive meshfree method based on the local maximum entropy approximants can yield very accurate solutions at an affordable cost for a phase-field model for biomembranes, we turn now to the (creeping) dynamics of vesicles embedded in a viscous fluid. The adaptive method proposed in Section 3.3 is adequate to analyze very accurately a given equilibrium configuration, but is not as well-suited to study quasi-statically equilibrium branches, as these exhibit buckling events, i.e. very large shape transitions for a small change in the enclosed volume for instance. The adapted grid for a given state cannot represent the solution for a very different state, as the high resolution is tailored to the initial state. Over-damped dynamics or gradient flows, even without a clear physical meaning [88, 89, 90], can be used to numerically obtain equilibrium shapes, and the method we proposed here can be interpreted in this vein. Furthermore, the dynamics of vesicles embedded in a viscous fluid is of interest by itself (see for instance [91] for a state of the art parametric method for vesicles combined with a boundary integral method for the Stokes equations with spherical harmonic approximants). Generally, the inertial effects can be disregarded, and here we consider the low Reynolds number limit.

Phase-field models for bio-membranes have been coupled with fluid dynamics by various authors [84, 83]. In all previous approaches, Eulerian framework was adopted, and a transport equation for the phase-field was set in place. If these models are combined with adaptivity, cumbersome mesh projection steps will be needed. In contrast, we propose here a meshfree Lagrangian method to study the dynamics of biomembranes embedded in a viscous fluid with the following features:

- The phase-field is viewed as a material property of the continuous medium, and the elastic energy changes as this field is pushed-forward by the deformation. The deformation rate produces viscous forces. The dynamics result from the balance of the configurational forces of the phase-field elastic energy and the viscous forces, subject to area or volume constraints.

- Due to the Lagrangian nature of the method and the stability of membrane structures, if the initial grid is adapted, the adaptivity automatically follows the sharp features of the solution, although sporadic remeshing steps may become necessary.

- The variational structure of the problem is fully preserved by the discretization schemes, which allows us to use robust incremental minimization implicit time-stepping schemes, which are non-linearly stable by construction.

- The smoothness of the basis functions allows us to treat in a straightforward way the second order derivatives of the elastic energy functional.

- The meshfree approximants can deal robustly with large deformations of the fluid/bio-membrane continuum. However, the basis functions can be updated by reconnecting the nodes along the simulation.

- The method is the same in 2D or in 3D, and is easily made parallel.

A visual description of the method is shown in Fig. 3.7. The proposed method shares common features with the optimal mass transport (OTM) method presented in [92].

3.4.1 Lagrangian phase-field model formulation

Consider a fixed fluid domain Ω, where a membrane is located and described at time $t = 0$ by a phase-field $\phi_0(\boldsymbol{x})$ (obtained from instance from an equilibrium calculations

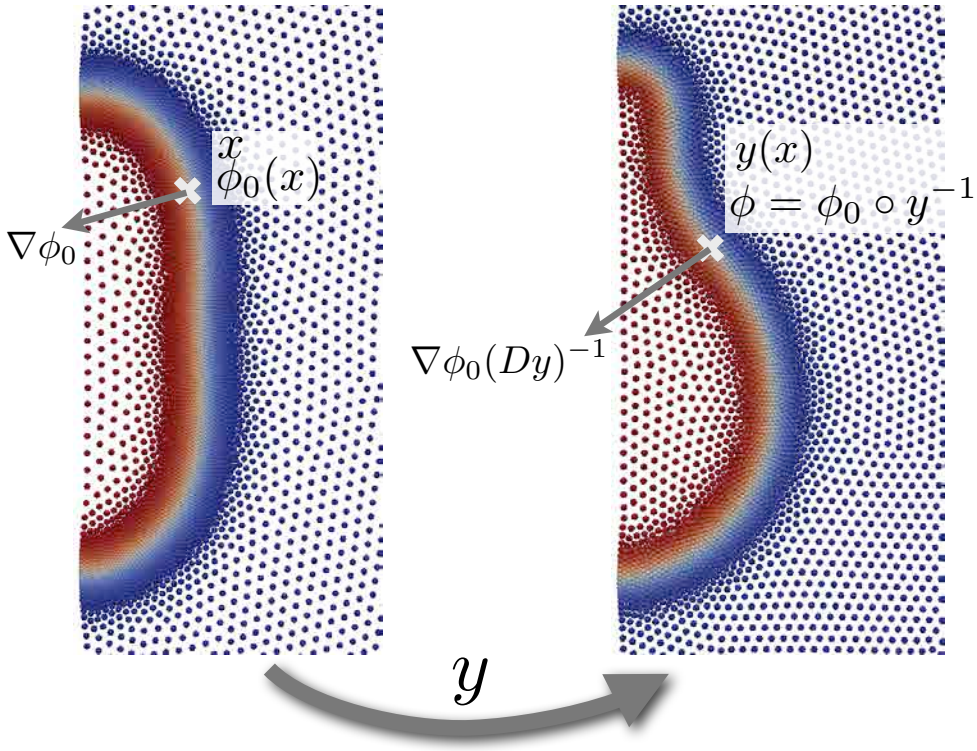


Figure 3.7: Main ideas behind the Lagrangian phase-field formulation. The background medium containing the viscous fluid and the smeared interface is rearranged by a deformation map $y(x)$, which deforms the phase-field, illustrated by a color map on the nodes of the computational grid. The phase-field is advected (pushed forward) as a material property as $\phi = \phi_0 \circ y$. The gradient of the deformed phase-field transforms as indicated, and as shown in the text, we can also compute $\Delta\phi$ as a push-forward of $\Delta\phi_0$. This allows us to write the Helfrich curvature energy in terms of y, and the viscous dissipation in terms of $\dot{y}$. Computationally, the deformation is discretized in terms of particle positions, indicated with colored circles, and the phase-field and its derivatives are sampled at fixed quadrature points in the reference configuration. As the Lagrangian simulation proceeds, the adaptivity follows the phase-field features.

as described in previous sections). Consider now a motion of the continuum medium, i.e. a smooth bijective mapping on Ω at each instant of time, $\boldsymbol{y}_t(\boldsymbol{x})$. Viewing the phase-field as a material property, attached to the material particles, it is pushed forward by the simulation following

$$\phi_t(\boldsymbol{x}) = \phi_0 \circ \boldsymbol{y}_t^{-1}(\boldsymbol{x}) = \phi_0 \left(\boldsymbol{y}_t^{-1}(\boldsymbol{x})\right). \tag{3.13}$$

From this point on, we omit the explicit dependence on t of the motion and the pushed-forward phase-field. The elastic energy of the membrane in terms of the phase-field can be computed as

$$E = \frac{3}{8\sqrt{2}}\frac{k}{2\epsilon}\int_\Omega \left[\epsilon\Delta\phi + \left(\frac{1}{\epsilon}\phi + C_0\sqrt{2}\right)\left(1-\phi^2\right)\right]^2 \, d\Omega. \tag{3.14}$$

The enclosed volume and surface are can be computed as

$$V = \frac{1}{2}\left(Vol(\Omega) + \int_\Omega \phi \, d\Omega\right) \tag{3.15}$$

and

$$A = \frac{3}{2\sqrt{2}}\int_\Omega \left[\frac{\epsilon}{2}|\nabla\phi|^2 + \frac{1}{4\epsilon}(\phi^2-1)^2\right] \, d\Omega. \tag{3.16}$$

To compute the spacial derivatives of the phase-field, we recall Eq. (3.13) and the inverse function theorem to obtain

$$\nabla\phi = \left(\nabla\phi_0 \mathbf{F}^{-1}\right) \circ \boldsymbol{y}^{-1}, \tag{3.17}$$

where $\mathbf{F} = \mathbf{D}\mathbf{y}$ is the deformation gradient. To compute the Laplacian of the pushed-forward phase-field, we resort to indicial notation and omit the composition with the deformation map or its inverse as it can be inferred from the context. From the

relation $\partial_i\phi = \partial_I\phi_0 F^{-1}_{Ii}$ we have

$$\partial^2_{ij}\phi \circ y = \partial^2_{IJ}\phi_0 F^{-1}_{Ii} F^{-1}_{Jj} + \partial_I\phi_0 \partial_j F^{-1}_{Ii}. \tag{3.18}$$

Now, from $F^{-1}_{Ik}F_{kJ} = \delta_{IJ}$, we obtain

$$\partial_j F^{-1}_{Ii} = -F^{-1}_{In}F^{-1}_{Ji}F^{-1}_{Kj}\partial_K F_{nJ} = -F^{-1}_{In}F^{-1}_{Ji}F^{-1}_{Kj}\partial^2_{JK} y_n. \tag{3.19}$$

In particular, we have

$$\Delta\phi \circ y = \partial^2_{IJ}\phi_0 F^{-1}_{Ii} F^{-1}_{Ji} + \partial_I\phi_0 \partial_i F^{-1}_{Ii}. \tag{3.20}$$

Thus, inserting these two equations into the above functionals and pulling-back the integration by the deformation map, it is clear that they can be interpreted as functions of the deformation mapping, depending parametrically on the initial phase-field:

$$E[\boldsymbol{y}] = \frac{3}{8\sqrt{2}}\frac{k}{2\epsilon}\int_\Omega \left[\epsilon\Delta\phi\circ\boldsymbol{y} + \left(\frac{1}{\epsilon}\phi_0 + C_0\sqrt{2}\right)\left(1-\phi_0^2\right)\right]^2 \det(\mathbf{F})\; d\Omega, \tag{3.21}$$

$$V[\boldsymbol{y}] = \frac{1}{2}\left(Vol(\Omega) + \int_\Omega \phi_0 \det(\mathbf{F})\; d\Omega\right), \tag{3.22}$$

and

$$A[\boldsymbol{y}] = \frac{3}{2\sqrt{2}}\int_\Omega \left[\frac{\epsilon}{2}|\nabla\phi_0\mathbf{F}^{-1}|^2 + \frac{1}{4\epsilon}(\phi_0^2-1)^2\right]\det(\mathbf{F})\; d\Omega \tag{3.23}$$

Lengthy but otherwise straightforward calculations allow us to compute the variations of these functionals with respect to the deformation. It is obvious that if the configuration mapping turns to be the identity, the expression for initial phase-field description is recovered. We can apply this push-forward to the remaining terms in our problem, such as the dissipation potential. To simplify the exposition of the method, we consider the Stokes equations for a slightly compressible fluid, i.e. a

penalized formulation of the incompressible Stokes equations.

Following standard continuum mechanics definitions, the Eulerian velocity field can be computed as

$$\mathbf{v} = \partial_t \boldsymbol{y} \circ \boldsymbol{y}^{-1}. \tag{3.24}$$

Consequently, the velocity gradient tensor can be written as

$$\nabla \mathbf{v} \circ \boldsymbol{y} = \dot{\mathbf{F}}\mathbf{F}^{-1}, \tag{3.25}$$

where $\dot{F}_{iI} = \partial_I \partial_t \boldsymbol{y}_i$, and the rate-of-deformation tensor in the Lagrangian domain as

$$\mathbf{d} \circ \boldsymbol{y} = \frac{1}{2}\left(\dot{\mathbf{F}}\mathbf{F}^{-1} + \mathbf{F}^{-T}\dot{\mathbf{F}}^{T}\right). \tag{3.26}$$

The Rayleigh dissipation potential for a compressible Newtonian fluid can therefore be written as [93]

$$\begin{aligned} Diss[\partial_t \boldsymbol{y}; \boldsymbol{y}] &= \mu \int_\Omega \mathbf{d} : \mathbf{d}\ d\Omega + \frac{\lambda}{2}\int_\Omega (\mathrm{d}iv\ \mathbf{v})^2\ d\Omega \\ &= \frac{\mu}{4}\int_\Omega \left|\dot{\mathbf{F}}\mathbf{F}^{-1} + \mathbf{F}^{-T}\dot{\mathbf{F}}^{T}\right|^2 (\det \mathbf{F})\ d\Omega \\ &\quad + \frac{\lambda}{2}\int_\Omega \left[trace(\dot{\mathbf{F}}\mathbf{F}^{-1})\right]^2 (\det \mathbf{F})\ d\Omega. \end{aligned} \tag{3.27}$$

where μ is the shear viscosity of the fluid, and by $Diss[\partial_t \boldsymbol{y}; \boldsymbol{y}]$ we highlight the parametric dependence of the functional on the current deformation $\boldsymbol{y}$. The coefficient λ can be interpreted here as a penalty parameter enforcing incompressibility approximately. For an incompressible Newtonian fluid, the second term above is replaced by the constraint

$$\mathrm{t}r(\dot{\mathbf{F}}\mathbf{F}^{-1}) = 0, \tag{3.28}$$

the linearization of the condition $\det \mathbf{F} = 1$.

We apply the following variational principle [94] to describe the motion of our coupled problem system, arising from the minimization of a generalized potential expressing the competition between elastic forces and bulk friction forces,

$$Diss[\partial_t \boldsymbol{y}; \boldsymbol{y}] + \delta E[\partial_t \boldsymbol{y}; \boldsymbol{y}], \tag{3.29}$$

with respect to $\partial_t \boldsymbol{y}$ subject to the constraints

$$\delta A[\partial_t \boldsymbol{y}; \boldsymbol{y}] = 0. \tag{3.30}$$

If the surrounding fluid is incompressible, the following constraint can be added

$$\delta V[\partial_t \boldsymbol{y}; \boldsymbol{y}] = 0. \tag{3.31}$$

3.4.2 Numerical approach

The numerical discretization of the variational principle 3.29 in spatial domain using LME approximants is straight-forward and it is detailed in Appendix B. However, we note here two additional issues regarding the time discretization and the reconnection strategies. While minimizing the action and then applying the discretization leads to a system of nonlinear differential algebraic equations that can be solved with standard algorithms, the system is stiff because of the nature of the curvature energy, and because of the presence of constraints. We find that standard numerical packages have serious difficulties in dealing with these equations, and require very small time-steps when the system is significantly out of equilibrium. Instead, we develop variational time-incremental integrators, which can robustly deal with large time-steps. Let us consider the simplest finite difference approximations for the rate of change of the

nodal positions

$$\dot{\boldsymbol{y}} \approx \frac{\boldsymbol{y}^{n+1} - \boldsymbol{y}^n}{\Delta t}, \tag{3.32}$$

and for the rate of change of the energy

$$\dot{E} = \frac{E(\boldsymbol{y}^{n+1}) - E(\boldsymbol{y}^n)}{\Delta t}. \tag{3.33}$$

We can then discretize in time the action, and given $\boldsymbol{y}^n$ find $\boldsymbol{y}^{n+1}$ by minimizing

$$\frac{1}{2}(\boldsymbol{y} - \boldsymbol{y}^n)^T \boldsymbol{K}(\boldsymbol{y}^n)(\boldsymbol{y} - \boldsymbol{y}^n) + \Delta t E(\boldsymbol{y}) \tag{3.34}$$

with respect to $\boldsymbol{y}$, subject to

$$A^h(\boldsymbol{y}) = A_0, \tag{3.35}$$

where K is the stiffness matrix resulting from the Galerkin discretization of the Stokes equations, providing a discrete dissipation potential that depends on $\boldsymbol{y}^n$, see Eq. 3.27. In the expression we have multiplied the action by Δt^2 and ignored the constant $E(\boldsymbol{y}^n)$ in Eq. (3.34). This method is related to the backward-Euler method, and many other variational time-integrators can be defined by choosing different time-discrete approximations of the action. The resulting nonlinear optimization program can be solved with a variety of methods. Here, we impose the constraints with Lagrange multipliers and solve the first order optimality conditions with Newton's method. We also note that, by construction, $E(\boldsymbol{y}^{n+1}) \leq E(\boldsymbol{y}^n)$, and therefore the method is endowed automatically with nonlinear stability. We note that adaptive time-stepping algorithms can be easily designed, for instance adapting Δt in such a way that ΔE is nearly constant. The adaptivity may also be driven by the number of iterations needed in the nonlinear solver.

As shown in Fig. 3.7, if the initial grid adapts to the features of the phase-field,

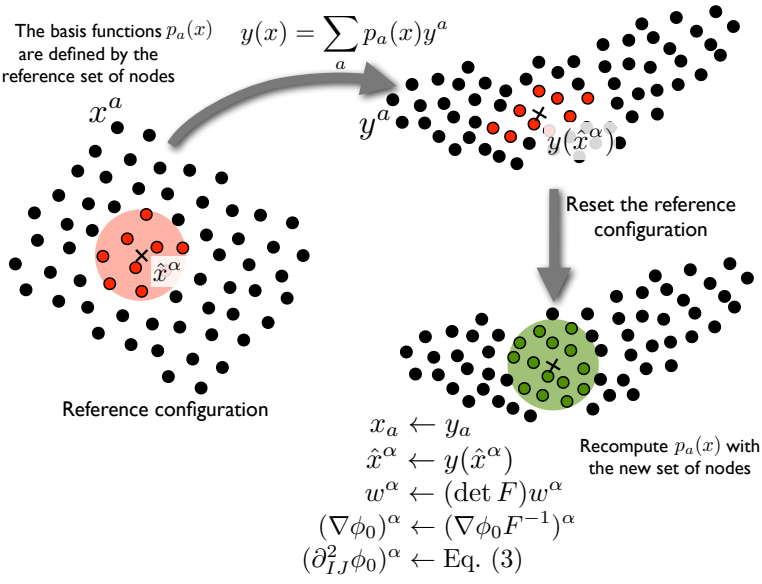


Figure 3.8: As the Lagrangian simulation proceeds, the deformation may significantly distort the domain. To avoid this, we periodically reset the reference configuration, as shown in the figure. This involves reseting the reference node position to the current position, recomputing the meshfree basis functions from the new node set, which involves new neighbor searches as indicated with the colored regions, and reseting the quadrature points $\hat{x}^\alpha$, the corresponding weights, and the reference phase-field first and second derivatives as indicated in the figure. Note that the reference phase-field value at the quadrature points, ϕ_0^α, does not need to be updated as the phase-field is a material property and the quadrature points keep their material identity.

adaptivity is advected by the Lagrangian map, and therefore local refinement along the dynamics is accomplished for free. The Lagrangian framework allows us to pull-back the successive states of the system to a reference configuration. Thus, we avoid the calculation of the meshfree basis functions in every step of the evolution. It has been shown that the meshfree method considered here can withstand significant deformation before the discretized deformation mapping ceases to be injective (i.e. the Jacobian determinant becomes negative at a quadrature point) [35]. However, we avoid coming close to this limit, which degrades the accuracy of the approximation, by reconnecting the nodes, recomputing the basis functions, and resetting the

reference configuration periodically along the simulation. These reconnection steps are seamless, as detailed in Fig. 3.8: they do not involve re- meshing, recomputing the background grid for quadrature, field projections, nor do they alter in any way the variational structure of the discrete equations, e.g. the nonlinear stability of the dynamics. In situations involving extreme Lagrangian deformations, particles may accumulate or cover insufficiently parts of the domain. In such cases, a full re-meshing and field projections are required.

3.4.3 Numerical results

As a sample of the proposed method performance, Fig. 3.9 shows the relaxation dynamics of an oblate vesicle brought out-of-equilibrium. The reduced volume, a ratio between the volume and area of the vesicle, is $v = 0.9$. We show the location of the nodes $y^a(t)$, and color-code them by the value of the phase-field. In this example, exhibiting moderate deformations, we do not reconnect in any way the nodes and therefore the evolution is purely Lagrangian, with the initial configurations as a reference configuration during the whole motion. The calculation proceeds robustly despite the distortions. It can be appreciated how the phase-field elastic energy maintains the transversal density of the nodes, and how the adapted region follows the features of the phase-field. We check the accuracy of this simulation with additional runs with a larger number of integration points, number of nodes, and different LME aspect ratio parameters γ.

The performance of the method is analyzed in Fig. 3.10. The left plot shows the non-dimensional energy $E^* = E/k$ and the non-dimensional time-step as a function of non-dimensional time $t^* = tk/(\mu R_0^3)$. The energy monotonically decreases as expected, converging towards the equilibrium energy calculated independently with a parametric method. As the process advances, the adaptive time-step grows to

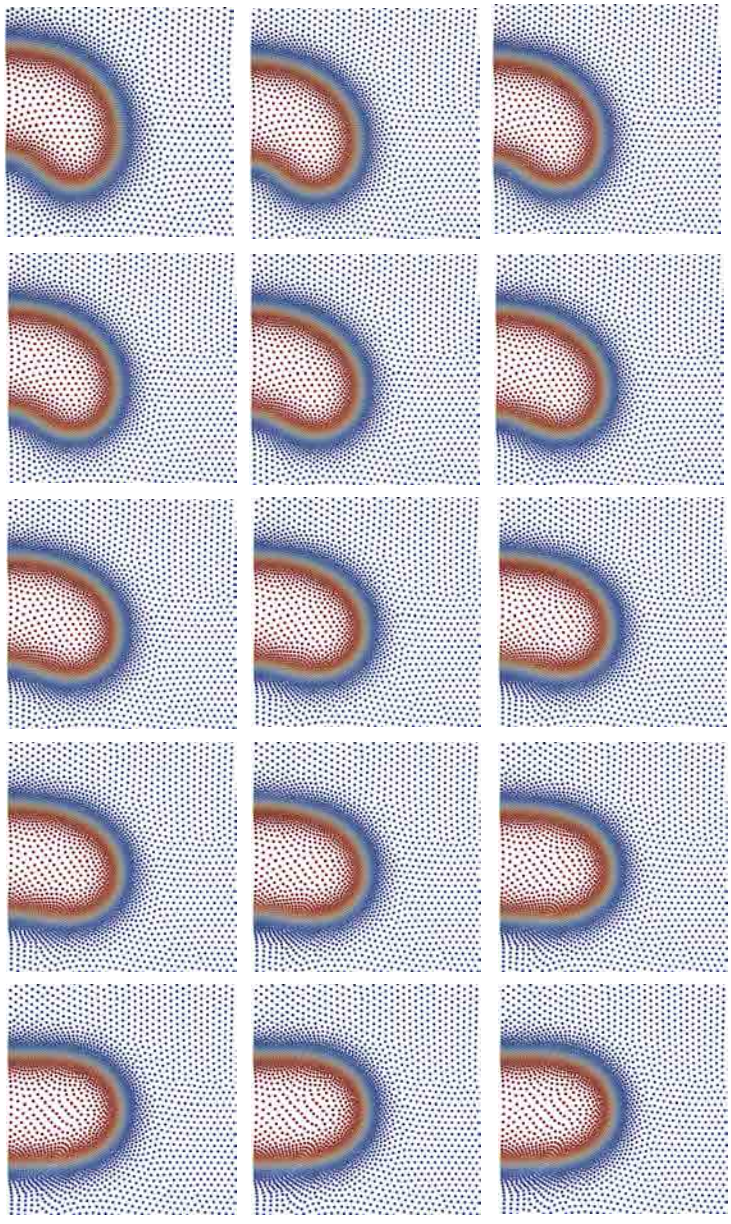


Figure 3.9: Relaxation dynamics of an oblate vesicle in a viscous fluid, initially brought out-of-equilibrium. We represent the time-evolution of the nodes $y^a(t)$, color-coded with the phase-field. The adapted grid has 6124 nodes.

roughly keep the energy decrement per time-step constant. Remarkably, the time-step changes by two orders of magnitude during the simulation. At the final stages, the time-step hits the maximum allowed size.

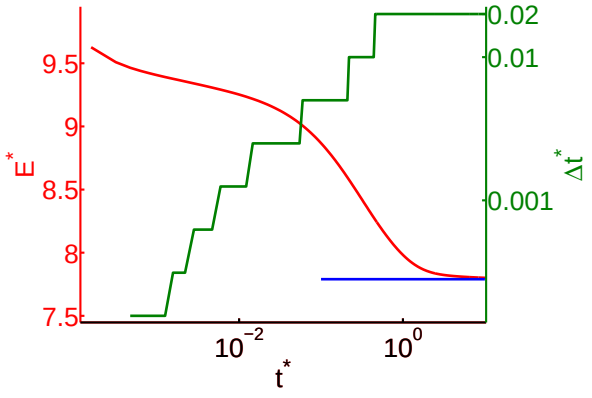


Figure 3.10: Energy relaxation and time adaptive strategy for the dynamics depicted in Fig. 3.9. Energy and time-step evolution, where time is represented in logarithmic scale. The blue horizontal line shows the equilibrium energy obtained independently with a parametric method based on B-Splines.

Table 3.3: Elastic energy and computational cost for different constant time-steps and methods (VTI: variational time-integration, FE: forward Euler). $t_1^* = 1.0 \cdot 10^{-3}$, $t_2^* = 1.1 \cdot 10^{-2}$.

Method	Δt^*	$E^*(t_1^*)$	$E^*(t_2^*)$	*steps*	*grad*	*hess*
VTI	$1.0 \cdot 10^{-2}$	9.374	9.243	1	2	2
VTI	$1.0 \cdot 10^{-3}$	9.374	9.240	10	20	10
VTI	$1.0 \cdot 10^{-4}$	9.374	9.239	100	200	100
FE	$1.0 \cdot 10^{-5}$	9.374	9.239	1000	1000	0

Although more sophisticated time-stepping schemes are possible, we compare the proposed variational time-integration (VTI) method with an explicit forward Euler (FE) method. It is computationally infeasible to perform the full relaxation dynamics with the forward Euler method, which imposes very stringent conditions on the time-step. Instead, we focus on a portion of the dynamics, and report the results

in Table 3.3. In the VTI method, we use Newton's method to numerically solve the optimization problem in Eq. (3.34), and for computational efficiency, update the Hessian matrix only once per time-step, not per iteration. However, for the largest time-step, we need to update the Hessian in each iteration for convergence. In all cases, Newton's method converges in two iterations. The table compares the VTI method with time-steps $\Delta t^* = 10^{-2}, 10^{-3}, 10^{-4}$, and the FE method with the largest time-step for stability in this interval, $\Delta t^* = 10^{-5}$. The accuracy is reported in terms of the energy at the end of the interval, and the computational cost in terms of gradient and Hessian evaluations. The table shows the ability of VTI to robustly take large time-steps with accurate results. In contrast, we find that for this nonlinear system, it is very difficult to stably adjust the time-step length in the FE method. We find that the VTI method provides a similar accuracy to the explicit method with time-steps between one and two orders of magnitude larger. This ratio is even more dramatic in the initial fast stages of the dynamics.

We next exercise the method in more challenging dynamics, involving large shape changes. Fig. 3.11 (left) shows a stomatocyte-discocyte dynamical transition. For the considered reduced volume $v = 0.6$, both a stomatocyte and a discocyte are metastable configurations, the latter having lower energy. We slightly displace the stomatocyte equilibrium configuration beyond the energy barrier, and then the system spontaneously evolves towards the discocyte configuration. The reference configuration is reset when large distortions occur as measured with the gradient of deformation mapping. In this simulation, the reference configuration is reset every 20 time-steps. The time-adaptive scheme allows us to efficiently track the entire transition, and by the end of the simulation the time-step is 2,048 times the initial time-step. Fig. 3.11 (right) shows the response of an prolate vesicle ($v = 0.8$) subject to an instantaneous change of spontaneous curvature from $C_0 = 0$ to $C_0\sqrt{A_0/(4\pi)} = 10.0$, which can be the result of exposing the bilayer to a different chemical environ-

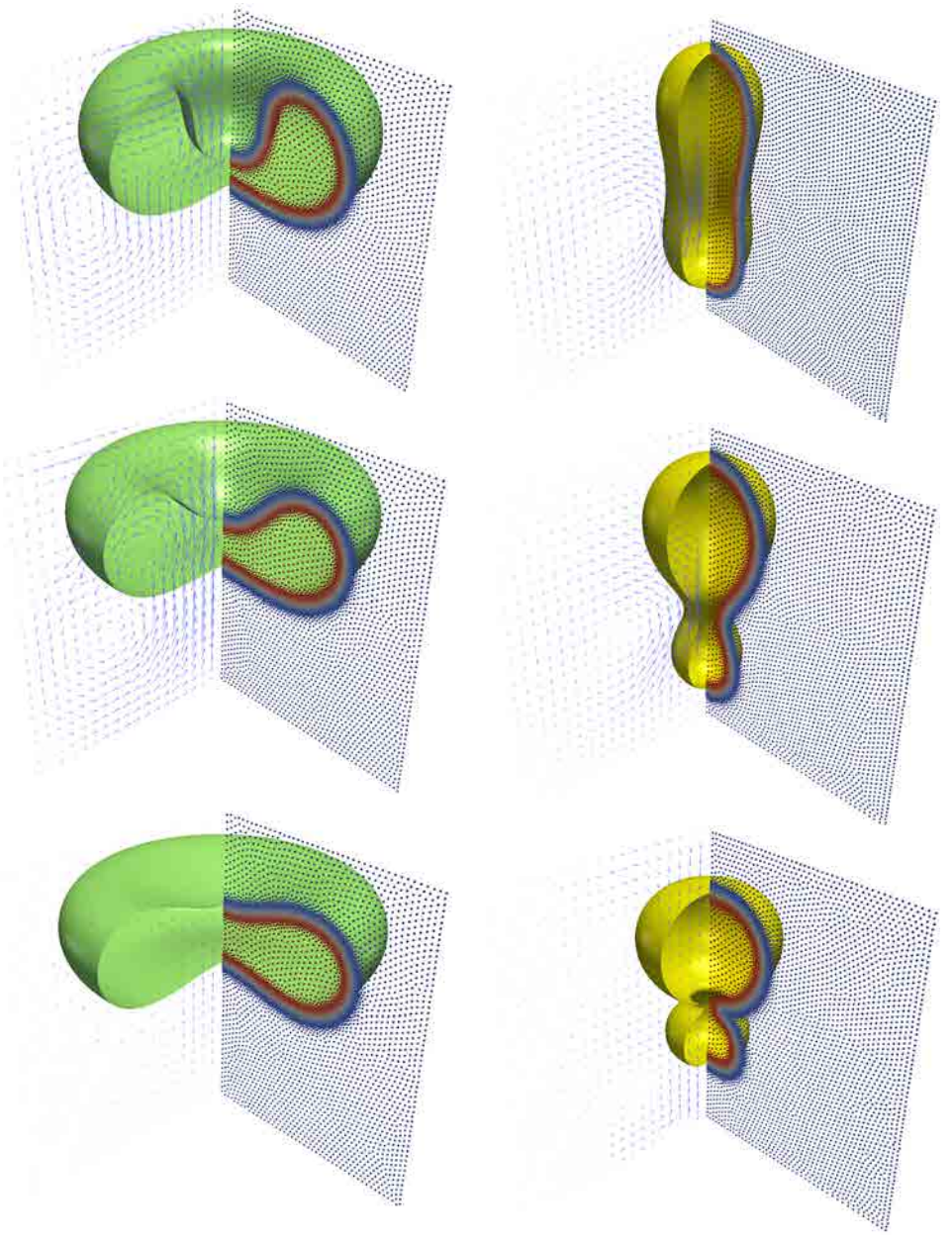


Figure 3.11: Left: Stomacyte-discocyte transition. Right: Prolate vesicle evolving after an instantaneous change of spontaneous curvature (6,124 nodes, constant area and volume). The points represent the nodes, color-coded with the phase-field, while the arrows depict the flow field in a symmetry plane.

ment [95, 96, 89]. The system evolves towards a configuration consisting of two dissimilar spheres connected by a narrow neck, which best adjusts to the imposed spontaneous curvature with the available volume. Both simulations run on a CVT adapted grid of 6,124 nodes.

Fig. 3.12 shows an even more dramatic shape change, in which a prolate vesicle is deflated from $v = 0.9$ to $v = 0.55$ and its spontaneous curvature increased to $C_0\sqrt{A_0/(4\pi)} = 12.0$, leading to an elongation and pearling transformation, widely observed in experiments [97]. The method robustly follows all the large shape deformations with an adapted grid of 12,650 nodes.

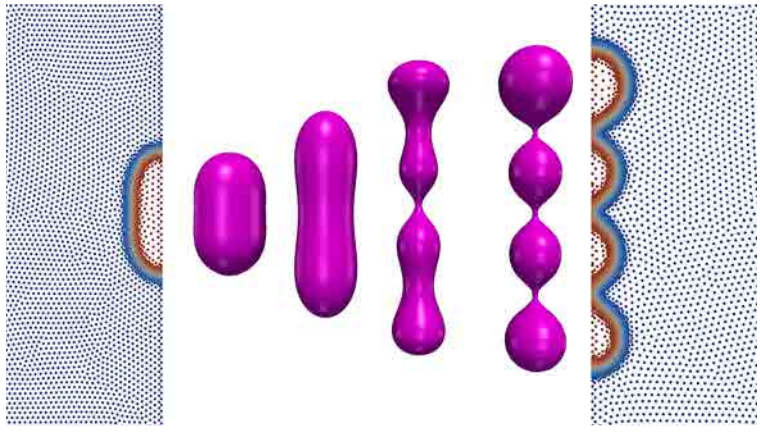


Figure 3.12: Relaxation dynamics of a constant area vesicle under combined volume decrease and spontaneous curvature increase (12,650 nodes, constant area).

For further discussion, more examples and numerical details see our paper Appendix B.

3.5 Complex biological processes : influence of kinetics and adhesion in vesicle shaping

3.5.1 Motivation

We present here an overview of our current projects in the biomembrane research line. We aim to apply our meshfree phase-field model to explain the formation of some important biomembrane structures from a mechanical point of view. Microscopy experiments have revealed the high degree of complexity of the spatial organization of biological systems. Conventionally, the shaping of membrane structures is attributed to biological regulation, involving membrane proteins and a tight control of lipid composition [98]. However, biological regulation should obey the laws of physics, which for membrane shaping involves deforming the bilayer and displacing the surrounding fluid. For example, the mitochondrium structure in Fig. 3.13 involves an external and internal membrane. Since the inner membrane has an enormous excess of area, it

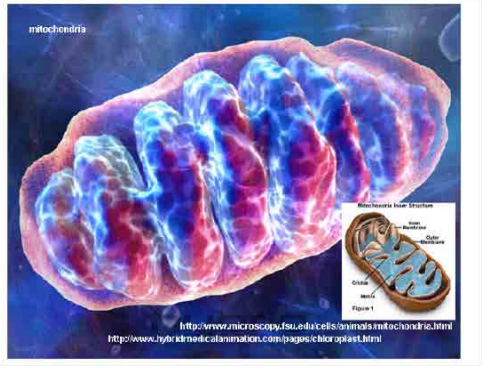


Figure 3.13: Representation of a mithocondrium external membrane and internal matrix. The internal matrix has been shown to change between several complex geometrical states experiencing merging and nucleation phenomena in the process. Although several chemical agents come into play, some shapes could be partially or completely explained by bare mechanical features, such as the combination of the elastic energy of the membrane, the dissipative fluid media and the adhesion between the vesicle surfaces. Sources: www.microscopy.fsu.edu, www.hybridmedicalanimation.com

shapes in folds and invaginations (cristae), which are fundamental for their function. Although there are chemical agents acting in matrix shaping, it is very likely that some phenomena (isolated or in combination) such as adhesion or confinement could explain by themselves the inner membrane shape before those chemical agents play their role. In general, membranes are confined to adjacent membranes, to external substrates such as the extra-cellular matrix, or to internal cytoskeletal structures, such as the acto-myosin cortex. Despite this generic confinement, the main model system for biomembrane shape transitions have been vesicles. Recent works have focused on the mechanics of membranes under confinement, i.e. adhered to a deformable substrate.

It has been shown in experiments that in biomimetic systems, for example, protrusions can arise from confined membranes upon straining the supporting surface or by adding lipids or peptides. A similar process is undergone by a cell where the plasma membrane bulges into microvesicles upon contraction of the underlying cortex. In [99], an experimental setup is presented to study confinement effects, consisting in a channel between a glass cover slip and a polydimethylsiloxane (PDMS) slab, coated with a uniform lipid bilayer. Using this original setup, it is possible to (1) subject a supported lipid membrane to a lateral strain by deforming (inflating or deflating) the PDMS sheet underneath the membrane and (2) modify the volume of interstitial liquid by controlling the osmolarity of the solution above the membrane. The results are shown in a phase diagram that summarizes the shape transformations in terms of the two variables (Fig. 3.14) i.e. the volume enclosed in the interstitial space between the bilayer and the substrate, and the compressive strain of the substrate.

As the substrate is progressively compressed, the membrane undergoes a process of relaxation by expelling from the adhered part of bilayer tubes containing the excess area. These thin, highly curved tubes are stabilized by osmotic effects, and

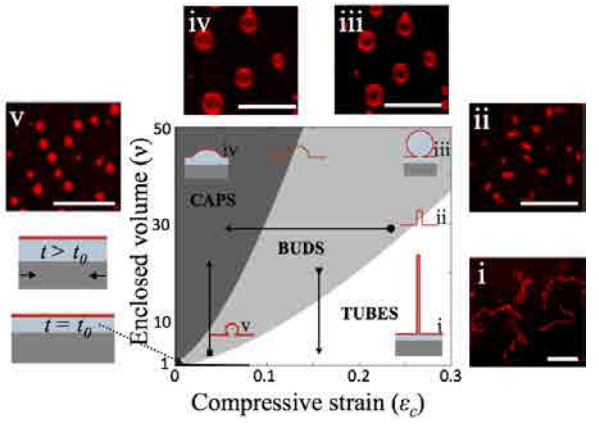


Figure 3.14: Morphological strain-volume phase diagram of confined lipid bilayers, derived theoretically for $R_0 = 4nm$ (a typical average half-distance between the protrusions in the experiments). We distinguish between a fully adhered planar bilayer (black region), and a bilayer with tubular protrusions, labeled TUBES (white area), or with spherical protrusions, labeled BUDS (light grey) and CAPS (dark grey). Source: [99]

can be inflated into spherical protrusions by subjecting the system to hypo-osmotic conditions. While tube formation had been previously explained by localized forces on the membrane and/or protein induced spontaneous curvature, these experiments show that tubes can spontaneously form by the mechanics of confinement.

Moving to highly dynamical observations, in [100] outward tubes have been observed in vesicles upon rapid lipid incorporation (fast area increase), where gradual changes and no localized structures such as tubular shapes are expected if the rate of lipid incorporation is slow. In [101], similar experiments were performed by dynamically incorporating cholesterol into vesicles, effectively increasing their surface area rapidly. In this case, vesicles were adhered to a substrate and tubes were inwards. These experiments suggest that dynamics can be a previously unexplored means of shaping membranes into highly curved structures, which may be later stabilized by biochemical regulation. (see Fig. 3.15).

Here, we want to investigate to which degree highly curves membrane structures,

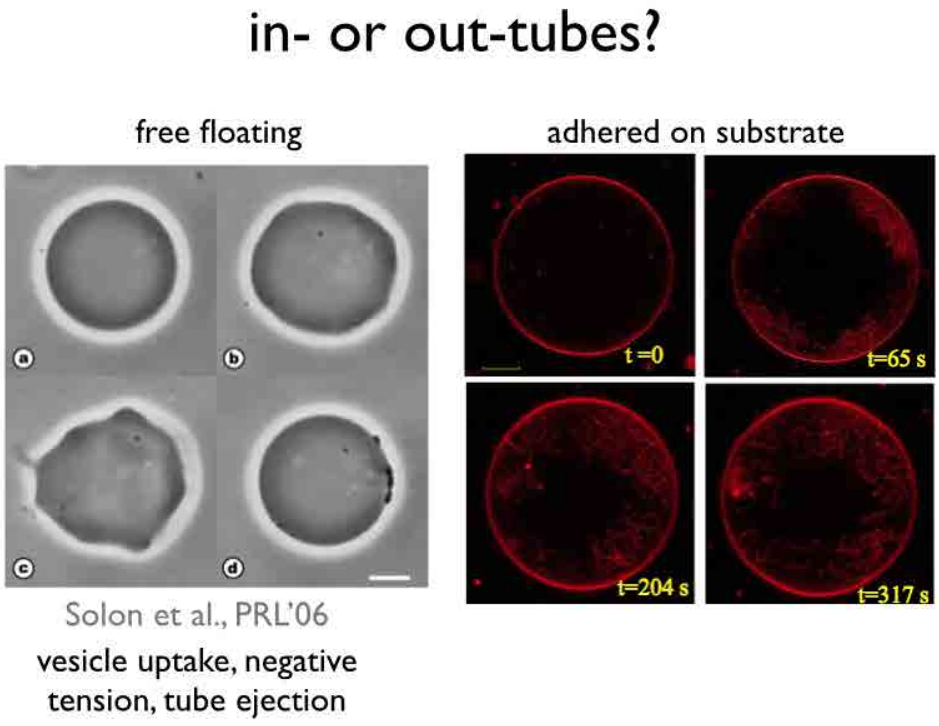


Figure 3.15: Left: Transformation of the shape of a negatively charged giant vesicle in contact with positive small vesicles. From an optically tense state, the amplitude of the fluctuations increases rapidly, and the vesicle is strongly deformed with outward thin tubular instabilities. After some time, the vesicle recovers its initial spherical shape and tension but with dense lipid aggregates on its surface bar, $10\mu m$). Source: [100] Right: transformation of a vesicle adhered to a substrate upon rapid lipid incorporation. Tubular instabilities show inward pattern. Source: [101]

e.g. tubes and invaginations, can arise merely from membrane mechanics. Motivated by the aforementioned recent experiments, we want to specifically address the role of kinetics and adhesion, which is a previously unexplored aspect.

3.5.2 Modeling adhesion

We introduce now the numerical framework to consider adhesion in our model. The bilayer is subjected to various forces (electrostatic, Van der Waals, structural) that define an attractive and repulsive behavior between the bilayer and a substrate/other bilayers. This combined potential defines the membrane adhesion, which is important in various situations. For example, tissue formation is based on mutual adhesion of cell membranes to a macromolecule network. Binding and unbinding of vesicles is also key to transport processes within the surface of cells and organelles within, which can be used for targeted drug delivering. Biosensors are also a good example of technology based on the binding of vesicles to surfaces. We take advantage of the energy variational structure to easily extend the global energy functional with an extra term accounting for the adhesion potential. We consider the adhesion by adding the following term,

$$\int_{\Omega} W(\mathbf{x}) F(\phi(\mathbf{x}))\ d\Omega, \tag{3.36}$$

where W is a classical adhesion potential depending on the spatial domain and F is a function needed in the phase-field model to localize W onto the membrane surface. F can be defined in various ways. Here we use a simple version proposed by [102],

$$F(\phi(\mathbf{x})) = \frac{3\sqrt{2}}{8} \frac{(\phi^2 - 1)^2}{\epsilon}. \tag{3.37}$$

We resort here to the Lennard-Jones (L-J) potential for W. The L-J potential is

a simple model that approximates the interaction between a pair of neutral atoms or molecules. A form of the potential was first proposed in 1924 by John Lennard-Jones [103]. The most common expression of the L-J potential is:

$$W(h) = 4\omega \left[\left(\frac{\delta}{h} \right)^{12} - \left(\frac{\delta}{h} \right)^{6} \right], \tag{3.38}$$

where ω is the potential well, h is the distance between the two interacting molecules and δ the distance at which the potential becomes zero. The first term accounts for the repulsive part (Pauli repulsion) and the second for the attractive one (Van der Waals forces).

Here we use an integrated version of the L-J potential with a linearized repulsive part, illustrated in Fig. 3.16. The attractive part expression accounts for an

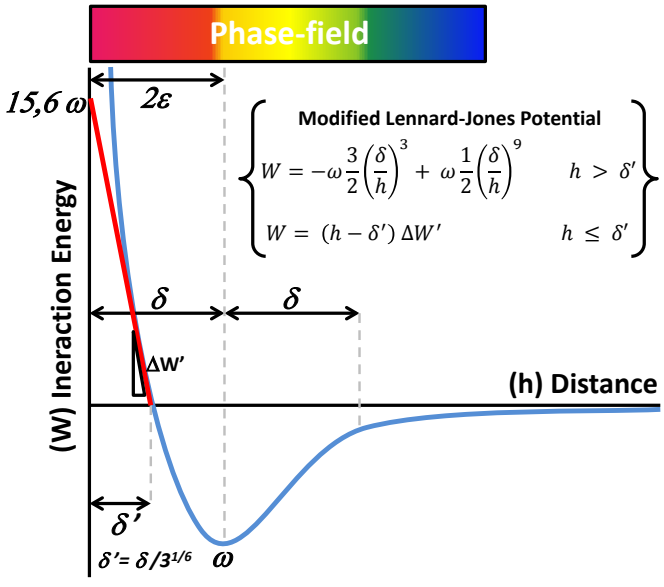


Figure 3.16: Lennard-Jones adhesion potential for a phase-field model of a biomembrane. The parameter δ determines the distance of the attractive well ω. δ is related to the phase-field parameter ϵ to be effective throughout the whole thickness of the vesicle. The repulsive part is linearized to avoid numerical problems due to its quick growth, improving the robustness of the iterative process.

integrated potential i.e. the result of the combined forces of a surface on a single atom. The linearized part of the expression is just a way of smoothing the numerical stiffness that introduces the repulsive part of the L-J potential, which quickly grows to infinite and generating numerical instability. On the other hand, and since we are using a phase-field model and thus the thickness of the membrane is not zero, a proper δ distance to the potential well ω has to be chosen. This value should extend enough to be able to capture the phase-field thickness, which is approximately 2ϵ. Setting the value to $\delta = 2\epsilon$ we ensure that the adhesion potential is acting on the full biomembrane interface, also leaving a layer 2ϵ thick of media in the repulsive contact surface.

The adhesion energy is introduced in our framework as a straight substrate wall and validated through classical theoretical results. The equilibrium shape of a vesicle is now determined by the interplay of elastic and adhesion energies. This competition establishes a phase diagram where two main states, namely *free state* and *bound state*, can be identified [80]. It turns out that, while the bending elastic energy is size independent, the adhesion is not. Therefore, one can work out a meaningful length scale, which determines the boundary between the states. This scale is defined in literature as a relation between the bending stiffness κ and the adhesion potential well ω:

$$R_c = \sqrt{\frac{\kappa}{2\omega}}. \tag{3.39}$$

This characteristic length corresponds to the inverse of the contact curvature of the vesicle meridian at the contact point with the substrate, that is $C_c = \frac{1}{R_c}$. If this length scale is comparable to the size of the vesicle ($R_c \sim R$) then the bending energy dominates and the membrane is free from the wall. As $R_c << R$ the adhesion energy takes over, the vesicle sticks to the wall and the contact curvature is set by

Eq. 3.39. One interesting theory that can be applied here to further understand these phenomena is the droplet theory. Droplets, in contrast with fluid vesicles, lack bending energy and so the adhesion and surface tension determine the final equilibrium shape. For $\kappa = 0$ the minimization of free energy leads to the Laplace equation with the Young-Dupré equation as boundary condition [104]. This equation states the expression for the contact angle ψ

$$\omega = \sigma(1 + cos\psi), \tag{3.40}$$

where σ refers to the Lagrange multiplier enforcing a constant area of the vesicle. Since in the presence of elastic bending energy $\kappa \neq 0$, a contact angle different from π would always imply an infinite energy. Therefore, the concept of contact angle looses in this case its meaning in an strict sense. However, when κ is very small (or ω is large in comparison, meaning $R_c << R$), the vesicle shape recovers that of a droplet with a rounded contact at scale R_c, and an effective contact angle ψ_{eff} can be defined (see Fig. 3.17). This contact angle in bounded states can be checked to measure the agreement of numerical results with theory. Along the same line, in the limit of vanishing κ, the relation between the two Lagrange multipliers σ (area) and p (volume) follows Laplace equation,

$$p = -2\sigma/R, \tag{3.41}$$

which also helps to validate the pressurized state which has been reported in adhered vesicles. The fact that an effective angle close to π is expectable in slightly deflated vesicles, means that those vesicles will present a higher σ and therefore a higher pressurized state. Our simulations in Fig. 3.17 approach the theoretical angles as the adhesion potential well ω increases and the vesicle attaches the substrate.

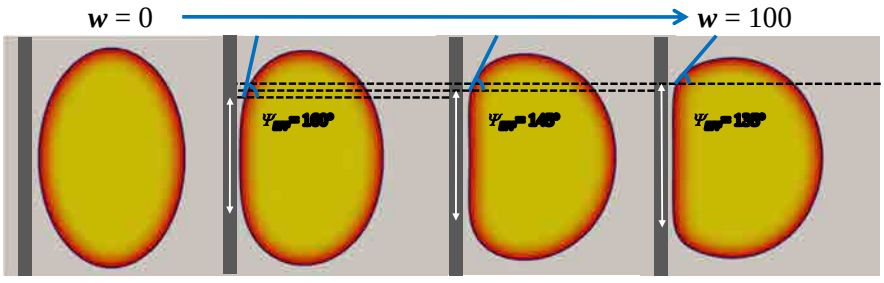


Figure 3.17: The *bounded* and *free* states depend on the scale given by relation between the potential well ω and the bending stiffness κ. As the characteristic radius of the vesicle R increases for a given κ and ω, the adhesion effect becomes more noticeable and it binds to the substrate. Geometrically defined effective angle and adhesion lengths change as the potential well increases for fixed reduced volume (ρ=0.85).

3.6 Kinetics and morphogenesis

Here we isolate the effect of kinetics arising in presence of the dissipative viscous media and a bounding confinement, and let adhesion as an effect to be added in future simulations. As we show in paper Appendix B, kinetic effects can become important by driving the transition shapes in presence of changes of volume and area in fast regimes. Area increase simulations are motivated by the dynamical lipid incorporation experiments reported in Fig. 3.15. In Fig. 3.18, the finite viscosity of the surrounding medium, together with the fast area increase rates, provide an effective confinement that penalizes long-wave geometry changes and favors localized deformation modes such as tubes. Higher area increase rates lead to higher number of stable tubes. In the processes studied, we can identify a number of key points. Initially, a large number of instabilities arise, more wiggly and energetic as high is the increase rate. Next, several relaxations take place, particularly for moderated rates, due to the impossibility to turn some of the buckling phenomena into tubular structures. Once a number of tubes is defined, the structure becomes stable and the tubes subsequently grow. In this propagation we can identify two separated stages.

In the first one, tubes grow at the expense of the area and volume still undeveloped

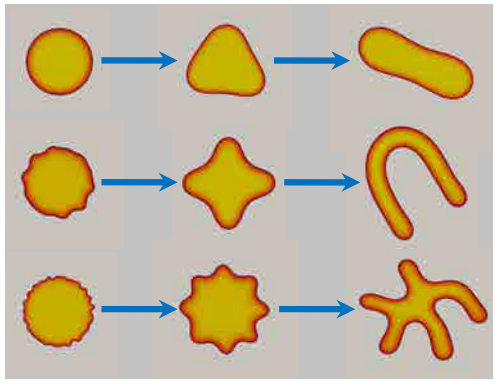


Figure 3.18: Simulation diagram showing the significance of kinetic effects in biomembrane shaping. Three branches coming from a single circle are explored by applying different area increase rates. The higher rates show more local buckling effects and generate more stable tubular structures. (19256 nodes grid, $\epsilon = 0.01$)

.

in the centre of the vesicle. In the second, the tubes are propagated by thinning, which means that tube diameter is reduced and the curvature of the tip increased. This generally leads to steepest rates in the overall elastic energy of the vesicle.

Additionally, we show in Fig. 3.19 and Fig. 3.20 two typical elastic energy patterns observed in the simulations. The first, corresponding to low/moderated area increase rates, shows the evolution of a circular vesicle towards a one single tubular structure. This kind of processes are identified by an initial high rate elastic energy increase due to the arising of several buckling features, and a series of subsequent relaxations that remove some of them towards smoother stable configuration. In Fig. 3.20, the high area increase rate generate a much quicker buckling sequence and most of initial features grow into tubes. In these patterns, the change of regime in the tubular growth, is clear by the change of rate in the elastic energy, which steeps when the inner volume is consumed and thinning appears. This behavior would be observed also in the moderate area increase rates if the simulation lasted long enough. Papers

regarding this part are in preparation.

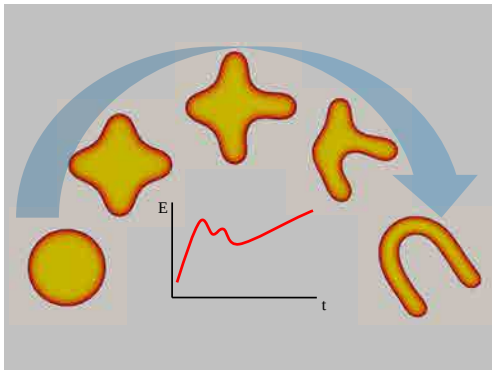


Figure 3.19: Energy profile for moderate area increase rates. The initial bucking events generate an increase in the elastic energy profile. At this moderate rate of area increase, not every instability is allowed to grow into a tubular structure, which leads to decreases in the profile corresponding to elastic relaxations. The remaining stable tubes advance then at smoother elastic energy rate of change.

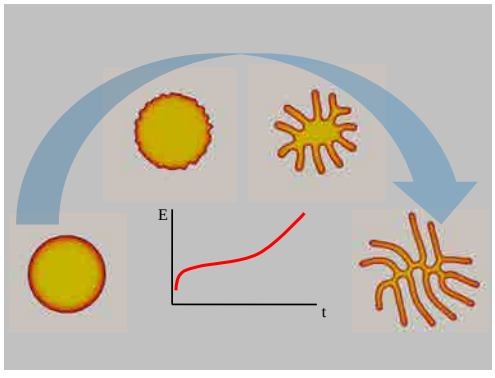


Figure 3.20: Energy profile for high area increase rates. After a quick and short increase due to initial instabilities, most of initial undulations due to buckling evolve into tubular structures. In the mid-term the tubes advance at constant thickness at the expense of the remaining volume at the centre, showing a smoother elastic energy increase rate. When the interior volume is consumed, the tubes grow by thinning, steeping the energy profile again.

Chapter 4

High Performance Computing

We present here an overview of the implementation of the numerical methodologies described in Chapter 3. As a necessary part of the project, we developed a $C++$ library with a number of routines to handle parallelism in a meshfree framework and proposed optimizations over the most demanding parts of our code. We describe in this chapter the core idea underlying these routines and briefly introduce the main contributions in Section 4.1, which are detailed in our paper Appendix C. A more technical view of the library itself is presented in Section 4.2 and relevant $C++$ code is presented in Appendix D.

4.1 Supercomputing: towards an efficient parallel sparse LME environment

As argued in Section 2.1, meshfree phase-field methods lead to high computational costs and, additionally, non-linear dynamics problems require a large number of iterations and/or time steps. The goal of the newly developed routines presented here is to speed-up critical parts of our code to overcome a possible bottleneck i.e. computational time and memory usage. Optimizing the most demanding routines and paral-

lelizing the code in a scalable way is therefore needed to use supercomputing facilities and reach reasonable computational times. For parallelization we chose to work in a distributed memory environment MPI, and use the package PETSc (Portable, Extensible Toolkit for Scientific Computation, see [105]) as a general routine wrapper. PETSc is a widely used scientific software devoted to handle structures like vectors and matrices in parallel, and comes with various scalable linear solvers. It also provides the user with several interfaces to useful external packages e.g. ParMetis [106] for partitioning and reordering routines.

We have focused in three main aspects to reach a working implementation of our methods. First, we have redesigned the way the meshfree sparse matrix and right hand side (RHS) of the system are created and assembled. As we show in the paper Appendix C, in meshfree methods these routines can become the bottleneck in computational time. The main contribution is presented in 4.1.1, where we propose a coarsening strategy over integration to mitigate the associated computational cost. Additional techniques to speed-up the creation and assembly routines can be consulted in the same paper. Second, we have implemented a parallel version of our routines in a new library in $C++$ that provides the user with very general and flexible classes that use PETSc as basis. These classes are able to manage the creation and filling routines in parallel for a meshfree method, among other functionalities. This is not a minor issue since PETSc is very well design and developed for mesh-based methods, but lacks actual routines for meshfree basis functions e.g. the creation of non-zero positions and assembly. Finally, some problems require a repetitive usage of the basis functions, hence their storage is advisable to reduce computational time. In high-order problems, large number of Gauss points and large supports can generate a bottleneck in the memory. In the paper Appendix C we also tackle this problem and propose an efficient storage strategy based on compressed structures that allows to recover the basis functions when needed. We particularize it for LME with excellent

results at the expense of little recovery overcost. In the following we briefly expand on the main contributions of this thesis in this matter.

4.1.1 Neighborhood coarsening algorithm

We introduce here an algorithm, shown in detailed in paper Appendix C, that optimizes the computation of the global sparse matrix in a general meshfree scheme. It is based on a coarsening strategy over integration points, e.g. Gauss points, and its impact is particularly noticeable within the routines leading to creation and the filling of the matrix and RHS of the system.

Regarding the matrices, we recall here that in a Galerkin discretization scheme the number of non-zeros is small in comparison with the total number of positions. These matrices are then called sparse and accept a variety of compression techniques that help to maintain the memory requirements below admissible limits. There are many methods for storing sparse matrices (see, for instance, [107] and [108]). We follow in our code the compressed sparse by row (CSR) storage, which is a proper choice for codes written in $C/C++$ due to its data structure. In CSR, a matrix is given in terms of three lists. The first list, ia, is an array of integers that stores the total number of nonzeros up to each row. Its dimension is the number of rows plus one, the first position being filled with a zero. The second and third lists are arrays of integers and doubles, ja and an, have as dimension the number of nonzeros in the matrix, and store the column index position and the associated matrix entry. We understand the sparse matrix structure creation as the collection of algorithms required to obtain the lists ia and ja, and the filling as the ones needed to obtain the values stored in an. Since unstructured grids and meshfree methods prevent the sparse matrix from having a clear a priori non-zero pattern, the creation and filling of these sparse structures is not straight-forward. We show in our paper Appendix

C how these routines become the most time consuming as the size of the problem increases.

The core of the optimization presented is to alleviate the computational cost of the system sparse matrix by coarse-graining the neighbor primal lists i.e. the lists that contain the nodes that influence a particular quadrature point. In the process also intervene the so called dual lists, which contain the quadrature points that are influenced by a particular node. We refer to our paper Appendix C for a formal definition of the neighbor lists. The key point is to generate a list for each cell/element of a defined coarsening mesh rather than one per Gauss point. The coarsening mesh provides us with a structure to group the primal lists of the Gauss points contained in the cell/element. Without loss of generality, a straightforward and natural choice for the coarsening mesh is the quadrature mesh cells/elements needed in most Galerkin meshfree methods to perform the numerical integration. In this way the complexity added by the increase of Gauss points due to accuracy requirements is removed and the neighbor lists are generated disregarding the number of integration points. We present next details of this procedure.

Once the coarsening mesh is set, we start with a neighbor search over the nodes defining the mesh. This allows us to obtain nodal-based primal lists rather than primal lists for quadrature points. To obtain the cell/element primal lists, the primal lists of its associated nodes are simply merged. More specifically, we define

$$\mathcal{N}_{el} = \bigcup_{a \in \mathcal{T}_{el}} \mathcal{N}^X_{\boldsymbol{x}_a}, \tag{4.1}$$

where $\mathcal{T}_{el}$ is an index set containing the nodal indexes of the el-th cell/element (e.g the mesh connectivity). Note that the $\mathcal{N}_{el}$ list is applicable to the totality of integration points inside the cell/element, regardless their number. This merging operation is negligible in terms of computational time, and give us the possibility to work from

now on with cell/element primal lists rather than with integration point based lists. We illustrate this concept in Fig. 4.1.

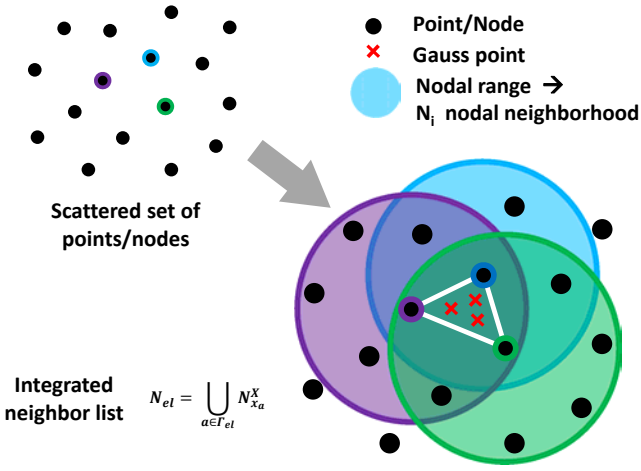


Figure 4.1: Integrated neighborhood concept. The new cell/element neighborhood is described by the union of nodal vertices lists of neighbors. The triangular elements given by the quadrature mesh are used here as background cell/element generator.

The creation and filling algorithms can be now based on cell/element neighbor lists, which greatly speeds-up the computations. The structure creation is simplified since only the nodes and cells/elements are involved in the whole procedure. Now the nonzero positions are identified by looping over cell/element neighbor lists instead of looping over Gauss points neighbor lists. The filling of the matrix benefits in two distinct ways. Firstly, the element-wise approach leads to cell/element dense matrices. These local matrices are efficiently filled since just a loop over the neighbor list of the cell/element and a loop over the cell/element Gauss points are required. Secondly, only one dense cell/element matrix is assembled into the global matrix, hence the memory access is improved, as illustrated in the lower part of Fig. 4.2.

This optimization maintains constant the computational time associated with

the matrix-pattern creation algorithm regardless the number of integration points used, see Fig. 4.3. This fact significantly alleviates one of the main disadvantages of meshfree methods, namely the large number of quadrature points needed as compared to piecewise polynomial approximants. Furthermore, the granularity of the element-level approach is better suited for parallel computing, minimizing memory access and limiting data exchange. The pseudo-code for the procedure is summarized in Algorithm 1.

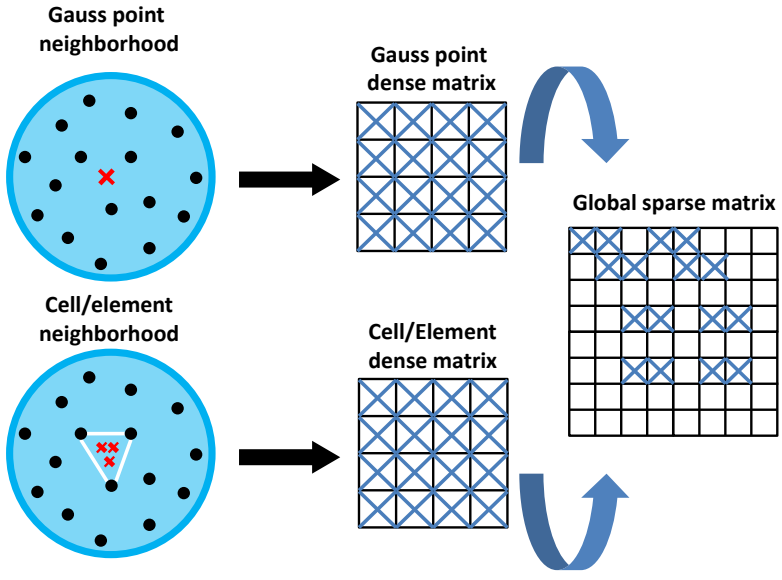


Figure 4.2: Filling algorithm transition from dense submatrices to global sparse matrix. Dense submatrices are generated from the neighborhood of an integration point (upper) or a cell/element (lower). Cell/element framework improves memory management since dense submatrix condenses information coming from several integration points.

4.1.2 Compressed meshfree basis functions storage

We present in paper Appendix C an optimization to overcome a possible memory limitation coming from the basis functions storage. Such memory difficulties arise

Algorithm 1 Pseudo-code for cell/element approach.

1: Compute adjacency lists for nodes $\mathcal{N}^X_{\boldsymbol{x}_a}$ and process cell/element lists $\mathcal{N}_{el} = \bigcup_{a \in \mathcal{T}_{el}} \mathcal{N}^X_{\boldsymbol{x}_a}$.
2: Compute shape functions p_a.
3: Sparse matrix structure ia, ja.
4: Fill sparse matrix structure an: cell/element loop based.

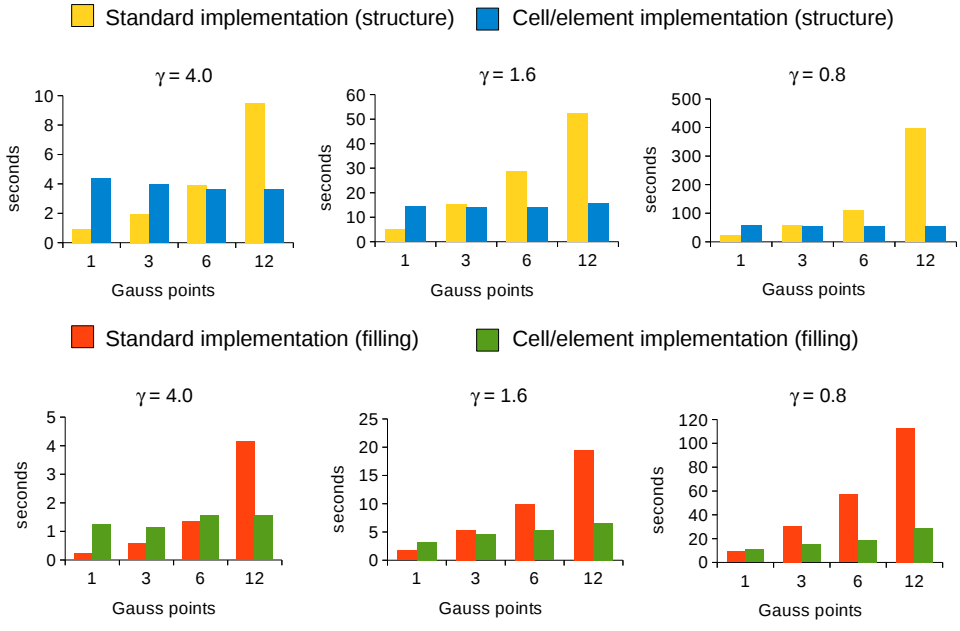


Figure 4.3: Structure creation (up) and filling (down) computational time vs Gauss points for increasing γ. Standard and new implementation are shown (left and right bars, respectively). DOF = 40,401.

when facing problems that require a repetitive usage of the values of the basis functions and their first and second derivatives. For example, when solving evolutions in time, incremental loading processes and problems that require non-linear iterative solvers. In these cases the values of the basis functions are used in a continuous way

and it becomes interesting not to compute them every time but to store and use them whenever needed. While this is optimal from a computational time perspective, the memory can become a bottleneck if we take into account the meshfree framework. In a meshfree method a standard implementation implies the storage of values, gradients and Hessians of the basis functions for every node in the neighborhood of every integration point, so the memory can become a major obstacle. To solve this problem we propose a partial storage of the basis functions based on proper structures that enable a quick recovery of their values.

For a general meshfree method, and considering the Galerkin approximation of a fourth-order PDE, the *full storage* of the basis functions requires $M_{FS} = L \cdot \bar{n} \cdot (1 + d + d(d+1)/2) = L \cdot \bar{n} \cdot \left(1 + \frac{3}{2}d + \frac{1}{2}d^2\right)$ doubles. In this equation, L is the total number of quadrature points, $\bar{n}$ is the mean cardinality of the primal lists, 1 accounts for the basis functions themselves, d for their gradients, and $d(d+1)/2$ for the Hessian, which is a symmetric matrix. In a for fourth-order PDE we typically have $\bar{n} \approx 65$ in 2D and $\bar{n} \approx 380$ in 3D, where a four cells/elements radius has been considered. In consequence, the memory requirements rapidly become unaffordable.

Focusing on LME approximants, we recall that the basis functions are obtained by means of a nonlinear optimization problem at each evaluation point with d unknowns, where d is the spatial dimension. This optimization problem yields the Lagrange multiplier associated with first-order consistency conditions. Once the Lagrange multiplier is known, an explicit expression for the basis functions, its gradient and its Hessian is explicit (see our paper Appendix C). Even if the nonlinear optimization problem is relatively easy to solve by Newton's method, it accounts for a significant part of the basis function evaluation time.

A straight-forward alternative to the *full storage* method would be to simply store the d reals in the Lagrange multiplier at each quadrature point. Analyzing in detail the structure of the explicit formulae for the basis functions and derivatives,

Table 4.1: Quantification and comparison of memory usage between the methods of full and optimal or compressed storage of local maximum-entropy basis functions and their derivatives. Here, $\bar{n}$ is the mean cardinality of primal lists, L is the number of Gauss points and d is the spatial dimension.

	Full storage	Optimal storage
Memory	$M_{FS} = L \cdot \bar{n} \cdot \left(1 + \frac{3}{2}d + \frac{1}{2}d^2\right)$ $\bar{n} \gg (2+d)$	$M_{OS} = L \cdot \left(2 + \frac{7}{2}d + 2d^2 + \frac{1}{2}d^3\right)$ $M_{OS} \approx L \cdot (2+d) \cdot \left(1 + \frac{3}{2}d + \frac{1}{2}d^2\right)$
Comparison	$M_{FS}/M_{OS} \approx \bar{n}/(2+d) \gg 1$	

it is easy to identify a set of matrices and vectors whose size is independent on $\bar{n}$, and some of which involve summations over $\bar{n}$. Thus, storing these arrays saves significant computation time at a limited memory cost. As detailed in the paper, this simple observation suggests the *optimal or compressed storage*, which only involves $M_{OS} = L \cdot \left(2 + \frac{7}{2}d + 2d^2 + \frac{1}{2}d^3\right) \approx L \cdot (2+d) \cdot \left(1 + \frac{3}{2}d + \frac{1}{2}d^2\right)$ doubles. As the mean cardinality is in general much greater than the spatial dimension, i.e. $\bar{n} \gg (2+d)$, from the ratio $M_{FS}/M_{OS} = \bar{n}/(2+d)$ it is clear that the memory usage decreases significantly when the compressed storage technique is used, as can be observed in Table 4.1.

4.1.3 Meshfree parallel sparse matrices in PETSc

Here we give an insight into the main challenges found in the process of writing the parallel $C++$ classes. PETSc is a well developed code with many useful routines ready-to-go and creation and filling routines can become quite direct if we work in a FEM-like environment since we have a connectivity that allows for a fast calculation of the non-zero positions. When having an structured mesh, PETSc is very efficient and provides the user with specific structures to exchange information between processes. Unfortunately, meshfree methods and unstructured grids are not that developed. We will focus here in the ideas underlying the assembly of the distributed

sparse matrix of the system in a meshfree method i.e. LME. Other routines, such as the ones leading to the RHS assembly, follow the same concepts.

Basically, PETSc distributes a matrix by partitioning sequentially its rows. Each process owns a certain domain corresponding to a sequential list of rows in the matrix. We assume the partitioning procedure has been applied in parallel e.g. by ParMetis. The objective of every thread is now to create and fill its local chunk on the global matrix. We start from a local collection of nodes and we proceed generating the integration points corresponding to this local node set. PETSc requires a careful preallocation of the memory and asks for a sparse CSR storage lists, which are local to the process, to maintain the scalability. Since some of the primal lists of the local Gauss points or cell/elements will be including neighbors from more than one nodal partition, it is required an information exchange between processes. We comment here on two different possibilities, both of them coded and tested throughout our simulations. In the former we focus on the minimal computational cost per process and rely on heavier information exchange, while in the latter we minimize the information exchange at the expense of increasing the computing load on every process.

The first routine takes advantage of PETSc internal exchange of information, which is direct, automatic and transparent to the user. Every thread starts by generating a connectivity using only its local integration points or cells/elements. That means that some of the dual lists created are not complete and will lack a number of Gauss points, which lay outside the partition. In the same way, some primal lists will have nodes corresponding to other processes. This generates an exchange of information, particularly near the boundaries. In this first procedure, every partition sends to the rest the part of the dual lists that they miss. In this way every thread is able to work with extended dual lists that generate right and complete non-zero positions for every row. The allocation is then achieved flawlessly

and PETSc runs with maximum efficiency. After allocation, the filling process is straightforward. PETSc matrices are designed in such a way that any position can be filled from any process, no matter if the row belongs or not to the process creating the value. Nevertheless, most of values should be created in the assembly partition to maintain scalability, which naturally happens if the repartitioning was correct.

Unfortunally, in our experience we found a problem with this set of routines when the filling algorithms start. If the communication band is not too large, as expected after a good repartitioning, PETSc routines make the assembly efficient and easy to implement at code level. However, PETSc creates a number of structures before starting this internal communication. This structures are perfect for mesh-based codes, where usually just one or two layers of neighbors define the exchange band, but they run into memory issues when this band becomes larger. In meshfree methods and when facing 3D problems , where this number of neighbors can increase considerably, the first routine fails. For this reason a second approach was developed. In this second approach we remove the information exchange in the filling process at the expense of increasing the computational load in every thread at the structure creation stage. We force the process not only to exchange information about the integration points lists but also to exchange the coordinates of the integration points themselves. In this way, every process expands its own integration point lists with that on the surrounding bands, and is able to create and fill its local structure by itself. The increase of computational cost is not important as long as the repartitioning is reasonable (the band is noticeably smaller than the own domain), and the gain is critical since the memory bottleneck is overcome. These routines and others can be checked in Appendix D.

4.2 A brief code overview

We present here the basics for our $C++$ code. The computation of LME approximants values, gradients and Hessians is performed by a $C++$ self-made library with several classes using ANN searcher [109] for neighbor searching and QHULL [110] and Metis [106] for different purposes e.g. triangulation, quadrature. This is called by a main code that uses the software PETSc as parallel core. PETSc main purpose in our code is the generation of data structures (matrices and vectors) in sparse mode (CSR) in a distributed way using MPI. PETSc provides an understandable way of declaring and filling this variables in an efficient pattern. Moreover, the library is appropriately extended with a number of interfaces to other useful external packages. In particular, we use it to easily call for Metis/Parmetis for parallel partitioning and reordering, and distributed memory solvers such as SuperLu or Mumps, which complete the set of direct and iterative solvers already embedded in PETSc. Partitioning is a fundamental issue in our code. Since meshfree methods in general, and LME approximants in particular (more as the aspect ratio parameter γ decreases), generate a wider communication band between partitions, a non-optimized distribution can fatally impact the overall performance. Reordering is also important when using direct solvers as LU to reduce the fill-in. The objective of the code is to generate an efficient and scalable program cand exploit the possibilities of $C++$ language to end up with a code as flexible as possible. In the following the basic structure of our code, conceptually illustrated in Fig. 4.4, is described.

From a technical point of view, the so called LME-Petsc-PDE library enables a particular problem main.cpp in $C++$ code to work with the main class *BasicPDE.cpp*, the reordering and the partitioning class *Reorder.cpp* and a user customized file called *operations.cpp*. The BasicPDE class creates a superstructure that is able to control the creation of several PETSc objects (matrices, vectors) and en-

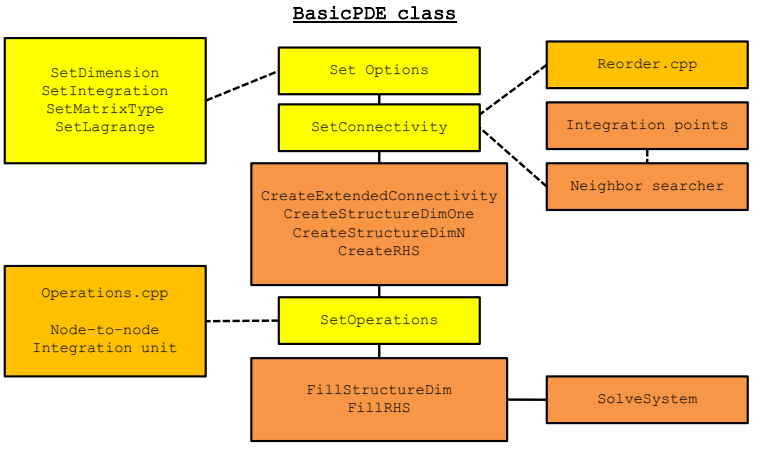


Figure 4.4: Concept structure of BasicPDE $C++$ class. Routines in yellow are information input methods. Continuous lines signal the general flux and the discontinuous ones mark subordinated methods, files or classes.

vironments (solvers). This over-control is necessary to ensure a correct matching of the parallelization applied to different objects that have to work together, as well as optimizing the usage of information throughout the whole numerical simulation. The *Reorder.cpp* class is a quite independent part of the code. It is meant to receive an arbitrary partition of nodes, manage the reordering and partitioning through ParMetis and reset each process with the new information. *Operations.cpp* is a file containing a set of filling operations. It can be particularized for a problem or meant to become a unique operations library that can grow with every project included. The BasicPDE class is prepared to work with two kind of operations. In a node-to-node operation, just a $(dim \times dim)$ matrix filling function has to be defined. Internally, the class uses this function to fill every node-to-node contribution to every local integration point or cell/element dense matrix (see Section 4.1.1). Simple mass or stiffness operations

can be defined as this. For operations involving more complex interactions between the nodes in the primal list, the BasicPDE class can also work directly with functions that define the local integration point or cell/element $(dim \times N) \times (dim \times N)$ dense matrix itself.

Regarding the procedural point of view (illustrated in Fig. 4.5), a general problem code starts reading an unstructured grid of points and possibly an initial phase-field. Each processor is then given an arbitrary partition of the grid (usually sequential) and works out a connectivity using a simple delaunay triangulation based in QHULL[110]. This information is then passed through the PETSc interface to ParMetis, which is able to repartition and reorder the grid in parallel. The output involves a series of mappings that can be used to migrate all necessary information, such as the phase-field associated to each node. With the new partitions, the integration elements are also assigned to the different processes and each process places the quadrature points over its partition. Never leaving the parallel scope, each thread computes the neighbor lists using the ANN searcher with CreateConnectivity(). Then follows the creation of the general matrix, the RHS and the solution vector that can be used afterwards in the PETSc solver integrated environment. The matrix structure creation starts with the routine CreateExtendedConnectivity(), which takes the neighbors list of the associated integration points and analyzes the partitions influenced by the list. It separates the nodes acting on each process and generate a bundle with the required integration points for each external process needing this information. Then every process sends and receives an extension for the Gauss point list, creating the extended lists described in Section 4.1.3. This will now generate a preliminary one-dimensional complete CSR local memory allocation with the routine CreateStructureDimOne(). The method CreateStructureDimN() extends easily these allocation to more dimensions if necessary. CreateRHS() completes the memory allocation by creating a distributed dense vector. Once the structure are set,

FillStructure() and FillRHS() use the calculated basis functions and the file *Operations.cpp* to compute the corresponding values and fill the non-zero positions. This routine generates the local matrix for each Gauss point owned by the process, and PETSc is able to send the exterior positions to their corresponding processes automatically. The class has similar routines to fill the RHS and generate the solution vector and easily applies boundary conditions and Lagrange multipliers in a parallelized fashion. Finally, these distributed objects can be used in a parallel solver, either a PETSc provided one, either one within an external package. Besides the classical Gauss point-wise strategy, the new cell/element approach shown in Section 4.1.1 has been also implemented and can be chosen as an option. The same goes for the zero-communication filling strategy presented in Section 4.1.3.

The rest of methods in the class correspond to an effort to enforce the flexibility of the code, motivated by the will of creating a user friendly software at the lab which can be easily adapted to solve a wide range of problems using LME or other meshfree methods in a parallel framework. The way it is organized, the user can just define a number of parameters with SetOptions(), e.g. the dimension of the problem, the type of the matrix or the size of the matrix, and give the grid as an input to get the PETSc objects created, allocated and ready to use. The filling of the matrix is also simplified for a user not familiar with the sparse assembly process since only the *Operations.cpp* file has to be modified with the particular problem operations. This flexibility has been tested through its application in different collaboration works within the lab, that we present in Chapter 5. We have conducted three periods in the Marenostrum III supercomputing facility [111] at Barcelona, in the context of the general project *Phase field modeling of biomembrane dynamics and crack propagation.* The assigned hours add up to more than 1200 Kh and we have successfully run grids of hundreds of thousands degrees of freedom using up to 1600 cores. Marenostrum III supercomputer runs at a peak performance of 1,1 Petaflops and holds 100.8 TB

of main memory. Node specifications: Homogeneous nodes 3,056 compute nodes 2x Intel SandyBridge-EP E5-2670/1600 20M 8-core at 2.6 GHz 8x4GB DDR3-1600 DIMMS (2GB/core) Heterogeneous Nodes 42 heterogeneous compute nodes 2x Intel SandyBridge-EP E5-2670/1600 20M 8-core at 2.6 GHz 2x Xeon Phi 5110 P 8x8GB DDR3-1600 DIMMS (4GB/core) 2 PB of disk storages Interconnection networks: Infiniband FDR10 Gigabit Ethernet Operating System: Linux - SuSe Distribution.

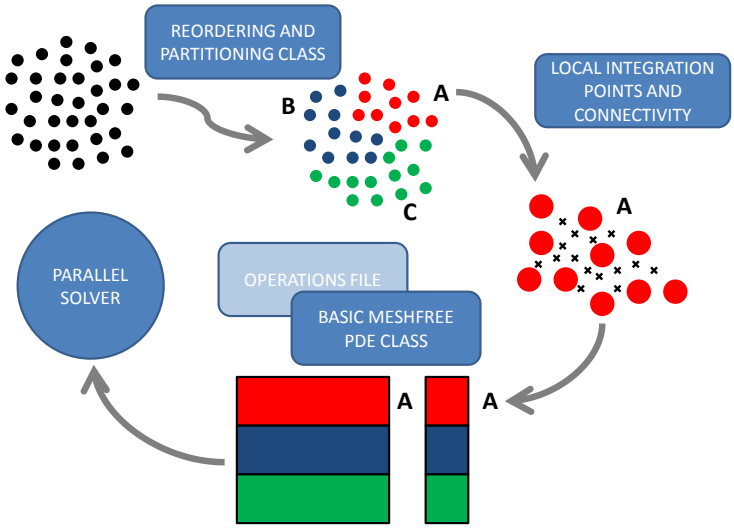


Figure 4.5: Generic main code structure. A unstructured grid of nodes (and possibly additional fields, such as a phase-field variable describing the initial state) is loaded and reordered and partitioned in parallel by the reorder class based on ParMetis. Every process advances generating its own integration points and connectivity. The sequential chunk of rows in the matrix (and RHS) for each one is then created and filled independently thanks to the BasicPDE class based on PETSc. The solver, integrated in PETSc or taken from an external package, is the final step of a common iteration.

Chapter 5

Other applications

Due to the flexible structure of the code, we have been able to apply and extend it to other problems sharing common numerical background. The author of this thesis has been involved in these projects by leading the computational implementation. We refer to the corresponding papers for more information.

5.1 Stabilization of Stokes equations with LME approximants

In anticipation to an incompressibility constraint in biomembrane models, an equal approximation method for velocities and pressure in the context of the incompressible Stokes equations with LME approximation schemes is studied in our paper [112]. The performance of the method is illustrated with classical benchmark tests, showing how the Stokes equations discretized with LME approximants can be effectively stabilized.

The Stokes problem can be formulated:

$$-\nu\triangle u + \nabla p = f \quad en\ \Omega, \tag{5.1}$$

$$\nabla \cdot u = 0 \quad en\ \Omega, \tag{5.2}$$

$$u = u_d \quad en\ \Gamma_d, \tag{5.3}$$

where u is the velocity, p the pressure, f the vector of body forces, ν the kinematic viscosity, u_d the Dirichlet boundary conditions and $\Omega \subset \mathbb{R}^d$.

Let be $V = H_0^1$ and $Q = L^2/\mathbb{R}$ the velocity and pressure spaces, respectively. Then, the weak formulation of the problem to find $u \subset V$ and $p \subset Q$ such that:

$$-\nu(\nabla\Delta u, \nabla\Delta v) - (p, \nabla\Delta \cdot v) = \langle f, v\rangle \ \ \forall v \in V, \tag{5.4}$$

$$(q, \nabla \cdot u) = 0 \ \ \forall q \in Q. \tag{5.5}$$

Defining,

$$\begin{aligned} a : V \times V \to R \quad & a(u,v) = \nu(\nabla u, \nabla v), \\ b : Q \times V \to R \quad & b(q,v) = (q, \nabla \cdot v), \\ l : V \to R \quad & l(v) = \langle f, v\rangle. \end{aligned} \tag{5.6}$$

The problem can be written as follows,

$$a(u,v) - b(p,v) = l(v) \ \ \forall v \in V, \tag{5.7}$$

$$b(q,u) = 0 \ \ \forall q \in Q. \tag{5.8}$$

Existence and uniqueness of solutions of this widely studied problem relies on the

the Ladyzhenskaya-Babuska-Brezzi (LBB) condition,

$$\inf_{q \in Q} \sup_{v \in V} \frac{b(q,v)}{\|q\|_Q \|v\|_V} \geq K_b > 0. \tag{5.9}$$

This condition holds true if $b(q,v) = (q, \nabla \cdot v), \forall q \in Q = L_2(\Omega)/\mathbb{R}$ and $\forall v \in V = H_0^1(\Omega)^d$ (Ladyzhenskaya), and therefore the continuous problem is well-posed. Unfortunately, when the equations are discretized with finite-dimensional spaces, the LBB condition can fail. In particular, using the same discretization space for both pressure and velocity results in a loss of stability, which is the cardinal issue of numerical methods for solving the Stokes problem.

The main strategies to deal with this obstacle are mixed formulations and the stabilization of equal approximation methods for the Stokes equations. The mixed formulations tackle the problem by seeking admissible pairs of spaces that fulfill the inf-sup condition at the discrete level. Stabilization techniques use a discretization based on a single space for both pressure and velocity, and add terms to the original weak form to give coercivity to the resulting matrix. We are interested in the coupled problem posed by the bending model of biomembranes and the Stokes flow in which they are immersed. LME approximants present nice characteristics to solve the phase-field governing the structure behavior and for simplicity we find convenient to use the same discretization space for the fluid problem. Stabilization techniques have undergone a large and satisfactory development in the FEM context [113]. We develop a LME stabilization method inspired in FEM stabilization ideas. Because of the differences between FEM and LME, the application of the FEM based methods is not direct and redefinition of parameters is needed. The discretization of the Stokes problem leads to the following system:

$$\begin{bmatrix} K & -D^T \\ D & 0 \end{bmatrix} \begin{bmatrix} U \\ P \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix}, \tag{5.10}$$

where K comes from the Laplacian velocity and it is definite positive, while D corresponds to the pressure terms and introduces the instability to the total matrix. To stabilize the system we need to add an extra term to the weak form [113]:

$$\int_\Omega \tau \mathcal{P}(w,q)\ R\ (u,p) d\Omega, \tag{5.11}$$

where $R\ (u,p)$ is the residual of the strong form of the problem (which ensures the consistency of the new weak form), τ is a parameter which controls the amount of the stabilization to be applied and $\mathcal{P}(w,q)$ is a partition of the differential operator. Different choices of this partition lead to different stabilization methods. To summarize the effect of the stabilization methods and to provide an integrated way of implementation in the code, we write the stabilization term as [114]:

$$\int_\Omega \tau_1(-\alpha\nu\Delta w + \beta_1 \nabla q)(-\nu\Delta u + \nabla p - f), \tag{5.12}$$

where alpha takes the values 1, 0 and -1, and beta 1 and -1. The different combinations of the values enable the user to switch from one stabilization method to another while maintaining the same framework. structure.

Since in LME we work with a set of points instead of a mesh, some redefinition of the parameter has to be worked out. In FEM, guidelines for the stabilization parameter τ are given in terms of the nodal spacing. In meshfree methods, this nodal spacing is usually interpreted as a measure of the effective support of the basis functions. Here we propose a Gauss point-wise parameter based in this idea, which is easy to implement and shows excellent performance.

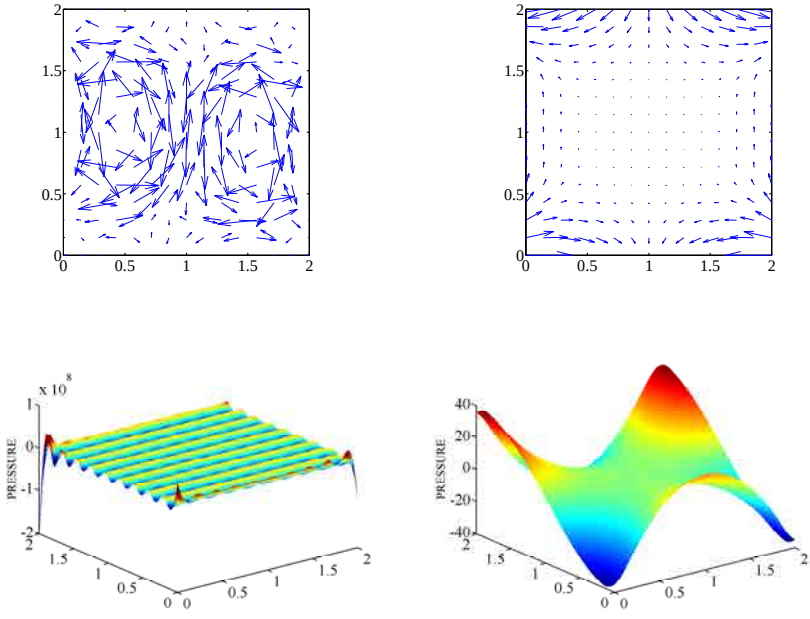


Figure 5.1: Velocity field for Colliding Flow: (top-left) Without stabilization. (top-right) Stabilized. Pressure field for Colliding Flow: (bottom-left) Without stabilization. (bottom-right) Stabilized. γ=1.0, DOF=1250.

$$\tau_1 = \frac{C}{\nu}\bar{\rho}^2, \tag{5.13}$$

where $\bar{\rho}$ stands for a weighted sort of mean applied upon the effective support sizes of the neighbors list $1, 2..., N$.

We select and apply the GLS-LME stabilization technique ($\alpha = 1$ and $\beta_1 = 1$) to the classical Poiseuille and Colliding flows benchmark tests for the Stokes problem. Since these tests have analytical solution we can accurately compare the results of the simulation. The velocity field of the Colliding flow problem is illustrated in the Fig. 5.1 without the application of stabilizing method and after stabilization. The analytical velocity field is recovered after the stabilization.

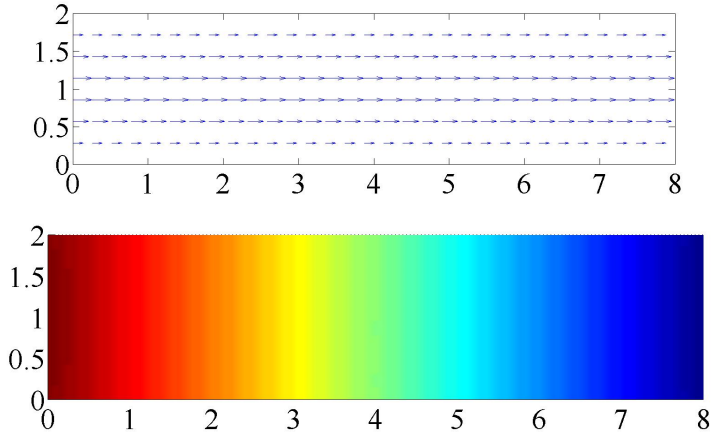


Figure 5.2: Poiseuille: stabilized velocity fields (top) and pressure (bottom) stabilized, γ=1.0, uniform set DOF=1152

The same behavior can be seen in the Fig. 5.1 where we plot the pressure for the solution without stabilization. Disproportionate values of pressure and oscillations are observed. This anomalous behavior disappears after the stabilization and the obtained solution recovers the smoothness, matching the analytical field. This fact is reflected in detail with the recovery of the optimal rate after the stabilization in the convergence charts for L2 norm. The results obtained for Poiseuille flow confirm the ones of the Colliding flow, showing identical behavior in its parameter dependence and convergence rates. The stabilized results are shown in Fig. 5.2.

Another set of simulations have been run over an unstructured grid in order to test the capability of the point-wise τ_1 parameter introduced before, with excellent results. This result enables the use of this fairly direct implementation parameter to manage adaptive processes which are needed in high demanding computations such as the ones in this thesis. Further information can be found in our paper [112].

5.2 A stabilized formulation for viscoplastic flow in metal forming

The FEM has been successfully applied to the simulation of metal forming [115, 116] and, thanks to the increasing computer power, it can provide excellent results with reasonable computational times. Thanks to its flexibility and robustness commercial FEM codes are a standard tool in the industry.

However, the main limitation of the FEM in this kind of application is that the quality of the results depends on the mesh. If a Lagrangian formulation is used the mesh moves with the material and, due to the high distortions, the numerical results loose their accuracy, unless remeshing-rezoning techniques are used. This step becomes very time consuming in 3D. Furthermore additional errors are introduced when the variables are mapped from the old mesh to the new one. Metal forming has been also studied with Eulerian and ALE formulations which involve drawbacks such as determining the geometry of the free surface of the flow in the former case and controlling the mesh motion in the latter. For these reasons, even if the FEM provides very good results in many applications, alternative techniques based on meshfree approximation schemes appear as an interesting alternative in the simulation of metal forming processes. Unfortunately, since meshfree methods are less mature than mesh-based methods, only few works applied these techniques to metal forming.

Since pioneer works on metal forming [117, 118] it is an accepted assumption to neglect elastic deformations and therefore treat the material as a non-newtonian viscoplastic fluid, in the so called flow formulation [119]. This aspect is discussed in detail in [120]. In the simulation of incompressible flows even if meshfree method are less sensitive to volumetric locking FEM it is still preferable to employ mixed pressure-velocity formulations. This poses some issues regarding the construction of a discretization that satisfies the inf-sup or LBB compatibility condition. While in the

FEM environment different type of shape functions, defined on the same elements, can be used for the velocity and the pressure, the problem becomes more complicated for meshfree methods. In the works based on the Natural Element Method, the approximants are used for the velocities and constant approximants directly defined on the Voronoi diagram are used for the pressure. However, even if the method performs well, some oscillation are still present. Here we propose a stabilized formulation based on the LME meshfree approximants. Since the same basis functions approximate the velocity and the pressure, we resort to stabilization to circumvent the LBB condition. As previously mentioned, a good approximation for metal forming problems is to treat the material as a non-newtonian incompressible viscoplastic fluid. Due to the analogy of the Stokes equations with a non-newtonian incompressible viscoplastic material, we extend to metal forming applications a stabilization approach proposed for fluid dynamics. Recently a family of consistent stabilization techniques for the FEM has been widely studied in the literature [113, 121, 122]. In [112] they have been also successfully applied to local maximum entropy schemes. In this work we propose a modification of the technique used in [121] that consists in penalizing the incompressibility equation with the gradient of the pressure. This approach recovers a strategy already proposed in [122].

In our work we validate the method with a classical metalforming benchmark (see Fig. 5.3), a cylindrical billet which is progressively flattened between two plates. The pressure and velocity in this example are simple and easy to analyize, and provide a proper frame to test the performance of our method, which provides excellent results. Further information can be found in our published paper [123].

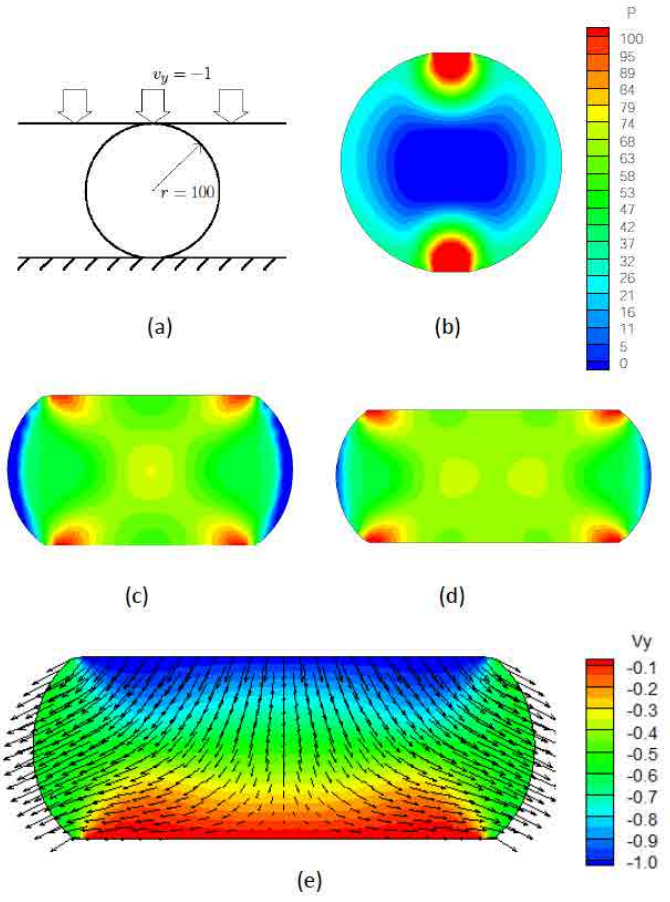


Figure 5.3: Cylindrical billet progressively flattened between two plates. (a) sketch of the geometry; (b-d) pressure at different time steps; (e) vertical velocity at the end of the process.(524 nodes and 100 time steps).

5.3 Computational evaluation of the flexoelectric effect in dielectric solids

Since its introduction by [124], flexoelectricity has been identified as an important electromechanical coupling in a wide variety of materials, including cellular membranes, liquid crystals, polymers, graphene, and piezoelectric and non-piezoelectric crystals. With the emergence of nanoscale fabrication and characterization, the interest in the flexoelectric effect has acquired a renewed vitality. See [125, 126, 127] for recent reviews. The flexoelectric effect describes the generation of an electric polarization induced by strain gradient:

$$P_i = \mu_{ijkl} \nabla_l \varepsilon_{jk}, \tag{5.14}$$

where $\boldsymbol{P}$ is the electric polarization, $\boldsymbol{\varepsilon}$ is the mechanical strain, and $\boldsymbol{\mu}$ is a fourth order flexoelectric tensor. Two features make flexoelectricity distinct from other electromechanical coupling mechanisms such as piezoelectricity. The first feature is its universality, due to the fact that a strain gradient can disrupt the inversion symmetry of the internal structure of a material, e.g. its crystalline structure, regardless of the lack of polarity of its undeformed configuration, hence inducing a polarization. As a result, the flexoelectric coefficients are generically non-zero for all dielectrics. The flexoelectric effect is prominent in materials with high dielectric constants such as ferroelectrics [128, 129, 130, 131]. Piezoelectricity is less universal since it can only appear in non-centrosymmetric crystals. The second distinguishing feature of flexoelectricity is its size-dependence, due to the scaling of strain gradients with structural size. Despite its universality, the flexoelectric effect is typically insignificant relative to piezoelectricity at macroscopic scales, and only manifests itself noticeably at the nanoscale. For this reason, the experimental observation of flexoelectricity is partic-

ularly difficult, which motivates the development of theoretical models to investigate this phenomenon.

Based on continuum models, the resulting fourth-order coupled system of partial differential equations (PDEs) has been approached with analytical solutions relying on simplifying assumptions and/or in very simple geometries. However, these assumptions may lead to under- or over-estimation of the flexoelectric effect. Furthermore, this effect can be more prominent in complex geometries favoring strain gradients, for which analytical solutions are not available. We are not aware of previous numerical approaches to solve the boundary value problems of flexoelectricity. The main difficulty is the fourth order nature of the PDEs of flexoelectricity, which demand at least C^1 continuous basis functions for a direct Galerkin method. Alternatively, mixed finite elements only requiring C^0 continuity and previously developed for strain gradient elasticity [132, 133, 134] could be applied to flexoelectricity. In consequence, we resort to local maximum-entropy (LME) meshfree approximants [135].

We summarize next a linear theory of flexoelectricity previously proposed in [136] and references therein. The electrical enthalpy density of a linear dielectric solid possessing piezoelectricity and flexoelectricity can be written as

$$\begin{aligned}\mathcal{H}(\varepsilon_{ij}, E_i, \varepsilon_{jk,l}, E_{i,j}) &= \frac{1}{2}\mathbb{C}_{ijkl}\varepsilon_{ij}\varepsilon_{kl} - e_{ikl}E_i\varepsilon_{kl} + f_{ijkl}E_i\varepsilon_{jk,l} \\ &+ d_{ijkl}E_{i,j}\varepsilon_{kl} - \frac{1}{2}\kappa_{ij}E_iE_j, \end{aligned} \tag{5.15}$$

where $E_i = -\phi_{,i}$ is the electric field, ϕ being the electric potential. The first energy term is the elastic potential, where $\mathbb{C}$ is the fourth-order tensor of elastic moduli. The piezoelectric coupling between the strain and electric field is through the second term with the third-order tensor of piezoelectricity $\mathbf{e}$. The last energy term is the electrostatic contribution, where $\boldsymbol{\kappa}$ is the second-order dielectric tensor. Here, our particular attention is on the third and fourth terms, which define the flexoelectric

behavior of the material. The term coupling the gradient of strain $\nabla\boldsymbol{\varepsilon}$ to the electric field is the direct flexoelectric coupling through the fourth-order tensor $\boldsymbol{f}$. Conversely, the gradient of electric field $\nabla\boldsymbol{E}$ is coupled to strain through the fourth-order tensor $\boldsymbol{d}$, introduced by [137] and termed converse flexoelectric effect. Using integration by parts, it has been shown that these flexoelectric energy terms can be expressed by only one term with one material tensor $\boldsymbol{\mu}$ [138]. The electrical enthalpy density in Eq. (5.15) is then rewritten as

$$\mathcal{H}(\varepsilon_{ij}, E_i, \varepsilon_{jk,l}) = \frac{1}{2}\mathbb{C}_{ijkl}\varepsilon_{ij}\varepsilon_{kl} - e_{ikl}E_i\varepsilon_{kl} - \mu_{ijkl}E_i\varepsilon_{jk,l} - \frac{1}{2}\kappa_{ij}E_iE_j, \tag{5.16}$$

where $\mu_{ijkl} = d_{iklj} - f_{ijkl}$. See [139] and [140] for recent accounts on the symmetry of the tensor of flexoelectric coefficients. The two forms of the enthalpy density in Eqs. (5.15) and (5.16) result in identical governing equations, and only the associated natural boundary conditions are different. We ignore strain gradient elasticity for simplicity and to isolate the effect of flexoelectricity, although as argued by [141], this may compromise the stability of the model in some regimes.

We present here two numerical examples. In the first, the electromechanical response of a cantilever beam due to flexoelectricity and the size-dependent elasticity behavior is studied. We evaluate this effect by defining the normalized Young's modulus Y' as:

$$Y' = \frac{\frac{1}{2}\int \boldsymbol{\varepsilon}_e : \mathbb{C} : \boldsymbol{\varepsilon}_e}{\frac{1}{2}\int \boldsymbol{\varepsilon}_f : \mathbb{C} : \boldsymbol{\varepsilon}_f}, \tag{5.17}$$

where $\boldsymbol{\varepsilon}_f$ and $\boldsymbol{\varepsilon}_e$ are the strains obtained from the simulations of the model with and without considering flexoelectricity, respectively. Fig. 5.4(a) presents Y' as a function of the normalized thickness h'. A similar size effect on the elastic behavior of ferroelectrics due to flexoelectricity has been reported [142]. The particular de-

formation of the beam due to flexoelectricity observed in the inset of Fig. 5.4(a) is examined in Fig. 5.4(b).

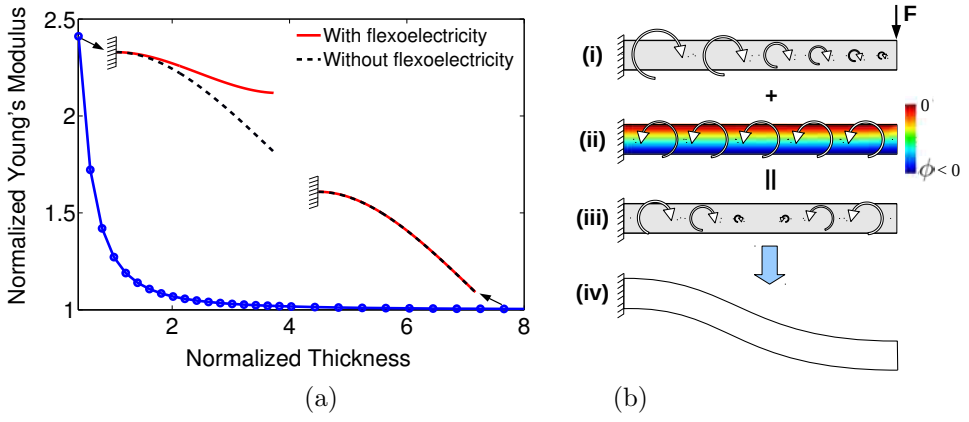


Figure 5.4: (a) Normalized Young's modulus Y' as a function of the normalized thickness h' for a non-piezoelectric material considering the closed circuit configuration. The insets show the deformation of the midline of the beam at two different length-scales, with and without flexoelectricity. The same mechanical load is applied for all the simulations in the figure. (b) Illustration of the deformation mechanism of the cantilever flexoelectric beam at small scales. The circular arrows show the moments induced by (i) the mechanical point load F and (ii) the converse flexoelectric effect. Due to the nearly uniform distribution of the electric potential along the beam, a uniform moment is induced due to the converse flexoelectric effect. (iii) The total moment distribution as the summation of the moments in (i) and (ii). The total moments lead to a peculiar deformation of the beam in (iv). The deformation is exaggerated for clarity.

The second example is a truncated pyramid. Under compression, it constitutes another setup to quantify the flexoelectric response of dielectric solids. The geometry of the truncated pyramid in plane strain and its boundary conditions are shown in Fig. 5.5(a). A force of magnitude F is applied uniformly at the top surface. The top and bottom surfaces have areas a_1 and a_2. Due to their different areas, the applied force generates different tractions at the top and bottom surfaces, resulting in a longitudinal strain gradient and thus generating a flexoelectric polarization. Here, we focus on a two-dimensional problem by considering a truncated triangle with unit

width.

Further information can be found in our published paper [143].

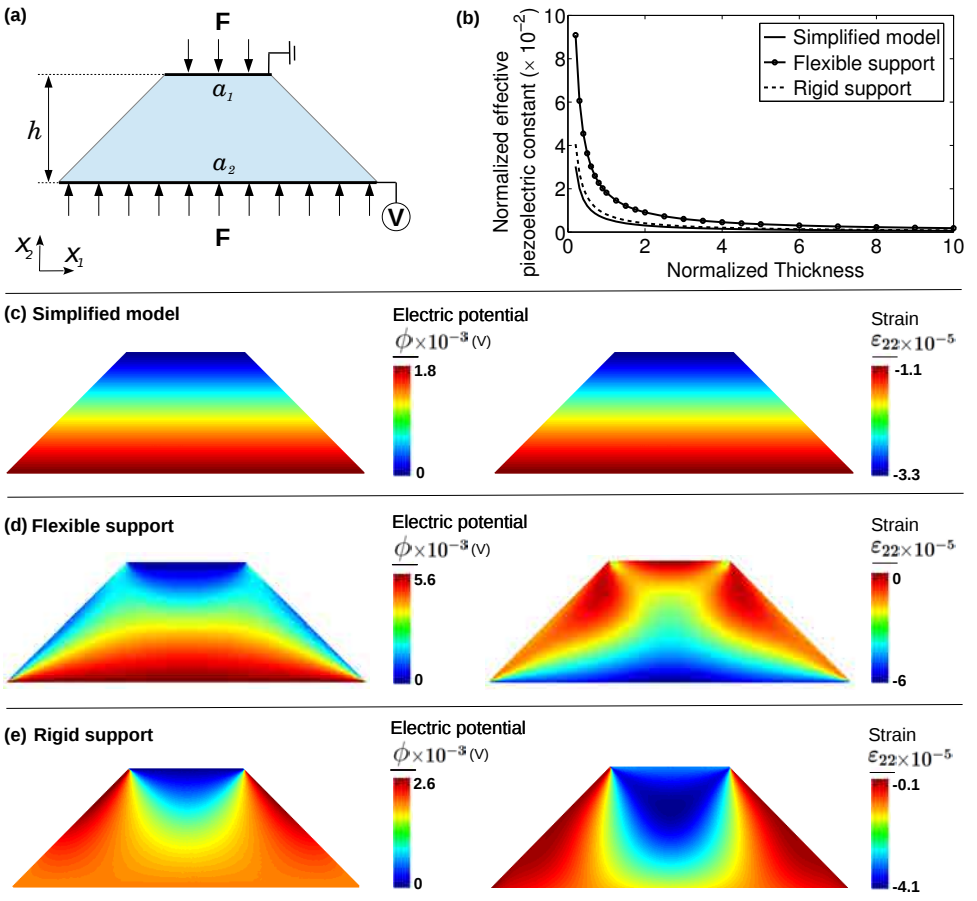


Figure 5.5: (a) Truncated pyramid in plane strain under the mechanical load F, uniformly distributed at the top surface. The top face has length a_1 and the bottom length a_2. The electric potential is fixed to zero at the top and is constant but unknown at the bottom. (b) Normalized effective piezoelectric constant e' as a function of the normalized thickness h' for a non-piezoelectric material, (c-e)(left) Distribution of the electric potential and (right) the strain ε_{22} obtained from the simplified analytical model (c) and the computational models with the flexible (d) and rigid (e) supports.

5.4 Fracture in brittle materials of anisotropic surface energy

Understanding how cracks choose their path of propagation, resulting in possibly complex crack patterns, is one of the outstanding problems in fracture mechanics. Patterns become more complex as we move from isotropic to anisotropic behaviors. The source of anisotropy may come from two different sources, namely the elastic anisotropy and the surface energy anisotropy. While the first source has often been addressed in literature, surface energy anisotropy remains less studied although it seems to bear a strong impact in crack patterns [144]. Its simulation, although challenging, may have interesting applications for industry since it is often produced during manufacturing processes (e.g. rolling polycrystalline materials, deep-drawing metals and alloys). Kinked and zig-zag patterns are experimentally observed in materials showing anisotropic surface energy (see Fig. 5.6).

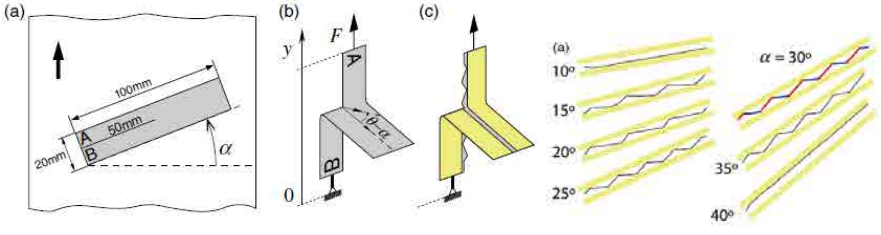


Figure 5.6: Zig-zag and kinked patterns in presence of anisotropic surface energy. Source: [144]

The current theoretical framework brittle fracture was initiated nearly a century ago by Griffith [145]. In this theory, crack propagation arises as a balance between the surface energy and the release of elastic energy; a crack will propagate in a direction given by the angle θ when the relation

$$G(\theta) \;=\; G_c \tag{5.18}$$

holds, where $G(\theta)$ is the elastic energy release rate for a crack along θ and G_c is the surface energy of the newly created crack faces.

Modeling of fracture mechanics involves the choice of a criterium for crack advance. Under quasi-static loading, several popular criteria have been appended to Griffith?s theory to determine the crack path, including (1) the principle of local symmetry, (2) the maximum energy release rate, (3) the minimum strain energy density and (4) the maximum hoop stress. While these criteria provide similar predictions for homogeneous isotropic materials (in fact, (1) and (2) coincide under certain conditions), they greatly differ when generalized to materials with anisotropic surface energy, in which the fracture toughness Gc./ is orientation dependent.

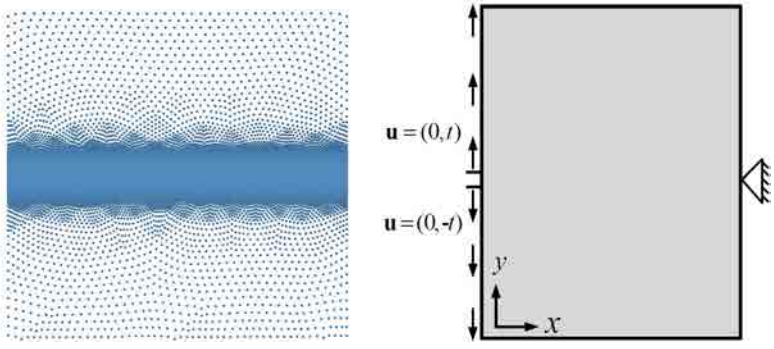


Figure 5.7: Adapted mesh for crack propagation (left). Geometrical description and boundary conditions for the benchmark problem (right).

Several have been proposed to model this issue, being the main ones the principle of local symmetry, the maximum energy release rate, the minimum strain energy density and the maximum hoop stress. These concepts give raise to very different crack patterns, which arises questions about its adequacy (check further details in [146, 147, 144]). In our work we resort to the variational approach, which considers

the propagation of the crack as driven by the minimization of a global energy. This energy takes into account the elastic energy and the crack surface energy. This has the advantage of removing the constraints of the classical Griffith theory, namely the existence of a pre-crack and a defined crack pattern. It can describe both short scale failure and macroscopic linear elasticity self-consistently [148]. We apply a phase-field model to easily introduce and substract energy additions in the potential. In this case, the phase-field v is set to $v = 1$ for the healthy material and $v = 0$ for the completely damaged (crack) zone. In the global functional, the elastic energy is thus modified by the phase-field and the phase-field modified by the displacement.

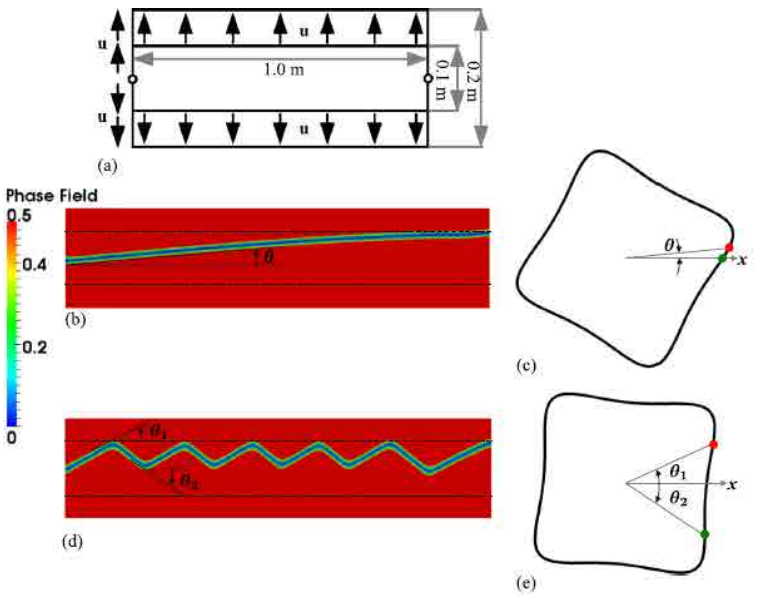


Figure 5.8: crack propagation guided along an allowed but high-energy direction (b, c) or along a forbidden direction (d, e). The red and green dots in (c) represent the initial and final crack orientation, while in (e) represent the two orientations of the sawtooth pattern.

We use the meshfree LME approximants due to the second order derivatives appearing in the terms accounting for anisotropy. The high adapted meshes we

use to capture the crack also call for the use of a meshfree method. We are running simulations in 2D with different boundary conditions and adapted grids, e.g. Fig. 5.7. We have successfully reproduced kinked cracks, as can be observed in the Fig. 5.8, and plan to extend this model to 3D. Further information in our published paper [149].

Chapter 6

Concluding remarks and future directions

6.1 Conclusions and future directions

We have demonstrated the suitability of the combination of phase-field modeling with meshfree methods to simulate a variety of high-order problems. We have focused in the simulation of biomembranes, where continuum mechanics description using parametrical sharp-interface models poses several obstacles that can be overcome with a diffuse interface approach. We have shown that the minimization of a variational scheme using a fourth-order phase-field model and highly adapted grids leads to validated equilibrium shapes within reasonable computational times. We have introduced the viscous fluid media with the Stokes equations and proposed a Lagrangian approach that considers the phase-field as a material property of the fluid. The competition of the dissipation potential and the bending elastic energy of the vesicle govern the dynamics and allow for simulations showing complex patterns with large deformations that the method is able to handle. We have extended the model with a term accounting for a Lennard-Jones adhesion potential and explored the role of kinetics in the morphogenesis of membrane structures. Meshfree phase-

field models, with high computational costs, demand for high adapted grids, adaptive time stepping and scalable parallel codes. We have presented a number of contributions in the implementation of meshfree methods in a high-performance computing environment. We showed how the creation and filling of sparse matrices can become the bottleneck in large scale simulations, and presented algorithms to improve its performance. The new algorithms involve the coarsening of the neighboring lists, minimizing the dependence of computational time with the number of quadrature points. We also presented a new strategy to efficiently store the basis functions and their derivatives in problems where they are repeatedly required, removing memory bottlenecks. The problems we solve within this work benefit from this optimizations and have run successfully on a supercomputing facility. We developed a flexible code in $C++$ using state-of-the-art supercomputing packages that permitted to tackle similar problems in a very direct way. We introduced the results obtained with the code in fluid mechanics, metal forming, flexoelectricity and fracture in brittle materials.

Regarding the future research lines, it is clear that the applications of the proposed methods can be extended to study many other biological processes. Cell motility, kinetics effects in combination with adhesion and the analysis of specific structures like organelle in living cells are just some of them. In the technical part, the code works in 3D but a further effort is to be done to become even more efficient and afford higher accuracies in tridimensional problems with evolutions which require a high number of time steps. In the phase-field model of biomembranes, for example, the need for an implicit method due to the stiffness of the system poses numerical obstacles when calculating lengthy and time consuming Hessians of the functional. Optimizations in this line are being carried on and should enable the simulation in 3D in the near future of the kind of problems we have already presented in 2D and axisymmetric 3D.

6.2 Publications

2014 B. Li, C. Peco, D. Millán, I. Arias and M. Arroyo. **Phase-field modeling and simulation of fracture in brittle materials with strongly anisotropic surface energy** International Journal for Numerical Methods in Engineering, DOI:10.1002/nme.4726, June 2014.

2014 C. Peco, D. Millán, and M. Arroyo. **Simulation of adhesion and confinement effects in biological structures with adaptive Lagrangian phase-field models.** In preparation.

2014 A. Abdollahi, C. Peco, D. Millán, I.Arias and M. Arroyo, **Computational evaluation of the flexoelectric effect in dielectric solids.** Journal of Applied Physics, 116(9):093502, 2014.

2013 C. Peco, D. Millán, A. Rosolen, and M. Arroyo. **Efficient implementation of meshfree Galerkin methods for large-scale problems with an emphasis on maximum entropy approximants.** Submitted to Computers and Structures.

2013 F. Greco, L. Filice, C. Peco, M. Arroyo . **A stabilized formulation with maximum entropy meshfree approximants for viscoplastic flow simulation in metal forming.** International Journal of Material Forming, DOI: 10.1007/s12289-014-1167-x, 2013.

2013 C. Peco, A. Rosolen, and M. Arroyo. **Estabilización de las ecuaciones de stokes con aproximantes locales de máxima entropía.** Submitted to RIMNI, 2013.

2013 A. Rosolen, C. Peco, and M. Arroyo. **An adaptive meshfree method for phase-field models of biomembranes. part i: Approximation with**

maximum-entropy basis functions. Journal of Computational Physics, 249(0):303–319, 2013.

2013 C. Peco, A. Rosolen, and M. Arroyo. **An adaptive meshfree method for phase-field models of biomembranes. part ii: A Lagrangian approach for membranes in viscous fluids.** Journal of Computational Physics, 249(0):320–336, 2013.

Bibliography

[1] R.F. Sekerka. Morphology: from sharp interface to phase field models. *Journal of Crystal Growth*, 264:530–540, 2004.

[2] I. Steinbach. Phase-field models in materials science. *Modelling and Simulation in Materials Science and Engineering*, 17:73001, 2009.

[3] G.A. Francfort and J.-J. Marigo. Revisiting brittle fracture as an energy minimization problem. *Journal of the Mechanics and Physics of Solids*, 46(8):1319–1342, 1998.

[4] A. Abdollahi and I. Arias. Phase-field modeling of the coupled microstructure and fracture evolution in ferroelectric single crystals. *Acta Materialia*, 59:4733–4746, 2011.

[5] M. Benes, V. Chalupecky, and K. Mikula. Geometrical image segmentation by the Allen-Cahn equation. *Applied Numerical Mathematics*, 51:187–205, 2004.

[6] D. Jacqmin. Calculation of Two-Phase Navier-Stokes Flows using Phase-Field Modeling. *Journal of Computational Physics*, 155(1):96–127, 1999.

[7] H. Gomez, L. Cueto-Felgueroso, and R. Juanes. Three-dimensional simulation of unstable gravity-driven infiltration of water into a porous medium. *Journal of Computational Physics*, 238:217–239, 2013.

[8] R. Dhote, H. Gomez, R. Melnik, and J. Zu. Isogeometric analysis of a dynamic thermo-mechanical phase-field model applied to shape memory alloys. *Computational Mechanics*, 53:1235–1250, 2014.

[9] G. Vilanova, I. Colominas, and H. Gomez. Capillary networks in tumor angiogenesis: From discrete endothelial cells to averaged phase-field descriptions via isogeometric analysis. *International Journal for Numerical Methods in Biomedical Engineering*, 29:1015–1037, 2013.

[10] Q. Du, C. Liu, and X. Wang. A phase field approach in the numerical study of the elastic bending energy for vesicle membranes. *Journal of Computational Physics*, 198:450–468, 2004.

[11] X. Wang. *Phase field models and simulations of vesicle bio-membranes*. PhD thesis, The Pennsylvania State University, 2005.

[12] J.S. Rowlinson. Translation of J. D. van der Waals' "the thermodynamik theory of capillarity under the hypothesis of a continuous variation of density". *Journal of Statistical Physics*, 20:197–200, 1979.

[13] L.D. Landau and I.M. Khalatikow. *The Selected Works of L.D. Landau*. Oxford: Pergamon, 1963.

[14] M Hillert. *A theory of nucleation for solid solutions, D.Sc Thesis*. Cambridge, MA: MIT Press, 1956.

[15] J.W. Cahn and J.E. Hillard. Free energy of a nonuniform system. i. interfacial free energy. *J. Chem. Phys.*, 28:258, 1958.

[16] J.S. Langer. Chance and matter, lectures on the theory of pattern formation. *Les Houches, session XLVI*, pages 692–711, 1987.

[17] F. Feng and W.S. Klug. Finite element modeling of lipid bilayer membranes. *Journal of Computational Physics*, 220(1):394–408, 2006.

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

[19] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005.

[20] H. Gomez, V.M. Calo, Y. Bazilevs, and T.J.R. Hughes. Isogeometric analysis of the Cahn-Hilliard phase-field model. *Computer Methods in Applied Mechanics and Engineering*, 197(49–50):4333–4352, 2008.

[21] M.J. Borden, T.J.R. Hughes, C.M. Landis, and C.V. Verhoosel. A higher-order phase-field model for brittle fracture: Formulation and analysis within the isogeometric analysis framework. *Computer Methods in Applied Mechanics and Engineering*, 273:100–118, 2014.

[22] A. Rosolen and M. Arroyo. Blending isogeometric analysis and maximum entropy meshfree approximants. *Comput. Methods. Appl. Mech. Eng.*, 264:95 – 107, 2013.

[23] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering*, 139(1–4):3–47, 1996.

[24] Vinh Phu Nguyen, Timon Rabczuk, Stéphane Bordas, and Marc Duflot. Meshless methods: A review and computer implementation aspects. *Mathematics and Computers in Simulation*, 79(3):763–813, December 2008.

[25] P. Lancaster and K. Salkauskas. Surfaces Generated by Moving Least Squares Methods. *Mathematics of Computation*, 37(155):141–158, 1981.

[26] J.J. Monaghan. An introduction to SPH. *Computer Physics Communications*, 48:89–96, 1988.

[27] W.K. Liu, S. Jun, and Y.F. Zhang. Reproducing kernel particle methods. *International Journal for Numerical Methods in Fluids*, 20:1081–1106, 1995.

[28] J.M. Melenk and I. Babuška. The partition of unity finite element method : Basic theory and applications. *Computer Methods in Applied Mechanics and Engineering*, 139(1–4):289–314, 1996.

[29] T. Belytschko, Y.Y. Lu, and L. Gu. Element free Galerkin methods. *International Journal for Numerical Methods in Engineering*, 37:229–256, 1994.

[30] J.S. Chen, C. Pan, C.T. Wu, and W.K. Liu. Reproducing kernel particle methods for large deformation analysis of nonlinear structures. *Computer Methods in Applied Mechanics and Engineering*, 139:195–227, 1996.

[31] J.S. Chen, C. Pan, C.M.O.L. Rogue, and H.P. Wang. A lagrangian reproducing kernel particle method for metal forming analysis. *Computational Mechanics*, 22:289–307, 1998.

[32] U.H. Combe and C. Korn. An adaptive approach with the element-free-galerkin method. *Computer Methods in Applied Mechanics and Engineering*, 162:203–222, 1998.

[33] C.A. Duarte and J.T. Oden. An h–p adaptive method using clouds. *Computer Methods in Applied Mechanics and Engineering*, 139:237–262, 1996.

[34] N. Sukumar. Construction of polygonal interpolants: a maximum entropy approach. *International Journal for Numerical Methods in Engineering*, 61(12):2159–2181, 2004.

[35] M. Arroyo and M. Ortiz. Local maximum-entropy approximation schemes: a seamless bridge between finite elements and meshfree methods. *International Journal for Numerical Methods in Engineering*, 65(13):2167–2202, 2006.

[36] F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: a new paradigm for thin-shell finite-element analysis. *International Journal for Numerical Methods in Engineering*, 47(12):2039–2072, 2000.

[37] C.J. Cyron, M. Arroyo, and M. Ortiz. Smooth, second order, non-negative meshfree approximants selected by maximum entropy. *International Journal for Numerical Methods in Engineering*, 79(13):1605–1632, 2009.

[38] F. Campelo. *Shapes in Cells. Dynamic instabilities, morphology, and curvature in biological membranes.* PhD thesis, Universitat de Barcelona, 2008.

[39] Q. Du, C. Liu, R. Ryham, and X. Wang. Energetic variational approaches in modeling vesicle and fluid interactions. *Physica D*, 238:923–930, 2009.

[40] H. Gomez, T.J.R. Hughes, X. Nogueira, and V.M. Calo. Isogeometric analysis of the isothermal Navier-Stokes-Korteweg equations. *Computer Methods in Applied Mechanics and Engineering*, 199(25–28):1828–1840, 2010.

[41] Q. Du and L. Zhu. Analysis of a mixed finite element method for a phase field bending elasticity model of vesicle membrane deformation. *Journal of Computational Mathematics*, 24(3):265–280, 2006.

[42] Q. Du and J. Zhang. Adaptive Finite Element Method for a Phase Field

Bending Elasticity Model of Vesicle Membrane Deformations. *SIAM J. Sci. Comput.*, 30(3):1634–1657, 2008.

[43] H.D. Ceniceros, R.L. Nós, and A.M. Roma. Three-dimensional, fully adaptive simulations of phase-field fluid models. *Journal of Computational Physics*, 229:6135–6155, 2010.

[44] S. Wise, J. Kim, and J. Lowengrub. Solving the regularized, strongly anisotropic Cahn–Hilliard equation by an adaptive nonlinear multigrid method. *Journal of Computational Physics*, 226(1):414–446, 2007.

[45] W.M. Feng, P. Yu, S.Y. Hu, Z.K. Liu, Q. Du, and L.Q. Chen. A Fourier Spectral Moving Mesh Method for the Cahn-Hilliard Equation with Elasticity. *Communications in Computational Physics*, 5(2–4):582–599, 2009.

[46] J.S. Lowengrub, A. Rätz, and A. Voigt. Phase-field modeling of the dynamics of multicomponent vesicles: Spinodal decomposition, coarsening, budding, and fission. *Physical Review E*, 79:31926, 2009.

[47] A. Voigt and T. Witkowski. Hybrid parallelization of an adaptive finite element code. *KYBERNETIKA*, 46:316–327, 2010.

[48] P. Yue, C. Zhou, J.J. Feng, C.F. Ollivier-Gooch, and H.H. Hu. Phase-field simulations of interfacial dynamics in viscoelastic fluids using finite elements with adaptive meshing. *Journal of Computational Physics*, 219(1):47–67, 2006.

[49] C. Zhou, P. Yue, J.J. Feng, C.F. Ollivier-Gooch, and H.H. Hu. 3D phase-field simulations of interfacial dynamics in Newtonian and viscoelastic fluids. *Journal of Computational Physics*, 229:498–511, 2010.

[50] L. Cueto-Felgueroso and J. Peraire. A time-adaptive finite volume method for

the Cahn-Hilliard and Kuramoto-Sivashinsky equations. *Journal of Computational Physics*, 227:9985–10017, 2008.

[51] R.J. Braun and B.T. Murray. Adaptive phase-field computations of dendritic crystal growth. *Journal of Crystal Growth*, 177:41–53, 1997.

[52] J. Rosam, P.K. Jimack, and A. Mullis. A fully implicit, fully adaptive time and space discretisation method for phase-field simulation of binary alloy solidification. *Journal of Computational Physics*, 225:1271–1287, 2007.

[53] W.M. Feng, P. Yu, S.Y. Hu, Z.K. Liu, Q. Du, and L.Q. Chen. Spectral implementation of an adaptive moving mesh method for phase-field equations. *Journal of Computational Physics*, 220(1):498–510, 2006.

[54] C.W. Lan and Y.C. Chang. Efficient adaptive phase field simulation of directional solidification of a binary alloy. *Journal of Crystal Growth*, 250:525–537, 2003.

[55] Z. Tan, K.M. Lim, and B.C. Khoo. An adaptive mesh redistribution method for the incompressible mixture flows using phase-field model. *Journal of Computational Physics*, 225:1137–1158, 2007.

[56] A. Rosolen, D. Millán, and M. Arroyo. On the optimum support size in meshfree methods: a variational adaptivity approach with maximum entropy approximants. *International Journal for Numerical Methods in Engineering*, 82(7):868–895, 2010.

[57] D. Millán, A. Rosolen, and M. Arroyo. Thin shell analysis from scattered points with maximum-entropy approximants. *International Journal for Numerical Methods in Engineering*, 85(6):723–751, 2011.

[58] A. Rosolen, D. Millán, and M. Arroyo. Second order convex maximum entropy approximants with applications to high order PDE. *International Journal for Numerical Methods in Engineering*, 94:150–182, 2013.

[59] M. Karlsson, K. Sott, M. Davidson, A. S. Cans, P. Linderholm, D. Chiu, and O. Orwar. Formation of geometrically complex lipid nanotube-vesicle networks of higher-order topologies. *Proceedings of the National Academy of Sciences*, 99(18):11573–11578, 2002.

[60] M. Edidin. Lipids on the frontier: a century of cell-membrane bilayers. *Nature Reviews Molecular Cell Biology*, 4(5):414–418, 2003.

[61] S. Semrau and T. Schmidt. Membrane heterogeneity – from lipid domains to curvature effects. *Soft Matter*, 5(17):3174–3186, 2009.

[62] H. Sprong, P. van der Sluijs, and G. van Meer. How proteins move lipids and lipids move proteins. *Nature Reviews Molecular Cell Biology*, 2(7):504–513, 2001.

[63] B. M. Discher, Y. Y. Won, D. S. Ege, J. C. M. Lee, F. S. Bates, D. E. Discher, and D. A. Hammer. Polymersomes: Tough vesicles made from diblock copolymers. *Science*, 284(5417):1143–1146, 1999.

[64] R. Dimova, S. Aranda, N. Bezlyepkina, V. Nikolov, K. A. Riske, and R. Lipowsky. A practical guide to giant vesicles. Probing the membrane nanoregime via optical microscopy. *Journal of Physics-Condensed Matter*, 18(28):S1151–S1176, 2006.

[65] E. Mabrouk, D. Cuvelier, L.L. Pontani, B. Xu, D. Levy, P. Keller, F. Brochard-Wyart, P. Nassoy, and Min-Hui Li. Formation and material properties of giant liquid crystal polymersomes. *Soft Matter*, 5(9):1870–1878, 2009.

[66] E. Reimhult, F. Höök, and B. Kasemo. Intact vesicle adsorption and supported biomembrane formation from vesicles in solution: influence of surface chemistry, vesicle size, temperature, and osmotic pressure. *Langmuir*, 19(5):1681–1691, 2003.

[67] M.˜Krausand W.˜Wintz, U. Seifert, and R. Lipowsky. Fluid vesicles in shear flow. *Physical Review Letters*, 77(17):3685–3688, 1996.

[68] L-T. Gao, X-Q. Feng, and H. Gao. A phase field method for simulating morphological evolution of vesicles in electric fields. *Journal of Computational Physics*, 228:4162–4181, 2009.

[69] T. Baumgart, S.T. Hess, and W.W. Webb. Imaging coexisting fluid domains in biomembrane models coupling curvature and line tension. *Nature*, 425:821–824, 2003.

[70] I.R. Cooke, K. Kremer, and M. Deserno. Tunable generic model for fluid bilayer membranes. *Physical Review E*, 72:011506, 2005.

[71] B.J. Reynwar, G. Illya, V.A. Harmandaris, M.M. Mueller, K. Kremer, and M. Deserno. Aggregation and vesiculation of membrane proteins by curvature-mediated interactions rid b-2958-2009 rid b-3677-2009. *Nature*, 447(7143):461–464, 2007.

[72] U. Seifert and S. A. Langer, S. A. Viscous modes of fluid bilayer membranes. *Europhysics Letters*, 23(1):71–76, 1993.

[73] U. Seifert. Configurations of fluid membranes and vesicles. *Advances in Physics*, 46(1):13–137, 1997.

[74] E. Evans and A. Yeung. Hidden dynamics in rapid changes of bilayer shape. *Chemistry and physics of lipids*, 73(1-2):39–56, 1994.

[75] P. Sens. Dynamics of nonequilibrium membrane bud formation. *Physical Review Letters*, 93(10):108103, 2004.

[76] P. Sens and M.S. Turner. Budded membrane microdomains as tension regulators. *Physical Review E*, 73(3):031918, 2006.

[77] P.B. Canham. The minimum energy of bending as a possible explanation of the biconcave shape of the human red blood cell. *Journal of Theoretical Biology*, 26(1):61–81, 1970.

[78] E. A. Evans. Bending resistance and chemically induced moments in membrane bilayers. *Biophys. J.*, 14:923–931, 1974.

[79] W. Helfrich. Elastic properties of lipid bilayers: theory and possible experiments. *Z. Naturforsch C*, 28(11):693–703, 1973.

[80] Udo Seifert and R Lipowsky. Adhesion and unbinding of vesicles. *Dynamical Phenomena at Surfaces, Interfaces, and Membranes*, pages 295–304, 1993.

[81] F. Campelo and A. Hernández-Machado. Dynamic model and stationary shapes of fluid vesicles. *The European Physical Journal E*, 20:37–45, 2006.

[82] G. Bellettini and L. Mugnai. Approximation of Helfrich's Functional via Diffuse Interfaces. *SIAM J. Math. Anal.*, 42:2402–2433, 2010.

[83] Q. Du. Phase field calculus, curvature-dependent energies, and vesicle membranes. *Philosophical Magazine*, 91:165–181, 2010.

[84] T. Biben, K. Kassner, and C. Misbah. Phase-field approach to three-dimensional vesicle dynamics. *Physical Review E*, 72(4):41921, October 2005.

[85] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Review*, 41(4):637–676, 1999.

[86] A.R. Conn, N.I.M. Gould, and P.L. Toint. A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM J. Numer. Anal.*, 28(2):545–572, 1991.

[87] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, USA, 1999.

[88] Q. Du, C. Liu, and X. Wang. Simulating the deformation of vesicle membranes under elastic bending energy in three dimensions. *Journal of Computational Physics*, 212:757–777, 2006.

[89] F. Campelo. Modeling morphological instabilities in lipid membranes with anchored amphiphilic polymers. *J. ˜Chem. ˜Biol.*, 2:65–80, 2009.

[90] A. Bonito, R.H. Nochetto, and S.M. Pauletti. Parametric FEM for geometric biomembranes. *Journal of Computational Physics*, 229(9):3171–3188, 2010.

[91] S.K. Veerapaneni, A. Rahimian, G. Biros, and D. Zorin. A fast algorithm for simulating vesicle flows in three dimensions. *Journal of Computational Physics*, 230:5610–5634, 2011.

[92] B. Li, F. Habbal, and M. Ortiz. Optimal Transportation Meshfree Approximation Schemes for Fluid and Plastic Flows. *International Journal for Numerical Methods in Engineering*, 83(12):1541–1579, 2010.

[93] J. Happel and H. Brenner. *Low Reynolds Number Hydrodynamics: with special applications to particulate media*. Martinus Nijhoff Publishers, 1983.

[94] H. Goldstein, C.P. Poole, and J.L. Safko. *Classical Mechanics*. Addison-Wesley, 2001.

[95] J.B. Fournier, N. Khalifat, N. Puff, and MI Angelova. Chemically triggered ejection of membrane tubules controlled by intermonolayer friction. *Physical Review Letters*, 102(1):18102, 2009.

[96] Nada Khalifat, Nicolas Puff, Stephanie Bonneau, Jean-Baptiste Fournier, and Miglena I Angelova. Membrane Deformation under Local pH Gradient: Mimicking Mitochondrial Cristae Dynamics. *Biophys. J.*, 95(10):4924–4933, 2008.

[97] J Sanborn, K Oglecka, R S Kraut, and A N Parikh. Transient pearling and vesiculation of membrane tubes under osmotic gradients. *Faraday Discussions*, DOI: 10.1039/C2FD20116J, 2013.

[98] Harvey T. McMahon and Jennifer L. Gallop. Membrane curvature and mechanisms of dynamic cell membrane remodelling. *Nature*, 438:590–596, 2005.

[99] Margarita Staykova, Marino Arroyo, Mohammad Rahimi, and Howard A. Stone. Confined bilayers passively regulate shape and stress. *Phys. Rev. Lett.*, 110:028101, 2013.

[100] Jérôme Solon, Jacques Pécréaux, Philippe Girard, Marie-Claude Fauré, Jacques Prost, and Patricia Bassereau. Negative tension induced by lipid uptake. *Phys. Rev. Lett.*, 97:098–103, 2006.

[101] Mohammad Rahimi, Margarita Staykova, Marino Arroyo, A.B. Subramaniam, and H.A. Stone. Dynamical remodelling of lipid bilayers upon cholesterol adsorption. *In preparation*, 2014.

[102] Jian Zhang, Sovan Das, and Qiang Du. A phase field model for vesicle–substrate adhesion. *Journal of Computational Physics*, 228(20):7837–7849, November 2009.

[103] J. E. Lennard-Jones. On the determination of molecular fields. *Proc. R. Soc. Lond. A*, 106:463–477, 1924.

[104] Udo Seifert and R Lipowsky. Adhesion of vesicles. *Physical Review A*, 42(8):4768–4771, 1990.

[105] Satish Balay, Jed Brown, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. Portable, extensible toolkit for scientific computation. `http://www.mcs.anl.gov/petsc`, 2013.

[106] George Karypis and Vipin Kumar. Metis-ParMetis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0.3. `http://www.cs.umn.edu/~metis`, 2009.

[107] Y. Saad. *Iterative Methods for Linear Systems*. PWS Publishing, Boston, 1996.

[108] V. Eijkhout. Distributed sparse data structures for linear algebra operations. Technical Report CS 92-169, Computer Science Department, University of Tennessee, 1992.

[109] M.D. Mount and S. Arya. A library for approximate nearest neighbor searching. `http://www.cs.umd.edu/~mount/ANN/`, 2010.

[110] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22:469–483, 1996.

[111] Barcelona supercomputing center (bsc). `http://www.bsc.es`, 2014.

[112] A. Rosolen C. Peco and M. Arroyo. Estabilización de las ecuaciones de stokes con aproximantes locales de máxima entropía. *Submitted to RIMNI*, 2014.

[113] R. Codina. Comparison of some finite element methods for solving the diffusion-convection-reaction equation. *Computer Methods in Applied Mechanics and Engineering*, 156:185–210, 1998.

[114] Teri Barth, Pavel Bochev, M A X Gunzburger, and John Shadid. A taxonomy of consistently stabilized finite element methods for the Stokes problem . *Siam J. Sci.Comp*, 25(5):1585–1607, 2004.

[115] S.I. Oh S. Kobayashi and T. Altan. *Metal forming and the finite-element method*. Oxford University Press, 1989.

[116] D. Peric and D. R. J. Owen. *Computational Modeling of Forming Processes*. Wiley and Sons, Ltd, 2004.

[117] O. C. Zienkiewicz. *Flow formulation for numerical solution of forming processes*. John Wiley, Chichester, 1984.

[118] O. C. Zienkiewicz and P. N. Godbole. Flow of plastic and visco-plastic solids with special reference to extrusion and forming processes. *International Journal for Numerical Methods in Engineering*, 8:1–16, 1974.

[119] J.L. Chenot and M. Bellet. The viscoplastic approach for the finite-element modelling of metal-forming processes. *Numerical Modelling of Material Deformation Processes*, pages 179–224, 1992.

[120] D. Bel E. Cueto M. Doblaré I. Alfaro, D. González and F. Chinesta. Recent advances in the meshless simulation of aluminium extrusion and other related forming processes. *Archives of Computational Methods in Engineering*, 13:3–43, 2006.

[121] Pavel Bochev and Max Gunzburger. An absolutely stable pressure-poisson stabilized finite element method for the stokes equations. *SIAM J. Numer. Anal.*, 42:1189–1207, 2004.

[122] F. Brezzi and J. Pitkaranta. On the stabilization of finite element approxima-

tions of the stokes equations. *Notes on Numerical Fluid Mechanics*, 10:11–19, 1984.

[123] F. Greco, L. Filice, C. Peco, and M. Arroyo. A stabilized formulation with maximum entropy meshfree approximants for viscoplastic flow simulation in metal forming. *International Journal of Material Forming DOI:10.1007/s12289-014-1167-x*, 2013.

[124] V. S. Mashkevich and K. B. Tolpygo. Electrical, optical and elastic properties of diamond type crystals. *I. Sov. Phys. JETP*, 5:435–439, 1957.

[125] T. D. Nguyen, S. Mao, Y.-W. Yeh, P. K. Purohit, and M. C. McAlpine. Nanoscale flexoelectricity. *Adv. Mater.*, 25(7):946–974, 2013.

[126] P. Zubko, G. Catalan, and A. K. Tagantsev. Flexoelectric effect in solids. *Annu. Rev. Mater. Res.*, 43:387–421, 2013.

[127] P. V. Yudin and A. K. Tagantsev. Fundamentals of flexoelectricity in solids. *Nanotech.*, 24(43):432001, 2013.

[128] G. Catalan, L. J. Sinnamon, and J. M. Gregg. The effect of flexoelectricity on the dielectric properties of inhomogeneously strained ferroelectric thin films. *J. Phys.: Cond. Matter*, 16(13):2253, 2004.

[129] P. Zubko, G. Catalan, A. Buckley, P. R. L. Welche, and J. F. Scott. Strain-gradient-induced polarization in SrTiO3 single crystals. *Phys. Rev. Lett.*, 99:167601, 2007.

[130] Wenhui Ma and L. Eric Cross. Flexoelectricity of barium titanate. *Appl. Phys. Lett.*, 88(23):232902, 2006.

[131] H. Lu, C.-W. Bark, D. Esque De Los Ojos, J. Alcala, C.B. Eom, G. Catalan, and A. Gruverman. Mechanical writing of ferroelectric polarization. *Science*, 335(6077):59–61, 2012.

[132] J.Y. Shu, W.E. King, and N.A. Fleck. Finite elements for materials with strain gradient effects. *Int. J. Numer. Meth. Eng.*, 44(3):373–391, 1999.

[133] E. Amanatidou and N. Aravas. Mixed finite element formulations of strain-gradient elasticity problems. *Comput. Methods. Appl. Mech. Eng.*, 191(15-16):1723–1751, 2002.

[134] S.I. Markolefas, D.A. Tsouvalas, and G.I. Tsamasphyros. Some c0-continuous mixed formulations for general dipolar linear gradient elasticity boundary value problems and the associated energy theorems. *Int. J. Solids Struct.*, 45(11–12):3255 – 3281, 2008.

[135] M. Arroyo and M. Ortiz. Local maximum-entropy approximation schemes: a seamless bridge between finite elements and meshfree methods. *Int. J. Numer. Meth. Eng.*, 65(13):2167–2202, 2006.

[136] M. S. Majdoub, P. Sharma, and T. Cagin. Enhanced size-dependent piezoelectricity and elasticity in nanostructures due to the flexoelectric effect. *Phys. Rev. B*, 79(11), 2009.

[137] R. D. Mindlin. Polarization gradient in elastic dielectrics. *Int. J. Solids Struct.*, 4(6):637 – 642, 1968.

[138] N. D. Sharma, C. M. Landis, and P. Sharma. Piezoelectric thin-film superlattices without using piezoelectric materials. *J. Appl. Phys.*, 108(2):024304, 2010.

[139] H Le Quang and Q C He. The number and types of all possible rotational symmetries for flexoelectric tensors. *Proc. Royal Soc. A*, 467:2369–2386, 2011.

[140] L. Shu, X. Wei, T. Pang, X. Yao, and C. Wang. Symmetry of flexoelectric coefficients in crystalline medium. *J. Appl. Phys.*, 110(10):104–106, 2011.

[141] S. Mao and Prashant K. Purohit. A flexoelectric reciprocal theorem and some boundary value problems. *Submitted*, 2014.

[142] M. Gharbi, Z. H. Sun, P. Sharma, and K. White. The origins of electromechanical indentation size effect in ferroelectrics. *Appl. Phys. Lett.*, 95(14):142901, 2009.

[143] A. Abdollahi, C. Peco, D. Millán, I. Arias, and M. Arroyo. Computational evaluation of the flexoelectric effect in dielectric solids. *Journal of Applied Physics*, 116:093502, 2014.

[144] Atsushi Takei, Benoît Roman, José Bico, Eugenio Hamm, and Francisco Melo. Forbidden directions for the fracture of thin anisotropic sheets: An analogy with the wulff plot. *Phys. Rev. Lett.*, 110:144301, Apr 2013.

[145] Griffith AA. The phenomena of rupture and flow in solids. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 221:163–198, 1921.

[146] V. Hakim and A. Karma. Laws of crack motion and phase-field models of fracture. *J. Mech. Phys Solids*, 57:342–368, 2009.

[147] M. Marder. Cracks cleave crystals. *EPL (Europhysics Letters)*, 66:364, 2004.

[148] J.J. Marigo G.A. Francfort. Revisiting brittle fracture as an energy minimization problem. *Journal of the Mechanics and Physics of Solids*, 46:1319–1342, 1998.

[149] B. Li, C. Peco, D. Millán, I. Arias, and M. Arroyo. Phase-field modeling and simulation of fracture in brittle materials with strongly anisotropic surface energy. *International Journal for Numerical Methods in Engineering DOI:10.1002/nme.4726*, 2014.

List of Figures

Appendix A
An adaptive meshfree method for phase-field models of biomembranes. Part I: approximation with maximum-entropy approximants.

A. Rosolen, C. Peco and M. Arroyo

Journal of Computational Physics,
249(0):303–319, 2013.

An adaptive meshfree method for phase-field models of biomembranes. Part I: approximation with maximum-entropy approximants

A. Rosolen[☆], C. Peco and M. Arroyo[*]

LaCàN, Universitat Politècnica de Catalunya-BarcelonaTech (UPC), Barcelona 08034, Spain

Abstract

We present an adaptive meshfree method to approximate phase-field models of biomembranes. In such models, the Helfrich curvature elastic energy, the surface area, and the enclosed volume of a vesicle are written as functionals of a continuous phase-field, which describes the interface in a smeared manner. Such functionals involve up to second-order spacial derivatives of the phase-field, leading to fourth-order Euler-Lagrange partial differential equations (PDE). The solutions develop sharp internal layers in the vicinity of the putative interface, and are nearly constant elsewhere. Thanks to the smoothness of the local maximum-entropy (*max-ent*) meshfree basis functions, we approximate numerically this high-order phase-field model with a direct Ritz-Galerkin method. The flexibility of the meshfree method allows us to easily adapt the grid to resolve the sharp features of the solutions. Thus, the proposed approach is more efficient than common tensor product methods (e.g. finite differences or spectral methods), and simpler than unstructured C^0 finite element methods, applicable by reformulating the model as a system of second-order PDE. The proposed method, implemented here under the assumption of axisymmetry, allows us to show numerical evidence of convergence of the phase-field solutions to the sharp interface limit as the regularization parameter approaches zero. In a companion paper, we present a Lagrangian method based on the approximants analyzed here to study the dynamics of vesicles embedded in a viscous fluid.

Keywords: maximum-entropy approximants, meshfree methods, adaptivity, phase field models, biomembranes, vesicles.

1. Introduction

Biomembranes are the fundamental separation structure in animal cells, and are responsible for the compartmentalization of the cell or for the transport of substances through cargo vesicles or tubes. They also play a key role in bio-mimetic engineered systems [1]. Their complex behaviour, rich physical properties, formation and dynamics have been objects of experimental and theoretical investigation for biologists, chemists and physicists during many years [2, 3]. Biomembranes are composed by several kinds of lipids self-assembled in a fluid bilayer, which

[☆]Current address: Institute for Soldier Nanotechnologies, MIT, Cambridge MA, USA.
[*]Correspondence to: marino.arroyo@upc.edu

presents a liquid behaviour in-plane and solid out-of-plane [4]. Vesicles are closed biomembranes, which play an important role in biophysical processes such as in the delivery of proteins, antibodies or drugs into cells, and separation of different types of biological macromolecules within cells. Vesicles serve as simplified models of more complex biological systems, and can be used to study the interaction between lipid bilayers and the surrounding medium, e.g. under osmotic stress [5], shear flow [6], or electrical fields [7]. Depending on the lipid composition, lipid bilayers can phase-separate forming multicomponent vesicles [8], which have also been the object of numerous studies as model systems for rafts.

Lipid bilayers can be modeled by very different techniques, depending on the focus. Atomistic [9] and coarse-grained [10] molecular dynamics (MD) can access molecular processes and the self-assembly. However, due to the slow relaxation of the bending modes, the computational cost of molecular simulations scales as L^6, where L is the lateral dimension of the system [11]. Even if coarse-grained MD simulations have been able to describe the collective dynamics of membrane patches of tens of nanometers, this sets a very stringent limit on the system sizes accessible with these methods. Other mesoscopic methods such as dynamically triangulated surfaces have been proposed to deal with intermediate scales [12]. On the other end of the spectrum, continuum mechanics has showed great success over the last decades in describing the equilibrium shapes of vesicles [4, 13, 14]. Continuum models have also helped understand the dynamics of fluctuations of bilayers [15], or the shape dynamics of membranes [16, 17]. Continuum mechanics models of biomembranes disregard atomic details, but still can incorporate many important effects such as the bilayer asymmetry, the spontaneous curvature, the diffusion of chemical species on the bilayer, or the dissipative mechanisms arising from the friction between the lipids [18]. Furthermore, these methods can easily access wide spans of time and length scales. The main drawback of these models is that they are usually formulated as complex nonlinear high-order partial differential equations (PDE). Here, we focus on the numerical approximation of a simple curvature model for biomembranes.

The Canham-Helfrich functional [19, 20] is a widely accepted continuum model for the curvature elasticity of fluid membranes, which explains to a large extent the observed morphologies of vesicles. This sharp interface model has been the basis of a number of numerical parametric approaches for the equilibrium analysis of axisymmetric and three-dimensional vesicles. The resulting equations for the parameterization are fourth-order nonlinear PDE. This functional is reparameterization invariant, which reflects mathematically the in-plane fluidity of lipid bilayers above the transition temperature. This feature poses numerical difficulties to parametric methods, since this invariance needs to be controlled to avoid serious mesh distortions [21, 22].

Phase-field counterparts of this model have been proposed and exercised numerically [23, 24, 25]. Although these methods increase the dimension of the problem, they naturally overcome the limitations of parametric methods when extreme shape, or even topology changes are present, and produce more robust simulations. Furthermore, these methods are more amenable to scalable parallel computations for complex systems, particularly when coupling it to the fluid mechanics of the ambient medium. Yet, the numerical solution of these models, again expressed mathematically as nonlinear fourth-order PDE, is challenging. Here, we propose to address high-order character of the equations and the sharp fronts they develop with an adaptive mesh-free method. We establish here the ability of the local maximum-entropy approximants [26]

to accurately and efficiently approximate equilibrium solutions of the phase-field model with a straight Ritz-Galerkin approach. In a companion paper [27], we propose a Lagrangian method to deal with the dynamics of vesicles embedded in a viscous fluid in the low Reynolds number limit, representative of most biological situations of interest.

The outline of the paper is as follows. Section 2 introduces the sharp interface and the phase-field models for the curvature elasticity of biomembranes, as well as a brief account of the numerical strategies to address these models. Section 3 describes the discretization of the phase-field functionals with the local maximum-entropy approximations schemes, the algorithm to find equilibrium solutions, and the method used to distribute the nodes. Numerical experiments to evaluate the performance of the approximants and the adaptive strategy are presented in Section 4. The final conclusions are collected in Section 5.

2. Sharp interface model, phase-field model, and its numerical treatment

2.1. Sharp interface model

In the sharp interface (S-I) approach, the membrane is a mathematical surface without thickness. The equilibrium shapes of vesicles minimize the Canham-Helfrich energy under area and enclosed volume constraints follow from

$$\begin{aligned} \text{(S-I model)} \qquad \text{Minimize} \quad & E(\Gamma) = \frac{k}{2}\int_\Gamma (H - C_0)^2 \; dS + k_G \int_\Gamma K \; dS \\ \text{subject to} \quad & V(\Gamma) = \frac{1}{3}\int_\Gamma \boldsymbol{x}\cdot\boldsymbol{n} \; dS = V_0 \\ & A(\Gamma) = \int_\Gamma dS = A_0, \end{aligned}$$

where Γ is the surface, k the bending rigidity, k_G the Gaussian bending rigidity, H the mean curvature, K the Gaussian curvature, $\boldsymbol{n}$ the normal to the surface, V_0 and A_0 are the prescribed volume and surface area, and C_0 is the spontaneous curvature. For surfaces of constant topology, the second integral in the curvature energy is a constant, and for this reason it is often ignored. We do not consider this term in the remainder of the paper, although is can be easily incorporated.

The area constraint comes from the near inextensibility of lipid bilayers under the usual applied forces. The volume can be regulated by osmotic effects, since biomembranes are semi-permeable. If the volume V_0 is smaller than the volume enclosed by a sphere of area A_0, then various equilibrium shapes are possible. For a given area and volume, there exist multiple equilibrium branches, as a consequence of the nonlinearity and non-convexity of the S-I model.

Various numerical methods have been proposed to solve the S-I model. Given the fact that the functional involves second derivatives of the parameterization, a direct Galerkin approach demands C^1 parameterizations. In 3D, this has been realized with subdivision finite elements [21, 28] and spherical harmonics [22]. Alternative formulations are amenable to C^0 finite elements [29, 30]. All these parametric approaches need to control the tangential motions of the mesh to avoid severe distortions.

2.2. Phase-field model

Phase-field models provide a powerful tool to tackle moving interface problems [31], and have been extensively used in physics and materials science (see [32, 33] and references therein). Recently, they are gaining popularity in a wide set of applications in applied science and engineering such as fracture [34, 35], microstructure formation and fracture evolution in ferroelectric materials [36], growth of thin films [37], image segmentation [38] and multi-phase flows [39], to mention a few.

The idea behind phase-field modeling is to replace the sharp description of the interface by a smeared continuous layer. To this end, an auxiliary field ϕ, called order parameter or phase-field, is introduced to represent the phases (e.g. inside and outside of the vesicle), and also the interface. The phase-field adopts distinct values, say -1 and +1, in each of the phases, and smoothly varies between these values in the diffuse interface. Typically, an energy functional expressed in terms of the phase-field models the physical phenomena at hand. Hence, the phase-field equation accomplishes two tasks at once: (1) it localizes the phase-field to represent a (smeared) interface, and (2) it encodes the interfacial physics. In sharp interface models, the geometric description of the interface is extrinsic to its physics.

The phase-field model for biomembranes proposed by Du *et al.* [23, 40] replaces the S-I model by:

$$
\begin{aligned}
\text{(P-F model)} \qquad \text{Minimize} \quad & E[\phi] = f_E \frac{k}{2\epsilon} \int_\Omega \left[\epsilon \Delta \phi + \left(\frac{1}{\epsilon} \phi + C_0 \sqrt{2} \right) \left(1 - \phi^2 \right) \right]^2 \, d\Omega \\
\text{subject to} \quad & V[\phi] = \frac{1}{2} \left(Vol(\Omega) + \int_\Omega \phi \, d\Omega \right) = V_0 \\
& A[\phi] = f_A \int_\Omega \left[\frac{\epsilon}{2} |\nabla \phi|^2 + \frac{1}{4\epsilon} (\phi^2 - 1)^2 \right] \, d\Omega = A_0 \\
& \phi|_{\partial\Omega} = -1,
\end{aligned}
$$

where ϵ is a small regularization parameter, $f_E = \frac{3}{8\sqrt{2}}$, $f_A = \frac{3}{2\sqrt{2}}$, Ω is the domain bounding the vesicle, and $\partial\Omega$ its boundary. The regions $\{\boldsymbol{x} : \phi(\boldsymbol{x}) > 0\}$ and $\{\boldsymbol{x} : \phi(\boldsymbol{x}) < 0\}$ represent, the inside and outside of the membrane, while the level set $\{\boldsymbol{x} : \phi(\boldsymbol{x}) = 0\}$ can be used to realize the position of the membrane.

Formal asymptotics [40], as well as rigorous mathematical analysis [41] (see also [42] for a review), provide the connection between the P-F model and the S-I mode when $\epsilon \to 0$. As this limit is never achieved in the numerical calculations, a modeling error is always present in practice. This model has been coupled with the Navier-Stokes equations in [43]. Similar ideas to couple phase-field models of biomembranes with fluid or other physical fields have been developed by other researchers as well [7, 24, 44, 45].

2.3. Numerical approaches for the phase-field functionals

The main advantage of the phase-field model is the unified treatment of the interfacial tracking and the mechanics, which potentially leads to simple, robust, scalable computer codes. This comes at the expense of a much higher computational cost, particularly if the modeling error with respect to the sharp interface limit needs to be small. Indeed, in can be seen that

the phase-field model produces solutions with the profile $\phi(\boldsymbol{x}) = \tanh\left[\frac{d(\boldsymbol{x})}{\sqrt{2}\epsilon}\right]$, where $d(\boldsymbol{x})$ is the distance to the interface. Resolving this profile requires a very fine discretization for small values of ϵ, but this high resolution is only required in the vicinity of the interface. Away from it, the phase-field is nearly constant. Hence, this problem naturally calls for adaptivity. Furthermore, a numerical method for the phase-field model needs to address the second-order derivatives in the energy and area functionals.

Traditional numerical methodologies like finite difference [23, 44] and spectral methods [43] have been used for phase-field models of biomembranes. Recently, isogeometric analysis [46], a Galerkin method based on tensor products of 1D NURBS approximants, has shown an excellent performance for the Cahn-Hilliard equation, handling successfully the sharp transitions of the solutions without spurious overshoots [47, 48]. Although these structured methods can handle higher-order operators, they have difficulties in adapting to localized features. C^0 finite element approaches can deal with the high-order character of the functional by reformulating the model as a system of second-order PDE [49] and are well suited for adaptivity [50], but suffer from poor accuracy for a given computational cost. A number of adaptive techniques have been developed for the Cahn-Hilliard model, including an adaptive multigrid finite-difference method [51, 52], a Fourier spectral moving-mesh method [53], an adaptive FEM with linear [45, 54, 55] and quadratic [56] shape functions after recasting the higher-order phase-field as a system of lower-order equations, and a finite volume approach for unstructured grids [57]. Adaptive methods based on finite differences [58, 59], Fourier spectral [60], or finite volumes [61, 62] have been proposed for other higher-order phase-field equations.

Here, we propose a Ritz-Galerkin method based on the local maximum-entropy meshfree approximants [26]. These meshfree approximants are:

- C^∞, and therefore handle without difficulties the high-order character of the functionals,
- non-negative, and therefore possess monotonicity properties, as B-Splines and NURBS successfully applied to Cahn-Hilliard models [47],
- ideally suited for local refinement and dynamic adaptivity, as the basis functions rely only on the vicinity of neighboring nodes, instead of a mesh.

3. Ritz-Galerkin approximation of the functionals with maximum-entropy schemes

We describe here the numerical approximation of the variational problem to obtain equilibrium axisymmetric configurations for biomembranes. To fix the rigid body displacements of the membrane along the axis of symmetry, we need to supplement the P-F model given above with the constraint

$$M[\phi] = \int_\Omega \phi\,(z - z_c)\;\, d\Omega = 0,$$

where z_c allows us to center the phase-field solution in the simulation box.

We discretize the equations with local maximum-entropy approximation schemes. These meshfree approximants are non-negative and satisfy up to first-order consistency conditions. They have been shown to accurately approximate fourth-order PDE, such as the Kirchhoff-Love

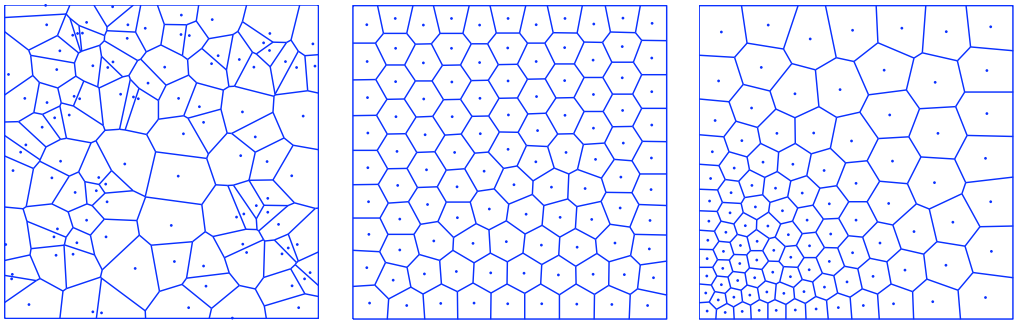


Figure 1: Voronoi tessellation for a random nodal distribution (left), CVT for a uniform density (center) and for a density function $\rho = 10 \exp\left[-2(x^2 + y^2)\right] + 0.1$ (right).

theory of thin shells [63, 64]. Second-order maximum entropy approximants have been developed [65, 66], and it has been shown that the linear approximants used here deliver comparable accuracy with a much simpler implementation. We follow a Ritz-Galerkin approach to approximate the variational formulation of the continuous problem by an algebraic optimization program, which we solve with an augmented Lagrangian method to impose the linear and nonlinear constraints, combined with L-BFGS and Newton-Rahpson nonlinear solvers. We locally adapt the node distribution to computationally afford very small values of ϵ by resorting to Centroidal Voronoi Tesselations (CVT) [67]. This method distributes nice grids of points obeying a prescribed density function, as illustrated in Figure 1. Here, we define the density functions such that the points are highly concentrated in the regions with high gradients of the phase-field (see Section 4.2).

3.1. Local maximum-entropy approximants

Meshfree methods define basis functions from a scattered set of nodes, not supported on a mesh as in traditional finite elements. The most popular meshfree approximants are based on the moving least squares (MLS) idea [68]. In recent years, the information theoretic concept of maximum-entropy has been put forth to develop polygonal approximants [69] and meshfree approximation schemes [26]. These maximum-entropy approximants present some advantages over MLS methods, such as their strict non-negativity, the straightforward imposition of boundary data, the robustness of their evaluation, or the simpler quadrature [65]. Moreover, the non-negativity and the linear reproducing conditions endow them with the structure of convex geometry [26], which enables the connection with other non-negative technologies like isogeometric analysis [46] or subdivision surfaces [70].

Maximum-entropy basis functions, denoted by $p_a(\boldsymbol{x}), a = 1, \ldots, N$ with $\boldsymbol{x} \in \mathbb{R}^d$, where d is the space dimension, are enforced to be non-negative and to fulfill the zeroth and first-order

consistency conditions

$$p_a(\boldsymbol{x}) \geq 0, \qquad \sum_{a=1}^{N} p_a(\boldsymbol{x}) = 1, \qquad \sum_{a=1}^{N} p_a(\boldsymbol{x})\ \boldsymbol{x}_a = \boldsymbol{x},$$

where the last equation allows us to identify the vectorial weights $\boldsymbol{x}_a$ with the positions of the nodes associated with each basis function.

The idea behind local maximum-entropy basis functions is to defined information-theoretical optimal approximants, only biased by locality, i.e. the property that the function approximation at a given point should depend on nodal values of nearby nodes. These approximants exhibit a (Pareto) compromise between two competing objectives, minimum width (locality) and entropy maximization (information theory optimality criteria), subject to the consistency constraints (reproducibility conditions). With these requirements, we write the following optimization program to select the approximants

$$\begin{aligned} \text{For fixed } \ \boldsymbol{x}, \ \text{minimize} \quad & \sum_{a=1}^{N} \beta_a p_a |\boldsymbol{x} - \boldsymbol{x}_a|^2 + \sum_{a=1}^{N} p_a \ln p_a \\ \text{subject to} \quad & p_a \geq 0, \quad a = 1, \dots, N \\ & \sum_{a=1}^{N} p_a = 1, \qquad \sum_{a=1}^{N} p_a \boldsymbol{x}_a = \boldsymbol{x}, \end{aligned}$$

where the non-negative nodal parameters $\beta_a = \gamma_a / h_a^2, \quad a = 1, \dots, N$ define the locality of the approximants [26, 71]. The dimensionless aspect ratio parameter γ_a characterizes the degree of locality of the basis function associated to the node $\boldsymbol{x}_a$, while h_a denotes a measure of the nodal spacing around node a. The local grid spacing h_a should be chosen to resolve the sharp features of the phase-field solutions, and should therefore be commensurate to ϵ. The basis functions become sharper and more local as the value of the dimensionless parameter γ_a increases, and the Delaunay approximants arise as specialized limits ($\gamma_a \geq 4$ in the practice), as illustrated in Figure 2 for a one-dimensional domain. In previous works, we characterized the behaviour of the approximants for problems involving higher-order derivatives, specifically for plates and thin-shells analysis [63, 66]. Typically, low values of γ_a lead to more accurate results for problems with smooth solutions, but also result in significantly more expensive calculations. This is due to the wider band-width and to the fact that more quadrature points are typically required. We found that the appropriate locality parameters are in the range $0.6 \leq \gamma \leq 1$, being $\gamma = 0.8$ the most convenient because it provides a good trade-off between computational cost and accuracy.

As detailed in [26], the optimization problem is smooth and convex, and admits a unique solution. An efficient solution follows from standard duality methods. Here, we just summarize the recipe for the calculation of the basis functions. By analogy with statistical mechanics, we define the partition function

$$Z(\boldsymbol{x}, \boldsymbol{\lambda}) = \sum_{b=1}^{N} \exp\left[-\beta_b |\boldsymbol{x} - \boldsymbol{x}_b|^2 + \boldsymbol{\lambda} \cdot (\boldsymbol{x} - \boldsymbol{x}_b)\right].$$

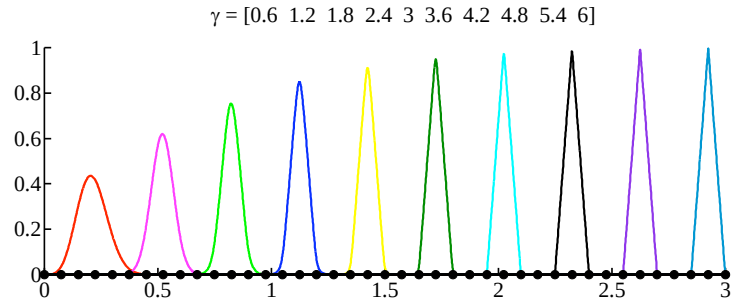


Figure 2: Seamless and smooth transition from meshfree to Delaunay affine basis functions. The transition is controlled by the non-dimensional nodal parameters γ_a, which here take linearly varying values from 0.6 (left) to 6 (right).

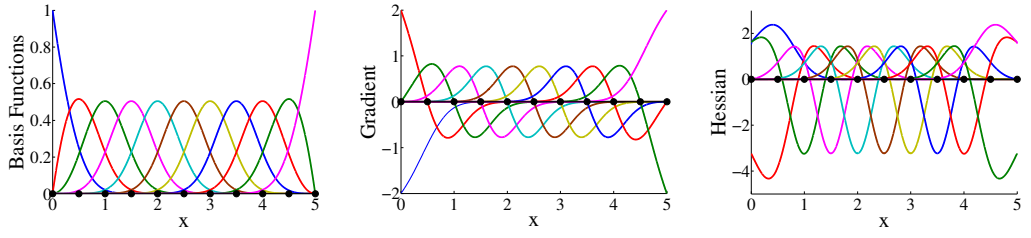


Figure 3: One-dimensional local maximum-entropy basis functions (left), and its first and second spatial derivatives (center-right) computed with a dimensionless parameter $\gamma = 0.8$.

At each evaluation point $\boldsymbol{x}$, the Lagrange multiplier for the linear consistency condition is the unique solution to a solvable, convex, unconstrained optimization problem

$$\boldsymbol{\lambda}^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{\lambda}\in\mathbb{R}^d} \ln Z(\boldsymbol{x},\boldsymbol{\lambda}).$$

This optimization problem with d unknowns is efficiently solved with Newton's method. Then, the basis functions adopt the form

$$p_a(\boldsymbol{x}) = \frac{1}{Z\left(\boldsymbol{x},\boldsymbol{\lambda}^*(\boldsymbol{x})\right)} \exp\left[-\beta_a|\boldsymbol{x}-\boldsymbol{x}_a|^2 + \boldsymbol{\lambda}^*(\boldsymbol{x})\cdot(\boldsymbol{x}-\boldsymbol{x}_a)\right].$$

We refer to [64, 71] for the expressions to compute the gradient $\nabla p_a(\boldsymbol{x})$ and the Hessian matrix $Hp_a(\boldsymbol{x})$ of the local maximum-entropy basis functions, which are illustrated in Figure 3 for a one-dimensional domain uniformly discretized and a dimensionless parameter $\gamma = 0.8$.

Some properties of the local maximum-entropy approximants, such as smoothnes and variation diminishing properties [26], are illustrated in Figure 4. These approximants also satisfy *ab initio* a weak Kronecker-delta property at the boundary of the convex hull of the nodes [26]. With this property, the imposition of essential boundary conditions in Galerkin methods

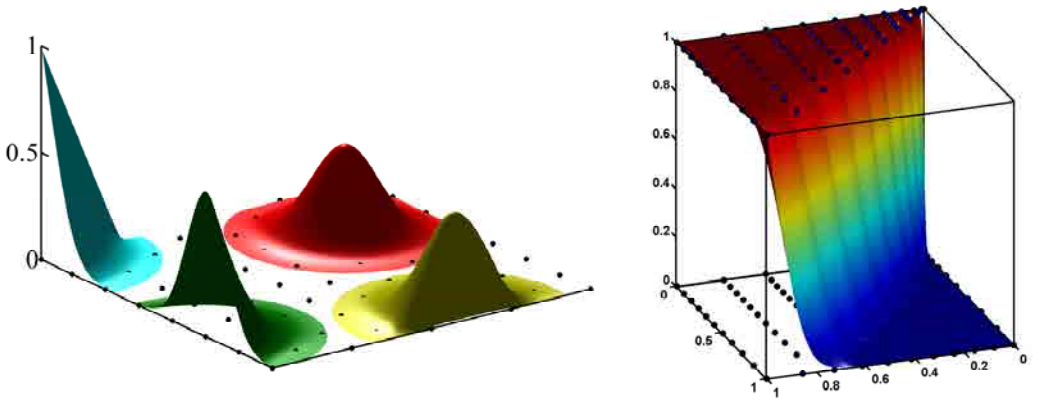


Figure 4: Illustration of non-negativity, smoothness and weak Kronecker-delta properties for two-dimensional local maximum-entropy basis functions (left), and the variation diminishing property (right).

is straightforward. Moreover, the approximants are multidimensional and lead to well behaved mass matrices [26]. We refer to [66] for a more detailed description of maximum-entropy approximants and their applications.

3.2. Discretization of the minimization problem

We consider the following expansion for the phase-field in terms of the basis functions

$$\phi(\boldsymbol{x}) \approx \phi_h(\boldsymbol{x}, \boldsymbol{\Phi}) = \sum_{a=1}^{N} p_a(\boldsymbol{x})\phi_a,$$

where $\boldsymbol{\Phi} = (\phi_1, \phi_2, ..., \phi_N)$ is an array containing the N nodal values of the phase-field, and insert this ansatz into the variational problem describing the P-F model to obtain the following algebraic optimization program:

$$\begin{aligned}
\text{Minimize} \quad & E_h(\boldsymbol{\Phi}) = E[\phi_h] = f_E \frac{k}{2\epsilon} \int_\Omega W_h^2 \; d\Omega \\
\text{subject to} \quad & V_h(\boldsymbol{\Phi}) = V[\phi_h] = \frac{1}{2}\left(Vol(\Omega) + \int_\Omega \phi_h \; d\Omega \right) = V_0 \\
& A_h(\boldsymbol{\Phi}) = A[\phi_h] = f_A \int_\Omega \left[\frac{\epsilon}{2} |\nabla \phi_h|^2 + \frac{1}{4\epsilon}(\phi_h^2 - 1)^2 \right] \; d\Omega = A_0 \\
& M_h(\boldsymbol{\Phi}) = M[\phi_h] = \int_\Omega \phi_h (z - z_c) \; d\Omega = 0 \\
& \phi_h|_{\partial\Omega} = -1,
\end{aligned} \tag{1}$$

where

$$W_h = \epsilon \Delta \phi_h + \left(\frac{1}{\epsilon}\phi_h + C_0\sqrt{2} \right)\left(1 - \phi_h^2\right).$$

The optimality conditions can be obtained from differentiating the Lagrangian function

$$\mathcal{L}(\boldsymbol{\Phi}, \boldsymbol{\nu}) = E_h(\boldsymbol{\Phi}) - \nu_A \left[A_h(\boldsymbol{\Phi}) - A_0\right] - \nu_V \left[V_h(\boldsymbol{\Phi}) - V_0\right] - \nu_M \left[M_h(\boldsymbol{\Phi}) - M_0\right],$$

where the area, volume and static moment constraints are maintained by the Lagrange multipliers $\boldsymbol{\nu} = (\nu_A, \nu_V, \nu_M)$. Physically, ν_A is a membrane tension and ν_V a pressure difference between the inside and the outside of the vesicle.

After defining a new set of variables $(\boldsymbol{\Phi}, \boldsymbol{\nu}) = (\phi_1, \phi_2, ..., \phi_N, \nu_A, \nu_V, \nu_M)$, the optimal solution of this saddle-point problem can be sought with the Newton-Raphson method applied to the nonlinear system of equations $\partial_{\boldsymbol{\Phi}}\mathcal{L} = 0$, $\partial_{\boldsymbol{\nu}}\mathcal{L} = 0$. However, this approach may lead to mere stationary points, not minimizers of the elastic energy (physically unstable equilibria). Furthermore, given the difficulty in setting good initial guesses for the Lagrange multipliers, this solution strategy is not robust.

A robust strategy that guarantees stable equilibria is based on the augmented Lagrangian method, which combines the standard Lagrangian with penalties. This method retains the exactness of the Lagrange multipliers method and the minimization principle of penalty methods. The minimization is performed iteratively on the phase-field variables only for frozen Lagrange multipliers, which are updated explicitly (see [72, 73] for further details). The augmented Lagrangian is

$$\begin{aligned}\mathcal{L}_A(\boldsymbol{\Phi}, \boldsymbol{\nu}) = {} & E_h(\boldsymbol{\Phi}) - \nu_A \left[A_h(\boldsymbol{\Phi}) - A_0\right] - \nu_V \left[V_h(\boldsymbol{\Phi}) - V_0\right] - \nu_M \left[M_h(\boldsymbol{\Phi}) - M_0\right] \\ & + \frac{1}{2\mu} \left|A_h(\boldsymbol{\Phi}) - A_0\right|^2 + \frac{1}{2\mu} \left|V_h(\boldsymbol{\Phi}) - V_0\right|^2 + \frac{1}{2\mu} \left|M_h(\boldsymbol{\Phi}) - M_0\right|^2.\end{aligned}$$

We solve the problem in two stages. First, we follow the augmented Lagrangian method to find an approximate minimizer consistent with the constraints with a coarse tolerance. Then, this approximation is refined with the regular Newton-Raphson method on the extended set of variables $(\boldsymbol{\Phi}, \boldsymbol{\nu})$. Since the initial guess for this second stage is very close to a minimizer, the algorithm never leads to unstable equilibria. The expressions to compute the gradients $\tilde{\boldsymbol{r}}(\boldsymbol{\Phi}, \boldsymbol{\nu})$ and $\tilde{\boldsymbol{r}}_A(\boldsymbol{\Phi}, \boldsymbol{\nu})$, as well as the hessians, of the Lagrangian and augmented Lagrangian, respectively, are given in Appendix A.

All the integrals in Eq. (1) and in its variations, see Appendix A, are approximated with numerical quadrature based on a background integration grid, as usually done in Galerkin meshfree methods (see [26] and references therein). Here, we consider Gaussian quadrature rules supported on the Delaunay triangulation associated with the set of nodes, although other specialized techniques are available [74].

4. Numerical Examples

The phase diagram for the equilibrium shapes of vesicles has been extensively studied (see [4, 75] and references therein). This diagram exhibits a number of equilibrium branches, including prolates, oblates, discocytes, or stomatocytes. The equilibrium shape for a given area, volume, and spontaneous curvature is not unique in general. For instance, upon deflation of an initially spherical vesicle without spontaneous curvature, the prolate-dumbbell and oblate-discocyte branches are possible, as illustrated in Figure 5. Mathematically, the shape transitions

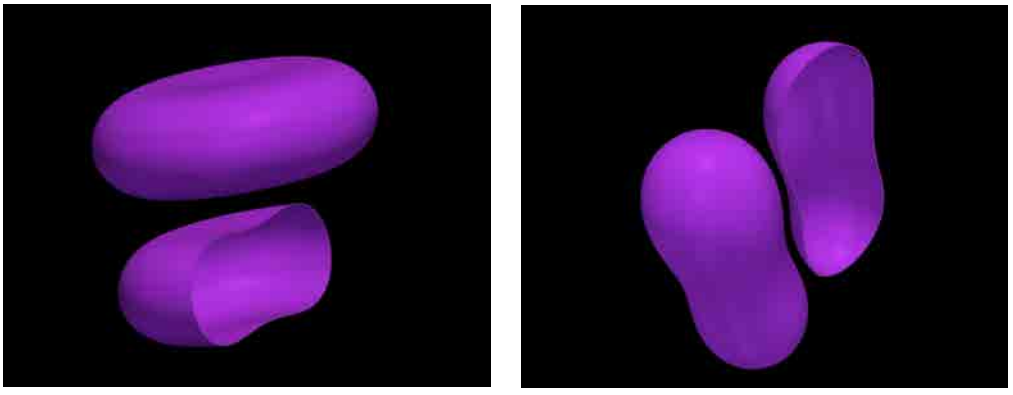


Figure 5: 3D view of discocyte (left) and dumbbell (right) equilibrium shapes.

and the equilibrium branches can be tracked by changing the volume constraint and solving for constrained minimizers. A number of equilibrium shapes for the oblate equilibrium branch are plotted in Figure 6. Each shape is an energy minimizer with fixed area and volume, after reducing by 5% the volume of the previous configuration. The computations are carried out with a uniform grid and a regularization parameter $\epsilon = 0.02$. In all the calculations, we take $C_0 = 0$ and $S_0 = 4\pi R^2$, with $R = 0.4$. The relative error in the energy is approximately 2% as compared to the sharp interface approach.

The accuracy of phase-field solutions relative to the sharp interface model is intrinsically linked to the regularization parameter ϵ, which in turn sets bounds on the required resolution of the computational grid. This motivates us to study two relevant aspects of the proposed approach: (i) the convergence as the number of points increases for a fixed regularization parameter ϵ and uniform grid, and (ii) the convergence to a sharp model as regularization parameter is decreased ($\epsilon \to 0$) and the grid of points is adapted.

To answer these questions, we analyze two specific equilibrium shapes, a discocyte and a dumbbell configuration, both of them with spontaneous curvature $C_0 = 0$. For the S-I model and the sphere, we have $A_{sphere} = 4\pi R^2 = 0.64\pi$, $V_{sphere} = \frac{4}{3}\pi R^3 \approx 0.08533\pi$ and $E_{sphere} = 2\pi$. The discocyte and dumbbell configurations are found by minimization of the curvature energy with constraints $A_0 = A_{sphere} = 0.64\pi$ and $V_0 = 0.8\ V_{sphere} \approx 0.06826\pi$, i.e. the volume of the sphere is reduced by 20%. The energies of the sharp interface model for the discocyte and dumbbell equilibrium shapes are $E_{discocyte} = 9.12657$ and $E_{dumbbell} = 8.71756$. These energies are computed with an overkill B-spline discretization of the S-I Model.

4.1. Convergence for fixed regularization parameter ϵ and uniform grids of points

Table 1 shows the numerical energies for the discocyte equilibrium shape considering different values of ϵ and several grids of points in a computational domain $\Omega = [0, 1.5] \times [0, 2]$. The identification code (O for the oblate-discocyte branch, P for the prolate-dumbbell branch) and the number of nodes for each grid are indicated in the first and the second column. As the grids

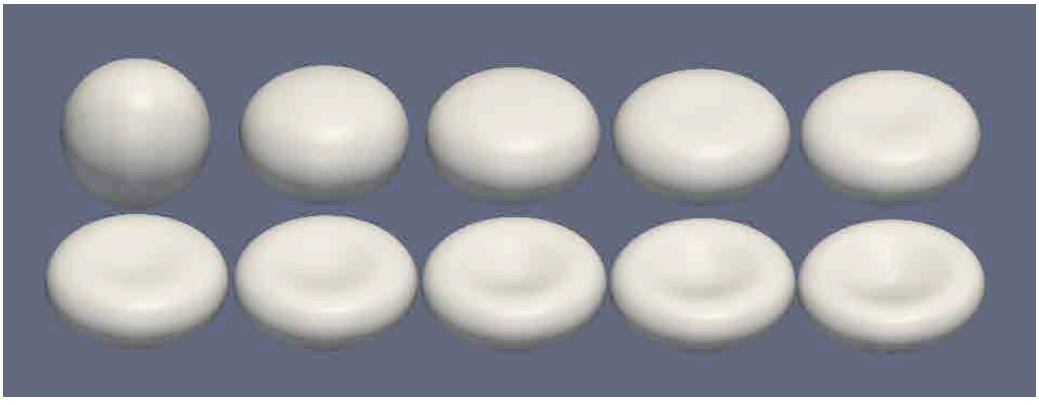


Figure 6: 3D views of the oblate equilibrium branch: each shape is computed by minimizing the energy and reducing by 5% the volume of the previous configuration.

Table 1: Energies of the discocyte equilibrium shape for different uniform grids of points and several values of ϵ. The size of the computational domain is $\Omega = [0, 1.5] \times [0, 2]$. Reference energy from a sharp interface simulation: $E_{discocyte} = 9.12657$.

ID	# nodes	$\bar{h}$	$\epsilon = 0.05$	$\epsilon = 0.04$	$\epsilon = 0.03$	$\epsilon = 0.02$	$\epsilon = 0.01$
O1	6124	0.024	9.71279	9.59056	–	–	–
O2	12271	0.017	9.72137	9.59446	9.43775	–	–
O3	24597	0.012	9.72671	9.59553	9.43483	9.29532	–
O4	49145	0.0084	9.73203	9.59786	9.43515	9.28938	–
O5	98388	0.0059	9.73536	9.59901	9.43481	9.28674	9.22082
O6	146545	0.0048	9.73716	9.59948	9.43422	9.28378	9.19139
O7	296344	0.0034	9.73989	9.60053	9.43437	9.28326	9.18627

Table 2: Energies of the dumbbell equilibrium shape for different uniform grids of points and several values of ϵ. Reference energy from a sharp interface simulation: $E_{dumbbell} = 8.71756$.

ID	# nodes	$\bar{h}$	$\epsilon = 0.05$	$\epsilon = 0.04$	$\epsilon = 0.03$	$\epsilon = 0.02$	$\epsilon = 0.01$
P1	6124	0.024	9.29504	9.15560	–	–	–
P2	12271	0.017	9.30167	9.15918	9.00361	–	–
P3	24597	0.012	9.30627	9.16106	9.00310	8.87045	–
P4	49145	0.0084	9.31053	9.16315	9.00362	8.86669	–
P5	98388	0.0059	9.31307	9.16407	9.00331	8.86445	8.81432
P6	146545	0.0048	9.31439	9.16421	9.00217	8.86005	8.77677
P7	296344	0.0034	9.31650	9.16512	9.00251	8.86033	8.77359

are not perfectly uniform (see Figure 7, for instance), the value of the average nodal spacing $\bar{h}$ is reported in the third column. The remaining columns show the energies computed for different values of the regularization parameter ϵ. We report the energies only when the transition profile is reasonably resolved, as decided by the relation $\epsilon > 2h$. Note the energy convergence from above as the number of points increases for each ϵ (columns). We can also observe how the value of the energy converges to the sharp interface value $E_{discocyte} = 9.12657$ as the parameter ϵ decreases.

In experiments not reported here, we consider the same problem in a slightly smaller domain $\Omega_1 = [0, 1]\times[0, 2]$. We find that for the larger values ϵ, the phase-field interacts with the boundary of the simulation box, resulting in higher energies. The influence of the domain size on the results further highlights the need for adaptivity, as local refinement makes it computationally affordable to increase significantly the size of the simulation box.

Table 2 reports the numerical energies for the dumbbell shape considering different values of ϵ and several refinements of the grid of points. We observe that the convergence both for ϵ and $\bar{h}$ presents the same behavior described for the discocyte shape.

4.2. Convergence as $\epsilon \to 0$ and adapted grids of points

As argued earlier, adaptivity is essential for numerical approaches based on phase-field models to be competitive. We now describe the node density function considered here to relocate the nodes following the CVT method. The phase-field is constant in a large part of the domain and presents a sharp variation in the thin region corresponding to the smeared interface. To capture this behavior, consider the density function

$$\rho(\boldsymbol{x}) = 1 + f\ |\nabla\phi(\boldsymbol{x})|$$

where f is an amplification factor. This heuristic density function allows us to obtain a uniform nodal distribution where the phase-field is constant, since $|\nabla\phi(\boldsymbol{x})| = 0$, and to locally concentrate in zones where the field changes abruptly. The factor f gives us the flexibility to increase/decrease the weight of the gradient, which in turn increases/decreases the local concentration of nodes.

A possible strategy for adaptivity is to solve the optimization problem with a coarse grid of points (and thus a large value of ϵ), apply CVT to redistribute the nodes concentrating them

Table 3: Energies of the discocyte equilibrium shape for several values of ϵ and uniform and adapted grids of 6,124 points. See Table B.6 for a description of the grids of points. Reference energy from a sharp interface simulation: $E_{discocyte} = 9.12657$.

ID	# nodes	$\epsilon = 0.04$	$\epsilon = 0.03$	$\epsilon = 0.025$	$\epsilon = 0.02$	$\epsilon = 0.015$	$\epsilon = 0.01$
O1	6124	9.59056	–	–	–	–	–
O11	6124	9.59678	9.44002	–	–	–	–
O12	6124	–	9.43506	9.35810	9.28849	–	–
O13	6124	–	–	9.35970	9.28701	9.22588	9.18703
O7	296344	9.60053	9.43437	9.35488	9.28326	9.22399	9.18627

around the interface, and compute the phase-field solution with a smaller ϵ for this new distribution of points. In practice, this strategy cannot be applied at once with a large amplification factor f. Indeed, the initial coarse grid provides an inaccurate phase-field solution, which in turn produces an inadequate relocation of the points. This ultimately constraints unphysically the phase-field solutions. A better strategy is to adapt the grid and reduce ϵ progressively, with moderate values of f.

Table 3 reports the bending energies of the discocyte equilibrium shape for uniform and adapted grids and several regularization parameters. The first and the last rows correspond to uniform meshes with 6,124 and 296,344 nodes, and are the same as those reported in Table 1. The other rows correspond to adapted grids with 6,124 nodes, obtained in each step of the progressive adaption of the grid and reduction of ϵ. The first column of the table gives an identification code for the grids of points. A description of the features of each grid is given in Table B.6, and some of the grids are shown in Figure 7. The smooth transition between the successive grids is apparent in the figure, as the value of ϵ is slowly decreased in each step, while f is increased to maintain the relative effect of the phase-field gradient. The minimum allowable value for the regularization parameter ϵ_{min} for a given grid is determined by the nodal spacing distribution, as detailed in Appendix B. As expected, the ability of adapted grids to accurately support sharp phase-field solutions at an affordable cost is noteworthy. Adapted grids grant the same accuracy (measured by the optimal energy) as uniform grids with a 50-fold reduction in the number of degrees of freedom for $\epsilon = 0.01$.

Figure 7 (bottom) shows the equilibrium phase-field for the grids referred to in Table 3 and shown in Figure 7 (top, center). It can be noticed that as the value of ϵ decreases, the thickness of the diffuse interface shrinks considerably. Figure 8 (a) illustrates the phase-field solution computed with grid O13 and $\epsilon = 0.01$; an abrupt transition can be observed between the inner ($\phi_h = 1$) and the outer ($\phi_h = -1$) regions. The superposition of the sharp interface solution with the zero phase-field level set ($\phi_h = 0$) is shown in Figure 8 (b). The two curves nearly lie on top of each other, illustrating numerically the connection between the phase-field and the sharp interface models.

Figure 8(c-e) shows cross sections of the phase-field solutions depicted in Figure 7 (bottom). The position of the cross section is indicated in Figure 8 (b) with a dashed-dotted cutline. The cross section corresponding to $\epsilon = 0.005$ is computed with an adapted grid of 24,597 nodes, as explained later. This figure highlights how the variation dimishing property (informally, the approximation is not more wiggly than the data) of the local, smooth, non-negative maximum-

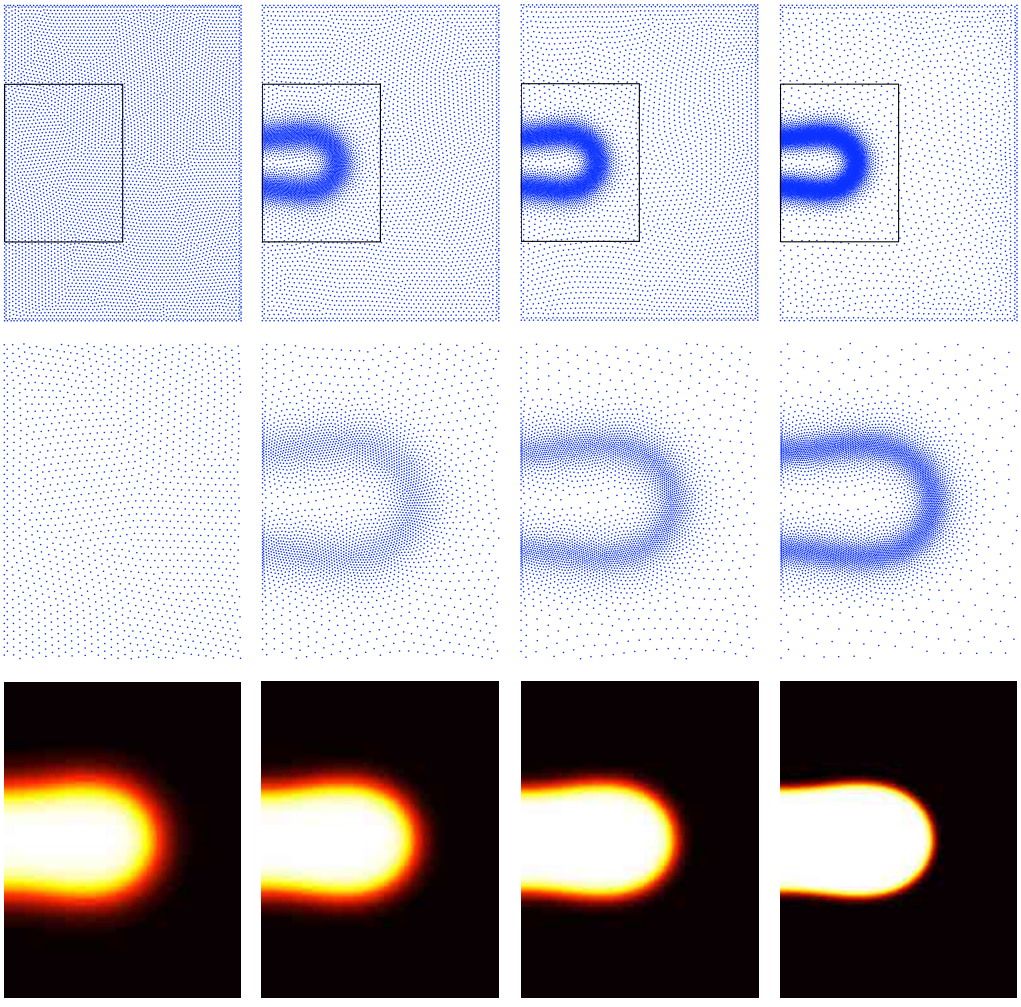


Figure 7: Discocyte equilibrium shape. Uniform and adapted grids of 6,124 points (top). From left to rigth: O1, O11, O12 and O13. Zoom of the areas indicated with black boxes (center). Phase-field (bottom). From left to rigth, the solutions correspond to $\epsilon = 0.04$, $\epsilon = 0.03$, $\epsilon = 0.02$, and $\epsilon = 0.01$. The values of energy for each solution are given in Table 3.

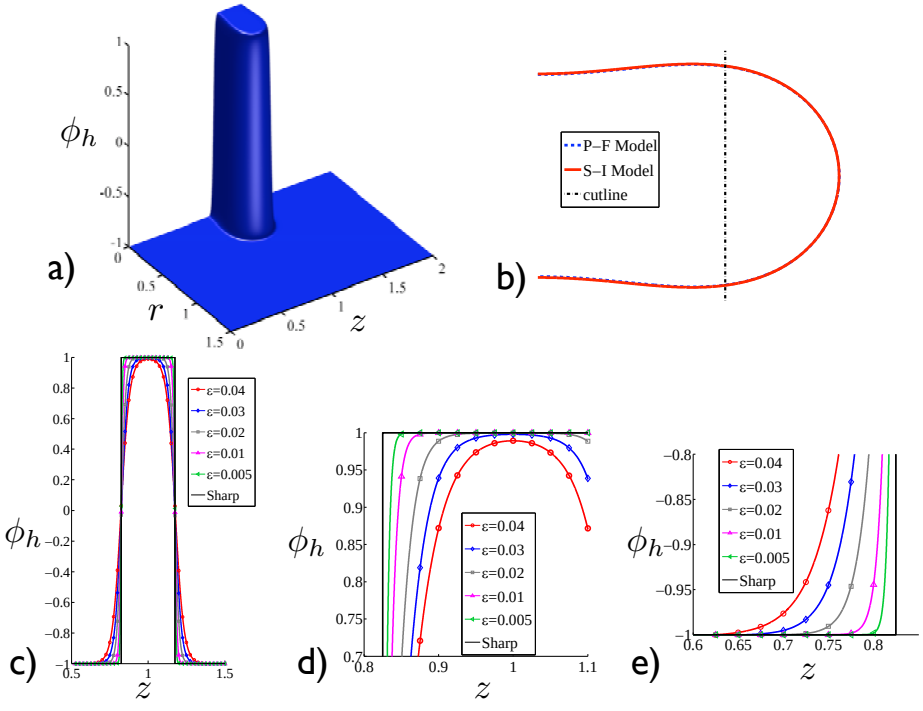


Figure 8: Phase-field solution for the discocyte equilibrium shape: (a) abrupt transition between inner ($\phi_h = 1$) and outer ($\phi_h = -1$) regions, and (b) superposition of the sharp interface solution and zero phase-field level set $\phi_h = 0$. The phase-field solution is obtained with an adapted grid of 6,124 nodes and $\epsilon = 0.01$. (c) Cross sections corresponding to the cutline indicated in (b) of the phase-field solutions with different values of ϵ. This plot, together with the zooms in (d) and (e), illustrates the absence of oscillations or overshoots near the interface, and how the interfacial thickness decreases as ϵ is reduced. The sharp interface solution is shown for comparison.

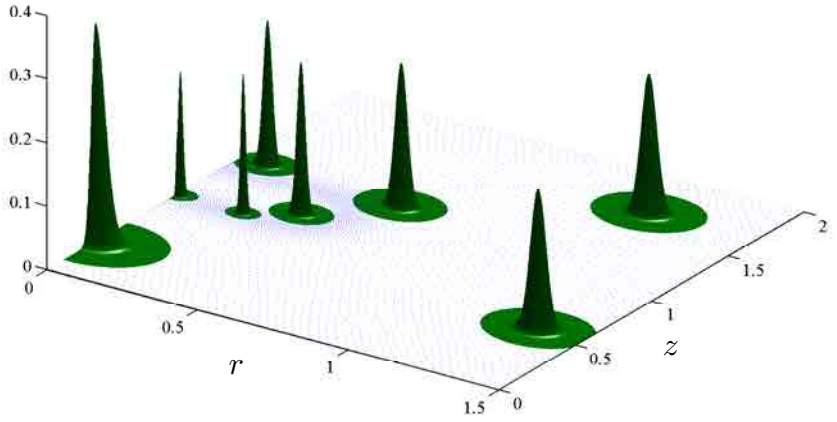


Figure 9: Illustration of the uniform aspect ratio of the basis functions, despite the strong non-uniformity of the nodal spacing (discocyte solution, N=6,124, grid O13).

Table 4: Relative error (%) measured in energy for the discocyte equilibrium shape and several values of the regularization parameter ϵ and different uniform (Un) and adapted (Ad) grids. The energy of the shape-interface model $E_{discocyte} = 9.12657$ is used as reference.

# nodes	Grid	$\epsilon = 0.03$	$\epsilon = 0.025$	$\epsilon = 0.02$	$\epsilon = 0.015$	$\epsilon = 0.01$	$\epsilon = 0.007$	$\epsilon = 0.005$
6124	Ad	3.38	2.55	1.76	1.09	0.66	–	–
12271	Ad	3.34	2.49	1.69	1.11	0.62	0.57	–
24597	Ad	3.37	2.51	1.69	1.12	0.63	–	0.43
296344	Un	3.37	2.50	1.72	1.07	0.65	–	–

entropy approximants results in monotone solutions of the phase-field PDE, devoid of spurious oscillations even for very sharp transitions. A selection of the basis functions for grid O13 are shown in Figure 9. The uniform aspect ratio of the interior basis functions is noteworthy, despite the strong non-uniformity of the grid. The monotonicity of the approximants does not immediately imply that the numerical solutions of the phase-field PDE is free of overshoots outside of the physically meaningful limits $-1 \leq \phi \leq 1$, but the numerical evidence suggests that this is the case. Further numerical analysis is required to clarify this issue. Again, the convergence of the phase-field solutions to the sharp interface stepped solution as $\epsilon \to 0$ is apparent. Similar conclusions were drawn from isogeometric simulations of the Cahn-Hilliard and isothermal Navier-Stokes-Korteweg phase-field equations, where similar smooth non-negative basis functions, albeit structured in nature, were used [47, 48].

We repeat the refinement experiments reported in Table 3 with grids of 12,271 and 24,597 nodes. The larger number of nodes allows us to resolve the phase-field model with $\epsilon = 0.007$ for the grid of 12,271 points, yielding $E_{\epsilon=0.007} = 9.17824$, and with $\epsilon = 0.005$ for the grid of 24,597 points, yielding $E_{\epsilon=0.005} = 9.16539$. Table 4 shows the relative errors in energy between the sharp interface solution and the the adapted phase-field solutions for different number of nodes

Table 5: Energies of the dumbbell equilibrium shape for several values of ϵ and uniform and adapted grids of 6,124 points. See Table B.6 for a description of the grids. Reference energy from a sharp interface simulation: $E_{dumbbell} = 8.71756$.

ID	# nodes	$\epsilon = 0.04$	$\epsilon = 0.03$	$\epsilon = 0.025$	$\epsilon = 0.02$	$\epsilon = 0.015$	$\epsilon = 0.01$
P1	6124	9.15559	–	–	–	–	–
P11	6124	9.16513	9.01027	8.93545	–	–	–
P12	6124	–	9.00358	8.92990	8.86381	8.80003	–
P13	6124	–	–	8.92452	8.86090	8.80834	8.77909
P7	296344	9.16512	9.00251	8.92706	8.86033	8.80628	8.77359

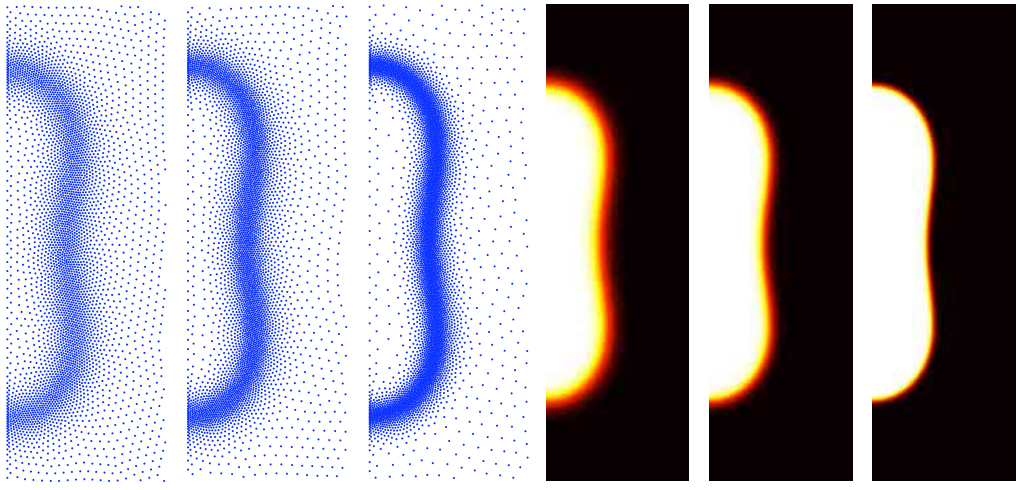


Figure 10: Distribution of points and phase-field density for adapted grids of 6,124 nodes (dumbbell equilibrium shape). The values of energy for each solution are indicated in the Table 5.

and several values of ϵ. It can noticed that, with our criterion to select ϵ_{min} for a given grid, the adapted grids resolve the width of the smeared interface, and the error depends basically on ϵ. Again, it is clear that the adaptive strategy can deliver very accurate solutions (error in the energy below 0.5%) for very small values of the regularization parameter ϵ with a reduced number of degrees of freedom.

We repeat the experiments for a dumbbell equilibrium shape. We observe the same behavior as reported in Table 5. Figure 10 illustrates adapted grids of 6,124 points with the corresponding phase-field solution for the regularization parameters $\epsilon = 0.03$, $\epsilon = 0.02$ and $\epsilon = 0.01$.

5. Conclusions

We have proposed an adaptive meshfree Ritz-Galerkin method to numerically approximate phase-field models of biomembranes. We have shown the ability of the proposed method, based

on local smooth non-negative approximants, to deal directly with the high-order character of the equations. Furthermore, adaptivity is very natural for a meshfree method, and proves essential to resolve the sharp features of the phase-field model at an affordable cost. We have shown that the adaptive method is able to resolve phase-field models with very small regularization parameter and numerically converge to the sharp interface limit. The method proposed here combines the adaptive capabilities of C^0 finite elements, which nevertheless require reformulating the fourth-order PDE as a system of second-order PDEs, hence introducing extra degrees of freedom, with the simplicity of tensor product methods, which do not require reformulations of the model.

An important issue in the adaptive strategy is to avoid excessive variations of the nodal spacing. Otherwise, the resulting meshfree basis functions can exhibit irregular features, which are difficult to integrate. CVT provides us with high quality graded distributions of points by designing an appropriate heuristic density function, although it can be computationally expensive. However, as discussed in a companion paper [27], in the proposed Lagrangian method for the dynamics of biomembranes in a viscous fluid, the CVT grid and its associated quadrature points and weights must only be computed once at the beginning of the calculation, and has a negligible cost overall. Furthermore, the strategy to adapt the nodes is not essential to the proposed method and other algorithms, such as octree methods, are more suitable and efficient to locally refine grids in 3D.

The calculations presented here are not practical in many situations of interest to assess the mechanics of vesicles and biomembranes in general, as these display very large and sometimes abrupt shape changes as the control parameters are changed. Locally refined grids impose very serious biases on the resolvable solutions, particularly when in a given optimization step, the system buckles to a distant equilibrium shape. In a companion paper [27] we present a Lagrangian method to deal with the coupled fluid-membrane overdamped dynamics, which exploits the virtues of the method presented here as the local refinement follows naturally with the Lagrangian flow the sharp features of the phase-field. This combination of methods shows promise for robust, scalable computations of complex membrane systems in three dimensions.

Acknowledgments

We acknowledge the support of the European Research Council under the European Community's 7th Framework Programme (FP7/2007-2013)/ERC grant agreement nr 240487, and of the Ministerio de Ciencia e Innovación (DPI2011-26589). MA acknowledges the support received through the prize "ICREA Academia" for excellence in research, funded by the Generalitat de Catalunya. CP acknowledges FPI-UPC Grant, FPU Ph. D. Grant (Ministry of Science and. Innovation, Spain) and Col·legi d'Enginyers de Camins, Canals i Ports de Catalunya for their support.

Appendix A. Derivatives for the optimization problem

In Section 3.2 we introduce a discretization for the continuum phase-field

$$\phi(\boldsymbol{x}) \approx \phi_h(\boldsymbol{x}, \boldsymbol{\Phi}) = \sum_{a=1}^{N} p_a(\boldsymbol{x})\phi_a,$$

where $p_a(\boldsymbol{x})$ denote the meshfree maximum-entropy approximants and $\boldsymbol{\Phi} = (\phi_1, \phi_2, ..., \phi_N)$ the array containing the N nodal values of the phase-field. The gradient and the hessian of the phase-field follow as

$$\nabla\phi(\boldsymbol{x}) \approx \nabla\phi_h(\boldsymbol{x}, \boldsymbol{\Phi}) = \sum_{a=1}^{N} \nabla p_a(\boldsymbol{x})\phi_a \quad \text{and} \quad H\phi(\boldsymbol{x}) \approx H\phi_h(\boldsymbol{x}, \boldsymbol{\Phi}) = \sum_{a=1}^{N} Hp_a(\boldsymbol{x})\phi_a.$$

The problem posed in Eq. (1) also requires the calculation of the Laplacian of the phase-field, whose expression in Cartesian coordinates is $\Delta\phi(\boldsymbol{x}) \approx \Delta\phi_h(\boldsymbol{x}) = \text{tr}\left[H\phi_h(\boldsymbol{x}, \boldsymbol{\Phi})\right]$. As we consider axisymmetric solutions, we use cylindrical coordinates, which result in

$$\Delta\phi(\boldsymbol{x}) \approx \Delta\phi_h(\boldsymbol{x}, \boldsymbol{\Phi}) = \frac{1}{r}\frac{\partial\phi_h}{\partial r} + \frac{\partial^2\phi_h}{\partial r^2} + \frac{\partial^2\phi_h}{\partial z^2}.$$

To compute the gradient of the Lagrangian and the augmented Lagrangian, we need the derivatives of E_h, V_h, A_h and M_h with respect to the nodal values $\boldsymbol{\Phi}$

$$\begin{aligned}
[\partial_{\boldsymbol{\Phi}} E_h]_a &= \frac{\partial E_h}{\partial \phi_a} = \; f_E \frac{k}{2\epsilon} \int_\Omega 2W_h \frac{\partial W_h}{\partial \phi_a} \; d\Omega, \\
[\partial_{\boldsymbol{\Phi}} V_h]_a &= \frac{\partial V_h}{\partial \phi_a} = \; \frac{1}{2} \int_\Omega p_a \; d\Omega, \\
[\partial_{\boldsymbol{\Phi}} A_h]_a &= \frac{\partial A_h}{\partial \phi_a} = \; f_A \int_\Omega \left[\epsilon \nabla\phi_h \cdot \nabla p_a + \frac{1}{\epsilon} p_a \phi_h (\phi_h^2 - 1) \right] \; d\Omega, \\
[\partial_{\boldsymbol{\Phi}} M_h]_a &= \frac{\partial M_h}{\partial \phi_a} = \; \int_\Omega p_a (z - z_c) \; d\Omega,
\end{aligned}$$

where

$$W_h = \epsilon\Delta\phi_h + \left(\frac{1}{\epsilon}\phi_h + C_0\sqrt{2}\right)\left(1 - \phi_h^2\right),$$

$$\frac{\partial W_h}{\partial \phi_a} = \epsilon\frac{\partial \Delta\phi_h}{\partial \phi_a} + \frac{p_a}{\epsilon} - p_a\phi_h\left(\frac{3}{\epsilon}\phi_h + 2C_0\sqrt{2}\right),$$

and

$$\frac{\partial \Delta\phi_h}{\partial \phi_a} = \frac{1}{r}\frac{\partial p_a}{\partial r} + \frac{\partial^2 p_a}{\partial r^2} + \frac{\partial^2 p_a}{\partial z^2}.$$

The calculation of the hessian of the Lagrangian and the augmented Lagrangian also requires the second derivatives of E_h, V_h, A_h and M_h with respect to $\boldsymbol{\Phi}$

$$\begin{aligned}
[\partial_{\boldsymbol{\Phi}}\partial_{\boldsymbol{\Phi}} E_h]_{ab} &= \frac{\partial^2 E_h}{\partial \phi_a \partial \phi_b} = f_E \frac{k}{2\epsilon} \int_\Omega 2 \left(\frac{\partial W_h}{\partial \phi_a}\frac{\partial W_h}{\partial \phi_b} + W_h \frac{\partial^2 W_h}{\partial \phi_a \partial \phi_b} \right) \; d\Omega, \\
[\partial_{\boldsymbol{\Phi}}\partial_{\boldsymbol{\Phi}} V_h]_{ab} &= \frac{\partial^2 V_h}{\partial \phi_a \partial \phi_b} = 0, \\
[\partial_{\boldsymbol{\Phi}}\partial_{\boldsymbol{\Phi}} A_h]_{ab} &= \frac{\partial^2 A_h}{\partial \phi_a \partial \phi_b} = f_A \int_\Omega \left[\epsilon \nabla p_a \cdot \nabla p_b + \frac{1}{\epsilon} p_a p_b (3\phi_h^2 - 1) \right] \; d\Omega, \\
[\partial_{\boldsymbol{\Phi}}\partial_{\boldsymbol{\Phi}} M_h]_{ab} &= \frac{\partial^2 M_h}{\partial \phi_a \partial \phi_b} = 0,
\end{aligned}$$

where

$$\frac{\partial^2 W_h}{\partial \phi_a \partial \phi_b} = -2 p_a p_b \left(\frac{3}{\epsilon} \phi_h + C_0 \sqrt{2} \right).$$

After defining a new set of variables $\tilde{\boldsymbol{x}} = (\boldsymbol{\Phi}, \boldsymbol{\nu}) = (\phi_1, \phi_2, ..., \phi_N, \nu_A, \nu_V, \nu_M)$, where $\boldsymbol{\nu}$ denotes the set of Lagrange multipliers, the gradient $\tilde{\boldsymbol{r}}(\tilde{\boldsymbol{x}})$ for the Lagrangian is given by

$$\tilde{\boldsymbol{r}}(\tilde{\boldsymbol{x}}) = \partial_{\tilde{\boldsymbol{x}}} \mathcal{L}(\tilde{\boldsymbol{x}}) = \left[\partial_{\boldsymbol{\Phi}} \mathcal{L}(\tilde{\boldsymbol{x}}) \; \partial_{\boldsymbol{\nu}} \mathcal{L}(\tilde{\boldsymbol{x}}) \right]^T,$$

where

$$\partial_{\boldsymbol{\Phi}} \mathcal{L}(\tilde{\boldsymbol{x}}) = \partial_{\boldsymbol{\Phi}} E_h(\boldsymbol{\Phi}) - \nu_A \partial_{\boldsymbol{\Phi}} A_h(\boldsymbol{\Phi}) - \nu_V \partial_{\boldsymbol{\Phi}} V_h(\boldsymbol{\Phi}) - \nu_M \partial_{\boldsymbol{\Phi}} M_h(\boldsymbol{\Phi}),$$

and

$$\partial_{\boldsymbol{\nu}} \mathcal{L}(\tilde{\boldsymbol{x}}) = \left[(A_h(\boldsymbol{\Phi}) - A_0) \;\; (V_h(\boldsymbol{\Phi}) - V_0) \;\; (M_h(\boldsymbol{\Phi}) - M_0) \right].$$

The hessian $\tilde{\boldsymbol{J}}(\tilde{\boldsymbol{x}})$ can be computed as

$$\tilde{\boldsymbol{J}}(\tilde{\boldsymbol{x}}) = \partial_{\tilde{\boldsymbol{x}}} \tilde{\boldsymbol{r}}(\tilde{\boldsymbol{x}}) = \partial_{\tilde{\boldsymbol{x}}} \partial_{\tilde{\boldsymbol{x}}} \boldsymbol{\mathcal{L}}(\tilde{\boldsymbol{x}}) = \begin{bmatrix} \partial_{\boldsymbol{\Phi}} \partial_{\boldsymbol{\Phi}} \mathcal{L}(\tilde{\boldsymbol{x}}) & \partial_{\boldsymbol{\Phi}} \partial_{\boldsymbol{\nu}} \mathcal{L}(\tilde{\boldsymbol{x}}) \\ \partial_{\boldsymbol{\nu}} \partial_{\boldsymbol{\Phi}} \mathcal{L}(\tilde{\boldsymbol{x}}) & \boldsymbol{0} \end{bmatrix},$$

where

$$\partial_{\boldsymbol{\Phi}} \partial_{\boldsymbol{\Phi}} \mathcal{L}(\tilde{\boldsymbol{x}}) = \partial_{\boldsymbol{\Phi}} \partial_{\boldsymbol{\Phi}} E_h(\boldsymbol{\Phi}) - \nu_A \partial_{\boldsymbol{\Phi}} \partial_{\boldsymbol{\Phi}} A_h(\boldsymbol{\Phi}) - \nu_V \partial_{\boldsymbol{\Phi}} \partial_{\boldsymbol{\Phi}} V_h(\boldsymbol{\Phi}) - \nu_M \partial_{\boldsymbol{\Phi}} \partial_{\boldsymbol{\Phi}} M_h(\boldsymbol{\Phi}),$$

$$\partial_{\boldsymbol{\Phi}} \partial_{\boldsymbol{\nu}} \mathcal{L}(\tilde{\boldsymbol{x}}) = \left[\partial_{\boldsymbol{\Phi}} A_h(\boldsymbol{\Phi}) \; \partial_{\boldsymbol{\Phi}} V_h(\boldsymbol{\Phi}) \; \partial_{\boldsymbol{\Phi}} M_h(\boldsymbol{\Phi}) \right],$$

and

$$\partial_{\boldsymbol{\nu}} \partial_{\boldsymbol{\Phi}} \mathcal{L}(\tilde{\boldsymbol{x}}) = \left[\partial_{\boldsymbol{\Phi}} \partial_{\boldsymbol{\nu}} \mathcal{L}(\tilde{\boldsymbol{x}}) \right]^T.$$

The gradient $\tilde{\boldsymbol{r}}_A(\boldsymbol{\Phi}, \boldsymbol{\nu}) = \partial_{\boldsymbol{\Phi}} \mathcal{L}_A(\boldsymbol{\Phi}, \boldsymbol{\nu})$ and the hessian $\tilde{\boldsymbol{J}}_A(\boldsymbol{\Phi}, \boldsymbol{\nu}) = \partial_{\boldsymbol{\Phi}} \partial_{\boldsymbol{\Phi}} \mathcal{L}_A(\boldsymbol{\Phi}, \boldsymbol{\nu})$ of the augmented Lagrangian with respect to the phase-field nodal values are

$$\begin{aligned} \tilde{\boldsymbol{r}}_A(\boldsymbol{\Phi}, \boldsymbol{\nu}) = \partial_{\boldsymbol{\Phi}} E_h(\boldsymbol{\Phi}) &- \left[\nu_A - \frac{A_h(\boldsymbol{\Phi}) - A_0}{\mu} \right] \partial_{\boldsymbol{\Phi}} A_h(\boldsymbol{\Phi}) \\ &- \left[\nu_V - \frac{V_h(\boldsymbol{\Phi}) - V_0}{\mu} \right] \partial_{\boldsymbol{\Phi}} V_h(\boldsymbol{\Phi}) - \left[\nu_M - \frac{M_h(\boldsymbol{\Phi}) - M_0}{\mu} \right] \partial_{\boldsymbol{\Phi}} M_h(\boldsymbol{\Phi}), \end{aligned}$$

and

$$\begin{aligned} \tilde{\boldsymbol{J}}_A(\boldsymbol{\Phi}, \boldsymbol{\nu}) = \partial_{\boldsymbol{\Phi}} \partial_{\boldsymbol{\Phi}} E_h(\boldsymbol{\Phi}) &+ \frac{1}{\mu} \partial_{\boldsymbol{\Phi}} A_h(\boldsymbol{\Phi}) \otimes \partial_{\boldsymbol{\Phi}} A_h(\boldsymbol{\Phi}) + \frac{1}{\mu} \partial_{\boldsymbol{\Phi}} V_h(\boldsymbol{\Phi}) \otimes \partial_{\boldsymbol{\Phi}} V_h(\boldsymbol{\Phi}) \\ &+ \frac{1}{\mu} \partial_{\boldsymbol{\Phi}} M_h(\boldsymbol{\Phi}) \otimes \partial_{\boldsymbol{\Phi}} M_h(\boldsymbol{\Phi}) - \left[\nu_A - \frac{A_h(\boldsymbol{\Phi}) - A_0}{\mu} \right] \partial_{\boldsymbol{\Phi}} \partial_{\boldsymbol{\Phi}} A_h(\boldsymbol{\Phi}). \end{aligned}$$

With the above expressions, the Newton-Raphson iterations follow directly,

$$\boldsymbol{\Phi}^{n+1} = \boldsymbol{\Phi}^n - \left[\tilde{\boldsymbol{J}}_A(\boldsymbol{\Phi}^n, \boldsymbol{\nu}^n) \right]^{-1} \tilde{\boldsymbol{r}}_A(\boldsymbol{\Phi}^n, \boldsymbol{\nu}^n)$$

in the first stage described in Section 3.2, and

$$\tilde{\boldsymbol{x}}^{n+1} = \tilde{\boldsymbol{x}}^n - \left[\tilde{\boldsymbol{J}}(\tilde{\boldsymbol{x}}^n) \right]^{-1} \tilde{\boldsymbol{r}}(\tilde{\boldsymbol{x}}^n)$$

in the second stage.

Table B.6: Description of the uniform and adapted grids used in the calculations.

ID	Grid	# nodes	Features
O1	Uniform	6124	$\bar{h} = 0.024$
O11	Adapted	6124	CVT starting from grid O1, with $f = 10$, and $\mathbf{\Phi}$ for $\epsilon = 0.04$
O12	Adapted	6124	CVT starting from grid O11, with $f = 100$, and $\mathbf{\Phi}$ for $\epsilon = 0.03$
O13	Adapted	6124	CVT starting from grid O12, with $f = 1000$, and $\mathbf{\Phi}$ for $\epsilon = 0.025$
O2	Uniform	12271	$\bar{h} = 0.017$
O21	Adapted	12271	CVT starting from grid O2, with $f = 10$, and $\mathbf{\Phi}$ for $\epsilon = 0.03$
O22	Adapted	12271	CVT starting from grid O21, with $f = 100$, and $\mathbf{\Phi}$ for $\epsilon = 0.025$
O23	Adapted	12271	CVT starting from grid O22, with $f = 1000$, and $\mathbf{\Phi}$ for $\epsilon = 0.015$
O3	Uniform	24597	$\bar{h} = 0.012$
O31	Adapted	24597	CVT starting from grid O3, with $f = 10$, and $\mathbf{\Phi}$ for $\epsilon = 0.03$
O32	Adapted	24597	CVT starting from grid O31, with $f = 100$, and $\mathbf{\Phi}$ for $\epsilon = 0.015$
O7	Uniform	296344	$\bar{h} = 0.0034$
P1	Uniform	6124	$\bar{h} = 0.024$
P11	Adapted	6124	CVT starting from grid P1, with $f = 10$, and $\mathbf{\Phi}$ for $\epsilon = 0.04$
P12	Adapted	6124	CVT starting from grid P11, with $f = 100$, and $\mathbf{\Phi}$ for $\epsilon = 0.03$
P13	Adapted	6124	CVT starting from grid P12, with $f = 1000$, and $\mathbf{\Phi}$ for $\epsilon = 0.025$
P7	Uniform	296344	$\bar{h} = 0.0034$

Appendix B. Progressive refinement of the grid

Table B.6 provides details about the progressive refinement of the grids presented in the paper. The adaptive process produces non-uniform nodal distributions. We use the nodal spacing as figure to measure the non-uniformity of a grid. The nodal spacing h_a can be understood as the average distance from a specific node $\boldsymbol{x}_a$ to the first ring of nearest neighbors $\boldsymbol{x}_b$, and it can be easily estimated with the information provided by the CVT. Indeed, as for a specific Voronoi cell Ω_a (associated to a node $\boldsymbol{x}_a$) we know all its adjacent Voronoi cells Ω_b (and thus the first ring of nodes $\boldsymbol{x}_b$), a good estimation of h_a can be obtained by computing the average distance among the node $\boldsymbol{x}_a$ and all its neighbors $\boldsymbol{x}_b$. The nodal spacing is also required to compute the basis functions (see Section 3.1) and to determine the transition parameter ϵ_{min}, as we explain later.

In Figure B.11 we illustrate the histograms for the nodal spacing distribution of uniform and adapted grids of 6,124 points corresponding to the discocyte equilibrium shape (see Table B.6 for the features of each grid). To facilitate the comparison between the different grids, we substract the nodal spacing of the uniform grid, i.e. $\bar{h} = 0.024$, to the nodal spacing of all the histograms. The top-left histogram corresponds to O1, and is strongly concentrated around zero because the grid is almost perfectly uniform. The other three histograms show the nodal spacing distribution for the adapted grids O11, O12 and O13. Note that the distributions exhibit two peaks, one associated to the smallest nodal spacing and the other to the largest one. The location and amplitude of these peaks change as the adaptivity algorithm concentrates further the nodes in a thin region near the interface (see Figure 7). The peak of the left increases its magnitude and becomes narrower, which means that the smallest nodal spacing decreases and a larger fraction

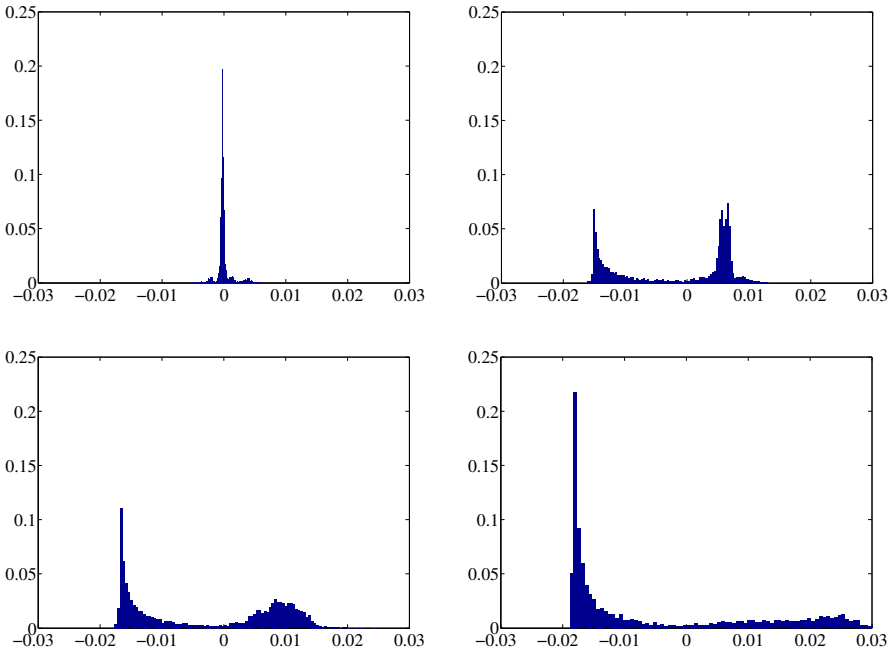


Figure B.11: Histograms of the nodal spacing distribution for different grids of 6124 points (discocyte equilibrium shape). The histograms are centered in $\bar{h} = 0.024$ and correspond to grids O1 (top-left), O11 (top-right), O12 (bottom-left) and O13 (bottom-right).

of the nodes is in the refined region. The peak of the right decreases its magnitude and becomes widespread, as fewer nodes suffice to describe the coarse region. The value of ϵ_{min} that a given grid can resolve is computed from the criterion $\epsilon_{min} \geq 2h_{min}$, where h_{min} is the nodal spacing of the left peak.

[1] M. Karlsson, K. Sott, M. Davidson, A.-S. Cans, P. Linderholm, D. Chiu, O. Orwar, Formation of geometrically complex lipid nanotube-vesicle networks of higher-order topologies, Proceedings of the National Academy of Sciences 99 (18) (2002) 11573–11578.

[2] M. Edidin, Lipids on the frontier: a century of cell-membrane bilayers, Nature Reviews Molecular Cell Biology 4 (5) (2003) 414–418.

[3] S. Semrau, T. Schmidt, Membrane heterogeneity – from lipid domains to curvature effects, Soft Matter 5 (17) (2009) 3174–3186.

[4] U. Seifert, Configurations of fluid membranes and vesicles, Advances in Physics 46 (1) (1997) 13–137.

[5] E. Reimhult, F. Höök, B. Kasemo, Intact vesicle adsorption and supported biomembrane formation from vesicles in solution: influence of surface chemistry, vesicle size, temperature, and osmotic pressure, Langmuir 19 (5) (2003) 1681–1691.

[6] M. K. W. Wintz, U. Seifert, R. Lipowsky, Fluid vesicles in shear flow, Physical Review Letters 77 (17) (1996) 3685–3688.

[7] L.-T. Gao, X.-Q. Feng, H. Gao, A phase field method for simulating morphological evolution of vesicles in electric fields, Journal of Computational Physics 228 (2009) 4162–4181.

[8] T. Baumgart, S. Hess, W. Webb, Imaging coexisting fluid domains in biomembrane models coupling curvature and line tension, Nature 425 (2003) 821–824.

[9] E. Lindahl, O. Edholm, Mesoscopic undulations and thickness fluctuations in lipid bilayers from molecular dynamics simulations, Biophysical Journal 79 (2000) 426–433.

[10] B. Reynwar, G. Illya, V. Harmandaris, M.M.Müller, K. Kremer, M. Deserno, Aggregation and vesiculation of membrane proteins by curvature-mediated interactions, Nature 447 (2007) 461–464.

[11] I. Cooke, K. Kremer, M. Deserno, Tunable generic model for fluid bilayer membranes, Physical Review E 72 (2005) 011506.

[12] H. Noguchi, G. Gompper, Dynamics of fluid vesicles in shear flow: Effect of membrane viscosity and thermal fluctuations, Physical Review E 72 (2005) 011901.

[13] S. Svetina, B. Zeks, Bilayer couple hypothesis of red cell shape transformations and osmotic hemolysis, Biochim. Biophys. Acta 42 (1983) 86–90.

[14] F. Julicher, R. Lipowsky, Shape transformations of vesicles with intermembrane domains, Physical Review E 53 (3) (1996) 2670–2683.

[15] U. Seifert, S. A. Langer, S. A., Viscous modes of fluid bilayer membranes, Europhysics Letters 23 (1) (1993) 71–76.

[16] P. Sens, Dynamics of nonequilibrium bud formation, Physical Review Letters 93 (10) (2004) 108103.

[17] M. Arroyo, A. DeSimone, Relaxation dynamics of fluid membranes, Phys. Rev. E 79 (3) (2009) 031915.

[18] M. Rahimi, M. Arroyo, Shape dynamics, lipid hydrodynamics, and the complex viscoelasticity of bilayer membranes, Physical Review E 86 (2012) 011932.

[19] P. Canham, The minimum energy of bending as a possible explanation of the biconcave shape of the human red blood cell, Journal of Theoretical Biology 26 (1) (1970) 61–81.

[20] W. Helfrich, Elastic properties of lipid bilayers: theory and possible experiments, Z. Naturforsch C 28 (11) (1973) 693–703.

[21] F. Feng, W. Klug, Finite element modeling of lipid bilayer membranes, Journal of Computational Physics 220 (1) (2006) 394–408.

[22] S. Veerapaneni, A. Rahimian, G. Biros, D. Zorin, A fast algorithm for simulating vesicle flows in three dimensions, Journal of Computational Physics 230 (2011) 5610–5634.

[23] Q. Du, C. Liu, X. Wang, A phase field approach in the numerical study of the elastic bending energy for vesicle membranes, Journal of Computational Physics 198 (2004) 450–468.

[24] T. Biben, K. Kassner, C. Misbah, Phase-field approach to three-dimensional vesicle dynamics, Physical Review E 72 (4) (2005) 041921.

[25] F. Campelo, Modeling morphological instabilities in lipid membranes with anchored amphiphilic polymers, J. Chem. Biol. 2 (2009) 65–80.

[26] M. Arroyo, M. Ortiz, Local maximum-entropy approximation schemes: a seamless bridge between finite elements and meshfree methods, International Journal for Numerical Methods in Engineering 65 (13) (2006) 2167–2202.

[27] C. Peco, A. Rosolen, M. Arroyo, An adaptive meshfree method for phase-field models of biomembranes. Part II: a Lagrangian approach for membranes in viscous fluids, Journal of Computational Physics ?? (2013) ??–??

[28] L. Ma, W. Klug, Viscous regularization and r-adaptive remeshing for finite element analysis of lipid membrane mechanics, Journal of Computational Physics 227 (11) (2008) 5816 – 5835.

[29] C. Elliott, B. Stinner, Modeling and computation of two phase geometric biomembranes using surface finite elements, Journal of Computational Physics 229 (18) (2010) 6585–6612.

[30] A. Bonito, R. Nochetto, S. Pauletti, Parametric FEM for geometric biomembranes, Journal of Computational Physics 229 (9) (2010) 3171–3188.

[31] L. Landau, On the theory of phase transitions, Gordon and Breach, 1937.

[32] R. Sekerka, Morphology: from sharp interface to phase field models, Journal of Crystal Growth 264 (2004) 530–540.

[33] I. Steinbach, Phase-field models in materials science, Modelling and Simulation in Materials Science and Engineering 17 (2009) 073001.

[34] G. Francfort, J.-J. Marigo, Revisiting brittle fracture as an energy minimization problem, Journal of the Mechanics and Physics of Solids 46 (1998) 1319–1342.

[35] C. Miehe, F. Welschinger, M. Hofacker, Thermodynamically consistent phase-field models of fracture: Variational principles and multi-field fe implementations, International Journal for Numerical Methods in Engineering 83 (2010) 1273–1311.

[36] A. Abdollahi, I. Arias, Phase-field modeling of the coupled microstructure and fracture evolution in ferroelectric single crystals, Acta Materialia 59 (12) (2011) 4733–4746.

[37] A. Rätz, A. Ribalta, A. Voigt, Surface evolution of elastically stressed films under deposition by a diffuse interface model, Journal of Computational Physics 214 (2006) 187–208.

[38] M. Benes, V. Chalupecky, K. Mikula, Geometrical image segmentation by the Allen-Cahn equation, Applied Numerical Mathematics 51 (2004) 187–205.

[39] D. Jacqmin, Calculation of two-phase Navier-Stokes flows using phase-field modeling, Journal of Computational Physics 155 (1999) 96–127.

[40] X. Wang, Phase field models and simulations of vesicle bio-membranes, Ph.D. thesis, Department of Mathematics, The Pennsylvania State University, Pennsylvania, USA (2005).

[41] G. Bellettini, L. Mugnai, Approximation of Helfrich's functional via diffuse interfaces, SIAM Journal on Mathematical Analysis 42 (2010) 2402–2433.

[42] Q. Du, Phase field calculus, curvature-dependent energies, and vesicle membranes, Philosophical Magazine 91 (2010) 165–181.

[43] Q. Du, C. Liu, R. Ryham, X. Wang, Energetic variational approaches in modeling vesicle and fluid interactions, Physica D 238 (2009) 923–930.

[44] F. Campelo, Shapes in Cells. Dynamic instabilities, morphology, and curvature in biological membranes, Ph.D. thesis, Universitat de Barcelona (2008).

[45] J. Lowengrub, A. Rätz, A. Voigt, Phase-field modeling of the dynamics of multicomponent vesicles: Spinodal decomposition, coarsening, budding, and fission, Physical Review E 79 (2009) 031926.

[46] T. Hughes, J. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (2005) 4135–4195.

[47] H. Gomez, V. Calo, Y. Bazilevs, T. Hughes, Isogeometric analysis of the Cahn-Hilliard phase-field model, Computer Methods in Applied Mechanics and Engineering 197 (2008) 4333–4352.

[48] H. Gomez, T. Hughes, X. Nogueira, V. Calo, Isogeometric analysis of the isothermal Navier-Stokes-Korteweg equations, Computer Methods in Applied Mechanics and Engineering 199 (2010) 1828–1840.

[49] Q. Du, L. Zhu, Analysis of a mixed finite element method for a phase field bending elasticity model of vesicle membrane deformation, Journal of Computational Mathematics 24 (2006) 265–280.

[50] Q. Du, J. Zhang, Adaptive finite element method for a phase field bending elasticity model of vesicle membrane deformations, SIAM J. Sci. Comput. 30 (3) (2008) 1634–1657.

[51] H. Ceniceros, R. Nós, A. Roma, Three-dimensional, fully adaptive simulations of phase-field fluid models, Journal of Computational Physics 229 (2010) 6135–6155.

[52] S. Wise, J. Kim, J. Lowengrub, Solving the regularized, strongly anisotropic Cahn–Hilliard equation by an adaptive nonlinear multigrid method, Journal of Computational Physics 226 (1) (2007) 414–446.

[53] W. Feng, P. Yu, S. Hu, Z. Liu, Q. Du, L. Chen, A Fourier spectral moving mesh method for the Cahn-Hilliard equation with elasticity, Communications in Computational Physics 5 (2–4) (2009) 582–599.

[54] A. Voigt, T. Witkowski, Hybrid parallelization of an adaptive finite element code, KYBERNETIKA 46 (2010) 316–327.

[55] P. Yue, C. Zhou, J. Feng, C. Ollivier-Gooch, H. Hu, Phase-field simulations of interfacial dynamics in viscoelastic fluids using finite elements with adaptive meshing, Journal of Computational Physics 219 (1) (2006) 47–67.

[56] C. Zhou, P. Yue, J. Feng, C. Ollivier-Gooch, H. Hu, 3d phase-field simulations of interfacial dynamics in Newtonian and viscoelastic fluids, Journal of Computational Physics 229 (2010) 498–511.

[57] L. Cueto-Felgueroso, J. Peraire, A time-adaptive finite volume method for the Cahn-Hilliard and Kuramoto-Sivashinsky equations, Journal of Computational Physics 227 (2008) 9985–10017.

[58] R. Braun, B. Murray, Adaptive phase-field computations of dendritic crystal growth, Journal of Crystal Growth 177 (1997) 41–53.

[59] J. Rosam, P. Jimack, A. Mullis, A fully implicit, fully adaptive time and space discretisation method for phase-field simulation of binary alloy solidification, Journal of Computational Physics 225 (2007) 1271–1287.

[60] W. Feng, P. Yu, S. Hu, Z. Liu, Q. Du, L. Chen, Spectral implementation of an adaptive moving mesh method for phase-field equations, Journal of Computational Physics 220 (1) (2006) 498–510.

[61] C. Lan, Y. Chang, Efficient adaptive phase field simulation of directional solidification of a binary alloy, Journal of Crystal Growth 250 (2003) 525–537.

[62] Z. Tan, K. Lim, B. Khoo, An adaptive mesh redistribution method for the incompressible mixture flows using phase-field model, Journal of Computational Physics 225 (2007) 1137–1158.

[63] D. Millán, A. Rosolen, M. Arroyo, Nonlinear manifold learning for meshfree finite deformation thin shell analysis, International Journal for Numerical Methods in Engineering 93 (7) (2013) 685–713.

[64] D. Millán, A. Rosolen, M. Arroyo, Thin shell analysis from scattered points with maximum-entropy approximants, International Journal for Numerical Methods in Engineering 85 (6) (2011) 723–751.

[65] C. Cyron, M. Arroyo, M. Ortiz, Smooth, second order, non-negative meshfree approximants selected by maximum entropy, International Journal for Numerical Methods in Engineering 79 (13) (2009) 1605–1632.

[66] A. Rosolen, D. Millán, M. Arroyo, Second order convex *maximum entropy* approximants with applications to high order PDE, International Journal for Numerical Methods in Engineering 94 (2) (2013) 150–182.

[67] Q. Du, V. Faber, M. Gunzburger, Centroidal Voronoi Tessellations: Applications and Algorithms, SIAM Review 41 (4) (1999) 637–676.

[68] P. Lancaster, K. Salkauskas, Surfaces generated by moving least squares methods, Mathematics of Computation 37 (155) (1981) 141–158.

[69] N. Sukumar, Construction of polygonal interpolants: a maximum entropy approach, International Journal for Numerical Methods in Engineering 61 (12) (2004) 2159–2181.

[70] F. Cirak, M. Ortiz, P. Schröder, Subdivision surfaces: a new paradigm for thin-shell finite-element analysis, International Journal for Numerical Methods in Engineering 47 (12) (2000) 2039–2072.

[71] A. Rosolen, D. Millán, M. Arroyo, On the optimum support size in meshfree methods: a variational adaptivity approach with maximum entropy approximants, International Journal for Numerical Methods in Engineering 82 (7) (2010) 868–895.

[72] A. Conn, N. Gould, P. Toint, A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds, SIAM J. Numer. Anal. 28 (2) (1991) 545–572.

[73] J. Nocedal, S. Wright, Numerical Optimization, Springer, USA, 1999.

[74] J.-S. Chen, C.-T. Wu, S. Yoon, Y. You, A stabilized conforming nodal integration for galerkin mesh-free methods, International Journal for Numerical Methods in Engineering 50 (2) (2001) 435–466.

[75] U. Seifert, K. Berndl, R. Lipowsky, Shape transformations of vesicles-phase-diagram for spontaneous-curvature and bilayer-coupling models, Physical Review A 44 (2) (1991) 1182–1202.

Appendix B
An adaptive meshfree method for phase-field models of biomembranes. Part II: A Lagrangian approach for membranes in viscous fluids.

C. Peco, A. Rosolen and M. Arroyo

Journal of Computational Physics,
249(0):320–336, 2013.

An adaptive meshfree method for phase-field models of biomembranes. Part II: a Lagrangian approach for membranes in viscous fluids

C. Peco, A. Rosolen[☆] and M. Arroyo[*]

LaCàN, Universitat Politècnica de Catalunya-BarcelonaTech (UPC), Barcelona 08034, Spain

Abstract

We present a Lagrangian phase-field method to study the low Reynolds number dynamics of vesicles embedded in a viscous fluid. In contrast to previous approaches, where the field variables are the phase-field and the fluid velocity, here we exploit the fact that the phase-field tracks a material interface to reformulate the problem in terms of the Lagrangian motion of a background medium, containing both the biomembrane and the fluid. We discretize the equations in space with maximum-entropy approximants, carefully shown to perform well in phase-field models of biomembranes in a companion paper. The proposed formulation is variational, lending itself to implicit time-stepping algorithms base on minimization of a time-incremental energy, which are automatically nonlinearly stable. The proposed method deals with two of the major challenges in the numerical treatment of coupled fluid/phase-field models of biomembranes, namely the adaptivity of the grid to resolve the sharp features of the phase-field, and the stiffness of the equations, leading to very small time-steps. In our method, local refinement follows the features of the phase-field as both are advected by the Lagrangian motion, and large time-steps can be robustly chosen in the variational time-stepping algorithm, which also lends itself to time adaptivity. The method is presented in the axisymmetric setting, but it can be directly extended to 3D.

Keywords: phase field models, biomembranes, vesicles, meshfree methods, variational methods, adaptivity

1. Introduction

Biomembranes self-assemble in a fluid, and often, the fluid mechanics are important in their dynamical behavior. Examples include the dynamics of vesicles in shear flows (see, e.g. [1, 2, 3, 4]), or the relaxation dynamics of membrane structures brought out-of-equilibrium [5, 6]. Describing explicitly the fluid surrounding biomembranes may also be useful in studying the interactions between membranes and other structures [7]. Here, we consider the simplest, yet very common and useful model of a biomembrane: an inextensible interface with curvature elasticity, given by the Helfrich energy. We ignore here the bilayer architecture, the monolayer

[☆]Current address: Institute for Soldier Nanotechnologies, MIT, Cambridge MA, USA.
[*]Correspondence to: marino.arroyo@upc.edu

extensibility, the surface viscosity, and the inter-monolayer friction, which can be important in some situations [8]. Our goal here is to develop a robust and efficient computational technique for biomembranes embedded in a viscous fluid, capable of handling arbitrarily large shape changes and the associated flows. We resort to phase-field models of biomembranes, and propose a non-conventional discretization of the membrane-fluid system. In a companion paper [9], we have shown that high-order phase-field models of biomembranes can be accurately approximated in a direct Galerkin approach with the maximum-entropy meshfree approximants [10], in an adaptive, accurate and efficient way. Here, we elaborate a Lagrangian method for the dynamics of vesicles embedded in a viscous fluid, which builds on the meshfree approximation of the phase-field equations. The proposed method shares common features with the optimal mass transport (OTM) method presented in [11].

Background

A number of models and numerical approaches have been proposed to analyze the hydrodynamics of fluid membranes in a viscous fluid. These include a mesoscopic model, combining a particle-based method for the fluid and a dynamically triangulated surface model for the membrane [3], which has been put forth to study the effect of membrane viscosity and thermal fluctuations in the dynamical behavior of vesicles in simple shear flow, as well as the behavior of vesicles and red blood cells in microcapillaries [12]. Other methods rely on conventional continuum mechanics models, e.g. [13, 14], where a sharp-interface Helfrich model coupled with a Lagrangian form of the Navier-Stokes Equations is discretized with finite elements. An alternative sharp-interface approach in three dimensions was presented in [15, 4], which relies on spherical harmonics representations of the vesicle shapes and fields on it, and on a boundary integral method for the Stokes flow. This method has been exercised in systems containing many interacting vesicles. Immersed boundary methods [16] represent an alternative approach to handle fluid-structure interaction, maintaining the Eulerian framework for the fluid media and the Lagrangian description for solid objects immersed in the flow. This family of methods have been applied to understand the hydrodynamic effects on fluid vesicles and biomembranes in [17, 18]. Phase-field models of vesicles [19] have also been coupled with the ambient hydrodynamics [20, 21], through an Eulerian description of the fluid with a source term of membrane elastic forces, and a transport equation to advect the phase-field representing the membrane. Alternative phase-field approaches to vesicles in a flow have been proposed in [22, 23], where the local area inextensibility was also accounted for, and in [24].

Phase-field models offer advantages when compared to sharp-interface models, in that they provide unified treatment of interface tracking and surface mechanics with a single partial differential equation (PDE) governing the phase-field. Phase-field approaches do not suffer from severe mesh distortions, and can easily deal with large deformations and even topology changes [22, 25] without demanding specific reparametrization techniques [13] or control of the tangential motions of the nodes [26]. In contrast, phase-field models are encoded by nonlinear PDEs, often high-order, which develop sharp features, and therefore present computational challenges. In phase-field models of biomembranes, an artificial length-scale ϵ governing the width of the smeared interface is introduced, and the sharp-interface limit is recovered as $\epsilon \to 0$ [19, 22, 27]. For phase-field models to accurately represent the sharp-interface limit, ϵ needs to be much

smaller than other relevant dimensions in the problem. Furthermore, this length-scale needs to be resolved by the computational grids, typically leading to expensive calculations. From a practical viewpoint, the high computational cost, associated with increasing the dimension of the problem and having to resolve numerically the (small) thickness of the smeared interface, can be outweighed by their simplicity, making them amenable to scalable parallel implementations. Besides, the computational cost can be considerably mitigated with spacial adaptivity [28, 29].

The proposed method

In previous phase-field approaches to the ambient hydrodynamics-biomembrane mechanics, the problem is formulated in a Eulerian frame, as a coupled system combining the fluid flow equations with a source term coming from curvature elasticity forces and the advection of the phase-field with the flow, in which the phase-field and the fluid velocity (and pressure) are the unknowns [22, 25]. In such approaches, adaptive strategies, not proposed so far, would require cumbersome grid projection steps. Since in the present situation the phase-field tracks a material interface, here we view the phase-field as a material property, and formulate a Lagrangian description of the problem in which the unknown is the Lagrangian motion of the background medium, containing both the fluid and the smeared interface (see Fig. 1 for an illustration). See [30] for a related approach. We particularize the model with the phase-field approach proposed by [19], and since biomembranes often operate in the limit of vanishing Reynolds number, describe the hydrodynamics with Stokes equations.

With the Lagrangian viewpoint, when discretized in space, the coupled membrane-fluid model becomes a nonlinear dissipative particle system, driven by curvature elasticity and dragged by a viscous force admitting a dissipation potential, whose dynamics minimize an action [31] subject to area and volume constraints. With the same spirit as variational integrators for Hamiltonian systems [32], we choose to discretize in time the action, and then derive by constrained minimization the discrete evolution equations from it, rather than discretizing in time the continuous evolution equations. The method results in time-incremental nonlinear minimization problems, as in modern treatments of dissipative processes in materials science following the seminal work in [33]. As a consequence, it is possible to overcome the stiffness of the dynamics (given by the fourth-order nature of the PDE) and take robustly large time-steps. Furthermore, the algorithm is automatically nonlinearly stable as the energy monotonically decreases.

If the initial grid adapts to the features of the phase-field, adaptivity is advected by the Lagrangian map, and therefore local refinement along the dynamics is accomplished for free (see Fig. 1). The Lagrangian framework allows us to pull-back the successive states of the system to a reference configuration. Thus, we avoid the calculation of the meshfree basis functions in every step of the evolution. It has been shown that the meshfree method considered here can withstand significant deformation before the discretized deformation mapping ceases to be injective (i.e. the Jacobian determinant becomes negative at a quadrature point) [10]. However, we avoid coming close to this limit, which degrades the accuracy of the approximation, by reconnecting the nodes, recomputing the basis functions, and resetting the reference configuration periodically along the simulation. These reconnection steps are seamless, as detailed later: they do not involve remeshing, recomputing the background grid for quadrature, field projections, nor do they alter in any way the variational structure of the discrete equations, e.g. the nonlinear stability of the

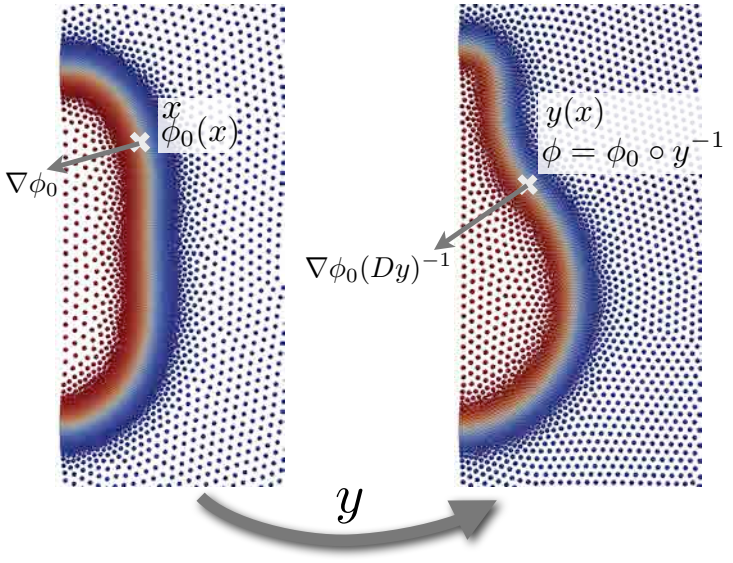


Figure 1: Main ideas behind the Lagrangian phase-field formulation. The background medium containing the viscous fluid and the smeared interface is rearranged by a deformation map $y(x)$, which deforms the phase-field, illustrated by a color map on the nodes of the computational grid. The phase-field is advected (pushed forward) as a material property as $\phi = \phi_0 \circ y$. The gradient of the deformed phase-field transforms as indicated, and as shown in the text, we can also compute $\Delta\phi$ as a push-forward of $\Delta\phi_0$. This allows us to write the Helfrich curvature energy in terms of y, and the viscous dissipation in terms of $\dot{y}$. Computationally, the deformation is discretized in terms of particle positions, indicated with colored circles, and the phase-field and its derivatives are sampled at fixed quadrature points in the reference configuration. As the Lagrangian simulation proceeds, the adaptivity follows the phase-field features.

dynamics. In situations involving extreme Lagrangian deformations, particles may accumulate or cover insufficiently parts of the domain. In such cases, a full re-meshing and field projections are required. We did not find the need to doing this, except in the example depicted in Figure 6.

From a purely numerical viewpoint, as exemplified later, accounting for the ambient fluid can help in devising adaptive strategies in space, resolving with detail the sharp and moving features of the phase-field, even if we are only interested in equilibria. Indeed, vesicles are prone to buckling events, i.e. large shape transformations under small changes of the enclosed volume or the spontaneous curvature as the system transitions between different metastable equilibrium branches. Consequently, if the grid is locally adapted to a given conformation [28] and the control parameters are slightly perturbed, the new energy minimizer may not be well described by the current mesh, strongly biased by the previous minimizer. This poses a serious challenge to adaptive phase-field methods based on free energy minimization. In contrast with possibly discontinuous equilibrium paths, dynamics are always continuous, making it possible to gradually adapt the resolution to the phase-field. Gradient flow dynamics, even without a

clear physical meaning [34, 27, 13], have been used to numerically obtain equilibrium shapes, and the adaptive method we proposed here can be used in this vein.

The structure of this paper is as follows. In Section 2, we derive the Lagrangian formulation for the phase-field membrane embedded in a viscous fluid, and obtain variationally the governing equations of the coupled dynamics. In Section 3, we propose the space and time discretization. In Section 4, we illustrate the method with several examples. Finally, we collect conclusions in Section 5.

2. Lagrangian phase-field formulation for biomembranes in a viscous fluid

2.1. Lagrangian form of the phase-field model

The formulation we present here is three-dimensional, and the particularization to axisymmetry is given in Appendix A. Consider a fixed fluid domain Ω, containing a fluid membrane described at time $t = 0$ by a phase-field $\phi_0(x)$. Such initial phase-field may result from an equilibrium calculation. Consider now a motion of the background continuum medium containing both the fluid and the smeared interface, i.e. a smooth bijective mapping on Ω at each instant of time, $y_t(x)$ [35, 36]. Viewing the phase-field as a material property, attached to the material particles, it is pushed forward by the motion following

$$\phi_t(x) = \phi_0 \circ y_t^{-1}(x) = \phi_0 \left(y_t^{-1}(x)\right). \tag{1}$$

From this point on, we omit the explicit dependence on t of the motion and the pushed-forward phase-field. Ignoring the Gaussian curvature term, the Helfrich elastic energy of the membrane in terms of the phase-field can be computed as [19]

$$E = \frac{3}{8\sqrt{2}}\frac{k}{2\epsilon}\int_\Omega \left[\epsilon\Delta\phi + \left(\frac{1}{\epsilon}\phi + C_0\sqrt{2}\right)\left(1-\phi^2\right)\right]^2 \, d\Omega,$$

where k is the bending stiffness of the bilayer, and ϵ is a regularization parameter controlling the width of the smeared interface. The enclosed volume and surface are can be computed as

$$V = \frac{1}{2}\left(Vol(\Omega) + \int_\Omega \phi \, d\Omega\right)$$

and

$$A = \frac{3}{2\sqrt{2}}\int_\Omega \left[\frac{\epsilon}{2}|\nabla\phi|^2 + \frac{1}{4\epsilon}(\phi^2-1)^2\right] \, d\Omega$$

To compute the spacial derivatives of the phase-field, we recall Eq. (1) and the inverse function theorem to obtain

$$\nabla\phi = \left(\nabla\phi_0 F^{-1}\right) \circ y^{-1}, \tag{2}$$

where $F = Dy$ is the deformation gradient, or $F_{iI} = \partial_I y_i$, where upper-case subindices denote indices or partial differentiation with respect to material (Lagrangian) coordinates, i.e. "x", while lower-case subindices refer to spacial (Eulerian) coordinates, i.e. "y". To compute the Laplacian of the pushed-forward phase-field, we resort to indicial notation and omit the composition with

the deformation map or its inverse as it can be inferred from the context. From the relation $\partial_i\phi = \partial_I\phi_0 F^{-1}_{Ii}$ we have

$$\partial^2_{ij}\phi \circ y = \partial^2_{IJ}\phi_0 F^{-1}_{Ii} F^{-1}_{Jj} + \partial_I\phi_0 \partial_j F^{-1}_{Ii}. \tag{3}$$

Now, from $F^{-1}_{Ik}F_{kJ} = \delta_{IJ}$, we obtain

$$\partial_j F^{-1}_{Ii} = -F^{-1}_{In}F^{-1}_{Ji}F^{-1}_{Kj}\partial_K F_{nJ} = -F^{-1}_{In}F^{-1}_{Ji}F^{-1}_{Kj}\partial^2_{JK} y_n.$$

In particular, we have

$$\Delta\phi \circ y = \partial^2_{IJ}\phi_0 F^{-1}_{Ii} F^{-1}_{Ji} + \partial_I\phi_0 \partial_i F^{-1}_{Ii}. \tag{4}$$

Thus, inserting these Eqs. (2,4) into the above functionals and pulling-back the integration by the deformation map to the Lagrangian domain, the elastic energy, enclosed volume, and surface area can be interpreted as functions of the deformation mapping alone, depending parametrically on the initial phase-field:

$$E[y] = \frac{3}{8\sqrt{2}}\frac{k}{2\epsilon}\int_\Omega \left[\epsilon\Delta\phi \circ y + \left(\frac{1}{\epsilon}\phi_0 + C_0\sqrt{2}\right)\left(1-\phi_0^2\right)\right]^2 \det(F)\; d\Omega, \tag{5}$$

$$V[y] = \frac{1}{2}\left(Vol(\Omega) + \int_\Omega \phi_0 \det(F)\; d\Omega\right), \tag{6}$$

and

$$A[y] = \frac{3}{2\sqrt{2}}\int_\Omega \left[\frac{\epsilon}{2}|\nabla\phi_0 F^{-1}|^2 + \frac{1}{4\epsilon}(\phi_0^2-1)^2\right]\det(F)\; d\Omega \tag{7}$$

Lengthy but otherwise straightforward calculations allow us to compute the variations of these functionals with respect to the deformation. We report directly the first and second derivatives of E, V and A after spacial Ritz-Galerkin discretization, required for the solution method, in Appendix B.

2.2. Lagrangian form of the fluid dissipation potential

To simplify the exposition of the method, we consider the Stokes equations for a slightly compressible fluid, i.e. a penalized formulation of the incompressible Stokes equations. The numerical treatment of the incompressible case with a stabilized maximum-entropy meshfree method is straightforward [37], but would distract from the main ideas of the present work.

Following standard continuum mechanics definitions, the Eulerian velocity field can be computed as

$$v = \partial_t y \circ y^{-1}.$$

Consequently, the velocity gradient tensor can be written as

$$\nabla v \circ y = \dot{F}F^{-1},$$

where $\dot{F}_{iI} = \partial_I \partial_t y_i$, and the rate-of-deformation tensor in the Lagrangian domain as

$$d \circ y = \frac{1}{2}\left(\dot{F}F^{-1} + F^{-T}\dot{F}^T\right).$$

The Rayleigh dissipation potential for a compressible Newtonian fluid can therefore be written as [38]

$$\begin{aligned} Diss[\partial_t y; y] &= \mu \int_\Omega d : d \; d\Omega + \frac{\lambda}{2} \int_\Omega (\text{div } v)^2 \; d\Omega \\ &= \frac{\mu}{4} \int_\Omega \left| \dot{F} F^{-1} + F^{-T} \dot{F}^T \right|^2 (\det F) \; d\Omega + \frac{\lambda}{2} \int_\Omega \left[\text{trace}(\dot{F} F^{-1}) \right]^2 (\det F) \; d\Omega. \end{aligned} \tag{8}$$

where μ is the shear viscosity of the fluid, and by $Diss[\partial_t y; y]$ we highlight the parametric dependence of the functional on the current deformation y. The coefficient λ can be interpreted here as a penalty parameter enforcing incompressibility approximately. For an incompressible Newtonian fluid, the second term above is replaced by the constraint

$$\text{tr}(\dot{F} F^{-1}) = 0, \tag{9}$$

the linearization of the condition $\det F = 1$.

2.3. Governing equations

The dynamics of the system can be obtained by minimizing the Rayleigh dissipation potential plus the rate of change of the elastic energy with respect to the variables expressing the rate of change of the system [31, 39, 8]. Here, the dynamics of the coupled membrane-fluid system are obtained by minimizing the functional

$$Diss[\partial_t y; y] + \delta E[\partial_t y; y]$$

with respect to $\partial_t y$ subject to the constraint

$$\delta A[\partial_t y; y] = 0.$$

To control the enclosed volume explicitly, without relying on the fluid (quasi)-incompressiblity, the following constraint can be added

$$\delta V[\partial_t y; y] = 0.$$

We note that this formulation enforces the global area preservation along the dynamics, while physically, local area preservation throughout the membrane is more meaningful, see [4, 8] and [23] for the phase-field modeling the local constraint. Again, for the sake of clarity, we stick here to the global constraint as in [21].

3. Discrete equations

3.1. Space discretization: function approximation and quadrature

We start from a node set $X = \{x^1, x^2, \dots, x^N\}$ adapted to the reference phase-field ϕ_0, and define the associated local maximum-entropy basis functions $p_a(x), a = 1, \dots, N$, for a given aspect ratio parameter γ [40, 9]. We also consider a set of quadrature points $\hat{X} = \{\hat{x}^1, \hat{x}^2, \dots, \hat{x}^Q\}$ and the associated quadrature weights $W = \{w^1, w^2, \dots, w^Q\}$, obtained for instance from a

triangulation of the node set X. The quadrature points and weights are only set up once in the calculation, and are subsequently transported by the motion. They are not altered by the node reconnection steps, see below. The gradient and the Hessian of the reference phase-field can be obtained through interpolation with the smooth meshfree basis functions:

$$\phi_0(x) = \sum_a p_a(x)\phi_0^a, \quad \partial_I\phi_0(x) = \sum_a \partial_I p_a(x)\phi_0^a, \quad \partial^2_{IJ}\phi_0(x) = \sum_a \partial^2_{IJ} p_a(x)\phi_0^a,$$

which only need to be evaluated at the quadrature points in $\hat{X}$ to yield the values ϕ_0^α, $\partial_I\phi_0^\alpha$, and $\partial^2_{IJ}\phi_0^\alpha$ for $\alpha = 1, 2, \ldots, Q$. If computer memory is not an issue, these objects only need to be evaluated once at the beginning of a simulation. The motion is represented numerically as

$$y_t(x) = \sum_a p_a(x) y^a(t). \tag{10}$$

At the initial instant we have $y^a(0) = x^a$, and as a result of the linear consistency of the approximants, $y_0(x) = x$. Furthermore, we have

$$\partial_t y = \sum_a p_a(x)\dot{y}^a, \quad F_{iI} = \sum_a \partial_I p_a(x) y_i^a, \quad \partial^2_{JK} y_n = \sum_a \partial_{JK} p_a(x) y_n^a.$$

Recalling Eqs. (2,4), and approximating the integrals by numerical quadrature, the functionals in Eqs. (5), (6) and (7) can be calculated, and expressed as functions of the nodal positions, $E^h(\boldsymbol{y})$, $V^h(\boldsymbol{y})$, and $A^h(\boldsymbol{y})$, where the array $\boldsymbol{y}$ collects all the nodal positions $y^a(t)$. These functions depend parametrically on $\hat{\phi}_0^\alpha$, $\partial_I\hat{\phi}_0^\alpha$, and $\partial^2_{IJ}\hat{\phi}_0^\alpha$ for $\alpha = 1, 2, \ldots, Q$.

Likewise, by replacing the numerical ansatz into Eq. (8), the dissipation potential can be written as

$$Diss^h(\dot{\boldsymbol{y}}; \boldsymbol{y}) = \frac{1}{2}\left[\mu K^\mu_{ai,bj}(\boldsymbol{y}) + \lambda K^\lambda_{ai,bj}(\boldsymbol{y})\right] \dot{y}_i^a \dot{y}_j^b,$$

where

$$K^\mu_{ai,bj}(\boldsymbol{y}) = \int_\Omega \left(\delta_{ij} F^{-1}_{Ik}\partial_I p_a F^{-1}_{Jk}\partial_J p_b + F^{-1}_{Ij}\partial_I p_a F^{-1}_{Ji}\partial_J p_b\right)(\det F)\, d\Omega,$$

and

$$K^\lambda_{ai,bj}(\boldsymbol{y}) = \int_\Omega F^{-1}_{Ii}\partial_I p_a F^{-1}_{Jj}\partial_J p_b (\det F)\, d\Omega.$$

Then, the dynamics of the resulting nonlinear dissipative particle system follow from minimizing

$$\frac{1}{2}\dot{\boldsymbol{y}}^T \boldsymbol{K}(\boldsymbol{y})\dot{\boldsymbol{y}} - \boldsymbol{f}_E(\boldsymbol{y})^T\dot{\boldsymbol{y}}$$

with respect to $\dot{\boldsymbol{y}}$ subject to

$$\left[\nabla A^h(\boldsymbol{y})\right]^T \dot{\boldsymbol{y}} = 0,$$

where elastic forces are defined as $\boldsymbol{f}_E(\boldsymbol{y}) = -\nabla E^h(\boldsymbol{y})$ and we note that by the chain rule $\dot{E} = -\boldsymbol{f}_E(\boldsymbol{y})^T\dot{\boldsymbol{y}}$. To account for the constraint, we write the Lagrangian

$$\mathcal{L}(\dot{\boldsymbol{y}}, \sigma; \boldsymbol{y}) = \frac{1}{2}\dot{\boldsymbol{y}}^T \boldsymbol{K}(\boldsymbol{y})\dot{\boldsymbol{y}} - \boldsymbol{f}_E(\boldsymbol{y})^T\dot{\boldsymbol{y}} + \sigma\left[\nabla A^h(\boldsymbol{y})\right]^T \dot{\boldsymbol{y}},$$

where σ is the membrane tension, which leads to the system

$$\begin{bmatrix} \boldsymbol{K}(\boldsymbol{y}) & \nabla A^h(\boldsymbol{y}) \\ \left[\nabla A^h(\boldsymbol{y})\right]^T & 0 \end{bmatrix} \begin{pmatrix} \dot{\boldsymbol{y}} \\ \sigma \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_E(\boldsymbol{y}) \\ 0 \end{pmatrix}.$$

This system of nonlinear differential algebraic equations can be solved with standard algorithms. The system is stiff because of the nature of the curvature energy, and because of the presence of constraints. We find that standard numerical packages have serious difficulties in dealing with these equations, and require very small time-steps when the system is significantly out of equilibrium. Instead, we develop next variational time-incremental integrators, which can robustly deal with large time-steps.

3.2. Variational time discretization

The time-discretization in the previous section is performed on time-continuous evolution equations derived from minimizing an action subject to constraints. Here, we adopt an alternative viewpoint, by first discretizing in time the action, and then minimizing the time-discrete action with respect to the configuration of the system at time-step $n+1$.

Let us consider the simplest finite difference approximations for the rate of change of the nodal positions

$$\dot{\boldsymbol{y}} \approx \frac{\boldsymbol{y}^{n+1} - \boldsymbol{y}^n}{\Delta t},$$

and for the rate of change of the energy

$$\dot{E} = \frac{E(\boldsymbol{y}^{n+1}) - E(\boldsymbol{y}^n)}{\Delta t}.$$

We can then discretize in time the action, and given $\boldsymbol{y}^n$ find $\boldsymbol{y}^{n+1}$ by minimizing

$$\frac{1}{2}(\boldsymbol{y} - \boldsymbol{y}^n)^T \boldsymbol{K}(\boldsymbol{y}^n)(\boldsymbol{y} - \boldsymbol{y}^n) + \Delta t E(\boldsymbol{y}) \tag{11}$$

with respect to $\boldsymbol{y}$, subject to

$$A^h(\boldsymbol{y}) = A_0,$$

where we have multiplied the action by Δt^2 and ignored the constant $E(\boldsymbol{y}^n)$ in Eq. (11). This method is related to the backward-Euler method, and many other variational time-integrators can be defined by choosing different time-discrete approximations of the action. The resulting nonlinear optimization program can be solved with a variety of methods. Here, we impose the constraints with Lagrange multipliers and solve the first order optimality conditions with Newton's method, although an augmented Lagrangian method, combined with line-search may be more robust at very large time-steps. Note that for large time-steps, the objective function in Eq. (11) is dominated by the curvature energy, and the system nearly minimizes this energy in one step. On the contrary, small time-steps give more weight to the viscous dissipation, penalizing changes in the configuration of the system. Even though $E(\boldsymbol{y})$ has in general a complex, non-convex landscape, for a sufficiently small time-step, the viscous dissipation makes the objective function convex, and hence the nonlinear optimization problem becomes easier

to solve. We also note that, by construction, $E(\boldsymbol{y}^{n+1}) \leq E(\boldsymbol{y}^n)$, and therefore the method is endowed automatically with nonlinear stability. Of course, the issue may be being able to numerically solve the nonlinear optimization problem.

We make the algorithm explicit in the dissipation matrix $\boldsymbol{K}(\boldsymbol{y})$, as otherwise the method is significantly more complex and most of the nonlinearity is in $E(\boldsymbol{y})$. In practical applications, we often update the Hessian of $E(\boldsymbol{y})$ once every time-increment, instead of in each iteration of Newton's method. We find that this significantly reduces the computational cost without affecting much the convergence of Newton's method. The treatment of the surface area constraint may be simplified by discretizing in time the linearized constraint, $\left[\nabla A^h(\boldsymbol{y}^n)\right]^T (\boldsymbol{y} - \boldsymbol{y}^n) = 0$. However, this option leads to significant drifts in the surface area for large time-steps. Finally, we note that adaptive time-stepping algorithms can be easily designed, for instance adapting Δt in such a way that ΔE is nearly constant. The adaptivity may also be driven by the number of iterations needed in the nonlinear solver.

3.3. Numerical quadrature and node reconnection

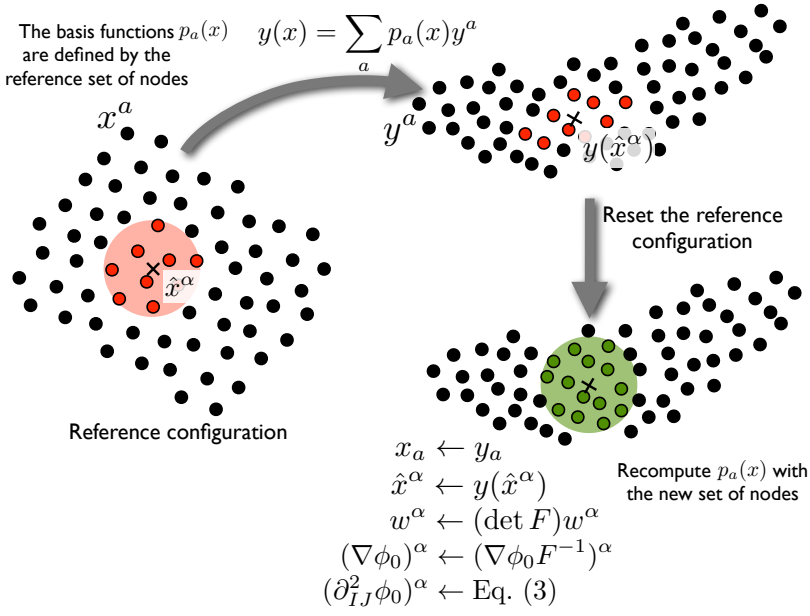


Figure 2: As the Lagrangian simulation proceeds, the deformation may significantly distort the domain. To avoid this, we periodically reset the reference configuration, as shown in the figure. This involves reseting the reference node position to the current position, recomputing the meshfree basis functions from the new node set, which involves new neighbor searches as indicated with the colored regions, and reseting the quadrature points $\hat{x}^\alpha$, the corresponding weights, and the reference phase-field first and second derivatives as indicated in the figure. Note that the reference phase-field value at the quadrature points, ϕ_0^α, does not need to be updated as the phase-field is a material property and the quadrature points keep their material identity.

As discussed in Fig. 2, the evolution most likely produces large distortions, which may even lead to non-injective deformation mappings in Eq. (10), i.e. negative Jacobian determinants $\det F$. Even if the deformation map remains injective, it is a good idea to avoid excessive distortions, which degrade the accuracy of the approximation. For this reason, we periodically reset the reference configuration, reconnect the nodes, and build new basis functions to parameterize the deformation maps from the new reference configuration. The reconnection can be done when a measure of the distortion (a norm of F) exceeds a threshold, or simply every a fixed number of time steps. Numerical experience shows that frequent reconnection leads to better numerical accuracy, but also that the method is very robust and can deliver acceptable results with very few reconnections. In practice, the frequency of reconnection can be set by weighing accuracy and efficiency, although objective criteria would be desirable.

Note carefully that the reconnection procedure does not require any projection of fields if the multiplicative structure of the composition of maps is exploited. Indeed, suppose that the deformation map in the motion at which we decide to reset the reference configuration is $y(x) = \sum_a p_a(x) y^a$, and its deformation gradient F. The new node set is simply $\{y^1, \dots, y^N\}$ and the new quadrature points are $y(\hat{x}^\alpha), \quad \alpha = 1, \dots, Q$. The value of the phase-field at these material points is simply the original value $\phi_0(\hat{x}^\alpha)$ since the phase-field is viewed as a material property. Its derivatives need to be updated with the formulas seen previously, but no new interpolation of the phase-field is needed and no new quadrature needs to be defined. The reset algorithm is sketched in Fig. 2. Remarkably, such reset of the reference configuration exactly preserves the elastic energy, area, and enclosed volume of the system, as can be understood from examining Eqs. (5,6,7).

4. Numerical examples

We present next a set of numerical simulations to test the proposed method. We first illustrate the general performance of the method with regards to space and time adaptivity. The former is automatic if the initial grid adapts to the interface, while for the second we set the time-step such that the energy decrement per step is roughly constant. We compare the proposed variational time integration with explicit Euler time-stepping. We then describe three representative simulations of relaxation dynamics of vesicles initially placed out-of-equilibrium, showing large shape changes, and requiring multiple node reconnections. Finally, we evaluate kinetic effects on the shape trajectory by deflating a vesicle at different rates. In all examples, we consider 6 integration points per cell (the triangles of the Delaunay triangulation of the the initial set of nodes) and an aspect ratio parameter for the maximum-entropy basis functions of $\gamma = 0.8$ [9]. The regularization parameter is chosen as about 1% of the size of the vesicle, i.e. $\epsilon/\sqrt{A_0/(4\pi)} = 0.01$.

Figure 3 shows (with a movie/with a collection of snapshots) the relaxation dynamics of an oblate vesicle brought out-of-equilibrium. The reduced volume, a non dimensional measure of the volume to area ratio, is $v = 0.9$. We show the location of the nodes $y^a(t)$, and color-code them by the value of the phase-field. In this example, exhibiting moderate deformations, we do not reconnect the nodes and therefore the method is purely Lagrangian, with the initial configurations as a reference configuration during the whole motion. The calculation proceeds robustly despite the large deformations. It can be appreciated how the phase-field elastic energy

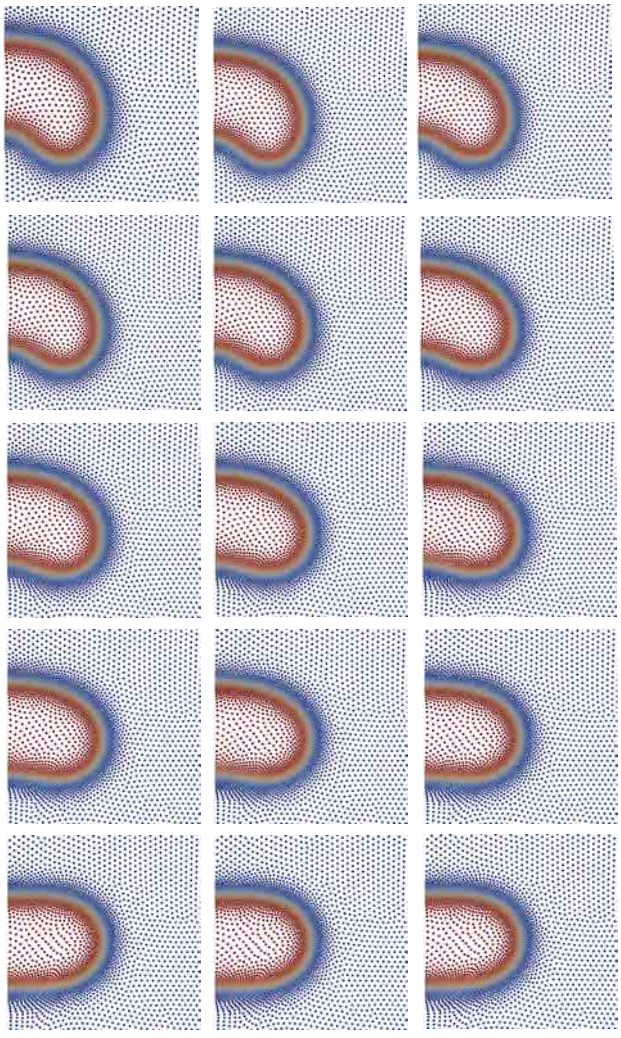


Figure 3: Relaxation dynamics of an oblate vesicle in a viscous fluid, initially brought out-of-equilibrium. We represent the time-evolution of the nodes $y^a(t)$, color-coded with the phase-field. The adapted grid has 6124 nodes.

maintains the transversal density of the nodes, and how the adapted region follows the features of the phase-field. We check the accuracy of this simulation with additional runs with a larger number of integration points, more nodes, and different γ parameters.

The performance of the method is analyzed in Fig. 4. The left plot shows the non-dimensional

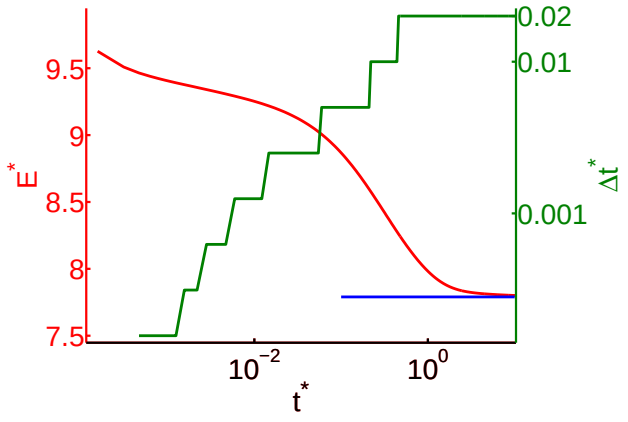


Figure 4: Energy relaxation and time adaptive strategy for the dynamics depicted in Fig. 3. Energy and time-step evolution, where time is represented in logarithmic scale. The blue horizontal line shows the equilibrium energy obtained independently with a parametric method based on B-Splines.

energy $E^* = E/k$ and the non-dimensional time-step as a function of non-dimensional time $t^* = tk/(\mu R_0^3)$. The energy monotonically decreases as expected, converging towards the equilibrium energy calculated independently with a parametric method. As the process advances, the adaptive time-step grows to roughly keep the energy decrement per time-step constant. Remarkably, the time-step changes by two orders of magnitude during the simulation. At the final stages, the time-step hits the maximum allowed size.

Table 1: Elastic energy and computational cost for different constant time-steps and methods (VTI: variational time-integration, FE: forward Euler). $t_1^* = 1.0 \cdot 10^{-3}$, $t_2^* = 1.1 \cdot 10^{-2}$.

Method	Δt^*	$E^*(t_1^*)$	$E^*(t_2^*)$	*steps*	*grad*	*hess*
VTI	$1.0 \cdot 10^{-2}$	9.374	9.243	1	2	2
VTI	$1.0 \cdot 10^{-3}$	9.374	9.240	10	20	10
VTI	$1.0 \cdot 10^{-4}$	9.374	9.239	100	200	100
FE	$1.0 \cdot 10^{-5}$	9.374	9.239	1000	1000	0

Although more sophisticated time-stepping schemes are possible, as compare the proposed variational time-integration (VTI) method with an explicit forward Euler (FE) method. It is computationally infeasible to perform the full relaxation dynamics with the forward Euler method, which imposes very stringent conditions on the time-step. Instead, we focus on a portion of the dynamics, and report the results in Table 1. In the VTI method, we use Newton's method to numerically solve the optimization problem in Eq. (11), and for computational efficiency, update the Hessian matrix only once per time-step, not per iteration. However, for the largest time-step, we need to update the Hessian in each iteration for convergence. In all cases, Newton's method converges in two iterations. The table compares the VTI method with time-steps $\Delta t^* =$

$10^{-2}, 10^{-3}, 10^{-4}$, and the FE method with the largest time-step for stability in this interval, $\Delta t^* = 10^{-5}$. The accuracy is reported in terms of the energy at the end of the interval, and the computational cost in terms of gradient and Hessian evaluations. The table shows the ability of VTI to robustly take large time-steps with accurate results. In contrast, we find that for this nonlinear system, it is very difficult to stably adjust the time-step length in the FE method. We find that the VTI method provides a similar accuracy to the explicit method with time-steps between one and two orders of magnitude larger. This ratio is even more dramatic in the initial fast stages of the dynamics.

We next exercise the method in more challenging dynamics, involving large shape changes. Figure 5 (left) shows a stomatocyte-discocyte dynamical transition. For the considered reduced volume $v = 0.6$, both a stomatocyte and a discocyte are metastable configurations, the latter having lower energy. We slightly displace the stomatocyte equilibrium configuration beyond the energy barrier, and then the system spontaneously evolves towards the discocyte configuration. The reference configuration is reset, as illustrated in Fig. 2, when large distortions occur as measured with the gradient of deformation mapping. In this simulation, the reference configuration is reset every 20 time-steps. The time-adaptive scheme allows us to efficiently track the entire transition, and by the end of the simulation the time-step is 2,048 times the initial time-step. Figure 5 (right) shows the response of an prolate vesicle ($v = 0.8$) subject to an instantaneous change of spontaneous curvature from $C_0 = 0$ to $C_0\sqrt{A_0/(4\pi)} = 10.0$, which can be the result of exposing the bilayer to a different chemical environment [5, 6, 27]. The system evolves towards a configuration consisting of two dissimilar spheres connected by a narrow neck, which best adjusts to the imposed spontaneous curvature with the available volume. Both simulations run on a CVT adapted grid of 6,124 nodes.

Figure 6 shows an even more dramatic shape change, in which a prolate vesicle is deflated from $v = 0.9$ to $v = 0.55$ and its spontaneous curvature increased to $C_0\sqrt{A_0/(4\pi)} = 12.0$, leading to an elongation and pearling transformation, widely observed in experiments [41]. The method robustly follows all the large shape deformations with an adapted grid of 12,650 nodes. This simulation requires frequent nodal reconnection, and even four complete re-meshing steps at later stages, in which a new grid is built and adapted to the current phase-field and the phase-field is projected onto the new grid.

Finally, we present a series of simulations highlighting kinetic effects. By subjecting a vesicle to fast changes (here a volume decrease rate), the system follows an out-of-equilibrium path that significantly deviates from the quasi-static response. We then fix the enclosed volume, and let the system relax towards equilibrium. In Fig. 7, we report the response of the system to three different volume decrease rates in terms of elastic energy evolution and shape at the instant of maximum energy for each evolution, which corresponds to the end of the deflection process. It can be observed that, due to the fluid dissipative forces, the faster the dynamics, the further apart is the shape at this instant from the equilibrium shape (D), eventually reached by all the simulations for $T^* \approx 1.00$. Also, the faster the rate, the larger the deviation between the elastic energy at this instant and the elastic energy in equilibrium. In principle, kinetic effects such as those reported here could assist in the transition to a different equilibrium branch, and bring the system to a qualitatively different equilibrium configuration. We are currently exploring such phenomena.

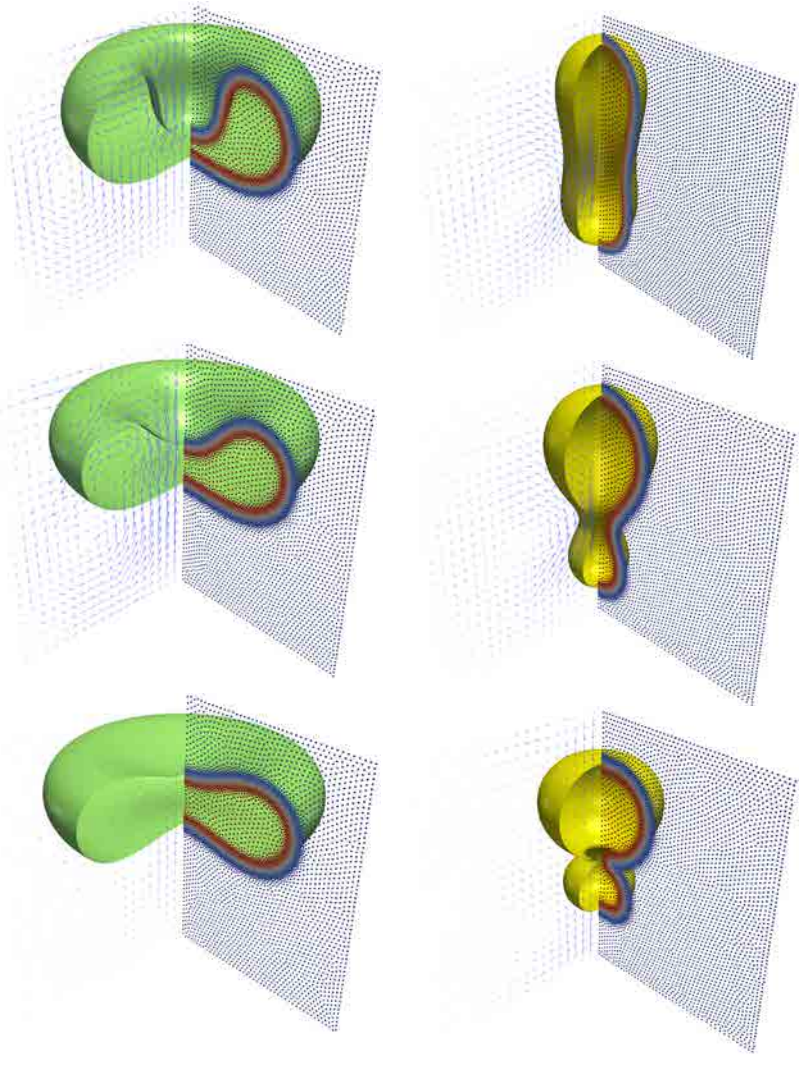


Figure 5: Left: Stomacyte-discocyte transition. Right: Prolate vesicle evolving after an instantaneous change of spontaneous curvature (6,124 nodes, constant area and volume). The points represent the nodes, color-coded with the phase-field, while the arrows depict the flow field in a symmetry plane.

5. Conclusions

We have proposed an adaptive meshfree Galerkin method to numerically approximate the dynamics phase-field models of biomembranes embedded in a viscous fluid. We have shown the

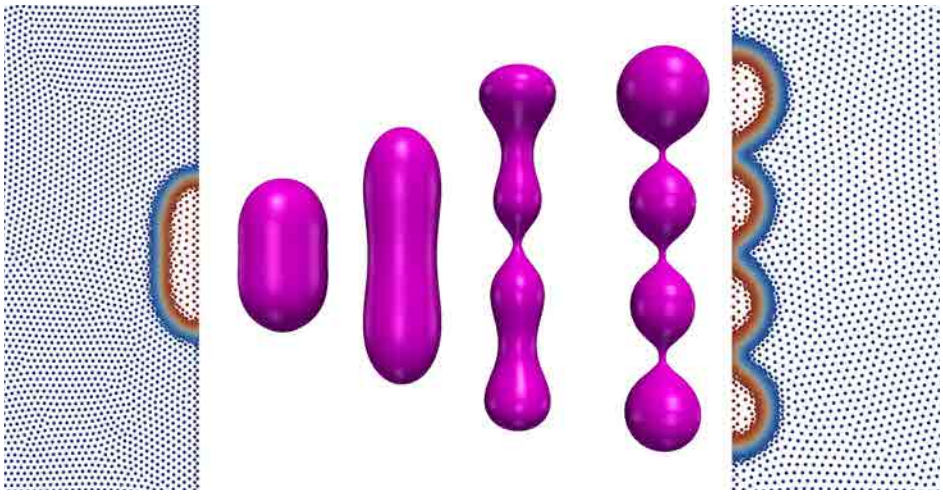


Figure 6: Relaxation dynamics of a constant area vesicle under combined volume decrease and spontaneous curvature increase (12,650 nodes, constant area).

ability of the proposed method, based on smooth approximants, to deal with the high order character of the equations in a direct manner. Furthermore, adaptivity is very natural for a meshfree method, and proves essential to resolve the sharp features of the phase-field model at an affordable cost. We have presented an original Lagrangian and variational formulation of the coupled fluid-membrane dynamics, which lends itself to efficient and robust time integrators based on time-incremental minimization problems. In this method, the local refinement follows naturally the sharp features of the phase-field. This combination of methods shows promise of robust, scalable computations of complex membrane systems in three dimensions, currently under development.

Acknowledgments

We acknowledge the support of the European Research Council under the European Community's 7th Framework Programme (FP7/2007-2013)/ERC grant agreement nr 240487, and of the Ministerio de Ciencia e Innovacin (DPI2011-26589). MA acknowledges the support received through the prize "ICREA Academia" for excellence in research, funded by the Generalitat de Catalunya. CP acknowledges FPI-UPC Grant, FPU Ph. D. Grant (Ministry of Science and. Innovation, Spain) and Col·legi d'Enginyers de Camins, Canals i Ports de Catalunya for their support.

Appendix A. Cylindrical coordinates

We consider cylindrical coordinates, but assume that there is no angular dependence of any function along the angular direction. We have $x = (R, Z, \Theta)$, and $y(x) = (r(R, Z), z(R, Z), \Theta)$.

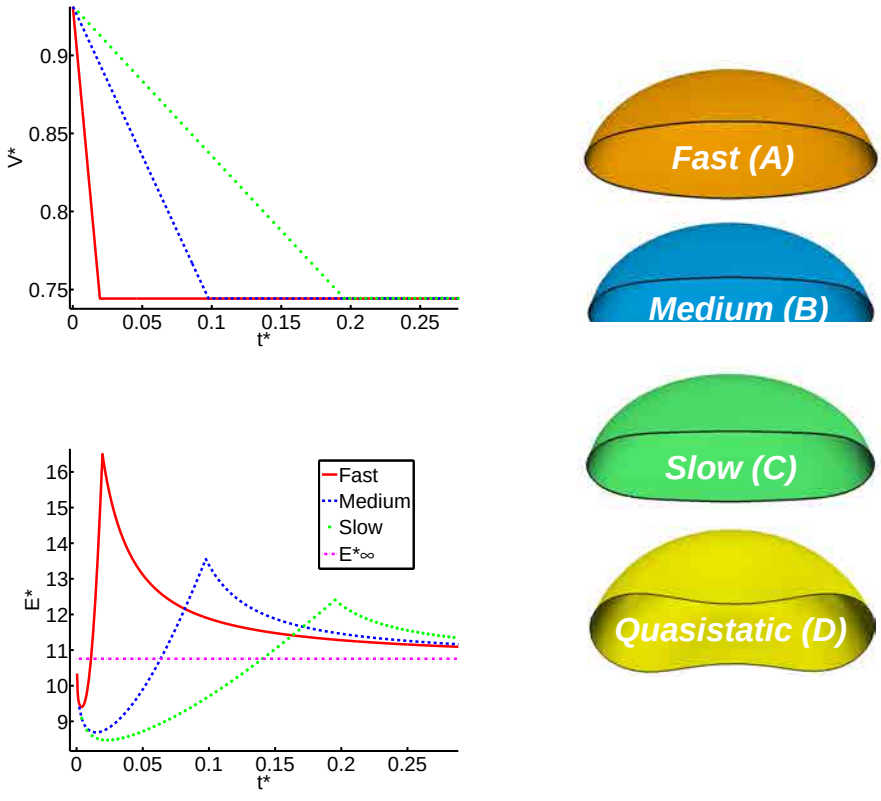


Figure 7: Kinetic effects. Top-left: Enclosed volume evolution for the three volume decrease rates considered. Bottom-left: Energy evolution for the three rates. Right: Shapes at the end of the volume reduction, therefore enclosing the same volume, and equilibrium shape (D) for this enclosed volume. (6,124 nodes, constant area).

The metric tensors of the reference and the deformed coordinate systems are

$$G_{IJ} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & R^2 \end{pmatrix}, \quad g_{ij} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & r^2 \end{pmatrix}.$$

It follows immediately that the volume element can be written as $dv = rdrdzd\theta$, $dV = RdRdZd\Theta$. The deformation gradient becomes

$$F^i{}_I = \begin{pmatrix} r_{,R} & r_{,Z} & 0 \\ z_{,R} & z_{,Z} & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

where the comma denotes partial differentiation. We denote by $\tilde{F}$ its first 2×2 minor. The Cauchy-Green deformation tensor can be written as $C^I{}_J = F^i{}_K F^j{}_J g_{ij} G^{IK}$ [35]. Consequently,

exploiting the block structure,

$$J = \sqrt{\det C^I{}_J} = (r/R) \det \tilde{F}.$$

Thus, we have

$$dv = J \; dV = r(\det \tilde{F}) \; dR dZ d\Theta.$$

For a function $\phi_0(R, Z)$, the gradient and the differential components coincide

$$\phi_{0,J} G^{IJ} = \phi_{0,I} = \begin{pmatrix} \phi_{0,R} \\ \phi_{0,Z} \\ 0 \end{pmatrix}.$$

By the chain rule, recalling that $\phi = \phi_0 \circ y^{-1}$, we have

$$\phi_{,i} = (F^{-1})^I{}_i \; \phi_{0,I},$$

which reduces in the (R, Z) plane to

$$\tilde{\phi}_{,i} = (\tilde{F}^{-1})^I{}_i \; \tilde{\phi}_{0,I},$$

owing to the block structure of the formation gradient and the fact that $\phi_{0,\theta} = 0$.

To compute covariant derivatives of vector fields and one-forms

$$v^i{}_{|j} = v^i{}_{,j} + \gamma^i_{jk} v^k, \qquad \alpha_{i|j} = \alpha_{i,j} - \gamma^k_{ij} \alpha_k,$$

we need the connection coefficients, which can be computed from

$$\gamma^i_{jk} = (1/2) g^{il} (g_{lj,k} + g_{lk,j} - g_{jk,l}).$$

The only non-zero components are

$$\gamma^r_{\theta\theta} = -r, \qquad \gamma^\theta_{r\theta} = \gamma^\theta_{\theta r} = 1/r.$$

For a vector field $v(r, z) = (v_r(r, z), v_z(r, z), 0)$, we can compute the covariant derivative

$$(\nabla v)^i{}_j = v^i{}_{|j} = \begin{pmatrix} v^r{}_{,r} & v^r{}_{,z} & 0 \\ v^z{}_{,r} & v^z{}_{,z} & 0 \\ 0 & 0 & v^r/r \end{pmatrix},$$

and taking the trace, its divergence

$$\text{div } v = v^r{}_{,r} + v^z{}_{,z} + v^r/r.$$

Now, from the above expressions, we can compute the Hessian of ϕ as

$$\phi_{|ij} = \phi_{,ij} - \gamma^l_{ij} \phi_{,l}.$$

Again, it has a block diagonal structure

$$\phi_{|ij} = \left(\begin{array}{c|c} \tilde{\phi}_{|ij} & 0 \\ \hline 0 & \phi_{|\theta\theta} \end{array}\right).$$

In the first 2×2 minor, the Cartesian structure given in Eq. (3) is preserved, while in the $\theta\theta$ component we have

$$\phi_{|\theta\theta} = r\phi_{,r} = r\left[(F^{-1})^R{}_r \phi_{0,R} + (F^{-1})^Z{}_r \phi_{0,Z}\right].$$

The Laplacian is computed correspondingly as

$$\Delta\phi = \phi_{|ij} g^{ij} = \Delta\tilde{\phi} + \phi_{,r}/r = \Delta\tilde{\phi} + (1/r)\left[(F^{-1})^R{}_r \phi_{0,R} + (F^{-1})^Z{}_r \phi_{0,Z}\right].$$

The Lagrangian expression of the rate-of-deformation tensor can be computed as [35]

$$2D_{IJ} = g_{ik}\left(V^k{}_{|I} F^i{}_J + V^i{}_{|J} F^k{}_I\right),$$

where the covariant derivative of the material velocity is defined as

$$V^i{}_{|J} = V^i{}_{,J} + \gamma^i_{jk} V^j F^k{}_J.$$

This tensor is simply

$$V^i{}_{|J} = \begin{pmatrix} V^r{}_{,R} & V^r{}_{,Z} & 0 \\ V^z{}_{,R} & V^z{}_{,Z} & 0 \\ 0 & 0 & V^r/r \end{pmatrix},$$

which leads to

$$2D_{IJ} = \left(\begin{array}{c|c} \tilde{V}^k_{,I}\tilde{F}^k_{,J} + \tilde{V}^k_{,J}\tilde{F}^k_{,I} & 0 \\ \hline 0 & 2rV^r \end{array}\right) = \left(\begin{array}{c|c} \dot{\tilde{F}}^T\tilde{F} + \tilde{F}^T\dot{\tilde{F}} & 0 \\ \hline 0 & 2rV^r \end{array}\right).$$

Now, noting that the Eulerian rate-of-deformation tensor can be computed as $d_{ij} \circ y = D_{IJ}(F^{-1})^I{}_i (F^{-1})^J{}_j$ [35], we have

$$2d_{ij} \circ y = \left(\begin{array}{c|c} \tilde{F}^{-T}\dot{\tilde{F}}^T + \dot{\tilde{F}}\tilde{F}^{-1} & 0 \\ \hline 0 & 2r\dot{r} \end{array}\right).$$

Its trace can be computed as

$$(\text{div } v) \circ y = (d_{ij} g^{ij}) \circ y = (1/2)\text{trace}(\tilde{F}^{-T}\dot{\tilde{F}}^T + \dot{\tilde{F}}\tilde{F}^{-1}) + \dot{r}/r,$$

while its norm squared is

$$|d \circ y|^2 = (d_{ij} d_{kl} g^{ik} g^{jl}) \circ y = (1/4)|\tilde{F}^{-T}\dot{\tilde{F}}^T + \dot{\tilde{F}}\tilde{F}^{-1}|^2 + (\dot{r}/r)^2.$$

Appendix B. Derivatives for gradient and Hessian of the energy, volume, and area

The main expressions and derivatives required to implement the proposed algorithm are presented in this appendix. We start with the derivatives of the motion, then move to nodal derivatives involving the gradient, and finish with the Hessian, used in Newton's method. Lighter gray symbols correspond to terms required in the axisymmetric formulation, which just need to be dropped in 3D.

Appendix B.1. Spacial derivatives of the motion

The motion is discretized as

$$y(x,t) = \sum_{a=1}^{N} p^a(x) y^a(t)$$

From now on, we ignore the arguments of the basis functions and nodal values for simplicity, $y = \sum_{a=1}^{N} p^a y^a$. We then have for the deformation gradient

$$F_{iI} = \sum_{a=1}^{N} \partial_I p^a y_i^a, \qquad \partial_J F_{iI} = \sum_{a=1}^{N} \partial_I \partial_J p^a y_i^a, \qquad \partial_J F_{Ii}^{-1} = -F_{Il}^{-1} F_{Ki}^{-1} \partial_J F_{lK}.$$

Appendix B.2. Nodal derivatives of the motion

Nodal derivative of F and F^{-1}

$$\partial_{y_k^b} F_{iI} = \sum_{a=1}^{N} \partial_I p^a \partial_{y_k^b} y_i^a = \partial_I p^b \partial_{y_k^b} y_i^b = \partial_I p^b \delta_{ik}, \qquad \partial_{y_k^b} F_{Ii}^{-1} = -F_{Il}^{-1} \partial_{y_k^b} F_{lK} F_{Ki}^{-1}.$$

Nodal derivative of ∇F

$$\partial_{y_k^b} \partial_J F_{iI} = \sum_{a=1}^{N} \partial_I \partial_J p^a \partial_{y_k^b} y_i^a = \partial_I \partial_J p^b \partial_{y_k^b} y_i^b = \partial_I \partial_J p^b \delta_{ik}.$$

Nodal derivative of ∇F^{-1}

$$\partial_{y_k^b} \partial_J F_{iI}^{-1} = -(\partial_{y_k^b} F_{Il}^{-1} F_{Ki}^{-1} \partial_J F_{iI} + F_{Il}^{-1} F_{Ki}^{-1} \partial_{y_k^b} \partial_J F_{iI} + F_{Il}^{-1} \partial_{y_k^b} F_{Ki}^{-1} \partial_J F_{iI}).$$

Nodal derivative of $\det F$

$$\partial_{y_k^b} \det F = \det F (F_{Ii}^{-1} \partial_{y_k^b} F_{iI})$$

Second nodal derivative of F^{-1}

$$\partial_{y_j^a} \partial_{y_k^b} F_{Ii}^{-1} = -(\partial_{y_j^a} F_{Il}^{-1} \partial_{y_k^b} F_{lK} F_{Ki}^{-1} + F_{Il}^{-1} \partial_{y_k^b} F_{lK} \partial_{y_j^a} F_{Ki}^{-1})$$

Second nodal derivative of ∇F^{-1}

$$\begin{aligned} \partial_{y_j^a} \partial_{y_k^b} \partial_J F_{Ii}^{-1} = -(&\partial_{y_j^a} \partial_{y_k^b} F_{Il}^{-1} F_{Ki}^{-1} \partial_J F_{lK} + \partial_{y_k^b} F_{Il}^{-1} \partial_{y_j^a} F_{Ki}^{-1} \partial_J F_{lK} + \partial_{y_k^b} F_{Il}^{-1} F_{Ki}^{-1} \partial_{y_j^a} \partial_J F_{lK} + \\ &\partial_{y_j^a} F_{Il}^{-1} F_{Ki}^{-1} \partial_{y_k^b} \partial_J F_{lK} + F_{Il}^{-1} \partial_{y_j^a} F_{Ki}^{-1} \partial_{y_k^b} \partial_J F_{lK} + \\ &\partial_{y_j^a} F_{Il}^{-1} \partial_{y_k^b} F_{Ki}^{-1} \partial_J F_{lK} + F_{Il}^{-1} \partial_{y_j^a} \partial_{y_k^b} F_{Ki}^{-1} \partial_J F_{lK} + F_{Il}^{-1} \partial_{y_k^b} F_{Ki}^{-1} \partial_{y_j^a} \partial_J F_{lK}) \end{aligned}$$

Second nodal derivative of $\det F$

$$\partial_{y_j^a} \partial_{y_k^b} \det F = \det F (F_{Ii}^{-1} \partial_{y_k^b} F_{iI})(F_{Mm}^{-1} \partial_{y_j^a} F_{mM}) - \det F (F_{Im}^{-1} \partial_{y_j^a} F_{mK} F_{Ki}^{-1} \partial_{y_k^b} F_{iI})$$

Appendix B.3. Derivatives of the phase-field

From the numerical discretization of the reference phase-field

$$\phi_0(x) = \sum_{a=1}^{N} p^a(x)\phi_0^a,$$

we have

$$\partial_I\phi_0 = \sum_{a=1}^{N} \partial_I p^a \phi_0^a, \qquad \partial_I\partial_J\phi_0 = \sum_{a=1}^{N} \partial_I\partial_J p^a \phi_0^a$$

Consequently, we can compute the gradient and Hessian (2×2 minor for axisymmetry or full tensor in 3D) of the deformed phase-field

$$\partial_i\phi = \partial_I\phi_0 F_{Ii}^{-1}, \qquad \partial_i\partial_j\phi = \partial_I\partial_J\phi_0 F_{Ii}^{-1}F_{Jj}^{-1} + \partial_I\phi_0\partial_J F_{Ii}^{-1}F_{Jj}^{-1}$$

Its Laplacian becomes

$$\Delta\phi = \partial_I\partial_J\phi_0 F_{Ii}^{-1}F_{Ji}^{-1} + \partial_I\phi_0\partial_J F_{Ii}^{-1}F_{Ji}^{-1} + \phi_{,r}/r.$$

Appendix B.4. Nodal derivatives of the energy, volume and area

Defining for convenience

$$C_1 = (\phi_0/\epsilon + c_0\sqrt{2})(1-\phi_0^2), \quad C_2 = \frac{1}{4\epsilon}(1-\phi_0^2)^2, \quad W = \epsilon\Delta\phi + C_1,$$

we have

$$E = f_E \int_{\Omega_0} W^2 \det F r d\Omega_0,$$

$$A = f_A \int_{\Omega_0} \left[\frac{\epsilon}{2}|\nabla\phi|^2 + C_2\right] \det F r d\Omega_0,$$

$$V = \frac{1}{2}\left[Vol(\Omega_0) + \int_{\Omega_0} \phi_0 \det F r d\Omega_0\right],$$

where $f_E = 3k/(16\sqrt{2}\epsilon)$ and $f_A = 3/(2\sqrt{2})$.

The gradient of the energy then follows as

$$\partial_{y_j^b}E = f_E \int_{\Omega_0} (2W\partial_{y_j^b}W \det F + W^2\partial_{y_j^b}\det F)r + W^2 \det F\partial_{y_j^b}r d\Omega_0,$$

where

$$\begin{aligned}\partial_{y_j^b}W/\epsilon \;=\; & \partial_{y_j^b}\partial_i\partial_i\phi = \partial_I\partial_J\phi_0\partial_{y_j^b}F_{Ii}^{-1}F_{Ji}^{-1} + \partial_I\partial_J\phi_0 F_{Ii}^{-1}\partial_{y_j^b}F_{Ji}^{-1} + \\ & \partial_I\phi_0\partial_{y_j^b}\partial_J F_{Ii}^{-1}F_{Ji}^{-1} + \partial_I\phi_0\partial_J F_{Ii}^{-1}\partial_{y_j^b}F_{Ji}^{-1} + \partial_{y_j^b}\phi_{,r}/r - (\phi_{,r}/r^2)\partial_{y_j^b}r\end{aligned}$$

The gradient of the area is

$$\begin{aligned}\partial_{y_j^b} A &= f_A \int_{\Omega_0} \left[\frac{\epsilon}{2}\partial_{y_j^b}|\nabla\phi|^2 \det F + \left(\frac{\epsilon}{2}|\nabla\phi|^2 + C_2\right)\partial_{y_j^b} \det F\right] r \\ &\quad + \left(\frac{\epsilon}{2}|\nabla\phi|^2 + C_2\right)\det F \partial_{y_j^b} r d\Omega_0,\end{aligned}$$

where,

$$\begin{aligned}\partial_{y_j^b}|\nabla\phi|^2 &= \partial_{y_j^b}\partial_i\phi\partial_i\phi = \partial_I\phi_0\partial_{y_j^b}F_{Ii}^{-1}\partial_J\phi_0 F_{Ji}^{-1} + \partial_I\phi_0 F_{Ii}^{-1}\partial_J\phi_0\partial_{y_j^b}F_{Ji}^{-1} \\ &= 2\partial_I\phi_0 F_{Ii}^{-1}\partial_J\phi_0\partial_{y_j^b}F_{Ji}^{-1}.\end{aligned}$$

Finally, the derivative of the volume is

$$\partial_{y_j^b} V = \frac{1}{2}\int_{\Omega_0} (\phi_0\partial_{y_j^b}\det F) r + \phi_0 \det F \partial_{y_j^b} r d\Omega_0.$$

We can compute the Hessian of the energy as

$$\begin{aligned}\partial_{y_i^a}\partial_{y_j^b} E &= f_E \int_{\Omega_0} (2W\partial_{y_i^a}\partial_{y_j^b}W \det F + 2\partial_{y_i^a}W\partial_{y_j^b}W\det F \\ &\quad + 2W\partial_{y_j^b}W\partial_{y_i^a}\det F + 2W\partial_{y_i^a}W\partial_{y_j^b}\det F + W^2\partial_{y_i^a}\partial_{y_j^b}\det F) r \\ &\quad + (2W\partial_{y_j^b}W\det F + W^2\partial_{y_j^b}\det F)\partial_{y_i^a} r + (2W\partial_{y_i^a}W\det F + W^2\partial_{y_i^a}\det F)\partial_{y_j^b} r d\Omega_0,\end{aligned}$$

where,

$$\begin{aligned}\partial_{y_i^a}\partial_{y_j^b} W/\epsilon &= \partial_I\partial_J\phi_0\partial_{y_i^a}\partial_{y_j^b}F_{Ik}^{-1}F_{Jk}^{-1} + \partial_I\partial_J\phi_0\partial_{y_j^b}F_{Ik}^{-1}\partial_{y_i^a}F_{Jk}^{-1} + \partial_I\partial_J\phi_0\partial_{y_i^a}F_{Ik}^{-1}\partial_{y_j^b}F_{Jk}^{-1} \\ &\quad + \partial_I\partial_J\phi_0 F_{Ik}^{-1}\partial_{y_i^a}\partial_{y_j^b}F_{Jk}^{-1} + \partial_I\phi_0\partial_{y_i^a}\partial_{y_j^b}\partial_J F_{Ik}^{-1}F_{Jk}^{-1} + \partial_I\phi_0\partial_{y_j^b}\partial_J F_{Ik}^{-1}\partial_{y_i^a}F_{Jk}^{-1} \\ &\quad + \partial_I\phi_0\partial_{y_i^a}\partial_J F_{Ik}^{-1}\partial_{y_j^b}F_{Jk}^{-1} + \partial_I\phi_0\partial_J F_{Ik}^{-1}\partial_{y_i^a}\partial_{y_j^b}F_{Jk}^{-1} \\ &\quad + \partial_{y_i^a}\partial_{y_j^b}\phi_{,r}/r - (1/r^2)\partial_{y_i^a}\phi_{,r}\partial_{y_j^b} r - (1/r^2)\partial_{y_j^b} r\partial_{y_i^a}\phi_{,r} + (2\phi_{,r}/r^3)\partial_{y_i^a} r\partial_{y_j^b} r.\end{aligned}$$

For the area, we have

$$\begin{aligned}\partial_{y_i^a}\partial_{y_j^b} A &= f_A \int_{\Omega_0} \left[\frac{\epsilon}{2}\partial_{y_i^a}\partial_{y_j^b}|\nabla\phi|^2\det F + \left(\frac{\epsilon}{2}|\nabla\phi|^2 + C_2\right)\partial_{y_i^a}\partial_{y_j^b}\det F \right. \\ &\quad \left. + \frac{\epsilon}{2}\partial_{y_j^b}|\nabla\phi|^2\partial_{y_i^a}\det F + \frac{\epsilon}{2}\partial_{y_i^a}|\nabla\phi|^2\partial_{y_j^b}\det F\right] r \\ &\quad + \left[\frac{\epsilon}{2}\partial_{y_i^a}|\nabla\phi|^2\det F + \left(\frac{\epsilon}{2}|\nabla\phi|^2 + C_2\right)\partial_{y_i^a}\det F\right]\partial_{y_j^b} r \\ &\quad + \left[\frac{\epsilon}{2}\partial_{y_j^b}|\nabla\phi|^2\det F + \left(\frac{\epsilon}{2}|\nabla\phi|^2 + C_2\right)\partial_{y_j^b}\det F\right]\partial_{y_i^a} r d\Omega_0,\end{aligned}$$

where,

$$\partial_{y_i^a}\partial_{y_j^b}|\nabla\phi|^2 = 2\partial_I\phi_0\partial_{y_i^a}F_{Ik}^{-1}\partial_J\phi_0\partial_{y_j^b}F_{Jk}^{-1} + 2\partial_I\phi_0 F_{Ik}^{-1}\partial_J\phi_0\partial_{y_i^a}\partial_{y_j^b}F_{Jk}^{-1}$$

Finally,

$$\partial_{y_i^a}\partial_{y_j^b} V = \frac{1}{2}\int_{\Omega_0} \phi_0\partial_{y_i^a}\partial_{y_j^b}\det F r + \phi_0\partial_{y_i^a}\det F\partial_{y_j^b} r + \phi_0\partial_{y_j^b}\det F\partial_{y_i^a} r d\Omega_0.$$

[1] M. K. W. Wintz, U. Seifert, R. Lipowsky, Fluid vesicles in shear flow, Physical Review Letters 77 (17) (1996) 3685–3688.

[2] U. Seifert, Fluid membranes in hdrodynamic flow fields: Formalism and an application to fluctuating quasispherical vesicles in shear flow., Eur. Phys. J. B 8 (1999) 405–415.

[3] H. Noguchi, G. Gompper, Dynamics of fluid vesicles in shear flow: Effect of membrane viscosity and thermal fluctuations, Physical Review E 72 (2005) 011901.

[4] S. Veerapaneni, A. Rahimian, G. Biros, D. Zorin, A fast algorithm for simulating vesicle flows in three dimensions, Journal of Computational Physics 230 (2011) 5610–5634.

[5] J. B. Fournier, N. Khalifat, N. Puff, M. I. Angelova, Chemically triggered ejection of membrane tubules controlled by intermonolayer friction, Phys. Rev. Lett. 102 (2009) 018102.

[6] N. Khalifat, N. Puff, S. Bonneau, J.-B. Fournier, M. I. Angelova, Membrane deformation under local ph gradient: Mimicking mitochondrial cristae dynamics, Biophys. J. 95 (10) (2008) 4924–4933.

[7] M. Staykova, M. Arroyo, M. Rahimi, S. H. A, Confined bilayers passively regulate shape and stress, Phys. Rev. Lett. 110 (2013) 028101.

[8] M. Rahimi, M. Arroyo, Shape dynamics, lipid hydrodynamics, and the complex viscoelasticity of bilayer membranes, Physical Review E 86 (2012) 011932.

[9] A. Rosolen, C. Peco, M. Arroyo, An adaptive meshfree method for phase-field models of biomembranes. Part I: approximation with maximum-entropy approximants, Journal of Computational Physics ?? (2013) ??–??

[10] M. Arroyo, M. Ortiz, Local maximum-entropy approximation schemes: a seamless bridge between finite elements and meshfree methods, International Journal for Numerical Methods in Engineering 65 (13) (2006) 2167–2202.

[11] B. Li, F. Habbal, M. Ortiz, Optimal transportation meshfree approximation schemes for fluid and plastic flows, International Journal for Numerical Methods in Engineering 83 (12) (2010) 1541–1579.

[12] J. L. McWhirter, H. Noguchi, G. Gompper, Flow-induced clustering and alignment of vesicles and red blood cells in microcapillaries, Proceedings of the National Academy of Sciences 106 (15) (2009) 6039–6043.

[13] A. Bonito, R. Nochetto, S. Pauletti, Parametric FEM for geometric biomembranes, Journal of Computational Physics 229 (9) (2010) 3171–3188.

[14] A. Bonito, R. H. Nochetto, M. S. Pauletti, Dynamics of biomembranes: Effect of the bulk fluid, Mathematical Modelling of Natural Phenomena 6 (2011) 25–43.

[15] S. Veerapaneni, D. Gueyffier, G. Biros, D. Zorin, A numerical method for simulating the dynamics of 3d axisymmetric vesicles suspended in viscous flows, Journal of Computational Physics 228 (19) (2009) 7233–7249.

[16] C. S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.

[17] D. V. Le, J. White, J. Peraire, K. M. Lim, B. C. Khoo, An implicit immersed boundary method for three-dimensional fluid-membrane interactions, Journal of Computational Physics 228 (2009) 8427–8445.

[18] Y. Kim, M.-C. Lai, Simulating the dynamics of inextensible vesicles by the penalty immersed boundary method, Journal of Computational Physics 229 (2010) 4840–4853.

[19] Q. Du, C. Liu, X. Wang, A phase field approach in the numerical study of the elastic bending energy for vesicle membranes, Journal of Computational Physics 198 (2004) 450–468.

[20] X. Wang, Phase field models and simulations of vesicle bio-membranes, Ph.D. thesis, Department of Mathematics, The Pennsylvania State University, Pennsylvania, USA (2005).

[21] Q. Du, C. Liu, R. Ryham, X. Wang, Energetic variational approaches in modeling vesicle and fluid interactions, Physica D 238 (2009) 923–930.

[22] T. Biben, K. Kassner, C. Misbah, Phase-field approach to three-dimensional vesicle dynamics, Physical Review E 72 (4) (2005) 041921.

[23] D. Jamet, C. Misbah, Towards a thermodynamically consistent picture of the phase-field model of vesicles: Local membrane incompressibility, Phys. Rev. E 76 (2007) 051907.

[24] M. Farshbaf-Shaker, H. Garcke, Thermodynamically consistent higher order phase field Navier-Stokes models with applications to biomembranes., Discrete Contin. Dyn. Syst., Ser. S 4 (2011) 371–389.

[25] Q. Du, Phase field calculus, curvature-dependent energies, and vesicle membranes, Philosophical Magazine 91 (2010) 165–181.

[26] L. Ma, W. Klug, Viscous regularization and r-adaptive remeshing for finite element analysis of lipid membrane mechanics, Journal of Computational Physics 227 (11) (2008) 5816 – 5835.

[27] F. Campelo, Modeling morphological instabilities in lipid membranes with anchored amphiphilic polymers, J. Chem. Biol. 2 (2009) 65–80.

[28] Q. Du, J. Zhang, Adaptive finite element method for a phase field bending elasticity model of vesicle membrane deformations, SIAM J. Sci. Comput. 30 (3) (2008) 1634–1657.

[29] A. Rosolen, D. Millán, M. Arroyo, Second order convex *maximum entropy* approximants with applications to high order PDE, International Journal for Numerical Methods in Engineering 94 (2) (2013) 150–182.

[30] E. Oñate, M. A. Celigueta, S. R. Idelsohn, F. Salazar, B. Suárez, Possibilities of the particle finite element method for fluid–soil–structure interaction problems, Computational Mechanics 48 (2011) 307–318.

[31] H. Goldstein, C. Poole, J. Safko, Classical Mechanics, Addison-Wesley, 2001.

[32] A. Lew, J. E. Marsden, M. Ortiz, M. West, Variational time integrators, Internat. J. Numer. Methods Engrg. 60 (2004) 153–212.

[33] M. Ortiz, E. A. Repetto, Nonconvex energy minimization and dislocation structures in ductile single crystals, Journal of the Mechanics and Physics of Solids 47 (1999) 397–462.

[34] Q. Du, C. Liu, X. Wang, Simulating the deformation of vesicle membranes under elastic bending energy in three dimensions, Journal of Computational Physics 212 (2006) 757–777.

[35] J. Marsden, T. Hughes, The mathematical foundations of elasticity, Prentice-Hall, 1983.

[36] T. Belytschko, W. Liu, B. Moran, Nonlinear Finite Elements for Continua and Structures, John Wiley & Sons, England, 2001.

[37] C. Peco, A. Rosolen, M. Arroyo, Stabilized analysis of Stokes's equations with local maximum entropy meshfree approximantsIn preparation.

[38] J. Happel, H. Brenner, Low Reynolds Number Hydrodynamics: with special applications to particulate media, Martinus Nijhoff Publishers, 1983.

[39] M. Arroyo, A. DeSimone, Relaxation dynamics of fluid membranes, Phys. Rev. E 79 (3) (2009) 031915.

[40] A. Rosolen, D. Millán, M. Arroyo, On the optimum support size in meshfree methods: a variational adaptivity approach with maximum entropy approximants, International Journal for Numerical Methods in Engineering 82 (7) (2010) 868–895.

[41] J. Sanborn, K. Oglecka, R. S. Kraut, A. N. Parikh, Transient pearling and vesiculation of membrane tubes under osmotic gradients, Faraday Discussions DOI: 10.1039/C2FD20116J.

Appendix C
Efficient implementation of meshfree Galerkin methods for large-scale problems with an emphasis on maximum entropy approximants.

C. Peco, D. Millán, A. Rosolen and M. Arroyo

Submitted to Computers and Structures

Efficient implementation of Galerkin meshfree methods for large-scale problems with an emphasis on maximum entropy approximants

Christian Peco, Daniel Millán, Adrian Rosolen and Marino Arroyo*

LaCàN, Universitat Politècnica de Catalunya (UPC), Barcelona 08034, Spain

Abstract

Galerkin meshfree methods are well-established to approximate partial differential equations (PDEs). However, since the support of the basis functions is not restricted to predefined elementary domains, the creation of sparse matrices and the assembly process is not straightforward. Furthermore, it is often necessary to evaluate the basis functions repeatedly in a computation, e.g. in nonlinear problems. As a result of the higher density of the connectivity graph in meshfree methods, the memory required to store the basis functions and its derivatives can quickly become unaffordable for large problems, particularly in higher order PDEs involving Hessians of the basis functions. The straightforward alternative, recomputing the basis functions every time they are needed, is computationally inefficient. Here, we show that it is possible to overcome or alleviate these two bottlenecks resorting to simple and effective algorithms. The first algorithm deals with the sparse matrix structure creation and filling. It relies on a cell/element-wise framework and it is easily made parallel. Its performance on a standard two-dimensional heat equation is compared against that of a classical implementation based on looping over quadrature points. The second algorithm stores only partial information of the basis functions, striking a balance between storage and computation. This optimization reduces considerably the memory usage at the expense of a minimum increment in the overall computational cost. We detail the data structures and provide pseudo-codes for the proposed algorithms, and exercise them in a Poisson problem and in a fourth order phase-field model. Both examples have been discretized with nonnegative and smooth local maximum entropy approximants.

Keywords: meshfree methods, local maximum entropy, sparse matrix efficient assembly, matrix structure creation, optimal memory storage, program optimization

*Correspondence to: marino.arroyo@upc.edu

1. Introduction

Meshfree methods have emerged in recent years as a viable alternative to finite elements in a number of applications, see [1, 2, 3, 4, 5] for a detailed review. These methods are based on basis functions that do not rely on a mesh. As a consequence, many of the requirements associated with the quality of the elements in traditional finite element method (FEM) are relaxed or disappear, but this extra flexibility raises new challenges in the numerical implementation [6]. Meshfree methods also present several advantages such as basis functions with high-order continuity, robustness in dramatic grid deformations [7, 8, 9], and easier local adaptivity [10, 11]. Galerkin meshfree methods require a quadrature mesh to perform numerical integration, commonly requiring a higher number of quadrature points to accurately integrate the weak form due to their nonpolynomial nature and nonelement-wise support [12, 13]. Additionally, most of the meshfree methods present an awkward treatment of essential boundary conditions due to nonsatisfaction of the Kronecker delta property

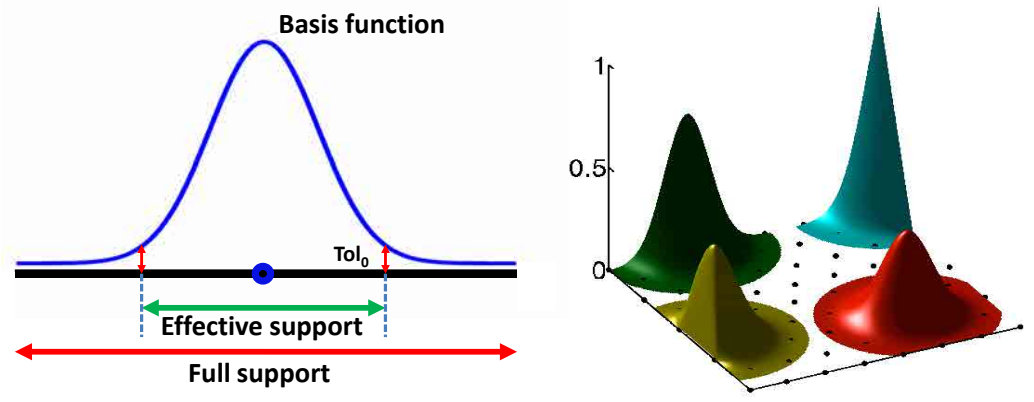


Figure 1: Full support of some meshfree basis functions, such as local maximum entropy approximants, covers the convex hull of the computational domain. The effective numerical support radius r_a is determined by a cutoff basis function value $\texttt{Tol}_0$ (left). Representation of two-dimensional LME approximants basis functions (right). Notice the noninterpolant character and the smoothness of the basis functions, and the fulfillment of a weak Kronecker-delta at the boundary of the convex hull.

Since smoothed particle hydrodynamics [15], a variety of techniques have emerged, such as reproducing kernel particle method [16], partition of unity finite element method [17], and element free Galerkin [18] to mention a few. We resort in this work to the local maximum entropy (LME) approximation schemes, a meshfree method inspired on information theory that generates nonnegative and smooth basis functions (see [19, 20, 21, 22, 23] for a detailed description,

properties and extensions). The capabilities of LME approximants have been examined in a variety of computational mechanics applications, such as linear and nonlinear elasticity [23, 24], plate [25] and thin-shell analysis [26, 27], convection-diffusion problems [28, 29], and phase-field models of biomembranes [30, 31] and fracture mechanics [32, 33, 34].

Like other meshfree methods, LME approximants involve a dilation or locality parameter that modulates their behavior and support. LME approximants show an exponential decay controlled by the locality parameter, and far from the boundaries they look like Gaussian weighted functions [35, 22]. Their effective support is controlled by setting a cut off or threshold value ($\texttt{Tol}_0$) below which the basis functions are taken numerically to be zero (see Fig. 1, Appendix A). The proper choice of the locality parameter is problem dependent and not easy in general, which has motivated a systematic studies for general meshfree methods [36] and for LME approximants [23] in particular. In LME approximations, the locality parameter is an aspect ratio parameter γ, which allows us to smoothly move from simplicial finite elements shape functions ($\gamma > 4.0$) to more spread out approximation schemes (e.g., $\gamma = 0.6$), as illustrated in Fig. 2. In general, broader functions lead to more accurate results for problems with smooth solutions at the expense of higher computational cost and worse matrix conditioning [20, 26].

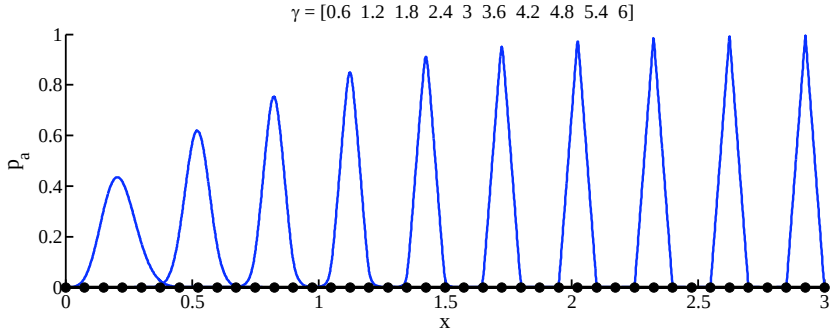


Figure 2: Seamless transition from spread-out meshfree to linear finite elements basis functions.

In contrast to conventional FEM, where the structure of the matrix is inherited from the mesh graph, stencils of meshfree schemes depend strongly on the aspect ratio parameter γ. In our experience, a noticeable run-time computational cost of meshfree methods is due to the creation of the sparse matrix structure and the assembly process, which can be specially harmful for iterative processes. These stages can be as expensive as the solver stage in two-dimensional problems and exceed it in three-dimensional ones. In a typical implementation of the assembly process in meshfree methods, the code loops over the quadrature points. The denser sparsity pattern and the large number of Gauss points required for accurate integration can make these methods unpractical for large-scale calculations. To overcome this issue, we propose here a set of algorithms based on a loop over cells/elements, as commonly done in FEM. We illustrate in this work how this simple approach reduces significantly the computational cost associated

with the matrix structure creation and the assembly process.

Additionally, a widespread practice (both in FEM and in meshfree methods) is to store in memory the basis functions and their derivatives for repetitive calculations required in nonlinear iterative solvers, incremental loading, or evolution in time. In FEM, this storage is insignificant because the basis functions of the parent element are mapped to each physical element. Since this is not the case in meshfree methods, the amount of memory and its access can become a bottleneck and substantially reduce the code efficiency, especially in large-scale problems. If meshfree basis functions are not stored in memory but recomputed every time, the computational cost can also increase significantly. To alleviate this issue, we propose here a strategy that is a trade-off solution between memory storage and computational time. The technique, based on a data structure that stores only partial information about the basis functions and an algorithm to reconstruct them when needed, reduces considerably the memory usage at the expense of a minimum increment in the overall computational cost. We illustrate and exploit this concept on LME approximants.

The paper is organized as follows. In Section 2 we review the basic technicalities for a meshfree method particularized to LME approximants and the classical implementation to approximate partial differential equations (PDEs). We then propose an algorithm to speed-up the matrix assembly and an algorithm for the compressed memory storage of LME approximants in Section 3. We extensively test our proposals with numerical examples in Section 4 and finish with some concluding remarks in Section 5.

2. A standard meshfree scheme

Let $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$, for $d = 1, 2, 3$, be an unstructured set of nodes used to describe a domain Ω, and $p_a(\boldsymbol{x})$ the meshfree basis function associated to the a-th node, for $a = 1, \ldots, N$. A continuous field Φ can be approximated as

$$\Phi(\boldsymbol{x}) = \sum_{a=1}^{N} p_a(\boldsymbol{x})\Phi_a,$$

where Φ_a stand for the nodal coefficients. Here we adopt the LME approximants as meshfree basis functions in a Galerkin method to approximate a general PDE. They are nonnegative, smooth, satisfy at least up to the first order consistency conditions and present a weak Kronecker-delta property. We rely on an integration mesh to define the quadrature points, typically through a Gauss-Legendre quadrature rule. We use simplicial meshes made of triangles/tetrahedra in 2D/3D, which are obtained via the library QHULL [37]. The procedure needed to compute the system matrix in a Galerkin meshfree approach requires mainly four steps: (i) neighborhood search, (ii) computation of the basis functions, (iii) creation of the sparse matrix structure and (iv) Gauss point-wise matrix filling. The pseudocode shown in Algorithm 1 summarizes these four steps. In the present work, we do not deal with solver performance. In the following we briefly extend on the computational implications of every step.

Algorithm 1 Pseudo-code for scheme based on a loop over quadrature points (see Section 2).

(i) Determine the neighborhood nodal index set $\mathcal{N}_{\boldsymbol{y}}^{X}$ for each Gauss point.
(ii) Compute shape functions (array p_a).
(iii) Construct sparse matrix structure (arrays *ia* and *ja*).
(iv) Fill sparse matrix (array *an*) with the quadrature point loop based algorithm.

The objective of step (i) is to compute the so-called neighbor lists, which can be interpreted as the counterpart of the mesh connectivity in FEM where the neighbor lists are given by the mesh itself. In a meshfree scheme this is made by specialized algorithms, i.e. neighbor searchers which identify the relationship between the quadrature points and the nodes. We will refer to the neighbor lists as *primal* and *dual lists* [26], which are complementary. In particular, a dual list identifies the quadrature points that are influenced by a particular node i.e. the quadrature points falling within the effective support of a nodal basis function. Conversely, the primal list contains the nodes that influence a particular quadrature point.

Formally, let $Y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_L\} \subset \Omega$ be a set of quadrature points. The dual list containing the nearest points from Y associated with a node $\boldsymbol{x}_a \in X$ can be defined as follows

$$\widetilde{\mathcal{N}}_{\boldsymbol{x}_a}^{Y} = \{k \in \{1, 2, .., L\} \mid |\boldsymbol{y}_k - \boldsymbol{x}_a| < r_a\},$$

where r_a is the effective support radius of the shape function associated to the a-th node (which is determined by a user defined cutoff value $\texttt{Tol}_0$, see Fig. 1). In the same way, the primal list containing the nodes from X associated with a particular quadrature point $\boldsymbol{y}_k \in Y$ is defined as

$$\mathcal{N}_{\boldsymbol{y}_k}^{X} = \{a \in \{1, 2, .., N\} \mid |\boldsymbol{y}_k - \boldsymbol{x}_a| < r_a\}.$$

The primal and dual lists are used afterwards in steps (ii), (iii) and (iv) to compute the basis functions, identify the nonzero positions in the sparse matrix and perform the assembly. Both lists can be obtained by simply invoking a neighbor searcher. The neighbor finding problem is standard, and comes in two flavors, namely finding the k-first neighbors or finding neighbors within a range. In our codes, we resort to the approximate nearest neighbor searching library [38], whose computational cost scales as $O(N \log N)$, where N is the number of nodes.

In step (iii) the nonzero elements in the global matrix are identified using algorithms that postprocess the neighbor lists. This information is critical to properly store the matrix in a sparse scheme and perform the filling. There are many methods for storing sparse matrices (see, for instance, [39] and [40]). We follow here the compressed sparse row (CSR) storage, which is a proper choice for codes written in C/C++ due to its memory layout. In CSR, a matrix is given in terms of three lists. The first list, *ia*, is an array of integers that stores the total number of nonzeros up to each row. Its dimension is the number of rows plus one, the first position being filled with a zero. The second and third lists are arrays of integers and doubles, *ja* and *an*, have as dimension the number of nonzeros in the matrix, and store the column index position and

the associated matrix entry. We understand the sparse matrix structure creation as the collection of algorithms required to obtain the lists *ia* and *ja*. An efficient way to compute this structure based on a loop over quadrature points is presented in Appendix C. The standard algorithm loops over the primal lists of the Gauss points associated to each nodal dual list, such that the nonzero entries of the sparse matrix are identified when two nodes appear together in at least one primal list. As a result, the sparse structure construction becomes increasingly expensive as the number of quadrature points and the support of the basis functions becomes larger.

In step (iv) the nonzero positions of the system sparse matrix (*an*) are filled in an operation dependent on the PDE. Pursuing a rational memory access, standard Galerkin meshfree algorithms rely on a loop over the Gauss points, each contributing with a local dense matrix. The number of rows of this local matrix is equal to the cardinality of the primal list $|\mathcal{N}_y^X| = n$ times the number of scalar fields in the problem. Again, the computing time of this step is directly penalized by the increase of quadrature points and by the support size of the basis functions. Furthermore, the local matrices have to be assembled into the global matrix. Since the global matrix is sparse, a search is required to identify the global position to be filled. This concept is illustrated in the upper part of Fig. 3.

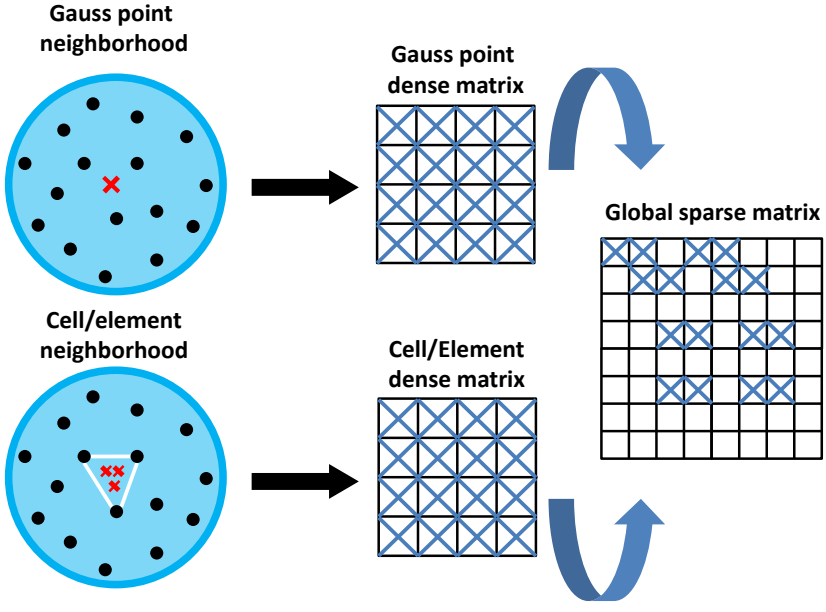


Figure 3: Filling algorithm from neighbor lists to global sparse matrix. Nodal list of neighbors (black dots) can be computed for an individual integration point (red X, top) or for a cell/element (white triangle, bottom). Dense submatrices are generated from these neighborhoods and assembled into the global matrix. Cell/element algorithm improves memory management since the resulting dense submatrix condenses information coming from several integration points.

3. Meshfree optimization concepts

We present here two optimizations to improve the efficiency when facing the bottlenecks described in Section 1. First we describe in Section 3.1 a neighborhood coarsening algorithm that considerably speed-up steps (iii) and (iv). Finally, a reduced storage strategy that mitigates the memory requirements when storing the basis functions is detailed in Section 3.2.

3.1. Neighborhood coarsening algorithm

A simple idea to alleviate the computational cost of the global sparse matrix is to coarse-grain the neighbor primal lists. The key point is to generate a list for each cell/element of a defined coarsening mesh rather than one per Gauss point. The coarsening mesh provides us with a structure to group the primal lists of the Gauss points contained in the cell/element. Without loss of generality, a straightforward and natural choice for the coarsening mesh is the quadrature mesh cells/elements needed in most Galerkin meshfree methods to perform the numerical integration. In this way the complexity added by the increase of Gauss points due to accuracy requirements is removed and the neighbor lists are generated disregarding the number of integration points. We present next details of this procedure.

Once the coarsening mesh is set, we start with a neighbor search over the nodes defining the mesh. This allows us to obtain nodal-based primal lists rather than primal lists for quadrature points. To obtain the cell/element primal lists, the primal lists of its associated nodes are simply merged. More specifically, we define

$$\mathcal{N}_{el} = \bigcup_{a \in \mathcal{T}_{el}} \mathcal{N}^X_{\boldsymbol{x}_a},$$

where $\mathcal{T}_{el}$ is an index set containing the nodal indexes of the el-th cell/element (e.g the mesh connectivity). Note that the $\mathcal{N}_{el}$ list is applicable to the totality of integration points inside the cell/element, regardless their number. This merging operation is negligible in terms of computational time, and give us the possibility to work from now on with cell/element primal lists rather than with integration point based lists. We illustrate this concept in Fig. 4. The examples shown in Section 4 use the quadrature mesh as coarsening mesh and follow the proposed unifying criterium.

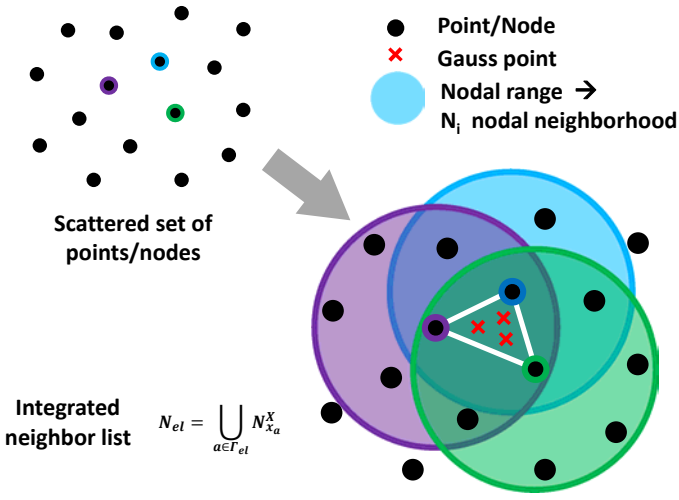


Figure 4: Integrated neighborhood concept. The new cell/element neighborhood is described by the union of nodal vertices lists of neighbors. The triangular elements given by the quadrature mesh are used here as background cell/element generator.

Vertex merging is a proper strategy when the support size r_a is large relative to the mesh size h_a, which is the usually the case in meshfree methods. Conceivably, it could be the case that the merging of the vertices lists would lead to some loss of information. A node could be influencing an integration point inside a cell/element without influencing any of the vertices containing it, e.g. in highly distorted triangles in 2D. If these unlikely events need to be absolutely ruled out, it is always possible to construct the unified lists by merging the neighbor lists of the Gauss points belonging to a cell/element. In our experience, however, this never happens when using a quadrature mesh; the agreement in numerical integration benchmarks is perfect, and for this reason we recommend the proposed vertex merging to create the cell/element lists.

The creation and filling algorithms can be now based on cell/element neighbor lists, which greatly speeds-up the computations. The structure creation is simplified since only the nodes and cells/elements are involved in the whole procedure. Now the nonzero positions are identified by looping over cell/element neighbor lists instead of looping over Gauss points neighbor lists. The filling of the matrix benefits in two distinct ways. Firstly, the element-wise approach leads to cell/element dense matrices. These local matrices are efficiently filled since just a loop over the neighbor list of the cell/element and a loop over the cell/element Gauss points are required. Secondly, only one dense cell/element matrix is assembled into the global matrix, hence the memory access is improved, as illustrated in the lower part of Fig. 3.

As we show later in Section 4.1, this optimization maintains constant the computational time associated with the matrix-pattern creation algorithm regardless the number of integration points used. This fact significantly alleviates one of the main disadvantages of meshfree methods, namely the large number of quadrature points needed as compared to piecewise polynomial

approximants. Furthermore, the granularity of the element-level approach is better suited for parallel computing, minimizing memory access and limiting data exchange. The pseudo-code for the procedure is summarized in Algorithm 2.

Algorithm 2 Pseudo-code for scheme based on a loop over cells/elements (see Section 3).

(i) Compute adjacency lists for nodes $\mathcal{N}^X_{\boldsymbol{x}_a}$ and process cell/element lists $\mathcal{N}_{el} = \bigcup_{a\in\mathcal{T}_{el}} \mathcal{N}^X_{\boldsymbol{x}_a}$.

(ii) Compute shape functions (array p_a).

(iii) Construct sparse matrix structure (arrays *ia* and *ja*).

(iv) Fill sparse matrix (array *an*) with the cell/element loop based algorithm.

3.2. Compressed meshfree basis functions storage

A standard practice in the numerical treatment of PDEs is to store in memory the basis functions and their derivatives at each Gauss point. This strategy decreases considerably the computational cost in problems involving nonlinear iterative solvers or evolution problems on Lagrangian meshes. While this storage is insignificant in FEM, in meshfree methods the amount of memory (as quantified later) and its access can become a bottleneck and substantially reduce the code efficiency. Here we propose a storage concept that is based on finding structures that optimally synthesize the basis function information at each integration point, striking a trade-off between memory storage and computation time. We generate data structures that store only partial information about the basis functions and an algorithm to reconstruct them when needed, reducing considerably the memory usage at the expense of a marginal increment in the overall computational cost.

For a general meshfree method, and considering the Galerkin approximation of a fourth-order PDE, the *full storage* of the basis functions requires $M_{FS} = L \cdot \bar{n} \cdot [1 + d + d(d+1)/2] = L \cdot \bar{n} \cdot \left(1 + \frac{3}{2}d + \frac{1}{2}d^2\right)$ doubles. In this equation, L is the total number of quadrature points, $\bar{n}$ is the mean cardinality of the primal lists, 1 accounts for the basis functions themselves, d for their gradients, and $d(d+1)/2$ for the Hessian, which is a symmetric matrix. In a for fourth-order PDE we typically have $\bar{n} \approx 65$ in 2D and $\bar{n} \approx 380$ in 3D. As a result, the memory requirements rapidly become unaffordable.

Focusing on LME approximants, we recall that the basis functions are obtained by means of a nonlinear optimization problem at each evaluation point with d unknowns, where d is the spatial dimension. This optimization problem yields the Lagrange multiplier associated with first-order consistency conditions. Once the Lagrange multiplier is known, an explicit expression for the basis functions, its gradient and its Hessian is explicit (see Appendix A). Even if the nonlinear optimization problem is relatively easy to solve by Newton's method, it accounts for a significant part of the basis function evaluation time.

A straight-forward alternative to the *full storage* method would be to simply store the d reals in the Lagrange multiplier at each quadrature point. Analyzing in detail the structure

Table 1: Quantification and comparison of memory usage between the methods of full and optimal or compressed storage of local maximum-entropy basis functions and their derivatives. Here, $\bar{n}$ is the mean cardinality of primal lists, L is the number of Gauss points and d is the spatial dimension.

	Full storage	Optimal storage
Total memory usage	$M_{FS} = L \cdot \bar{n} \cdot \left(1 + \frac{3}{2}d + \frac{1}{2}d^2\right)$ $\bar{n} \gg (2 + d)$	$M_{OS} = L \cdot \left(2 + \frac{7}{2}d + 2d^2 + \frac{1}{2}d^3\right)$ $M_{OS} \approx L \cdot (2 + d) \cdot \left(1 + \frac{3}{2}d + \frac{1}{2}d^2\right)$
Comparison	$M_{FS}/M_{OS} \approx \bar{n}/(2 + d) \gg 1$	

of the explicit formulae for the basis functions and derivatives, it is easy to identify a set of matrices and vectors whose size is independent on $\bar{n}$, and some of which involve summations over $\bar{n}$. Thus, storing these arrays saves significant computation time at a limited memory cost. As detailed in Appendix B, this simple observation suggests the *optimal or compressed storage*, which only involves $M_{OS} = L \cdot \left(2 + \frac{7}{2}d + 2d^2 + \frac{1}{2}d^3\right) \approx L \cdot (2 + d) \cdot \left(1 + \frac{3}{2}d + \frac{1}{2}d^2\right)$ doubles. As the mean cardinality is in general much greater than the spatial dimension, i.e. $\bar{n} \gg (2 + d)$, from the ratio $M_{FS}/M_{OS} = \bar{n}/(2 + d)$ it is clear that the memory usage decreases significantly when the compressed storage technique is used, as can be observed in Table 1.

In Section 4.2 we apply this strategy to a fourth-order PDE requiring the storage of the values, gradients and Hessians of the LME basis functions. We quantify the memory usage and computational time devoted to evaluate the basis functions for both the full and for the optimized storage implementations.

4. Numerical examples

The performance of the optimizations presented in Section 3 are studied here in two boundary-value problems. We focus in the four steps presented in Section 2 and leave aside the solver stage. As we specify in Section 1, in our experience the analized steps can be comparable in computational time to the solver in 2D problems and exceed it in 3D. In the first example the neighbor coarsening procedure from Section 3.1 is applied to solve a 2D heat diffusion PDE, whereas the compressed basis functions storage detailed in Section 3.2 is exercised in a nonlinear fourth-order phase-field PDE. Both problems use uniform grids that ensure a quite constant number of nodal neighbors for every integration point and facilitate the comparison. In the first example we use the quadrature mesh and the vertex merging approach to generate the unified primal lists, as proposed in Section 2. These stages can be as expensive as the solver stage in two-dimensional problems and exceed it in three-dimensional ones, particularly in problems with vectorial fields.

4.1. The neighborhood coarsening algorithm applied to a heat equation

We exercise the neighborhood coarsening algorithm on a benchmark heat equation in 2D, which is a scalar problem. The sparse matrix structure creation and assembly using the proposed

neighborhood coarsening for scalar and vectorial problems is detailed in Appendix C, along with a description of the data structures and a C/C++ pseudo-code.

The diffusion boundary problem is defined as follows:

$$\begin{aligned} \Delta T &= f, \quad \text{in } \Omega, \\ T &= T_0, \quad \text{on } \Gamma_D, \end{aligned}$$

where T is the temperature field, $\Omega = (0,1) \times (0,1)$ the domain, f is an arbitrary source of heat and T_0 is the prescribed temperature on the Dirichlet's boundary Γ_D. We consider $T_0 = 0$ on $\Gamma_D = \partial\Omega$ and assume a source $f = 2y$. The solution obtained using LME approximants with a uniform grid of points of 100×100 is depicted in Fig. 5.

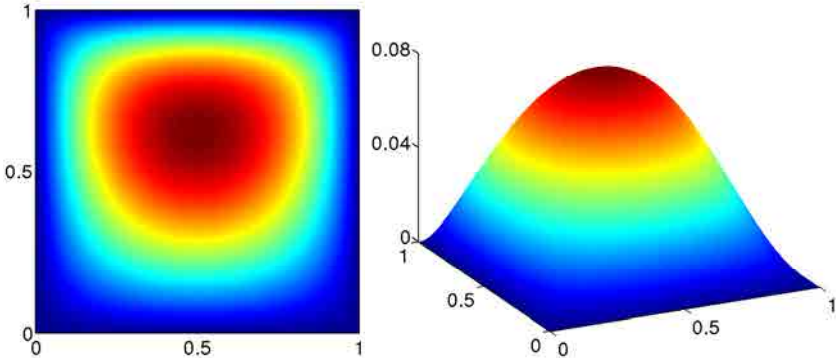


Figure 5: 2D and 3D views of the solution for a heat equation with a source. We use LME approximants, a uniform mesh of 100×100 nodes, $\gamma = 1.6$ and 6 Gauss points per triangular cell/element.

The entries of the stiffness matrix take the standard form

$$K_{ab} = \int_\Omega \nabla p_a \cdot \nabla p_b \, d\Omega,$$

where Ω cannot be reduced to a set of elements as in FEM. A quadrature rule is defined on a background integration mesh over the whole domain (that may or may not coincide with the coarsening mesh) as

$$K_{ab} = \sum_{k=1}^{L} \nabla p_a(\boldsymbol{y}_k) \cdot \nabla p_b(\boldsymbol{y}_k) \, \omega_k,$$

where ω_k stand for the Gauss points quadrature weights in physical space.

The performance of the algorithm based on looping over quadrature points depends on the number of nodes used to discretize the domain, the number of Gauss points per quadrature cell and the size of the support of the basis functions (linked to the aspect ratio parameter γ). In Fig. 6 we show a representative performance, reporting the computational time spent in

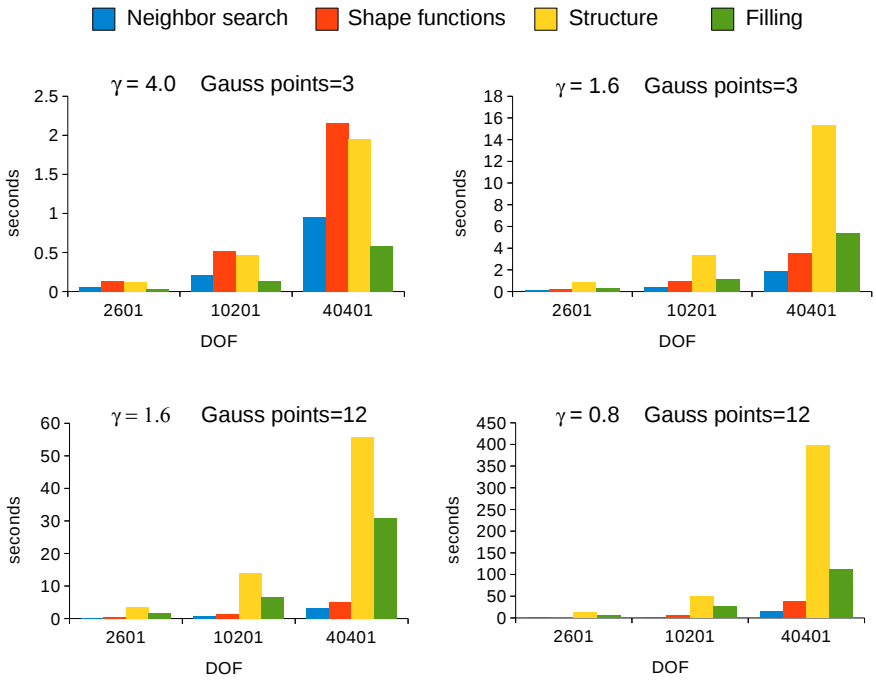


Figure 6: Computational time vs grid size for different values of Gauss points per cell and γ. From left to right, bars correspond to stages: (i) neighborhood search, (ii) shape functions, (iii) matrix structure creation and (iv) matrix structure filling.

the main four stages of the algorithm described in Section 2, i.e. (i) neighborhood search, (ii) shape functions, (iii) matrix structure creation and (iv) matrix structure filling. The plots show computational time vs degrees of freedom (number of nodes) for different combinations of the aspect ratio parameter $\gamma = 0.8, 1.6, 4.0$ and three and twelve Gauss points per element. The left-upper chart shows FEM-like LME approximants ($\gamma = 4.0$) with three points per integration element. In this case, the bottleneck in the shape functions calculation. In the other plots we can see how increasing the support size (decreasing γ) or/and increasing Gauss points per element dramatically rises the cost of structure creation and structure filling in comparison with the FEM-like shape functions. The upper charts in Fig. 7 illustrate the computing time growth for different number of Gauss points, whereas the lower charts depict the growth when changing the parameter γ. The results of both figures highlight the need for speed-up techniques in steps (iii) and (iv) when the size of the system increase for spread-out basis functions ($\gamma = 0.8$, large support) that require accurate numerical integration.

We proceed then to analyze the proposed cell/element scheme and review the performance of critical stages (iii) and (iv). In the upper part of Fig. 8 the computational time vs number of Gauss points is shown for $\gamma = 0.8, 1.6, 4.0$. The juxtaposed bars in the figures compare the stan-

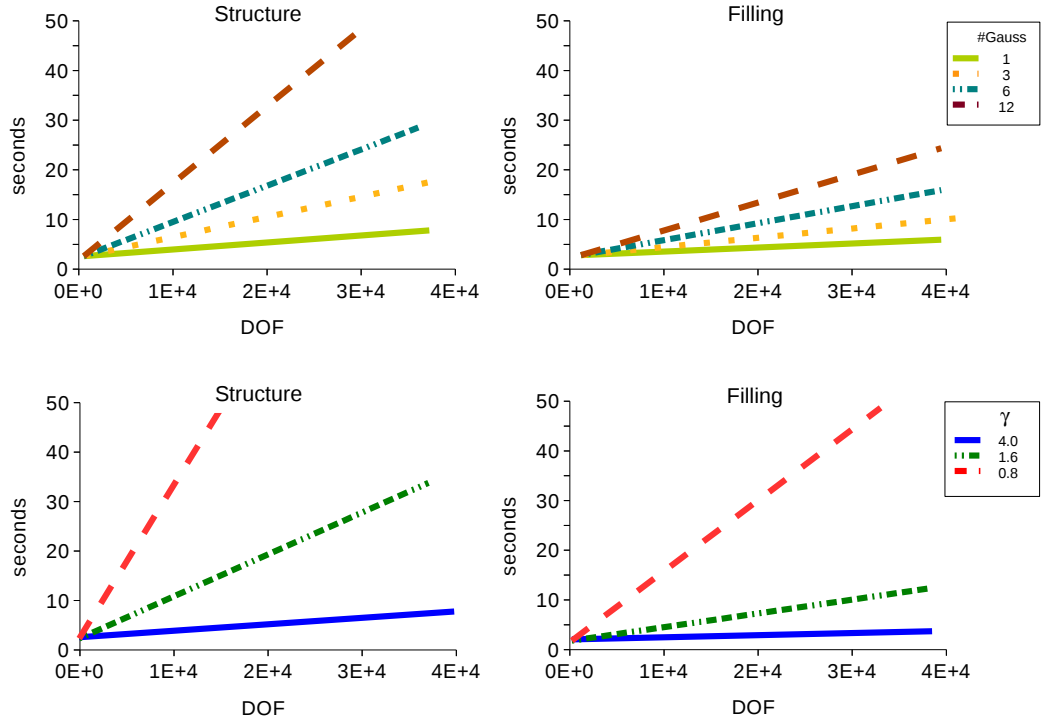


Figure 7: Comparison between growth for matrix structure creation (left) and filling algorithms (right) with the grid size for different values of Gauss points per cell (top, γ=1.6) and for different values of parameter γ (bottom, Gauss points = 6).

dard and the new implementations. We can see how the gain in performance grows as the support size increases, giving greater speed-ups as γ decreases e.g. ten times faster for twelve Gauss points and $\gamma = 0.8$. Notice also that the matrix structure creation is completely insensitive in the new implementation to the number of quadrature points per element, see Fig. 8. No speed-up is observed when a small number of integration points is used. We show in the lower panels of Fig. 8 the filling algorithm computational time vs number of Gauss points for $\gamma = 0.8, 1.6, 4.0$. We observe the same pattern of speed-ups when γ decreases. Notice that although the speed-up is considerable, the filling operations do depend on the number of quadrature points in the nee algorithm, but far less critically than in the standard implementation. Nevertheless, the filling time is greatly reduced with the proposed approach, particularly for large supports i.e. five times smaller for $\gamma = 0.8$ and twelve Gauss points.

Finally, the growth of computational time as a function of system size is presented in Fig. 9. The standard implementation using twelve Gauss points is shown as reference. We conclude

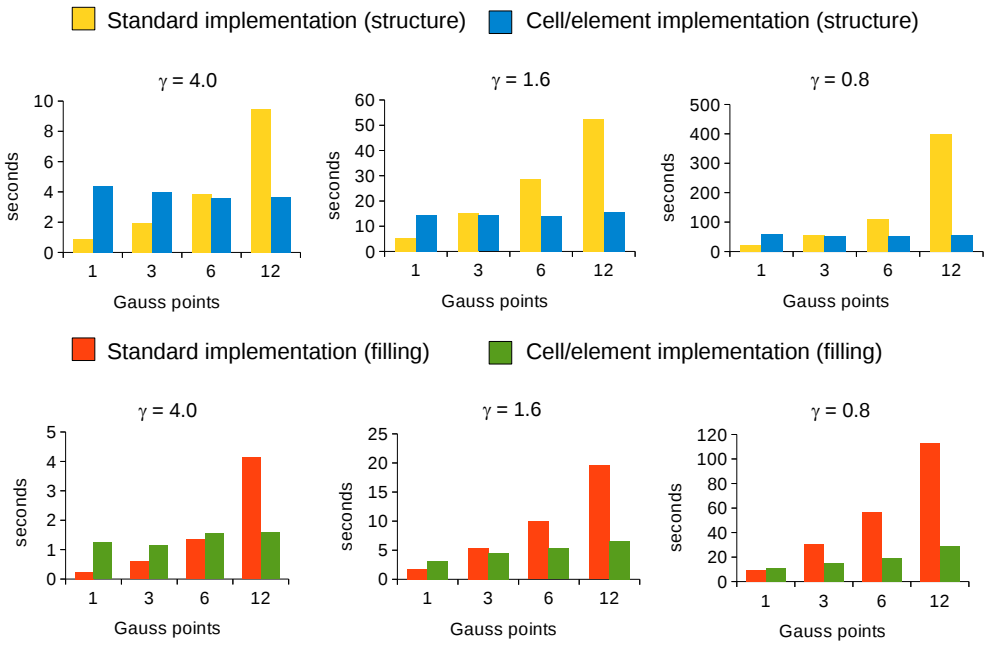


Figure 8: Structure creation (top) and filling (bottom) computational time vs Gauss points for decreasing γ (in-

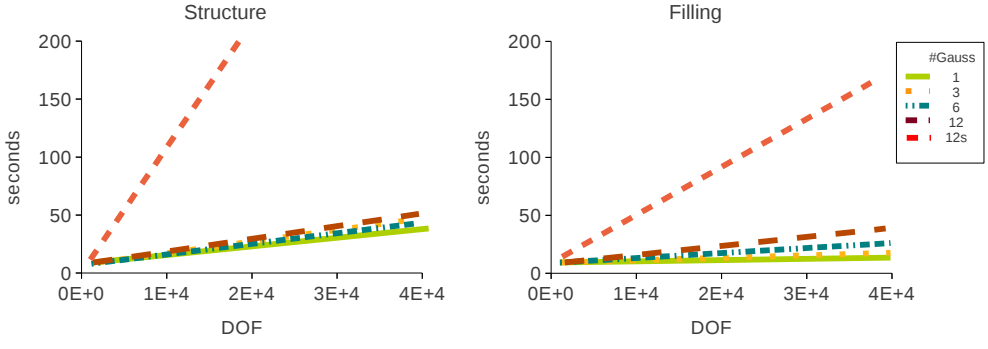


Figure 9: Growth of the computational time as a function of the size of the system for the matrix structure creation (left) and filling (right), using the new implementation different numbers of Gauss points per cell and $\gamma = 0.8$. For comparison purposes, the standard implementation using 12 Gauss points (legend 12s) is presented.

4.2. The compressed meshfree basis functions storage applied to a phase-field fracture model

We present here the results of the proposed memory storage strategy for the LME approximants. We compare the full storage and optimized schemes in a fourth-order PDE problem requiring the values, gradients and Hessians of the basis functions. In a variational model of fracture, the phase-field PDE results from the following functional

$$\int_\Gamma G_c \, d\Gamma = \int_\Omega G_c \left[\frac{(1-\phi)^2}{4l_0} + \frac{l_0}{2}|\nabla\phi|^2 + \frac{l_0^3}{4}\Delta\phi^2 \right] d\Omega,$$

where G_c is the critical fracture energy density, ϕ the phase-field, and l_0 the parameter controlling the width of the approximation of the crack. We illustrate a typical solution in Fig. 10. More details about this particular model can be found in [41].

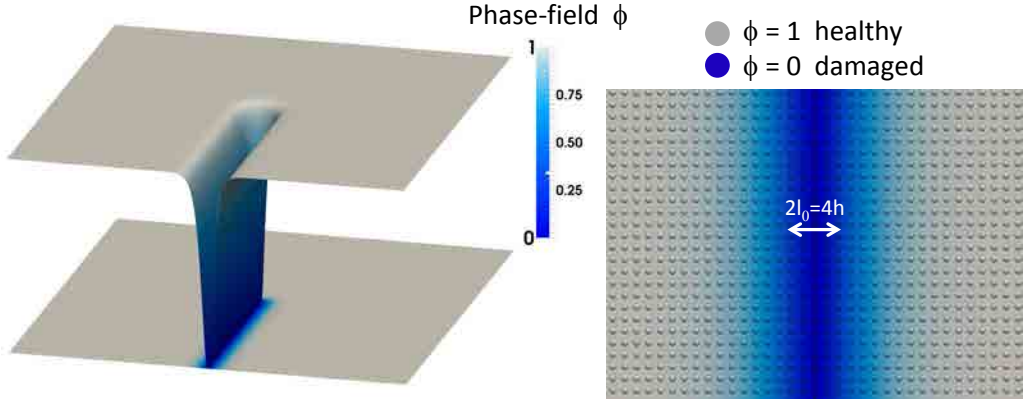


Figure 10: Fourth-order phase-field solution for a crack. Phase-field values range from 1 to 0 signaling the progressive damaging of the material (left). The ratio between the crack width parameter l_0 and the nodal spacing h is 2 (right).

We focus on the amount of doubles that need to be stored when using a standard and the optimized scheme, and also on the impact on computational time of the structure filling routine for the global matrix. The latter requires retrieving the stored basis functions in the usual approach, and partially recomputing them in the optimized storage approach. The structure creation step is completely independent on evaluation/retrieval of the basis functions, and for this reason we do not report it here. As can be observed in the left panel of Fig. 11, the optimized storage strategy decreases the memory requirements by an order of magnitude; the ratio of memory requirements is about 20. For this two-dimensional problem we use $\gamma = 1$, leading to a mean value of 72 neighbors per integration point. Here we use six Gauss points by element.

We analyze now the computational time invested in the filling of the global matrix. We can observe in the right panel of Fig. 11 that the memory optimized storage is marginally slower than the standard routine. The extra operations to retrieve the basis functions and its derivatives,

see Appendix A, is partially compensated by a more efficient access to the memory, resulting in running time increments of about 10%.

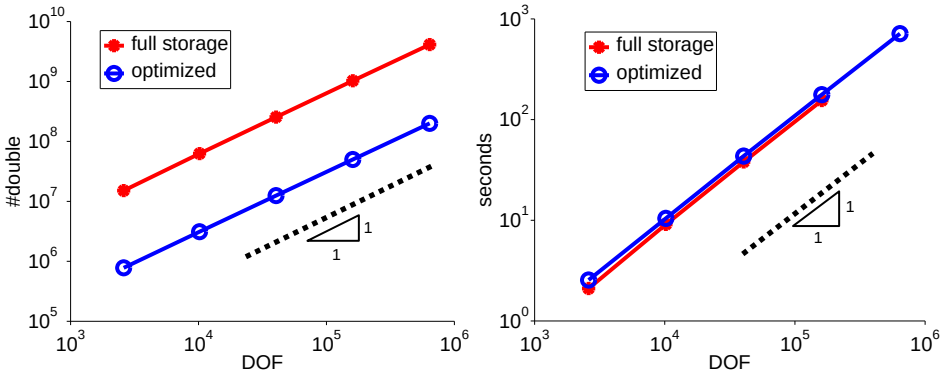


Figure 11: Comparison between the full storage and optimized implementations. The plots show the number of doubles stored in the global matrix vs the number of degrees of freedom (left) and the computational time invested in the filling and assembly of the global matrix vs the number of degrees of freedom (right).

5. Conclusions

We have presented two optimization procedures to mitigate two fundamental bottlenecks in Galerkin meshfree methods: matrix assembly and basis functions storage. We have shown how the sparse structure creation and filling of the system matrix become critical in a meshfree context when either the support size of the basis functions or the number of integration points increases. We have introduced a simple coarse-graining procedure for matrix structure creation and filling, where we change from an integration point perspective to one based on cells/elements. As a result of this optimization, the dependence of the computational time on the number of integration points is completely severed in the sparse structure creation and dramatically decreased in the matrix assembly. We tested the new implementation on a 2D heat diffusion PDE, speeding-up ten times the structure creation and five times the filling in the case of twelve Gauss points and $\gamma = 0.8$. Furthermore, our analysis of a scalar 2D problem suggests that the connectivity coarsening procedure should become particularly effective in 3D vectorial boundary-value problems. Additionally, a compressed memory storage for LME approximants has been introduced to alleviate memory requirements. We have shown how this methodology can recover with minimal computational overhead the basis functions, gradients and Hessians that are repeatedly required in large-scale nonlinear or evolution problems, hence reducing drastically the amount of memory by 20−fold for a scalar fourth-order PDE in 2D.

Further research in Galerkin meshfree methods should focus on tridimensional problems and the study of proper parallelization algorithms for supercomputing. We have successfully parallelized the presented techniques using state-of-the-art scientific codes such as PETSc (portable,

extensible toolkit for scientific computation library, [42]) and ParMetis [43] for reordering and partitioning. Our current experience on a supercomputing facility further highlights the importance of optimizations such as those presented here in large-scale vectorial problems in 3D. Models showing an intrinsic high computational cost such as the phase-field approaches can particularly benefit from this concept due to the easy parallelization of the algorithms presented. The approximation of phase-field models with LME in biomembrane dynamics [30, 31] and fracture mechanics [33, 34] are successful examples of these optimization procedures.

Acknowledgments

We acknowledge the support of the European Research Council under the European Community's 7th Framework Programme (FP7/2007-2013)/ERC grant agreement nr 240487, and of the Ministerio de Ciencia e Innovación (DPI2011-26589). CP acknowledges FPI-UPC Grant and FPU Ph. D. Grant (Ministry of Science and Innovation, Spain).

Appendix A. Optimal storage of local maximum-entropy approximants

We review here the calculation of local maximum-entropy (LME) basis functions and their derivatives. We represent spatial gradients of scalar functions by ∇, and we denote by $D\boldsymbol{f}(\boldsymbol{x})$ the matrix of partial derivatives for vector-valued functions. The subindexes a and b refer to nodes. Summation is not implied for repeated node indices (see [20, 23, 26] for further explanation).

Let X be a set of N scattered nodes $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$, where $d = 1, 2, 3$ is the spatial dimension, and their associate set of locality parameters $\{\beta_1, \beta_2, \dots, \beta_N\} \subset \mathbb{R}$. Given a point $\boldsymbol{x}$, recall that the primal list $\mathcal{N}_{\boldsymbol{x}}^X$ contains the indices of the nodes affecting $\boldsymbol{x}$. The evaluation of the basis function corresponding to the nodal point a is computed as

$$p_a(\boldsymbol{x}) = \frac{\exp[-\beta_a|\boldsymbol{x} - \boldsymbol{x}_a|^2 + \boldsymbol{\lambda}^* \cdot (\boldsymbol{x} - \boldsymbol{x}_a)]}{Z(\boldsymbol{x})}, \tag{A.1}$$

where $Z(\boldsymbol{x})$ is a partition function

$$Z(\boldsymbol{x}) = \sum_{a \in \mathcal{N}_{\boldsymbol{x}}^X} \exp[-\beta_a|\boldsymbol{x} - \boldsymbol{x}_a|^2 + \boldsymbol{\lambda}^* \cdot (\boldsymbol{x} - \boldsymbol{x}_a)],$$

and the Lagrange multiplier $\boldsymbol{\lambda}^*$ is the minimizer of the cost function $\ln Z(\boldsymbol{x}, \boldsymbol{\lambda})$ [20], that is

$$\boldsymbol{\lambda}^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{\lambda} \in \mathbb{R}^d} \ln Z(\boldsymbol{x}, \boldsymbol{\lambda}).$$

The first spatial derivatives of the basis functions (gradient) are computed as [20, 23]

$$\nabla p_a(\boldsymbol{x}) = p_a \Big[\boldsymbol{r}_\beta - \boldsymbol{M}_a(\boldsymbol{x} - \boldsymbol{x}_a)\Big] \ \in \mathbb{R}^d, \tag{A.2}$$

where

$$\boldsymbol{r}_\beta(\boldsymbol{x}) = 2 \sum_{b \in \mathcal{N}_{\boldsymbol{x}}^X} \beta_b p_b(\boldsymbol{x} - \boldsymbol{x}_b) \ \in \mathbb{R}^d, \quad \boldsymbol{M}_a = 2\beta_a \boldsymbol{I} - D\boldsymbol{\lambda} \ \in \mathbb{R}^{d\times d}, \quad D\boldsymbol{\lambda}(\boldsymbol{x}) = \Big(\boldsymbol{J}_\beta - \boldsymbol{I}\Big)\boldsymbol{J}^{-1} \ \in \mathbb{R}^{d\times d},$$

$$\boldsymbol{J}_\beta(\boldsymbol{x}) = 2 \sum_{b \in \mathcal{N}_{\boldsymbol{x}}^X} \beta_b p_b \, (\boldsymbol{x} - \boldsymbol{x}_b) \otimes (\boldsymbol{x} - \boldsymbol{x}_b) \ \in \mathbb{R}^{d\times d}, \text{ and } \quad \boldsymbol{J}(\boldsymbol{x}) = \sum_{b \in \mathcal{N}_{\boldsymbol{x}}^X} p_b(\boldsymbol{x} - \boldsymbol{x}_b) \otimes (\boldsymbol{x} - \boldsymbol{x}_b) \ \in \mathbb{R}^{d\times d}.$$

The second spatial derivatives of the basis functions (Hessian matrix) can be written as [26]

$$\begin{aligned} Hp_a(\boldsymbol{x}) = \ & p_a \Big[\boldsymbol{r}_\beta - \boldsymbol{M}_a(\boldsymbol{x} - \boldsymbol{x}_a)\Big] \otimes \Big[\boldsymbol{r}_\beta - \boldsymbol{M}_a(\boldsymbol{x} - \boldsymbol{x}_a)\Big] \\ & + p_a \Big[\boldsymbol{r}_\beta \otimes \boldsymbol{r}_\beta + \boldsymbol{r}_\beta \otimes \boldsymbol{j}_a + \boldsymbol{j}_a \otimes \boldsymbol{r}_\beta + \Big(\boldsymbol{r}_\beta \cdot \boldsymbol{j}_a\Big)\boldsymbol{I}\Big] \\ & + p_a \Big[2(\bar{\beta} - \beta_a)\boldsymbol{I} - \boldsymbol{Q} - \boldsymbol{j}_a \cdot \boldsymbol{T}\Big] \ \in \mathbb{R}^{d\times d}, \end{aligned} \tag{A.3}$$

where

$$\boldsymbol{j}_a = \boldsymbol{J}^{-1}(\boldsymbol{x} - \boldsymbol{x}_a) \ \in \mathbb{R}^d, \quad \boldsymbol{Q}(\boldsymbol{x}) = \sum_{b \in \mathcal{N}_{\boldsymbol{x}}^X} p_b \boldsymbol{M}_b(\boldsymbol{x} - \boldsymbol{x}_b) \otimes \boldsymbol{M}_b(\boldsymbol{x} - \boldsymbol{x}_b) \ \in \mathbb{R}^{d\times d},$$

$$\bar{\beta}(\boldsymbol{x}) = \sum_{b \in \mathcal{N}_{\boldsymbol{x}}^X} \beta_b p_b \ \in \mathbb{R}, \text{ and } \quad \boldsymbol{T}(\boldsymbol{x}) = \sum_{b \in \mathcal{N}_{\boldsymbol{x}}^X} p_b(\boldsymbol{x} - \boldsymbol{x}_b) \otimes \boldsymbol{M}_b(\boldsymbol{x} - \boldsymbol{x}_b) \otimes \boldsymbol{M}_b(\boldsymbol{x} - \boldsymbol{x}_b) \ \in \mathbb{R}^{d\times d\times d}.$$

Appendix B. Quantification of memory usage

We quantify here the memory usage for two different strategies to store local maximum-entropy basis functions and their derivatives: the *full storage* and the *optimal* or *compressed storage* methods.

The basis functions are usually computed and stored in memory for a given a set of L quadrature points $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_L\} \subset \mathbb{R}^d$ and the associated set of primal lists $\{\mathcal{N}^X_{\boldsymbol{y}_1}, \mathcal{N}^X_{\boldsymbol{y}_2}, \ldots, \mathcal{N}^X_{\boldsymbol{y}_L}\}$ (see Section 2). By defining as $n_k = |\mathcal{N}^X_{\boldsymbol{y}_k}|$ the cardinality corresponding to the primal list of the quadrature point $\boldsymbol{y}_k$, we can construct the set of cardinalities $\{n_1, n_2, \ldots, n_L\} \subset \mathbb{R}$. To simplify the calculations, we define the mean cardinality as $\bar{n} = (\sum_{k=1}^{L} n_k)/L$.

The *full storage* (FS) method demands a massive usage of memory because basis functions and first and second derivatives associated to all the nodal points, and evaluated at all the quadrature points, need to be stored in memory. The calculation of memory usage is straightforward from the analysis of Eqs. A.1, A.2 and A.3 (see Table B.2 for a summary): $M_{FS} = L \cdot \bar{n} \cdot (1 + d + d(d+1)/2) = L \cdot \bar{n} \cdot \left(1 + \frac{3}{2}d + \frac{1}{2}d^2\right)$ doubles, where here and elsewhere we exploit the symmetry of matrices (here the Hessian) to reduce storage. On the other hand, the *optimal* or *compressed storage* (OS) method only requires the storage of some specific variables associated to the quadrature points. We quantify the memory usage of this method in Table B.2: $M_{OS} = L \cdot \left(2 + \frac{7}{2}d + 2d^2 + \frac{1}{2}d^3\right) \approx L \cdot (2+d) \cdot \left(1 + \frac{3}{2}d + \frac{1}{2}d^2\right)$ doubles. As the mean cardinality is regularly much greater than the spatial dimension, i.e., $\bar{n} \gg (2+d)$, from the ratio $M_{FS}/M_{OS} = \bar{n}/(2+d)$ we can conclude that the memory usage decreases significantly when the compressed storage technique is used.

Appendix C. Data structure and specialized algorithms

We detail here the data structures and algorithms proposed to handle more efficiently sparse matrices in the context of meshfree methods. The data structures are specifically designed to store the neighborhood index sets for particular "entities" (elements, nodes, or quadrature points). The algorithms described are responsible for the creation of the sparse matrix structure and the assembly process.

Appendix C.1. Data structure to store neighbor lists

The data structure to store neighbor lists is inspired in the compressed sparse row storage format (see Section 2) and consists of two arrays, one indicating the number of neighboring points to an entity (*pointer_array*), and the other containing the index or identification number of each one of these points (*index_array*). Depending on the kind of assembly process, we need to construct at least two of the following four neighborhood index sets:

- Primal lists: set of neighboring nodes to each quadrature point. These lists, stored in the arrays *is_n* and *js_n*, are required for the assembly process based on a loop over the quadrature points (see Section 2).

Table B.2: Quantification of memory usage for two methods that store local maximum-entropy basis functions and their derivatives. The *optimal* or *compressed storage* (OS) technique needs approximately $L\cdot(2+d)\cdot\left(1+\frac{3}{2}d+\frac{1}{2}d^2\right)$ doubles, while the *full storage* (FS) method demands $L\cdot\bar{n}\cdot\left(1+\frac{3}{2}d+\frac{1}{2}d^2\right)$ doubles, where L is the number of quadrature points, d the spatial dimension, and $\bar{n}\gg(2+d)$ the mean cardinality of the primal lists. The ratio M_{FS}/M_{OS} shows that memory usage decreases significantly when the compressed storage technique is used.

	Full storage		Optimal storage	
	Variable	Memory usage	Variable	Memory usage
Basis functions	p_a	$L\cdot\bar{n}$	$Z(\boldsymbol{x})$	L
			$\boldsymbol{\lambda}(\boldsymbol{x})$	$L\cdot d$
First spatial derivatives	∇p_a	$L\cdot\bar{n}\cdot d$	$\boldsymbol{r}_\beta(\boldsymbol{x})$	$L\cdot d$
			$D\boldsymbol{\lambda}(\boldsymbol{x})$	$L\cdot d\cdot(d+1)/2$
Second spatial derivatives	Hp_a	$L\cdot\bar{n}\cdot d\cdot(d+1)/2$	$\bar{\beta}(\boldsymbol{x})$	L
			$\boldsymbol{J}(\boldsymbol{x})$	$L\cdot d\cdot(d+1)/2$
			$\boldsymbol{Q}(\boldsymbol{x})$	$L\cdot d\cdot(d+1)/2$
			$\boldsymbol{T}(\boldsymbol{x})$	$L\cdot d\cdot d\cdot(d+1)/2$
Total memory usage	$M_{FS}=L\cdot\bar{n}\cdot\left(1+\frac{3}{2}d+\frac{1}{2}d^2\right)$ $\bar{n}\gg(2+d)$		$M_{OS}=L\cdot\left(2+\frac{7}{2}d+2d^2+\frac{1}{2}d^3\right)$ $M_{OS}\approx L\cdot(2+d)\cdot\left(1+\frac{3}{2}d+\frac{1}{2}d^2\right)$	
Comparison	$M_{FS}/M_{OS}\approx\bar{n}/(2+d)\gg 1$			

- Dual lists: set of the neighboring quadrature points to each nodal point (see Section 2 for details). These lists, stored in the arrays *in_s* and *jn_s*, are dual to the primal lists.

- Lists of the neighboring nodal points to each cell/element, stored in the arrays *ie_n* and *je_n*. These sets, defined in Section 3.1, are needed for the assembly process based on a loop over the cell/elements.

- Lists of the neighboring cell/elements to each nodal point (arrays *in_e* and *jn_e*). These lists are dual to those contained in the set of arrays *ie_n* and *je_n*.

We use here the primal lists to explain how the information of the neighborhood index sets is stored in the arrays *pointer_array* and *index_array*, which in this work are respectively referred as is_n and js_n. Given a set of L quadrature points $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_L\} \subset \mathbb{R}^d$ and the associated set of primal lists $\{\mathcal{N}^X_{\boldsymbol{y}_1}, \mathcal{N}^X_{\boldsymbol{y}_2}, \ldots, \mathcal{N}^X_{\boldsymbol{y}_L}\}$, where d is the spatial dimension, $\mathcal{N}^X_{\boldsymbol{y}_k} = \{a \in \{1, 2, .., N\} \mid p_a(\boldsymbol{y}_k) > \texttt{Tol}_0\}$ the primal list for the quadrature point $\boldsymbol{y}_k$, N the total number of nodes, $\texttt{Tol}_0$ a numerical tolerance, and $p_a(\boldsymbol{y}_k)$ the evaluation at the point $\boldsymbol{y}_k$ of the basis function corresponding to the node a, the information stored in the arrays is the following:

- is_n: the component $p + 1$ of this array is defined as $is\_n(p + 1) = \sum_{k=1}^{p} |\mathcal{N}^X_{\boldsymbol{y}_k}|$. In other words, the element (or position) $p + 1$ of the array contains the summation of the cardinalities of the primal lists associated with the first p quadrature points. Note that the first component is always zero and the length of the array is $\dim(is\_n) = L + 1$.

- js_n: this array, which stores consecutively in memory all the primal lists, is defined as $js\_n = (\mathcal{N}^X_{\boldsymbol{y}_1}, \mathcal{N}^X_{\boldsymbol{y}_2}, \ldots, \mathcal{N}^X_{\boldsymbol{y}_L})$. The length of the array is $\dim(js\_n) = \sum_{k=1}^{L} |\mathcal{N}^X_{\boldsymbol{y}_k}|$, where the cardinality can be different for each quadrature point. Note that the order of the quadrature points is important and, in general, $\dim(js\_n) \ll N \cdot L$.

Appendix C.2. Algorithms for matrix structure creation and assembly process

The algorithms implemented to create the sparse matrix structure and the assembly process are presented here in a C/C++ pseudo-code (declarations are left out for the sake of clarity). The three routines detailed in the subsequent sections are:

- CreateElementBasicStructure1D(): this algorithm creates the arrays $ia1$ and $ja1$ of the sparse matrix structure for the case in which the physical field is scalar. The neighbor lists is_n and js_n are used in the method based on a loop over the quadrature points, and the lists *ie_n*, *je_n* in the cell/element scheme.

- CreateStructureND(): extension of the previous algorithm to the n-dimensional case, i.e., when the physical field is vectorial. The arrays created are denoted by *ia* and *ja*.

- FillStructureND(): algorithm to fill the array *an* by executing the operations implemented in the pointer function $*pfunction$. The arrays *ia* and *ja* are needed in the assembly process to loop over the rows and columns of the sparse matrix.

Appendix C.2.1. CreateElementBasicStructure1D()

```
/* 1. Method based on a loop over the quadrature points */
// Input data:
//   - is_n and js_n: lists of the neighboring nodes to the quadrature points
//   - in_s and jn_s: lists of the neighboring quadrature points to the nodal points

// Creation of list ia1 for the case in which the physical field is scalar
iwa=new int[nPts];  // nPts: number of nodal points
for (i=0;i<nPts;i++) iwa[i]=0;  // auxiliary arrays
sumrow=0; l_ia1=nPts+1; ia1=new int[nPts+1]; ia1[0]=0;  // auxiliary arrays

// loop over matrix rows
for (i=0;i<nPts;i++){
    // loop over the neighboring quadrature points to a nodal point
    for (j=in_s[i];j<in_s[i+1];j++){
        // loop over the neighboring nodes to a quadrature point
        for (k=is_n[jn_s[j]];k<is_n[jn_s[j]+1];k++) iwa[js_n[k]]=1;
    }
    // loop over all the nodes
    for (kk=0;kk<nPts;kk++){ sumrow+=iwa[kk]; iwa[kk]=0; }
    ia1[i+1]=ia1[i]+sumrow;
    sumrow = 0;
}

// Creation of list ja1 for the case in which the physical field is scalar
l_ja1=ia1[nPts];
ja1=new int[ia1[nPts]];
std::set<int> row_list;
std::set<int>::const_iterator

// loop over matrix rows
for (i=0;i<nPts;i++){
    // loop over the neighboring quadrature points to a nodal point
    for (j=in_s[i];j<in_s[i+1];j++){
        // loop over the neighboring nodes to a quadrature point
        for (k=is_n[jn_s[j]];k<is_n[jn_s[j]+1];k++) row_list.insert(js_n[k]);
    }
    sit (row_list.begin()),
    send(row_list.end());
    for (kk=0;sit!=send;++sit,kk++) ja1[ia1[i]+kk]=*sit;
    row_list.clear();
}
```

```
/* 2. Method based on a loop over cell/elements */
// Input data:
//   - ie_n and je_n: lists of the neighboring nodal points to the elements
//   - in_s and jn_s: lists of the neighboring quadrature points to the nodal points

// Creation of dual lists in_e, jn_e
in_e=new int[nPts+1];
jn_e=new int[ie_n[nElem]];
for (i=0;i<nPts+1;i++) in_s[i]=0;
for (i=0;i<nElem;i++) for (j=ie_n[i];j<ie_n[i+1];j++) in_e[je_n[j]+1]+=1;
for (i=0;i<nPts;i++) in_e[i+1]+=in_e[i];

count=new int[nPts];
for (i=0;i<nPts;i++) count[i]=0;

// loop over elements
for (i=0;i<nElem;i++) {
    // loop over the neighboring nodes to an element
    for (j=ie_n[i];j<ie_n[i+1];j++){
        jn_e[in_s[je_n[j]]+count[je_n[j]]]=i;
        count[je_n[j]]+=1;
    }
}

// Creation of list ia1 for the case in which the physical field is scalar
iwa=new int[nPts];
for (i=0;i<nPts;i++) iwa[i]=0;
sumrow=0;
l_ia1=nPts+1; ia1=new int[nPts+1]; ia1[0]=0;

// loop over the rows
for (i=0;i<nPts;i++){
    // loop over the neighboring elements to a node
    for (j=in_e[i];j<in_e[i+1];j++){
        pos=ie_n[jn_e[j]];
        for (k=pos;k<ie_n[jn_e[j]+1];k++) iwa[je_n[k]]=1;
    }
    for (kk=0;kk<nPts;kk++){
        sumrow+=iwa[kk];
        iwa[kk]=0;
    }
    ia1[i+1]=ia1[i]+sumrow;
```

```
}

// Creation of list ja1 for the case in which the physical field is scalar
l_ja1=ia1[nPts];
ja1=new int[ia1[nPts]];
std::set<int> row_list;
std::set<int>::const_iterator

// loop over the rows
for (i=0;i<nPts;i++){
    // loop over the neighboring elements to a node
    for (j=in_e[i];j<in_e[i+1];j++){
        pos=ie_n[jn_e[j]];
        for (k=pos;k<ie_n[jn_e[j]+1];k++) row_list.insert(je_n[k]);
    }
    sit (row_list.begin()),
    send(row_list.end());
    for (kk=0;sit!=send;++sit,kk++) ja1[ia1[i]+kk]=*sit;
    row_list.clear();
}
```

Appendix C.2.2. CreateStructureND()

```
l_ia=(l_ia1-1)*nDim+1;
ia=new int[l_ia];
l_ja=l_ja1*nDim*nDim;
ja=new int[l_ja];
an=new double[l_ja];  // matrix array

// Creation of ia for the case in which the physical field is vectorial
ia[0]=0;
for (i=0;i<l_ia1-1;i++){
    size=ia1[i+1]-ia1[i];
    for (j=0;j<nDim;j++) ia[nDim*i+1+j]=ia[nDim*i]+(j+1)*(size*nDim);
}

// Creation of ja for the case in which the physical field is vectorial
for (i=0;i<l_ia1-1;i++){
    size=ia1[i+1]-ia1[i];
    for (j=0;j<size;j++){
        siseJ=nDim*ja1[ia1[i]+j];
        for (k=0;k<nDim;k++){
            for (kk=0;kk<nDim;kk++) ja[ia[nDim*i+k]+(nDim*j+kk)]=siseJ+kk;
        }
```

```
    }
}
```

Appendix C.2.3. FillStructureND()

```
/* 1. Method based on a loop over the quadrature points */

M=new double[nDim*nNNMax*nDim*nNNMax];  // quadrature point local matrix

for (k=0;k<sPts;k++){  // loop over quadrature points (sPts is the number of Gauss points)
    size=is_n[k+1]-is_n[k];
    for (i=0;i<nDim*nNNMax*nDim*nNNMax;i++) M[i]=0.0;
    for (i=0;i<size;i++){  // loop over neighbors
        for (j=0;j<size;j++){  // loop over neighbors
            for (ii=0;ii<nDim*nDim;ii++) A[ii]=0.0;
            (*pfunction)(A,parameters,shape_functions);  // operation --> get matrix A
            // fill local matrix M
            for (ii=0;ii<nDim;ii++)
                for (jj=0;jj<nDim;jj++)
                    M[(nDim*i+ii)*(size*nDim)+(nDim*j)+jj]=A[ii*nDim+jj];
        }
    }
}

// fill global sparse matrix an with quadrature point contribution
rows=nDim*size;
for (i=0;i<rows;i++){
    if (symmetric) j_ini=i;  // symmetric
    else j_ini=0;
    inc_i=i%nDim;
    base_row=(int)(i/nDim);  // floor row
    genrow=js_n[is_n[k]+base_row]*nDim+inc_i;

    for (j=j_ini;j<rows;j++){
        nc_j=j%nDim;
        base_col=(int)(j/nDim);  // floor row
        gencol=js_n[is_n[k]+base_col]*nDim+inc_j;

        for (kk=ia[genrow];kk<ia[genrow+1];kk++){
            if (ja[kk]==gencol){
               an[kk]+=M[i*rows+j];
               break;
            }
        }
```

```
    }
}

/* 2. Method based on a loop over cell/elements */

M=new double[nDim*nNNMax*nDim*nNNMax];  // cell/element local matrix

for (k=0;k<nElem;k++){  // loop over elements
    size=ie_n[k+1]-ie_n[k];
    for (i=0;i<nDim*nNNMax*nDim*nNNMax;i++) M[i]=0.0;
    for (i=0;i<size;i++){  // loop over neighbors
        for (j=0;j<size;j++){  // loop over neighbors
            for (ii=0;ii<nDim*nDim;ii++) A[ii]=0.0;
            (*pfunction)(A,parameters,shape_functions);  // operation --> get matrix A
            // fill local matrix
            for (ii=0;ii<nDim;ii++)
                for (jj=0;jj<nDim;jj++)
                    M[(nDim*i+ii)*(size*nDim)+(nDim*j)+jj]=A[ii*nDim+jj];
        }
    }
}

// fill global sparse matrix an with quadrature point contribution
rows=nDim*size;
for (i=0;i<rows;i++){
    if (symmetric) j_ini=i;  // symmetric
    else j_ini=0;
    inc_i=i%nDim;
    base_row=(int)(i/nDim);  // floor row
    genrow=je_n[ie_n[k]+base_row]*nDim+inc_i;

    for (j=j_ini;j<rows;j++){
        nc_j=j%nDim;
        base_col=(int)(j/nDim);  // floor row
        gencol=je_n[ie_n[k]+base_col]*nDim+inc_j;

        for (kk=ia[genrow];kk<ia[genrow+1];kk++){
            if (ja[kk]==gencol){
               an[kk]+=M[i*rows+j];
               break;
            }
        }
    }
```

```
}
```

References

1. Belytschko, T., Krongauz, Y., Organ, D., Fleming, M., Krysl, P.. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering* 1996;**139**(1):3–47.

2. Li, S., Liu, W.. Meshfree and particle methods and their applications. *Applied Mechanics Reviews* 2002;**55**(1):1–34.

3. Fries, T., Matthies, H.. Classification and overview of meshfree methods. Tech. Rep.; Institute of Scientific Computing, Technical University Braunschweig, Germany; July 2004.

4. Huerta, A., Belytschko, T., Fernández-Méndez, S., Rabczuk, T.. *Meshfree Methods*; vol. 1 of *Encyclopedia of Computational Mechanics. E. Stein and R. de Borst and T.J.R. Hughes (eds.)*; chap. 10. John Wiley & Sons, Ltd.; 2004, p. 279–309.

5. Fasshauer, G.. *Meshfree Methods*; chap. 2. Handbook of Theoretical and Computational Nanotechnology. M. Rieth and W. Schommers (eds.). American Scientific Publishers; 2006, p. 33–97.

6. Nguyen, V.P., Rabczuk, T., Bordas, S., Duflot, M.. Meshless methods: A review and computer implementation aspects. *Mathematics and Computers in Simulation* 2008; **79**(3):763–813.

7. Chen, J., Pan C.and Wu, C., Liu, W.. Reproducing kernel particle methods for large deformation analysis of nonlinear structures. *Computer Methods in Applied Mechanics and Engineering* 1996;**139**:195–227.

8. Chen, J., Pan, C., Rogue, C., Wang, H.. A lagrangian reproducing kernel particle method for metal forming analysis. *Computational Mechanics* 1998;**22**:289–307.

9. Li, B., Habbal, F., Ortiz, M.. Optimal transportation meshfree approximation schemes for fluid and plastic flows. *International Journal for Numerical Methods in Engineering* 2010;**83**(12):1541–1579.

10. Combe, U., Korn, C.. An adaptive approach with the element-free-galerkin method. *Computer Methods in Applied Mechanics and Engineering* 1998;**162**:203–222.

11. Duarte, C., Oden, J.. An h–p adaptive method using clouds. *Computer Methods in Applied Mechanics and Engineering* 1996;**139**:237–262.

12. Dolbow, J., Belytschko, T.. Numerical integration of the galerkin weak form in meshfree methods. *Computational Mechanics* 1999;**23**:219–230.

13. Babuška, I., Banerjee, U., Osborn, J., Li, Q.. Quadrature for meshless methods. *International Journal for Numerical Methods in Engineering* 2008;**76**:1434–1470.

14. Fernández-Méndez, S., Huerta, A.. Imposing essential boundary conditions in mesh-free methods. *Computer Methods in Applied Mechanics and Engineering* 2004;**193**(12–14):1257–1275.

15. Monaghan, J.. An introduction to SPH. *Computer Physics Communications* 1988;**48**:89–96.

16. Liu, W., Jun, S., Zhang, Y.. Reproducing kernel particle methods. *International Journal for Numerical Methods in Fluids* 1995;**20**:1081–1106.

17. Melenk, J., Babuška, I.. The partition of unity finite element method : Basic theory and applications. *Computer Methods in Applied Mechanics and Engineering* 1996;**139**(1–4):289–314.

18. Belytschko, T., Lu, Y., Gu, L.. Element free Galerkin methods. *International Journal for Numerical Methods in Engineering* 1994;**37**:229–256.

19. Sukumar, N.. Construction of polygonal interpolants: a maximum entropy approach. *International Journal for Numerical Methods in Engineering* 2004;**61**(12):2159–2181.

20. Arroyo, M., Ortiz, M.. Local maximum-entropy approximation schemes: a seamless bridge between finite elements and meshfree methods. *International Journal for Numerical Methods in Engineering* 2006;**65**(13):2167–2202.

21. Sukumar, N., Malsch, E.. Recent advances in the construction of polygonal finite element interpolants. *Archives of Computational Methods in Engineering* 2006;**13**(1):129–163.

22. Sukumar, N., Wright, R.. Overview and construction of meshfree basis functions: From moving least squares to entropy approximants. *International Journal for Numerical Methods in Engineering* 2007;**70**(2):181–205.

23. Rosolen, A., Millán, D., Arroyo, M.. On the optimum support size in meshfree methods: a variational adaptivity approach with maximum entropy approximants. *International Journal for Numerical Methods in Engineering* 2010;**82**(7):868–895.

24. Ullah, Z., Coombs, W., Augarde, C.. An adaptive finite element/meshless coupled method based on local maximum entropy shape functions for linear and nonlinear problems. *Computer Methods in Applied Mechanics and Engineering* 2013;**267**:111–132.

25. Hale, J., Baiz, P.. A locking-free meshfree method for the simulation of shear-deformable plates based on a mixed variational formulation. *Computer Methods in Applied Mechanics and Engineering* 2012;**241-244**:311–322.

26. Millán, D., Rosolen, A., Arroyo, M.. Thin shell analysis from scattered points with maximum-entropy approximants. *International Journal for Numerical Methods in Engineering* 2011;**85**(6):723–751.

27. Millán, D., Rosolen, A., Arroyo, M.. Nonlinear manifold learning for meshfree finite deformation thin shell analysis. *International Journal for Numerical Methods in Engineering* 2013;**93**(7):685–713.

28. Nissen, K., Cyron, C., Gravemeier, V., Wall, W.. Information-flux method: a meshfree maximum-entropy petrov-galerkin method including stabilised finite element methods. *Computer Methods in Applied Mechanics and Engineering* 2012;**241-244**:225–237.

29. Wu, C., Young, D., Hong, H.. Adaptive meshless local maximum-entropy finite element method for convection-diffusion problems. *Computational Mechanics* 2014;**53**:189–200.

30. Rosolen, A., Peco, C., Arroyo, M.. An adaptive meshfree method for phase-field models of biomembranes. Part I: approximation with maximum-entropy approximants. *Journal of Computational Physics* 2013;**249**:303–319.

31. Peco, C., Rosolen, A., Arroyo, M.. An adaptive meshfree method for phase-field models of biomembranes. Part II: a Lagrangian approach for membranes in viscous fluids. *Journal of Computational Physics* 2013;**249**:320–336.

32. Amiri, F., Anitescu, C., Arroyo, M., Bordas, S., Rabczuk, T.. XLME interpolants, a seamless bridge between XFEM and enriched meshless methods. *Computational Mechanics* 2014;**53**:45–57.

33. Amiri, F., Millán, D., Shen, Y., Rabczuk, T., Arroyo, M.. Phase-field modeling of fracture mechanics in linear thin shells. *Theoretical and Applied Fracture Mechanics* 2014;**69**:102–109.

34. Li, B., Peco, C., Millán, D., Arias, I., Arroyo, M.. Phase-field modeling and simulation of fracture in brittle materials with strongly anisotropic surface energy. *International Journal for Numerical Methods in Engineering* 2014;:DOI: 10.1002/nme.4726.

35. Arroyo, M., Ortiz, M.. *Meshfree Methods for Partial Differential Equations III*; vol. 57 of *Lecture Notes in Computational Science and Engineering*; chap. Local Maximum-Entropy Approximation Schemes. Springer; 2007, p. 1–16.

36. Du, Q., Gunzburger, M., Ju, L.. Meshfree, probabilistic determination of point sets and support regions for meshless computing. *Computer Methods in Applied Mechanics and Engineering* 2002;**191**(13-14):1349–1366.

37. Bradford Barber, C., Dobkin, D.P., Huhdanpaa, H.. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software* 1996;**22**:469–483.

38. Mount, M., Arya, S.. A library for approximate nearest neighbor searching. http://www.cs.umd.edu/~mount/ANN/; 2010.

39. Eijkhout, V.. Distributed sparse data structures for linear algebra operations. Technical Report CS 92-169; Computer Science Department, University of Tennessee; 1992.

40. Saad, Y.. *Iterative Methods for Linear Systems*. PWS Publishing, Boston; 1996.

41. Borden, M., Hughes, T., Landis, C., Verhoosel, C.. A higher-order phase-field model for brittle fracture: Formulation and analysis within the isogeometric analysis framework. *Computer Methods in Applied Mechanics and Engineering* 2014;**273**:100–118.

42. Balay, S., Brown, J., Buschelman, K., Gropp, W.D., Kaushik, D., Knepley, M.G., et al. Portable, extensible toolkit for scientific computation. http://www.mcs.anl.gov/petsc; 2013.

43. Karypis, G., Kumar, V.. Metis-ParMetis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0.3. http://www.cs.umn.edu/~metis; 2009.

Appendix D
Meshfree Parallel Algorithms

The algorithms implemented to handle the matrix assembly in parallel are presented here in a C/C++ and PETSc pseudo-code (declarations are left out for the sake of clarity). We limit ourselves to the routines detailed in the subsequent sections:

- CreateExtendedConnectivity(): this algorithm takes the local connectivity lists (is_n, js_n) of each process integration points set and extend them with the communication bands, obtaining is_n_ex and js_n_ex. In this way every process will be able to locally compute the non-zero positions and create a local sparse matrix structure in one dimension.

- CreateSructureDimOne(): generates in parallel $ia1$ and $ja1$ of the sparse matrix structure for the case in which the physical field is scalar. The neighbor lists is_n_ex and js_n_ex are used in the method based on a loop over the quadrature points, and the lists ie_n_ex, je_n_ex in the cell/element scheme.

- CreateStructureDim(): extension of the previous algorithm to the n-dimensional case, i.e., when the physical field is vectorial. The arrays created are denoted by ia and ja.

- FillStructureDimN(): algorithm to fill the matrix A by executing the operations implemented in the pointer function $*pfunction$. The arrays ia and ja are needed in the assembly process to loop over the rows and columns of the sparse matrix. The space is exactly preallocated through PETSc routines.We include the algorithm to fill the values corresponding to the Lagrange multipliers in the global matrix.

CreateExtendedConnectivity()

```
// Input data:
//   - is_n and js_n: lists of the neighboring nodes to the quadrature points
```

```
//   - l_is_n: length of is_n list
//   - low, high: range of nodes/rows which owns the process
//   - map_part: the global mapping from node to partition

general_count = new int [num_procs];
// the integration point info is to be sent to a particular process  (1-yes 0-no)
count_is_aux= new int [num_procs];
// cumulative number of integration points sent to each process
block_count= new int [num_procs];
// prevents from double counting an integration point.
for (i=0;i<num_procs;i++){general_count[i]=0;count_is_aux[i]=0;}

// counting extra integration points for each process
for (i=0;i<l_is_n-1;i++){// loop in integration points
    for (k=0;k<num_procs;k++){block_count[k]=0;}
    for (j=is_n[i];j<is_n[i+1];j++){// loop in nodes
        node_proc=map_part[js_n[j]];
        if (((js_n[j]<low) || (js_n[j]>=high)) && (block_count[node_proc]==0)){
            general_count[node_proc]=1;
            block_count[node_proc]=1;
            count_is_aux[node_proc] += 1;
        }
    }
}


// integration list for each processor
count_is= new int [num_procs+1]; count_is[0]=0;
for (i=1;i<num_procs+1;i++) count_is[i] = count_is[i-1] + count_is_aux[i-1];
```

```
// cumulative count
count_js=new int [count_is[num_procs]];
counter=new int[num_procs];
for (i=0;i<num_procs;i++) counter[i]=0;// fill count_js
for (i=0;i<l_is_n-1;i++){// loop integration points
        for (k=0;k<num_procs;k++) block_count[k]=0;
        for (j=is_n[i];j<is_n[i+1];j++){// loop nodes
                node_proc=map_part[js_n[j]];
                if (((js_n[j]<low) || (js_n[j]>=high)) && (block_count[node_proc]==0)){
                block_count[node_proc]=1 ;
                count_js[count_is[node_proc]+counter[node_proc]] = i;
                counter[node_proc] +=1;
               }
        }
}
// count_is, count_js send to each processor the correspondent line of is_n,js_n
// allocate space for count_is
sdispls=new int[num_procs];
rdispls=new int[num_procs];
recv_cnts=new int[num_procs];
MPI_Alltoall(count_is_aux, 1, MPI_INT,recv_cnts, 1, MPI_INT, PETSC_COMM_WORLD);
//each process knows now how many integration points will come from every process
int l_is_n_loc=0;
for (i=0;i<num_procs;i++) l_is_n_loc += recv_cnts[i];
is_n_loc=new int[l_is_n_loc];
// store of number of nodes coming for every integration point received
// create is_n_aux with number of nodes seen by every integration point
int l_is_n_aux, integration_p, counter_int;
l_is_n_aux=count_is[num_procs];//number of integration points to send
```

```
is_n_aux=new int[l_is_n_aux];
counter_int=0;
for (i=0;i<num_procs;i++){
    for (j=count_is[i];j<count_is[i+1];j++){//points in process
        integration_p=count_js[j];
        is_n_aux[counter_int] = is_n[integration_p+1]-is_n[integration_p];
        counter_int++;
    }
}
// send info regarding is_n_aux
for (i=0;i<num_procs;i++) sdispls[i]=count_is[i];
rdispls[0]=0;
for (i=1;i<num_procs;i++) rdispls[i]=rdispls[i-1]+recv_cnts[i-1];
MPI_Alltoallv(is_n_aux, count_is_aux, sdispls,MPI_INT,
is_n_loc,recv_cnts,rdispls, MPI_INT, PETSC_COMM_WORLD);
//create js_n_aux
int l_js_n_aux=0;
for (i=0;i<l_is_n_aux;i++) l_js_n_aux += is_n_aux[i];
js_n_aux=new int[l_js_n_aux];
counter_int=0;
for (i=0;i<num_procs;i++){//process
    for (j=count_is[i];j<count_is[i+1];j++){// local integration points
        integration_p=count_js[j];
        for (k=is_n[integration_p];k<is_n[integration_p+1];k++){// local nodes
            js_n_aux[counter_int] = js_n[k];//aadimos lista de nodos
            counter_int++;
        }
    }
}
```

```
// send info regarding js_n_aux
int l_js_n_loc=0;
for (i=0;i<l_is_n_loc;i++) l_js_n_loc += is_n_loc[i];
js_n_loc=new int[l_js_n_loc];
count_is_n_aux=new int[num_procs];
for (i=0;i<num_procs;i++) count_is_n_aux[i]=0;
for (i=0;i<num_procs;i++){
    for (j=count_is[i];j<count_is[i+1];j++){// local integration points
        count_is_n_aux[i] += is_n_aux[j];
    }
}
sdispls[0]=0;
for (i=1;i<num_procs;i++) sdispls[i]= sdispls[i-1] + count_is_n_aux[i-1];
recv_cnts_ac=new int[num_procs+1];
recv_cnts_ac[0]=0;
for (i=1;i<num_procs+1;i++) recv_cnts_ac[i] = recv_cnts_ac[i-1]+recv_cnts[i-1];
count_is_n_loc=new int[num_procs];
for (i=0;i<num_procs;i++) count_is_n_loc[i]=0;
for (i=0;i<num_procs;i++) {
    for (j=recv_cnts_ac[i];j<recv_cnts_ac[i+1];j++) count_is_n_loc[i] += is_n_loc[j];
}
rdispls[0]=0;
for (i=1;i<num_procs;i++) rdispls[i]= rdispls[i-1] + count_is_n_loc[i-1];
MPI_Alltoallv(js_n_aux, count_is_n_aux, sdispls,MPI_INT,
js_n_loc, count_is_n_loc, rdispls, MPI_INT, PETSC_COMM_WORLD);
// with is_n_loc and js_n_loc extend the original lists
// is_n_ex dimension and fill
l_is_n_ex = l_is_n + l_is_n_loc;
is_n_ex=new int [l_is_n_ex];
```

```
for (i=0;i<l_is_n;i++) is_n_ex[i]=is_n[i];
for (i=l_is_n;i<l_is_n_ex;i++) is_n_ex[i] = is_n_ex[i-1] + is_n_loc[i-l_is_n];
// js_n_ex dimension and fill
l_js_n_ex = l_js_n + l_js_n_loc;
js_n_ex=new int [l_js_n_ex];
for (i=0;i<l_js_n;i++) js_n_ex[i]=js_n[i];
for (i=l_js_n;i<l_js_n_ex;i++) js_n_ex[i]=js_n_loc[i-l_js_n];
```

CreateStructureDimOne()

```
// Input data:
//   - is_n_ex and js_n_ex: extended neighbor lists
//   - low, high: range of nodes/rows which owns the process

//Creation of ia1
iwa=new int[nPts];
for (i=0;i<nPts;i++) iwa[i]=0;// auxiliar array
l_ia1=loc_nPts+1; ia1=new int[l_ia1]; ia1[0]=0;
ia1_d=new int[l_ia1]; ia1_d[0]=0; ia1_od=new int[l_ia1]; ia1_od[0]=0;

for (i=0;i<loc_nPts;i++){// loop in local rows
    for (j=in_s[i];j<in_s[i+1];j++){//integration points
        for(k=is_n_ex[jn_s[j]];k<is_n_ex[jn_s[j]+1];k++) iwa[js_n_ex[k]]=1;
    }
for (kk=0;kk<low;kk++){sumrow_left += iwa[kk];
iwa[kk]=0;
}
for (kk=low;kk<high;kk++){sumrow_d += iwa[kk];// in-diagonal
iwa[kk]=0;
```

```
}
for (kk=high;kk<nPts;kk++){sumrow_right += iwa[kk];
iwa[kk]=0;
}
sumrow_od = sumrow_left + sumrow_right;// off-diagonal
sumrow=sumrow_d+sumrow_od;// total

ia1_d[i+1]=ia1_d[i]+sumrow_d;
ia1_od[i+1]=ia1_od[i]+sumrow_od;
ia1[i+1]=ia1[i]+sumrow;
sumrow_left=0; sumrow_right=0; sumrow_d=0; sumrow_od=0; sumrow=0;
}

// Creation of ja1
l_ja1=ia1[loc_nPts];
ja1=new int[l_ja1];
std::set<int> row_list;
for (i=0;i<loc_nPts;i++){// loop in local rows
    for (j=in_s[i];j<in_s[i+1];j++){// integration points
        for(k=is_n_ex[jn_s[j]];k<is_n_ex[jn_s[j]+1];k++)
          row_list.insert(js_n_ex[k]);
    }
     std::set<int>::const_iterator
        sit (row_list.begin()),
        send(row_list.end());
        kk=0;
        for(;sit!=send;++sit) {ja1[ia1[i]+kk]= *sit; kk++;}
        row_list.clear();
```

CreateStructureDimN()

```
// Input data:
//   - ia1, ja1: one-dimensional sparse matrix structure
//   - low, high: range of nodes/rows which owns the process
//   - num_lagrange: number of Lagrange multiplier constraints

l_ia=(l_ia1-1)*nDim+1;
if (myrank==(num_procs-1)) l_ia += num_lagrange;
ia=new int[l_ia];
ia_d=new int[l_ia];
ia_od=new int[l_ia];
// ia
ia[0]=0.0;
for (i=0;i<l_ia1-1;i++){
    size=ia1[i+1]-ia1[i];
    size_d=ia1_d[i+1]-ia1_d[i];
    size_od=ia1_od[i+1]-ia1_od[i];
    for (j=0;j<nDim;j++){
        ia[nDim*i+1+j]=ia[nDim*i]+(j+1)*(size*nDim);
        ia_d[nDim*i+1+j]=ia[nDim*i]+(j+1)*(size_d*nDim);
        ia_od[nDim*i+1+j]=ia[nDim*i]+(j+1)*(size_od*nDim);
        }
    }
// ja
for (i=0;i<l_ia1-1;i++){
    size=ia1[i+1]-ia1[i];
    for (j=0;j<size;j++){
        for (k=0;k<nDim;k++){
```

```
            for (kk=0;kk<nDim;kk++){
            ja[ia[nDim*i+k]+(nDim*j+kk)]= nDim*ja1[ia1[i]+j]+kk;
            }
        }
    }
}
int n_cols;
n_cols= nDim*nPts;//basic number of cols
int diff=0;int count;
if (num_lagrange>0){
    for (i=0;i<l_ia_aux-1;i++){// in regular matrix lines
    diff=0;
    row=nDim*low + i;
    size_lag=ia[i+1]-ia[i];
    diff=size_lag-original_size[i];

    if (diff>0){
        count=0;
        for (j=0;j<num_lagrange;j++){
            for (k=0;k<long_lagrange_nodes[j];k++){
                if (lagrange_nodes[j][k]==row){
                    ja[ia[i] + original_size[i] + count]= n_cols + j;
                    count++;
                    break;
                }
            }
        }
    }
}
```

```
//add last lines to last process
if (myrank==num_procs-1){
    for (j=0;j<num_lagrange;j++){
        for (k=0;k<long_lagrange_nodes[j];k++)
            ja[ia[l_ia_aux-1 + j] + k] = lagrange_nodes[j][k];
        ja[ia[l_ia_aux + j] - 1] = n_cols + j;//add diagonal zero
        }
    }
}

max_ia=0;
// local maximum ia
for (i=0;i<l_ia-1;i++) if ((ia[i+1]-ia[i])>max_ia) max_ia=ia[i+1]-ia[i];
if (num_procs==1) MatSeqAIJSetPreallocationCSR(A,ia,ja,PETSC_NULL);
else MatMPIAIJSetPreallocationCSR(A,ia,ja,PETSC_NULL);
```

FillStructureDim()

```
// Input data:
//   - is_n and js_n: lists of the neighboring nodes to the quadrature points
//   - low, high: range of nodes/rows which owns the process

MatZeroEntries(A);
int *global_indexes_M=NULL;
global_indexes_M=new int[nNNMax*nDim];
M=new double [nDim*nNNMax*nDim*nNNMax];

for (k=0;k<sPts;k++){// loop in local integration points
```

```
//shape functions of integration point
maxent->ComputeBasisFunctions( k );
ps  = maxent->GetShapeFunctions();
dps =  maxent->GetGradients();
hps =  maxent->GetHessians();
size=is_n[k+1]-is_n[k];
for (i=0;i<nDim*nNNMax*nDim*nNNMax;i++) M[i]=0.0;
// initialize to 0.0 for integration point

// Option A: node-to-node operations
for (i=0;i<size;i++)
    for (j=0;j<size;j++)
    p_g = k;//integration point number
    wpos_g[0]=w_s[k];//weight
    for (ii=0;ii<sDim;ii++) wpos_g[1 + ii] = x_s[sDim*k + ii];
    //coordinates of integration point
    sparam for node-to-node operations
    sparam[0]=ps[i];
    sparam[1]=ps[j];
    for (ii=0;ii<sDim;ii++){
    sparam[2+ii]=dps[sDim*i+ii];
    sparam[2+sDim+ii]=dps[sDim*j+ii];
    }
    for (ii=0;ii<sDim*sDim;ii++){
    sparam[2+2*sDim+ii]=hps[sDim*sDim*i+ii];
    sparam[2+2*sDim+sDim*sDim+ii]=hps[sDim*sDim*j+ii];
    }
    //call to correspondent operation --> get matrix A
    for (ii=0;ii<nDim*nDim;ii++) A_g[ii]=0.0;
```

```
        (*pfunction)(A_g,nDim, iparam,dparam, sparam, p_g, wpos_g);
        //fill local neighbour matrix M
        for (ii=0;ii<nDim;ii++) for (jj=0;jj<nDim;jj++)
        M[(nDim*i+ii)*(size*nDim)+(nDim*j)+jj] = A_g[ii*nDim+jj];
    }
       // Option B: integration point operations
       (*pfunction_b)(M, &value_out, iparam, dparam,
       x_n, is_n, js_n, ps, dps, hps, p_g, wpos_g);
}
int pos=0;
// fill global sparse matrix an with integration point contribution
// create vector with global positions
int rows=nDim*size;
int row;
for (i=0;i<size;i++){
    pos=js_n[is_n[k]+i];//node global
    for (j=0;j<nDim;j++) global_indexes_M[nDim*i+j]=nDim*pos+j;
MatSetValues(A,rows,global_indexes_M,rows,global_indexes_M,M,ADD_VALUES);
// send local M matrix directly into the global matrix

if (num_lagrange>0){
int row,col;
// columns in regular rows
for (i=0;i<l_ia_aux-1;i++){
    row = nDim*low + i;
    for (j=0;j<num_lagrange;j++){
    col = nPts*nDim + j;
        for (k=0;k<long_lagrange_nodes[j];k++){
                if (lagrange_nodes[j][k]== row) {
```

```
                MatSetValues(A,1,&row,1,&col,&lagrange_values[j][k],ADD_VALUES);
                break;
                }
        }
    }
}
// last lines
int m_rows;
if (myrank==num_procs-1){
    for (i=0;i<num_lagrange;i++){
     //insert line
      m_rows= nDim*nPts + i ;//basic number of cols + lagrange line
      MatSetValues(A,1,&m_rows,long_lagrange_nodes[i],lagrange_nodes[i],
      lagrange_values[i],ADD_VALUES);
    }
}
MatAssemblyBegin(A,MAT_FINAL_ASSEMBLY);
MatAssemblyEnd(A,MAT_FINAL_ASSEMBLY);
```