Differential geometric bifurcation problems in `pde2path` – algorithms and tutorial examples

Alexander Meiners, Hannes Uecker

Institut für Mathematik, Universität Oldenburg, D26111 Oldenburg, alexander.meiners@uni-oldenburg.de, hannes.uecker@uni-oldenburg.de

August 20, 2024

Abstract

We describe how some differential geometric bifurcation problems can be treated with the MATLAB continuation and bifurcation toolbox `pde2path`. The basic setup consists in solving the PDEs for the normal displacement of an immersed surface $X \subset \mathbb{R}^3$ and subsequent update of X in each continuation step, combined with bifurcation detection and localization, followed by possible branch switching. Examples treated include some minimal surfaces such as Enneper's surface and a Schwarz–P–family, some non–zero constant mean curvature surfaces such as liquid bridges and nodoids, and some 4th order biomembrane models. In all of these we find interesting symmetry breaking bifurcations. Some of these are (semi)analytically known and thus are used as benchmarks.

Contents

1 Introduction

There are various algorithms and toolboxes for the numerical continuation and bifurcation analysis for solutions of partial differential equations (PDEs), for instance `AUTO` [DCF+97], Coco [DS13], `BifurcationKit.jl` [Vel20], and `pde2path` [Uec21, Uec24]. In its standard setup, `pde2path` is for PDEs for functions $u : \Omega \times \Lambda \to \mathbb{R}^N$, where $\Omega \subset \mathbb{R}^d$ is a fixed domain, $d = 1, 2$, or 3, $N \in \mathbb{N}$, and $\Lambda \subset \mathbb{R}^p$ is a parameter domain, or for time–dependent functions $u : I \times \Omega \times \Lambda \to \mathbb{R}^N$, $I \subset \mathbb{R}$, which then includes the continuation and bifurcation of time periodic orbits. Essentially, this also applies to `BifurcationKit.jl`, while wrt PDEs `AUTO` originally focuses on 1D boundary value problems, and Coco in principle allows great flexibility by delegating the PDE definition/discretization to the user.

However, none of these packages seem directly applicable to differential geometric PDEs in parametric form, which deal directly with manifolds, e.g., surfaces in 2D, which are not graphs over a fixed domain. There are well established numerical methods for the discretization of such PDEs, for instance the surface finite element method (surface FEM) [DE13], but there seem to be few algorithms or packages which combine these with continuation and bifurcation. Two notable exceptions are the algorithm from [Bru18], and the `SurfaceEvolver`, for which bifurcation aspects are for instance discussed in [Bra96].

Here we present an extension of `pde2path` aimed at geometric PDE bifurcation problems. We focus on constant mean curvature (CMC) surfaces, which are not necessarily graphs, and with, e.g., the mean curvature, or the area or enclosed volume as the primary bifurcation parameter. See Fig. 1 for a preview of the type of solutions we compute. For X a two dimensional surface immersed in $\mathbb{R}^3$, we for instance want to study the parameter dependent problem

$$H(\cdot) - H_0 = 0, \tag{1a}$$
$$V(X) - V_0 = 0, \tag{1b}$$

possibly with boundary conditions (BCs) in (a), where $H(X)$ is the mean curvature at each point of X, and $V(X)$ is the (properly defined) volume enclosed by X. The system (1) is obtained for minimizing the area $A(X)$ under the volume constraint $V(X) = V_0$, i.e., as the Euler–Lagrange equations for minimizing the energy

$$E(X) = A(X) + H_0(V(X) - V_0), \tag{2}$$

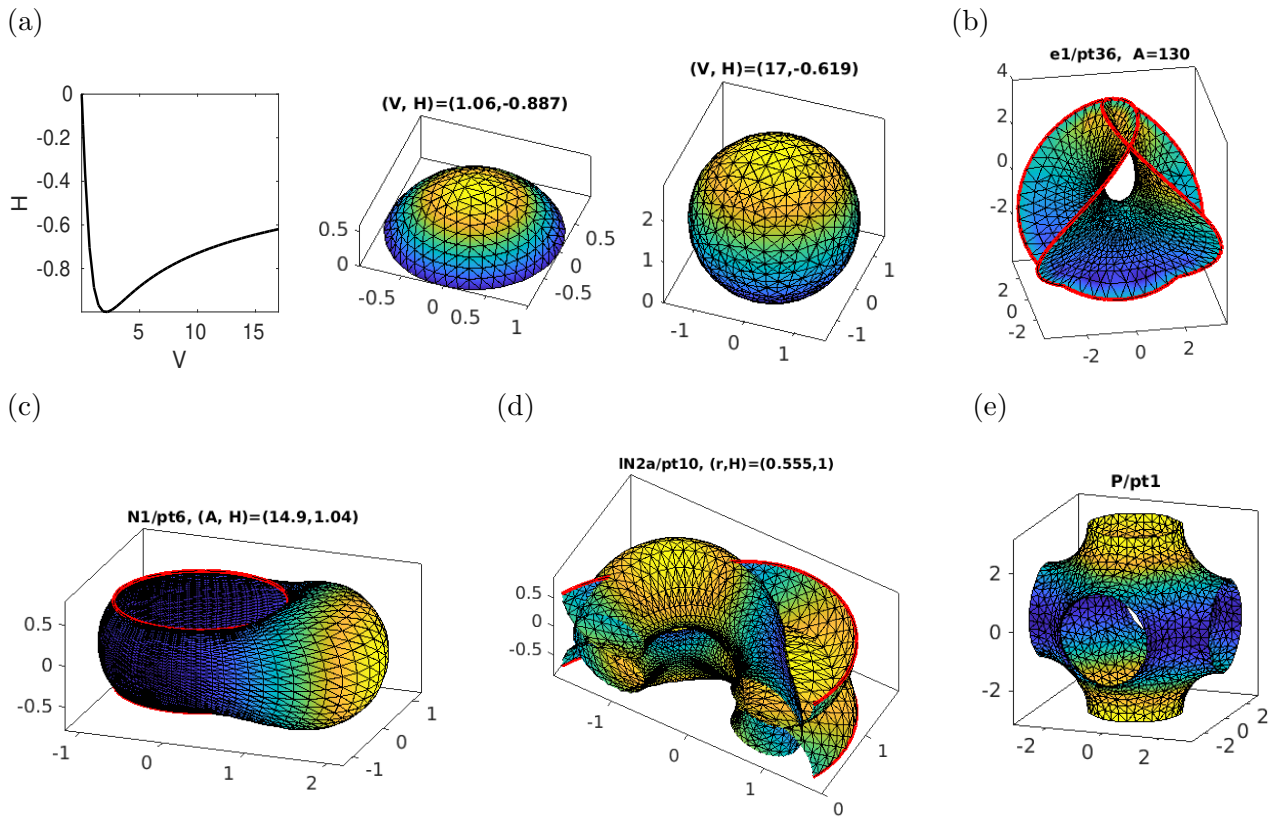

Figure 1: Preview of solutions (solution branches) we compute. (a) H over V for spherical caps, and sample solutions, §3.1. The colors indicate u in the last step, yellow>blue, and thus besides giving visual structure to X indicate the "direction" of the continuation. H is negative since here N is the outer normal. (b) Enneper minimal surface (a bounded part, with the boundary shown in red), §3.2.3. (c) A liquid bridge between two circles, with excess volume and hence after a symmetry breaking bifurcation, §3.3.3. (d) A nodoid with periodic BCs, cut open for plotting, §3.4. (e) A Schwarz P surface, §3.5.1. Samples (b)–(e) are each again from branches of solutions of the respective problems, see Figures 11, 12,17, and 18 for the bifurcation diagrams.

and $V_0 \in \mathbb{R}$ typically plays the role of an "external continuation parameter", while H_0, which for instance describes a spatially constant pressure difference for interfaces between fluids, is "free".

Following [Bru18], our setting for (1) and generalizations is as follows. Let X_0 be a surface satisfying (1) for some V_0 and H_0, and define a new surface via $X = X_0 + u\,N_0$, $u : X_0 \to \mathbb{R}$ with suitable boundary conditions, where $N_0 : X_0 \to \mathbb{S}^2$ is (a choice of) the unit normal vector field of X_0. Then (1) reads

$$G(u, \widetilde{H}) = H(X) - \widetilde{H} = 0 \text{ (with boundary conditions if applicable)}, \tag{3a}$$

which is a quasilinear elliptic equation for u coupled to the volume constraint

$$q(u) = V(X) - \widetilde{V}. \tag{3b}$$

Thus,

$$\text{after solving (3) for } u, \widetilde{H}, \widetilde{V} \text{ we can update } X_0 = X_0 + uN_0,\ H_0 = \widetilde{H}, V_0 = \widetilde{V}, \text{ and repeat.} \tag{4}$$

Importantly, *solution branches* of (3) may move back and forth in parameter space (passing through folds), and hence we do not consider them in "natural" parameterization $\lambda \mapsto u(\cdot, \lambda)$, but in arclength $s \mapsto (u(s), \lambda(s))$ with a (dummy) arclength parameter $s \in \mathbb{R}$. Assuming that after spatial discretization $u \in \mathbb{R}^n$ and that we have one free parameter $\lambda \in \mathbb{R}$, the basic idea is as follows. Given a current

solution $(u_n, \lambda_n) \in \mathbb{R}^{n+1}$, and a tangent vector $\tau \in \mathbb{R}^n$ to the already computed part of the branch, we make a predictor $(u^1, \lambda^1) = (u_n, \lambda_n) + \tau$, and solve the extended system

$$\mathcal{H}(u, \lambda) = \begin{pmatrix} G(u, \lambda) \\ p(u, \lambda, s) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \mathbb{R}^{n+1}, \tag{5}$$

where p is used to make s an approximation of arclength on the branch. The standard choice is

$$p(u, \lambda, s) := \xi \langle u_0', u(s) - u_0 \rangle + (1-\xi)\lambda_0'(\lambda(s) - \lambda_0) - (s - s_0). \tag{6}$$

Here $0 < \xi < 1$ is a weight, typically chosen as $\xi = 1/n$, and τ_0 is assumed to be normalized in the weighted norm

$$\|\tau\|_\xi := \sqrt{\langle \tau, \tau \rangle_\xi}, \qquad \left\langle \begin{pmatrix} u \\ \lambda \end{pmatrix}, \begin{pmatrix} v \\ \mu \end{pmatrix} \right\rangle_\xi := \xi \langle u, v \rangle + (1 - \xi)\lambda\mu. \tag{7}$$

For fixed s and $\|\tau_0\|_\xi = 1$, $p(u, \lambda, s) = 0$ thus defines a hyperplane perpendicular (in the inner product $\langle \cdot, \cdot \rangle_\xi$) to τ_0 at distance $\mathrm{ds} := s - s_0$ from (u_n, λ_n). Solving (5) (typically by a Newton method) thus means solving (3) on that hyperplane, which in particular allows to go around folds.

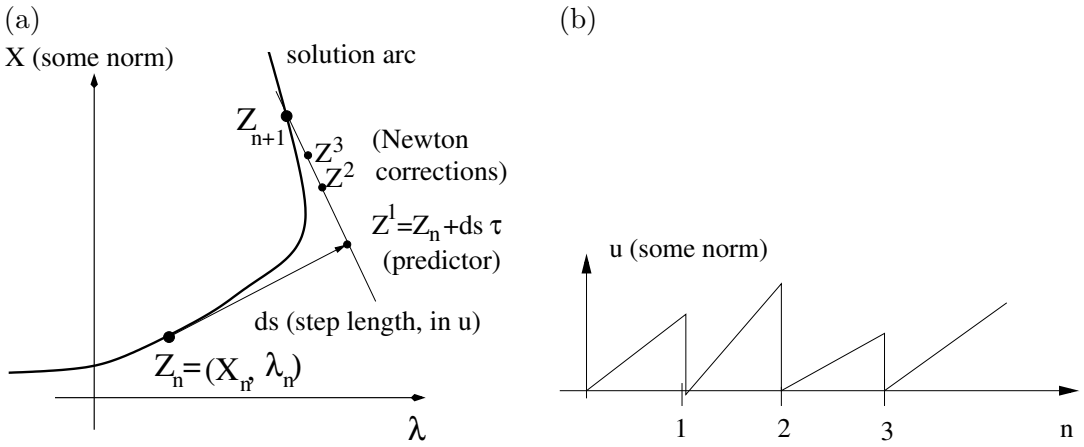

Figure 2: (a) Sketch of arclength continuation algorithm in the `Xcont` setting. (b) Illustration of the "solutions" u_n, which are only meaningful together with X_{n-1} via $X_n = X_{n-1} + u_n N_{n-1}$.

This is illustrated in Fig.2. Recall that the quantity to compute is $X = X_0 + uN$, which gives the system $G(u, \lambda) = 0$ of type (**??**) for u. The main difference between the `Xcont` setting and the legacy setting of `pde2path` is that for `Xcont` the solution u_n in the nth continuation step is only meaningful together with X_{n-1}. In particular, after forming the new predictor Z_n^1 we initialize the Newton–loop for u_{n+1} with $u_{n+1} = 0$, as illustrated in Fig. 2(b). Nevertheless we keep writing $G(u, \lambda) = 0$ for the defining equation, i.e., do not explicitly display the base manifold X which changes from step to step.

We generally compute (approximate), e.g., the mean curvature H from a surface FEM discretization of X, see §2.3. Alternatively, we may assume a parametrization $\phi : \Omega \to \mathbb{R}^3$ of X over some bounded domain $\Omega \subset \mathbb{R}^2$, and compute, e.g., H from a classical 2D FEM mesh in Ω, although this is generally more complicated and less robust than using surface meshes, and we mainly review it for completeness in App. A.3. Both approaches can and usually must be combined with adaptive mesh refinement and coarsening as X changes. Both can also be applied to other geometric PDEs, also of higher order, for instance fourth order biomembrane model, see §4. In this case, the analog of (3a) can be rewritten as a system of (2nd order) PDEs for a vector valued u, and the same ideas apply.

Our work comes with a number of demos which are subdirectories of `pde2path/demos/geomtut`, see Table 1. The rather large number of demos is aimed at showing versatility, and, more importantly,

is due to our own needs for extensive testing, in particular of various BCs, and of different mesh handling strategies. Table 2 summarizes acronyms and notation used throughout, and Fig.3 explains the basic installation steps for `pde2path`. See also [dWDR+23] for a quick overview of installation and usage of `pde2path`, and of all demos coming with `pde2path`, and [RU19] or [Uec21, Chapters 5 and 6] for getting started with `pde2path` via simple classical PDEs.

Table 1: Demo directories in `pde2path/demos/geomtut`. The first two and last three are rather introductory and not dealing with bifurcations.

directory	remarks
`spcap1`	Spherical caps via surface meshes, introductory demo.
`bdcurve`	Experiments on minimal surfaces with different boundary curves.
`enneper`	Bifurcation from Enneper's surface, closely related to `bdcurve`.
`nodDBC`	Nodoids with Dirichlet BCs, including so called liquid bridges.
`nodpBC`	Nodoids with periodic BCs.
`TPS`	Triply Periodic Surfaces, here Schwarz P.
`vesicles`	Closed vesicles as critical points of the Helfrich functional, a 4th order problem.
`biocaps`	Disk type solutions as a variant of `vesicles`.
`biocyl`	Helfrich cylinders with clamped BCs as a variant of `biocaps`.
`parabol`	A paraboloid, to test meshing and mesh adaptation
`spheres`	Continuation of spheres, and tests for VPMCF, §A.1.
`hemispheres`	Continuation of hemispheres on a supporting plane, and VPMCF, §A.2.
`spcap2`	Spherical caps via 2D FEM in the preimage, §A.3.

Table 2: Notations and acronyms; for given X_0, quantities of $X = X_0 + uN_0$ will also be considered as functions of u, e.g., $A(u) = A(X_0 + uN_0)$.

X	surface immersed in $\mathbb{R}^3$	$N = N(X)$	surface unit normal
$A{=}A(X){=}A(u)$	area of X, resp. of $X{=}X_0{+}uN_0$	$V = V(X)$	(algebraic) volume, e.g., (17)
$H = H(X)$	mean curvature, e.g., (15)	$K = K(X)$	Gaussian curvature
$G(u, \lambda) = 0$	generic form of a PDE such as (3a), λ as a generic parameter	$\mathrm{ind}(X)$	index, i.e., number of unstable eigenvalues of linearization
$L = \partial_u H(u)$	Jacobi op. (with BCs)	$q(u, \lambda) = 0$	generic constraint such as (3b)
BC	boundary condition	DBC/NBC	Dirichlet/Neumann BC
pBC	periodic BC	PC	phase condition
BP/FP	branch/fold point	CMC	constant mean curvature
TPS	triply periodic surface	TPMS	triply periodic minimal surface
MCF	mean curvature flow	VPMCF	volume preserving MCF

Remark 1.1 Here we focus on stationary problems of type (1), which give critical points of the volume preserving mean curvature flow (VPMCF). A time t dependent 2D manifold $X(t) \subset \mathbb{R}^3$ deforms by mean curvature flow (MCF) if, assuming the correct sign for H, i.e., $H > 0$ for X bounding a convex body and N the inner normal,

$$\dot{X} = -H(X)N. \tag{8}$$

This is the L^2 gradient flow for the area functional $A(X)$, and can be considered as a quasilinear parabolic PDE, at least on short times. For closed and compact X there always is finite time blowup (generically by shrinking to a "spherical point"), and we refer to [Man11] for an introduction to this huge field, which inter alia heavily relies on maximum (comparison) principles.

- Make a directory, e.g., `myp2p` anywhere in your path. Download `pde2path` from [Uec24] to `myp2p` and unpack, which gives you the `pde2path` home directory `myp2p/pde2path`.
- In MATLAB, change into `pde2path/` and call `setpdepath` to make the libraries available. (We also recommend `ilupack` [Bol11], which is not used here but otherwise in `pde2path` for large scale problems).
- Test, i.e: change directory into `pde2path/demos/geomtut/spcap1`, load the script `cmds1.m` into the editor (i.e., type `edit cmds1.m` at the command line). To get an understanding what command does what, we then recommend to run `cmds1.m` in "cell–mode", i.e., to proceed "cell–by–cell".
- Find a demo that is closest to the problem you want to study; copy this demo directory to a new directory `myproblem/` or any other name (we recommend not as a subdirectory of `pde2path` but *somewhere else*, for instance in a subdirectory `myproblems` of `myp2p`. In `myproblem`, modify the relevant files (usually at least `*init` and `cmds`) and explore.

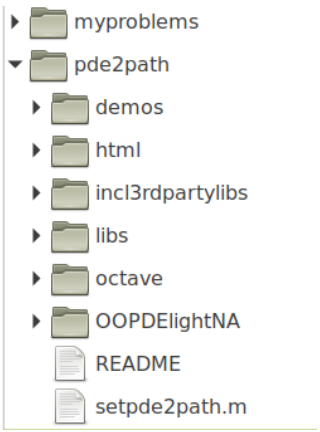

Figure 3: Installation of `pde2path` (version 3.1), and a "typical" directory structure of `myp2p`.

The VPMCF reads

$$\dot{X} = -(H(X) - \overline{H})N, \quad \overline{H} = \frac{1}{A(X)} \int_X H \, \mathrm{d}S, \tag{9}$$

and for closed X conserves the enclosed volume $V(X)$. For non–closed X one typically studies Neumann type BCs on "support planes", see, e.g., [Har13], and in most cases the analysis is done near axisymmetric states such as spheres, spherical caps, and cylinders. In general, the existence and regularity theory for (9) is less well understood than for (8) due to the lack of general maximum principles for (9).

Our notion of stability of solutions of (1) (indicated by thick lines in bifurcation diagrams, while branches of unstable solutions are drawn as thinner lines) refers to (9) if we have an active volume constraint such as (3b), and to (8) if not, with the exception of the fourth order problems in §4, see Remark 4.1. Moreover, by $\mathrm{ind}(X)$ we denote the number of unstable eigenvalues of the linearization of (the discretization of) (3), including the constraints (if active).

We also provide very basic setups to numerically integrate (8) and (9) by explicit Euler stepping. This often has to be combined with mesh adaptation, and in this case A does not necessarily decrease monotonously for MCF. Moreover, our VPMCF typically conserves V only up to 0.5% error. Thus, both are not necessarily efficient or highly accurate, but can be used to generate initial guesses for the continuation of steady states of (1). See §3.1, §3.2.3 (MCF) and §A.1, §A.2 (VPMCF) for examples, and, e.g., [BNP10, BGN20, BGNZ22] for much more sophisticated numerical algorithms for geometric flows including (8) and (9), and detailed discussion. ⌋

The remainder of this tutorial is organized as follows: In §2 we review some differential geometric background, and the `pde2path` data structures and functions to deal with geometric PDEs. In §3 we discuss the main 2nd order demos, and §4 deals with the 4th order biomembrane demos `vesicles` and `biocaps`. In §5 we give a summary, and an outlook on ongoing and future work. In §A we comment on the further demos `spheres` and `hemispheres`, which do not show bifurcations but deal with VPMCF and Neumann type free BCs, and present a classical FEM setup for spherical caps, and in §B we discuss the `biocyl` demo. See also [MU24] for supplementary information (movies) on some of the rather complicated bifurcation diagrams we obtain.

2 Geometric background, and data structures

2.1 Differential geometry

We briefly review the geometric PDE setup, and recommend [Des04, Tap16, UY17] for further background, among many others.

Throughout, let Σ be a 2D connected compact orientable manifold, with coordinates x, y, and possibly with boundary $\partial\Sigma$, and for some $\alpha \in (0,1)$ immersed by $X \in C^{2,\alpha}(\Sigma, \mathbb{R}^3)$. By pulling back the standard metric of $\mathbb{R}^3$ we obtain the first and second fundamental forms on Σ expressed via X as

$$g = \begin{pmatrix} g_{11} & g_{12} \\ g_{12} & g_{22} \end{pmatrix} = \begin{pmatrix} \|X_x\|^2 & \langle X_x, X_y\rangle \\ \langle X_x, X_y\rangle & \|X_y\|^2 \end{pmatrix}, \qquad h = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix} = \begin{pmatrix} \langle X_{xx}, N\rangle & \langle X_{xy}, N\rangle \\ \langle X_{xy}, N\rangle & \langle X_{yy}, N\rangle \end{pmatrix}, \tag{10}$$

with unit normal N, which we consider as a field on Σ, or locally on X, which will be clear from the context. The mean curvature H then is

$$H = \frac{1}{2}\frac{h_{11}g_{22} - 2h_{12}g_{12} + h_{22}g_{11}}{g_{11}g_{22} - g_{12}^2}, \tag{11}$$

which is the mean of the minimal and maximal normal curvatures κ_1 and κ_2. The Gaussian curvature is

$$K = \kappa_1\kappa_2. \tag{12}$$

The sign of H depends on the orientation of X, i.e., on the choice of N. A sphere has positive H iff N is the inner normal. The Gaussian curvature does not depend on N or any isometry of Σ (Gauß' Theorema egregium).

A generalization of the directional derivative of a function f to vector fields or tensors is the covariant derivative ∇_Z for some vector field Z on X. For a vector field Y, the covariant derivative in the j'th coordinate direction is defined as $\nabla_j Y_i := \frac{\partial Y_i}{\partial x_j} + \Gamma^i_{jk} Y_k$, and for a 1-form ω we have $\nabla_j \omega_i := \frac{\partial \omega_i}{\partial x_j} - \Gamma^i_{jk}\omega_k$, with the Christoffel symbols $\Gamma^i_{jk} = \frac{1}{2}g^{il}(\partial_{x_j} g_{kl} + \partial_{x_k} g_{jl} - \partial_{x_l} g_{jk})$, where g^{ij} are the entries of g^{-1}, and where we use Einstein's summation convention, i.e., summation over repeated indices. The covariant derivative is linear in the first argument, giving a general definition of $\nabla_Z Y$ with some vector field Z, and if f is a function on X, then

$$\nabla_Z f = \langle g\nabla f, Z\rangle_{\mathbb{R}^2}. \tag{13}$$

Throughout we are dealing with surfaces (2 dimensional manifold immersed into $\mathbb{R}^3$), hence the gradient ∇ is the *surface gradient*, i.e., the usual gradient $\nabla_{\mathbb{R}^d}$ in $\mathbb{R}^3$ projected onto the tangent space,

$$\nabla f = \nabla_{\mathbb{R}^3} f - \langle \nabla_{\mathbb{R}^3} f, N\rangle N, \tag{14}$$

which later will be needed to (formulate and) implement phase conditions, and, e.g., Neumann type BCs. This also gives the *Laplace Beltrami operator* via

$$\Delta f = g^{ij}\nabla_i \nabla_j f,$$

which then also applies to general tensors. Using the Gauß–Weingarten relation $\frac{\partial^2 X}{\partial x_i \partial x_j} = \Gamma^k_{ij}\frac{\partial X}{\partial x_k} + h_{ij}N$ we obtain

$$\Delta X = g^{ij}\nabla_i\nabla_j X = g^{ij}\left(\frac{\partial^2 X}{\partial x_i \partial x_j} - \Gamma^i_{jk}\frac{\partial X}{\partial x_k}\right) = g^{ij}h_{ij}N = 2H(X)N = 2\vec{H}(X),$$

where $\vec{H}(X)$ is called the mean curvature vector, and

$$H(X) = \frac{1}{2} \langle \Delta X, N \rangle . \tag{15}$$

The area of X is

$$A(X) = \int_X \mathrm{d}S, \tag{16}$$

and, based on Gauß' divergence theorem, the (algebraic) volume is

$$V(X) = \frac{1}{3} \int_X \langle X, N \rangle \, \mathrm{d}S. \tag{17}$$

If X is a closed manifold bounding $\Omega \subset \mathbb{R}^3$, i.e., $\partial\Omega = X$, and N the outer normal, then $V(X) = |\Omega|$ is the physical volume. If X is not closed, then we typically need to add a third of the flux of $\vec{x}$ through the open ends to $V(X)$ (see the examples below).

We denote the set of all immersed surfaces with the same boundary γ by

$$\mathcal{N}_\gamma = \{X : X \text{ is an immersed surface as above and } \partial X = \gamma\}. \tag{18}$$

The following Lemma states that all immersions $Y \in \mathcal{N}_\gamma$ close to X are graphs over X determined by a function u as $Y = X + uN$, which justifies our numerical approach (4). The condition that Y has the same boundary as X in general cannot be dropped, as obviously motions of ∂X tangential to X cannot be captured in the form $X + uN$; see §3.2.1 for an illustration.

Lemma 2.1 *[KPP17]. For $X \in C^{2,\alpha}(\Sigma, \mathbb{R}^3)$ with boundary $\partial X = \gamma$ there exists a neighborhood $U \subset C^{2,\alpha}(\Sigma, \mathbb{R}^3)$ of X such that for all $Y \in U \cap \mathcal{N}_\gamma$ there exists a diffeomorphism $\phi : \Sigma \to \Sigma$ and a $u \in C^{2,\alpha}(\Sigma)$ such that*

$$Y \circ \phi = X + u\, N. \tag{19}$$

Assume that a CMC surface X_0 with boundary $\partial X_0{=}\gamma$ and volume $V(X_0){=}V_0$ belongs to a family of CMC surfaces X_t, $t \in (-\varepsilon, \varepsilon)$ for some $\varepsilon > 0$. For example, the spherical caps S_t from Fig. 1(a) with the boundary $\gamma = \{(x, y, 0) \in \mathbb{R}^3 : x^2 + y^2 = 1\}$ are a family of CMC immersions fully described by the height $t \in \mathbb{R}$. By Lemma 2.1, the parameter t uniquely defines u in a small neighborhood, i.e., $X_t = X + u\, N$, and the system of equations for u reads

$$H(u) - H_0 = 0, \tag{20}$$

for some $H_0 \in \mathbb{R}$, where we abbreviate $H(u) = H(X + u\, N)$, etc. If we consider variational vector fields at X_0 in the form $\psi = \dfrac{\partial X_t}{\partial t}|_{t=0} = uN$, and additionally assume that $X_t \in \mathcal{N}_\gamma$, then necessarily

$$u|_{\partial X} = 0, \text{ Dirichlet boundary conditions (DBCs).} \tag{21}$$

Such X_t are called *admissible variations* in [Lóp13, §2.1], and we have the following results on derivatives of A and V.

Lemma 2.2 *[Lóp13, §2.1] For an admissible one parameter variation X_t of $X \in C^{2,\alpha}(\Sigma)$ and variational vector fields $\psi = \frac{\partial X}{\partial t}\big|_{t=0} = uN$ the functions $t \mapsto A(t) = A(X_t)$ and $t \mapsto V(t) = V(X_t)$ are*

smooth, and

$$V'(0)=\int_{X_0} u\,\mathrm{d}S,\quad A'(0)=-2\int_{X_0} H_0 u\,\mathrm{d}S,\ \textit{and}\ A''(0)=-\int_{X_0}(\Delta u+\|S_0\|^2 u)u\,\mathrm{d}S, \tag{22}$$

where $\|S_0\|^2=4H_0^2-2K_0$ *with the Gaussian curvature* K_0*. Thus*

$$\frac{\mathrm{d}}{\mathrm{d}t}H(X_t)\Big|_{t=0}=-\Delta u-\|S_0\|^2 u, \tag{23}$$

and the directional derivative (23) *is given by the self-adjoint Fredholm operator* L *on* $L^2(X_0)$ *with*

$$L=\partial_u H(0)=-\Delta-\|S_0\|^2,\quad \textit{with DBCs.} \tag{24}$$

Remark 2.3 a) The operator in (24) without BCs is called *Jacobi operator*, and a nontrivial kernel function is called a *Jacobi field* on $X=X_0$. An immersion X with a Jacobi field satisfying the BCs is called *degenerate*. The Fredholm property allows the use of the Crandall-Rabinowitz bifurcation result [CR71]: Given a C^1 branch $(t_0-\varepsilon,t_0+\varepsilon)\ni t\mapsto X_t$, if X_t is non–degenerate for $t\in(t_0-\varepsilon,t_0)\cup(t_0,t_0+\varepsilon)$, and if at t_0 a *simple* eigenvalue $t\mapsto\mu_0(t)$ crosses transversally, i.e., $\mu(t_0)=0$, $\mu_0'(t_0)\neq 0$, then a branch $\widetilde{X}_t$ bifurcates at t_0.

See also [KPP17] for a formulation via Morse indices $\mathrm{ind}(X_t)$=number of negative eigenvalues of L, counted with multiplicity, used to find bifurcation points in families of nodoids, which we shall numerically corroborate in §3.3.2. An equivariant version can be found in [KPS18, Theorem 5.4], applied to bifurcations of triply periodic minimal surfaces, for which linearizations always have a trivial 5 dimensional kernel due to translations and rotations, see §3.5 for numerical illustration. See also [GS02, Hoy06, Kie12] and [Uec21, Chapters 2 and 3] for general discussion of Crandall–Rabinowitz type results, and of Krasnoselski type results (odd multiplicity of critical eigenvalues, based on degree theory), including equivariant versions.

b) Besides the (zero) DBCs (21) corresponding to a fixed boundary $\partial X=\gamma$, we shall consider so called free boundaries of Neumann type. This means that $\partial X\subset\Gamma$, where $\Gamma\subset\mathbb{R}^3$ is a fixed 2D support manifold (e.g., a plane), and that X intersects Γ orthogonally. Following [Ros08], we summarize the second derivative of A in this case as follows: if h_Γ is the second fundamental form of Γ, and $\psi=uN$, then $N|_{\partial X}$ is tangent to Γ, such that $h_\Gamma(N,N)$ is well defined, and

$$A''(0)=-\int_{X_0}(\Delta u+\|S_0\|^2 u)u\,\mathrm{d}S-\int_{\partial X_0} h_\Gamma(N,N)u^2\,\mathrm{d}s. \tag{25}$$

Note that the term $\int_{\partial X_0}\ldots$ in (25) vanishes if Γ is a plane and hence $h_\Gamma\equiv 0$.

c) The formulas (22)–(24) translate to our discrete computational setting in a straightforward way, see §2.3.1. However, given some X_0, we compute $X=X_0+uN_0$ via Newton loops for iterates u_n with derivatives (of V, A and H) evaluated at $X_n=X_0+u_nN_0$, and hence the formulas are accordingly adjusted. ⌋

2.2 Default data and initialization of a `pde2path` struct `p`

Before explaining the modifications needed for the geometric problems, we briefly review the standard setup of `pde2path`, and as usual assume that all problem data is contained in the MATLAB struct `p` as in problem. In the standard FEM setting this includes the object `p.pdeo` (with sub–objects `fem` and `grid`), which provides methods to generate FEM meshes, to code BCs, and to assemble FEM matrices `M` (mass matrix) and `K` (e.g., Laplacian), or directly a rhs G. Typical initializations and first continuation steps in the FEM setup (for semilinear problems) then run as follows, where steps 1,2 and 5 are usually combined into an init-function.

1. Call `p=stanparam()` to initialize most fields in `p` with default values (see source of `stanparam.m` for default fields and values).
2. Call a `pdeo` constructor, for instance `p.pdeo=stanpdeo2D(p,vararg)`, where here and in the following `vararg` stands for variable arguments.
3. In a function `oosetfemops` (in the current directory), use `p.pdeo.assema` to generate a mass matrix `p.mat.M` and a stiffness matrix `p.mat.K` (typically corresponding to $-\Delta$), and possibly further FEM matrices, e.g., for BCs.
4. Use `p.mat.M` and `p.mat.K` in a function `r=sG(p,u)` to encode the PDE, and optionally the Jacobian in `Gu=sGjac(p,u)` (usually recommended, but numerical Jacobians are also supported). The input argument `u` contains the "PDE unknowns" u and the parameters appended at the end. If required by the problem, similarly create a function `q=qf(p,u)` for the constraints as in (3b), and a function `qu=qder(p,u)` for the derivatives of `qf`.
5. Initialize `p.u` with a first solution (or a solution guess, to be corrected in a Newton loop).
6. Call `p=cont(p)` to (attempt to) continue the initial solution in some parameter, including bifurcation detection, localization, and saving to disk.
7. Call `p=swibra(dir,bpt,newdir)` to attempt branch switching at branch point `bpt` in directory `dir`; subsequently, call `p=cont(p)` again, with saving in `newdir`.
8. Perform further tasks such as fold or branch–point continuation; use `plotbra(dir,pt,vararg)` to plot bifurcation diagrams, and `plotsol(dir,pt,vararg)` to plot sample solutions.

Remark 2.4 The rhs, Jacobian, and some further functions needed/used to run `pde2path` on a problem `p`, are interfaced via function handles in `p.fuha`. For instance, you can give the function encoding your rhs G any name, e.g., `myrhs`, with signature `res=myrhs(p,u)`, and then set `p.fuha.sG=@myrhs`, but you can also simply keep the "standard names" `sG` and `sGjac` and encode these in the respective problem directory. For many handles in `p.fuha` there are standard choices which we seldomly modify, e.g., `p.fuha.headfu=@stanheadfu` (the header for printouts). Functions for which the "default choice" is more likely to be modified include, e.g.,

- `p.fuha.outfu=@stanbra`, signature `out=stanbra(p,u)`, branch output;
- `p.fuha.ufu=@stanufu`, signature `[p,cstop]=refufu(p,brout,ds)`, "UserFUnction"; this is called after each successful continuation step, and in its default setting just gives printout and checks if the (primary) continuation parameter is still in the prescribed range. In some of the demos we modify `stanufu` to a function `refufu` which, e.g., checks the mesh-quality and adapts the mesh if necessary.
- `p.fuha.lss=@lss`, signature `[x,p]=lss(A,b,p)`, linear systems solver $x = A^{-1}b$.
  Other options include, e.g., `lssbel` (bordered elimination) and `lssAMG` (preconditioned GMRES using `ilupack` [Bol11]).

During continuation, the current solution is plotted via `plotsol(p)`, and similarly for a posteriori plotting (from disk). The behavior of `plotsol` is controlled by the subfields of `p.plot` (and possible auxiliary arguments), and if `p.plot.pstyle=-1`, then `plotsol` immediately calls a function `userplot`, to be user–provided. Such user functions naturally must be in the MATLAB–path, typically in the current problem directory, which MATLAB scans first when looking for a file. We sometimes also exploit this to overload `pde2path` library functions that need modifications for a given problem. ⌋

2.3 `pde2path` setup for discrete differential geometry

2.3.1 Discrete differential geometry FEM operators

We recall a few discrete differential geometry operators from [MDSB03, Jac13], and shall use implementations of them from the `gptoolbox` [Jac24]. Given a triangulation $\texttt{X} \in \mathbb{R}^{n_p\times 3}$ (point coordinates) and $\texttt{tri} \in \mathbb{R}^{n_t\times 3}$ (triangle corner indices) of X, and the piecewise linear element "hat" functions

$\phi_i : X \to \mathbb{R}$, $\phi_i(X_j) = \delta_{ij}$, we have

$$\int \nabla\phi_i \nabla\phi_j \,\mathrm{d}S = -\frac{1}{2}(\cot\alpha_{ij} + \cot\beta_{ij}) =: L_{ij}, \tag{26}$$

where α_{ij} and β_{ij} are the angles opposite the edge e_{ij} from point X_i to point X_j. For $u : X \to \mathbb{R}$, $u = \sum_{i=1}^{n_p} u_i\phi_i$, this yields the FEM stiffness matrix Lu corresponding to the Laplace–Beltrami operator $-\Delta u$ weighted by the mass matrix M. In [MDSB03] it is explained that for geometric problems, with possibly rather distorted triangles, instead of the full mass matrix M^{full} with

$$M^{\text{full}}{}_{ij} = \int \phi_i\phi_j \,\mathrm{d}S, \tag{27}$$

the Voronoi mass matrix

$$M = \operatorname{diag}(A_1, \ldots, A_{n_p}), \tag{28}$$

should be expected to give better approximations, see also Fig. 4. Here, $A_i = \sum_{j=1}^{n_i} A_m(T_j)$ is the area of the Voronoi region at node i, where T_j, $j = 1, \ldots, n_i$ are the adjacent triangles, and $A_m(T)$ is a "mixed" area: For non–obtuse T, $A_m(T)$ is the area of the rhomb with corners in X_i, in the midpoints of the edges adjacent to X_i, and in the circumcenter of T, while for obtuse T we let $A_m(T) := |T|/2$ if the angle at X_i is obtuse, and $A_m(T) := |T|/4$ else. Altogether, this yields the approximation

$$-\Delta u = M^{-1}Lu, \tag{29}$$

where M from (28) is diagonal, and L and M are evaluated very efficiently via `cotmatrix` and `massmatrix` from the `gptoolbox`, see Table 3.

However, as we always consider our problems such as (3) in weak form, we let $\mathtt{H} = -\frac{1}{2}\langle LX, N\rangle$, where for the vertex normals N we can use `per_vertex_normals`, and the weak form of, e.g., $H - H_0 = 0$ then is

$$-\langle LX, N\rangle - 2MH_0 = 0, \tag{30}$$

again with Voronoi M. Alternatively, we use `[k,H,K,M]=discrete_curvatures(X,tri)`, where `K` and $\mathtt{k} = (k_1, k_2)$ are the (weighted, i.e., weak) discrete Gaussian and principal curvatures per vertex; these are computed from a discrete version of the Gauß–Bonnet theorem.[1] Namely

$$\mathtt{K}(X_i) = 2\pi - \sum_{j=1}^{n_i} \theta_j, \quad (\text{and } k_1 = H + \sqrt{D} \text{ and } k_2 = H - \sqrt{D}), \tag{31}$$

where the θ_j are the angles at X_i, and where the discriminant $D = H^2 - K$ (which is non-negative in the continuous case) in the discrete case is set to 0 if negative. An approximations of K is then obtained (cheaply, since M is diagonal) from

$$K = M^{-1}\mathtt{K}. \tag{32}$$

For notions of convergence of discrete differential geometry objects and operators we refer to [War08], which considers approximations of a smooth manifold X by shape regular triangulations X_h

[1] On a manifold X with boundary ∂X we have $\int_X K \,\mathrm{d}S + \int_{\partial X} \kappa_g \,\mathrm{d}s = 2\pi\chi(X)$ where $\chi(X)$ is the Euler characteristic of X, and κ_g is the geodesic curvature of ∂X. This will play an important role for the biomembranes in §4. The discrete formula (31) is used at interior points of X, while at boundary points X_i it is modified to $K(X_i) - \pi$.

in the sense of Hausdorff distance $\mathrm{dist}(X_h, \mathcal{X}) \to 0$ as $h \to 0$. Here and in the following $h = \max_{T \in \mathtt{tri}} h(T)$, where $h(T)$ means the maximal edge length of triangle T, and shape regular means that the mesh distortions

$$\delta_{\mathrm{mesh}} := \max_{T \in \mathtt{tri}} (h(T)/r(T)) \quad \text{(max edge-length over in–radius)}, \tag{33}$$

are bounded.[2] Then, the following convergences are equivalent:

$$\begin{array}{ll} \text{(a)} & \text{normals } \|N - N_h\|_\infty \to 0, \\ \text{(b)} & \text{volumes (area) } \|\,\mathrm{d}S - \mathrm{d}S_h\|_\infty \to 0, \\ \text{(c)} & \text{Laplace-Beltrami operators } \|\Delta - \Delta_h\|_{\mathrm{op}} \to 0, \end{array} \tag{34}$$

where $\|\cdot\|_{\mathrm{op}}$ is the norm in $L(H_0^1(X), H^{-1}(X))$ and $\mathrm{d}S_h$ and Δ_h in (34) are to be understood in a metric pullback sense. A fourth notion of convergence equivalent to those in (34) and in fact used for the proof of the equivalence in (34) is metric convergence (suitably defined). In particular, in general the discrete mean curvature H_h obtained from $H_h(X_h) = \frac{1}{2}\langle \Delta_h X_h, N_h\rangle$ only converges as a functional to $H(X)$, not as function. A famous counterexample that (hence) none of the convergences from (34) follows from $\mathrm{dist}(X_h, X) \to 0$ alone is the lantern of Schwarz.[3] Nevertheless, while numerical experiments in [Xu04] show that a variety of natural schemes for Δ in general do not converge, [Xu04, Theorem 2.1] states that with Voronoi M and at valence six nodes (six neighbors)

$$M^{-1}L = \Delta + \mathcal{O}(h^2); \tag{35}$$

see also [XX09] for function space convergence as $h \to 0$ of various schemes for H_h and K_h.

In Fig.4 (and Fig. 5) we give an illustration of the (function space) error and convergence behavior of our discrete $H = \frac{1}{2}M^{-1}\langle LX, N\rangle$ based on (29), and of K from (32) on discretizations of the unit sphere obtained from subdivision and projection, with 2 (a) resp. 3 (b) subdivisions. See `geomtut/hemispconv/` for the MATLAB source. Here N=outer normal, hence $H = -1$ and $K = 1$ are the exact values, and the two left columns indicate the convergence for H, but also that the node valence plays a role on these otherwise very regular meshes.[4] However, the last column shows that using M^{full} in this example, i.e., $H_{\mathrm{full}} = \frac{1}{2}M^{\mathrm{full}^{-1}}\langle LX, N\rangle$ gives a significant error (and similarly in K), and in fact no convergence at the valence 5 nodes.

Figure 5(a) shows the pointwise convergence of H and K (away from the boundary) for a hemisphere discretized by subdivision and projection with j=2, ..., 5 steps; here, "$\|H+1\|_\infty, \alpha = -1.03$" in the legend means $\|H+1\|_\infty \sim Cn^\alpha$ with α the best linear fit for the log–log plot. Since $n \sim h^{-2}$, (a) shows that here (35) also holds at the valence 5 nodes, and that also for K we get the same convergence. In (b) we show the data for the L^2 norm, in which H_{full} and K_{full} also converge, but with much slower rates than H and K.

Finally, (c) shows a different experiment: after initialization with the hemispheres from (a), we solve the discretized problem $H + 1 = 0$ with DBCs $\partial X = S^1$, i.e., $u = 0$ for the nodes associated to ∂X, and plot the deviation of X from the spherical shape. The convergence rate for $\||X| - 1\|$ is approximately $n^{-5/3}$, both for X and X_{full} (obtained with M^{full}), and in both $\|\cdot\|_\infty$ and in $\|\cdot\|_2$. Thus, while H_{full} is in general not accurate, here the solution X_{full} is. In any case, in the following

[2] δ_{mesh} will also be one of our criteria for mesh adaptation in the numerics; for an equilateral triangle (best case) $\delta_{\mathrm{mesh}} = 2\sqrt{3} \approx 3.46$, and for a "standard" triangle (right angled isoceles), $\delta_{\mathrm{mesh}} \approx 4.83$. As a rule of thump, meshes with $\delta_{\mathrm{mesh}} \le 10$ have little distortion, and $\delta_{\mathrm{mesh}} \le 50$ is still OK. See also [She02] for a very useful discussion of mesh quality (in the planar setting, and in 3D).

[3] In a nutshell, this is a 2D version of $u_n(x) := \frac{1}{n}\sin(nx) \to 0$ in L^∞ without convergence of $u_n'(x) = \cos(nx)$ in any proper function space, but $u_n'(x) \to 0$ in the sense of distributions.

[4] Euler's polyhedron formula yields that triangulations with all nodes of valence 6 do not exist, see, e.g., [BF67].

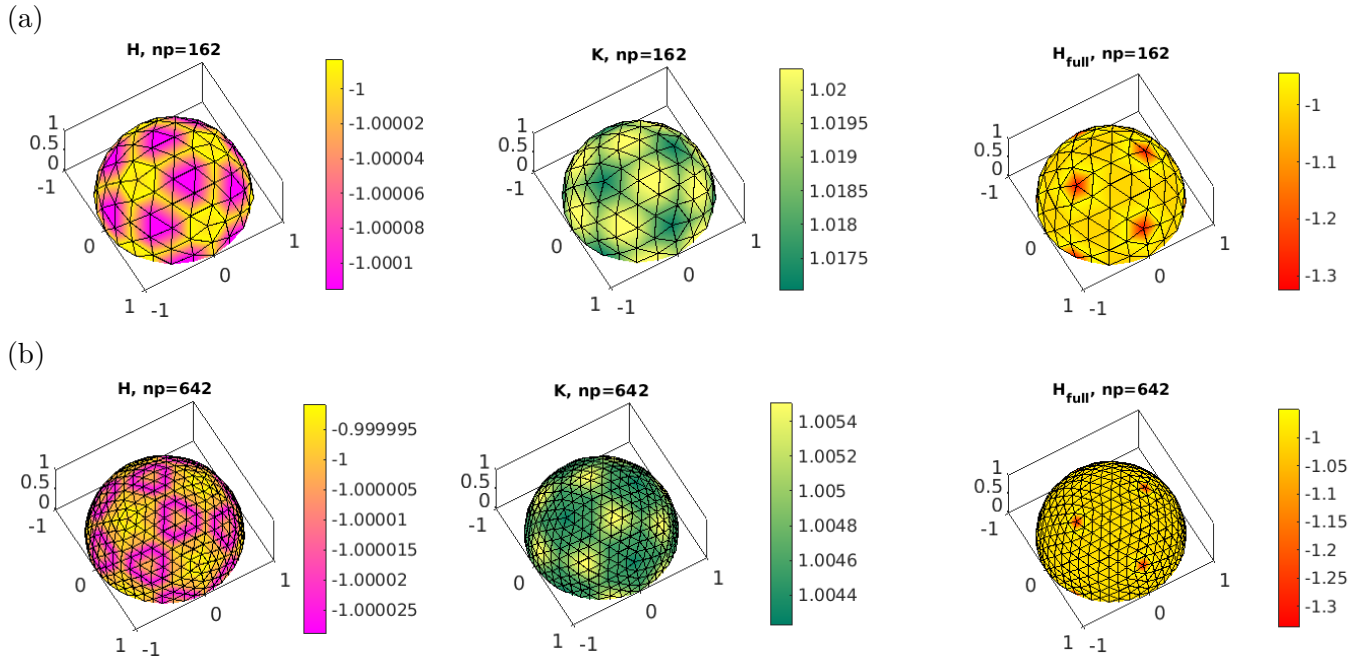

Figure 4: Discrete H (and K) on (coarse) meshes of the unit sphere (plots cropped). Two left columns: Convergence for $H = -\frac{1}{2}M^{-1}\langle LX, N\rangle$ and $K = M^{-1}\mathtt{K}$ with Voronoi M. Right column: No convergence for H (and similar for K) at valence 5 nodes when using M^{full}.

we always use the Voronoi mass matrix M and generally recommend this. See also Example 2.7 for another convergence experiment, where we also explain different options for mesh refinement, and where we display some lingering effects of the valence of nodes on solution accuracy.

2.3.2 The `pde2path` library `Xcont`

Table 3 lists the main (for us) functions from the `gptoolbox` [Jac24], which we interface by functions from the `pde2path` library `Xcont`. The most important new data for continuation of a surface X are `p.X` and `p.tri`, which essentially replace the data in `p.pdeo.grid`. The most important switch, which also modifies the behavior of some standard `pde2path` functions, is

$$\texttt{p.sw.Xcont} = \begin{cases} 0 & \text{legacy setting (no X)}, \\ 1 & \text{switch on X–continuation (default)}, \\ 2 & \text{refined setting for X–continuation.} \end{cases} \tag{36}$$

The difference between `p.sw.Xcont`=1 and `p.sw.Xcont`=2 is as follows: For `p.sw.Xcont`=1 we update `p.X` after convergence of the Newton loop, i.e., set `p.X=p.X+u*N0, p.up=u` (for plotting, see Remark 2.5(b)), and `u(1:p.np)=0` (zeroing out u for the next continuation step). For `p.sw.Xcont`=2 we do all this after each successful Newton-step, such that we obtain slightly different (more accurate) Jacobians as we also have a new N. A side-effect is that the last update in the Newton loop and hence the `p.up` may be very small and will in general not represent the "direction" of continuation. For `p.sw.Xcont`=1, `p.up` will contain the complete step, and therefore we choose this as default. For the spectral computations (bifurcation detection and localization) `p.sw.Xcont`=1 vs `p.sw.Xcont`=2 makes no difference as all spectral computations are done after convergence of the Newton loops, i.e., after the final update of `p.X`.

Some main functions from the `pde2path` library `Xcont` are listed in Table 4, and can be grouped as follows:

1. Functions directly interfacing the `gptoolbox` such as `N=getN(p)` which in the default setting just calls `N=per_vertex_normals(p.X,p.tri)`. These are meant as easy–to–change interfaces, as for a given problem it may be desired to, e.g., change the orientation of X. For this, make a

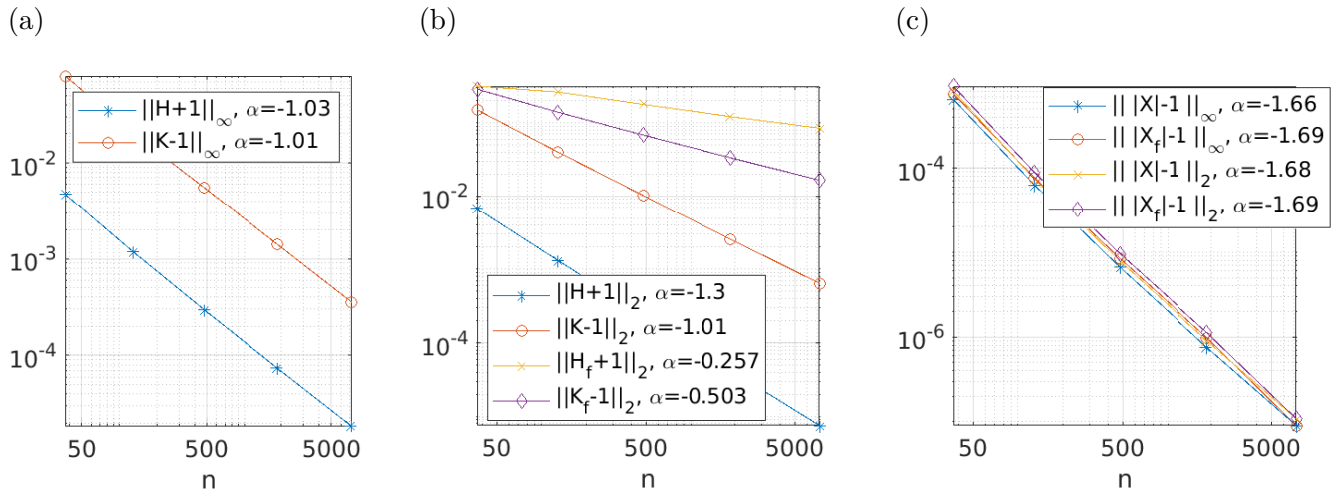

Figure 5: (a,b) Convergence rates for H and K on discretized hemipsheres. (c) Convergence rates for the spherical deviation $|X| - 1$ of solutions of $H + 1 = 0$.

Table 3: Main functions used from [Jac24] and [Sch22, FS21], and some small additions/mods. Here $\texttt{X} \in \mathbb{R}^{n_p \times 3}$ are the point coordinates of the triangulation, and $\texttt{tri} \in \mathbb{R}^{3 \times n_t}$ the triangulation as rows of point numbers of triangles.

function	remarks
`N=per_vertex_normals(X,tri)`	normals; should be interfaced as `N=getN(p,X)`, see Table 4 to possibly flip sign at one place;
`L=cotmatrix(X,tri)`	cotangent discrete Laplace–Beltrami
`M=massmatrix(X,tri,type)`	mass matrix, with type 'full', 'barycentric' or 'voronoi'.
`K=discrete_gaussian_curvature(X,tri)`	Gaussian curvature K.
`[k,H,K,M]=discrete_curvatures(X,tri)`	principal curvatures $k = (k_1, k_2)$, and H, K, M.
`[X,tri,DBC,NBC]` `=TrefineRGB(X,tri,DBC,NBC,elist)`	refinement, with DBC/NBC the Dirichlet/Neumann boundary nodes, and elist the triangles to refine. Interfaced via `p=refineX(p,sig);` where `sig` is the usual factor of triangles to refine, and where after refinement `p.u` and `p.tau` are interpolated to the new mesh. See also `TcoarsenRGB`, with similar signature.

local copy of `getN.m` and there set `N=-per_vertex_normals(p.X,p.tri)`.

2. "Template" functions to compute $V(u)$ (corresponding to $V(X) = V(X_0 + uN_0)$ in (17)), and $A(u)$ from (16), and wrappers to use them in constraints such as $\texttt{q} = \texttt{qfV(p,u)} = V(u) - V_0$, and their derivative, such as $\texttt{qu} = \texttt{qjacV(p,u)} = \partial_u V$. (Since V is a scalar function, here we use "`jac`" in a loose sense.)
3. Template convenience functions such as `pplot` for plotting `p.X`, and `updX` for updating $X = X_0 + uN_0$, cf. (4), and overloads of `pde2path` library function adapted to X continuation, e.g., `getGupde`, and a prototype `oosetfemops`, which for X–continuation usually should only serve as a dummy needed by other `pde2path` functions.
4. Functions related to mesh handling such as `refineX`, `coarsenX` and `degcoarsenX`, and associated elements–to–refine–selectors such as `e2rsA` (based on triangle areas) and `e2rshape` (based on triangle shapes). Additionally there are `retrigX` and `moveX`, based on [PS04], see Remark 2.6.
5. The template `geomflow` for geometric flows such as MCF, interfaced in `p.fuha.flowf`.

All of these functions, in particular from 2. and 3. are "templates"; for a given problem it may be necessary or useful to make local copies of some of these functions in the problem directory and change them there. Additionally, there is convenience function `p=stanparamX(p)`, which (re)sets some `pde2path` parameters to typical values for X–continuation. Another important interface to the `gptoolbox` is

Table 4: Main functions from the library `Xcont`; see sources for argument lists and detailed comments, and Remark 2.5 for further comments.

function	remarks
`p=stanparamX(p)`	convenience function to set `pde2path` parameters to "standard" values for X–continuation; to be called *after* `p=stanparam()` during initialization.
`getN`	interface to `per_vertex_normals`; to flip orientation, make local copy with `function N=getN(p,X); N=-per_vertex_normals(X,p.tri); end`
`getA, getV`	get area A and volume V according to (16) and (17).
`qV, qjacV`	V as a constraint q for continuation, and its derivative $\partial_u q$.
`qA, qjacA`	A as a constraint q for continuation, and its derivative $\partial_u q$.
`c2P`	triangle center to point (nodes) interpolation matrix.
`cmcbra`	default branch output in our demos below.
`pplot`	plot of `p.X`, usually called from `userplot`; see Remark 2.5b).
`oosetfemops`	usually only needed as a dummy (since called by `pde2path` library functions).
`updX`	update `p.X` after successful Newton steps.
`plotHK`	given `p.X`, plot $H(X)$ and $K(X)$ (mostly for checking).
`e2rsA, e2rsAi`	element–to–refine–selector choosing triangles with large areas; `e2rsAi` chooses triangles with small areas, intended for coarsening.
`e2rsshape1`	Select triangles according to (`??`). See also `e2rsshape*`.
`meshqdat`	mesh quality data, see (38).
`refineX`	mesh refinement of `p.X`; selecting triangles via `p.fuha.e2rs` and calling `TrefineRGB` (or `Trefinelong` if `p.sw.rlong==1` for bisection of only longest edges of triangles).
`coarsenX`	coarsening of `p.X`, same syntax as `refineX`; i.e., selecting triangles via `p.fuha.e2rs`. Inverse to `refineX`, as only triangles from prior refinement can be coarsened (to their common ancestors).
`degcoarsenX`	coarsening of mesh by removing degenerate triangles (via `gptoolbox`).
`retrigX`	retriangulation of X, based on [PS04]; using the adjacency in `p.trig` to generate a new (Delauney) triangulation of X while keeping the surface structure.
`moveX`	retriangulate and move points of X based on [PS04] and the associated code.
`geomflow`	simple explicit time integrator for $\dot{X}=fN$, where f=`p.fuha.flowf`, e.g., $f(X)=-H$ for mean curvature flow, implemented in `mcff` (with DBCs).

6. `getM(p)`, which for scalar problems should call `M=massmatrix(p.X,p.tri,'Voronoi')`, and for vector valued problems should build the system mass matrix from such "scalar" `M`, cf. §4. However, for compatibility with the non–X–setting, `getM` is *not* part of `Xcont`, but needs a local copy in each (X–continuation) problem directory.

Remark 2.5 a) As is, `out=cmcbra(p,u)` puts the data

$$\text{out=[pars;V;A;E;meshq]} \tag{37}$$

into `out`, where `pars` of length $m = \texttt{length(u)} - \texttt{p.nu}$ is the (user defined) parameter vector of the problem, V and A are volume and area of X, and $E = A + \texttt{par(1)}V$, cf. (2), which assumes that H_0 is at `par(1)`, and is an *active* continuation parameter. Next, `meshq` $= (\delta_{\text{mesh}}, a_{\max}, a_{\min}, h_{\max}, h_{\min})$ computed in `meshq=meshqdat(p)`, with δ_{mesh} the mesh distortion (33), and $a_{\max}$ and $a_{\min}$ are the max and min of the triangle areas, and $h_{\max,\min}$ are the largest and smallest edge lengths in the triangulation.

`out=p.fuha.outfu` is appended to `bradat(p,u)=[count;type;ineg;lam;err;L2]`[5], and we list

[5]six values, where `count`=step-counter, `type` $\in \{0,1,2,3\}$ (regular point, BP, FP, Hopf point), `ineg`=number of unstable eigenvalues (if `p.sw.spcalc` > 0), `lam`=value of primary continuation parameter, and `err` and `L2` are *not* meaningful in the `Xcont` setting, see `bradat.m` for details)

the component `c` for branch plotting for `p.fuha.outfu=@cmcbra` (with m the number of parameters):

c	1...m	m+1	m+2	m+3	m+4	m+5	m+6	m+7	m+8
meaning	`pars`	V	A	E	δ_{mesh}	$a_{\max}$	$a_{\min}$	$h_{\max}$	$h_{\min}$

(38)

Thus, to plot, e.g., V over the *active* bifurcation parameter from a computed branch `b1` into figure `fnr`, use `plotbra(b1,fnr,m+1,varargin)`, where `varargin` gives `many` options for colors, labels, etc. Similarly, to plot δ_{mesh}, use `plotbra(b1,fnr,m+4,varargin)`. Moreover, `c` in `plotbra(b1,fnr,c,varargin)` can be a vector, and to, e.g., plot δ_{mesh} over V use `c=[m+1, m+4]`.

b) `pplot(p,fignr)` (or `pplot(dir,pt,fignr)`, where `p` is loaded from `dir/pt`) colors `p.X` by `p.up`, which contains u from the last continuation step and hence codes (together with N) the "continuation direction" (if `p.sw.Xcont=1`), or just the last Newton update (if `p.sw.Xcont=2`), see (36). The behavior of `pplot` can further be controlled by settings in the *global* structure `p2pglob`.[6] For instance, `p2pglob.tsw` controls the titles, e.g., (V, H) for `tsw=1`, and (A, H) for `tsw=2`, which assumes that the parameters are ordered as (H, V, A). For flexibility, if `tsw` > 4, then `pplot` searches the current directory for a function `mytitle` to generate a title, see, e.g., demo `geomtut/enneper`. Again, `pplot` is a template, if necessary to be modified in the problem directory for customized plots, but all demos from `geomtut/` only use the given library version.

c) In all demos below we use indices `p.idx` and `p.idN` of boundary points to set boundary conditions. In this, `p.idx` should be thought as points for Dirichlet BCs and associated to `p.DBC` (the corresponding *edges*, updated in `refineX` and `coarsenX`), and `p.idN` as Neumann BCs with edges `p.NBC`. All of these can also be empty (for instance if X is closed, or has only one type of boundary), and again, the use of `p.idx, p.idN, p.DBC` and `p.NBC` should only be seen as a template, for instance to be modified if a given problem has several different boundaries. ⌋

Remark 2.6 a) For surface meshes (`X,tri`), mesh adaptation, i.e., refinement and coarsening, seems even more important than for standard (non–parametric) problems, because well behaved initial triangulations (well shaped triangles of roughly equal size) may deteriorate as X changes. The case of growing spherical caps in Fig. 1(a) is rather harmless as triangle sizes grow but shapes stay intact, and can easily be dealt with by refinement of the largest triangles. For this, in `e2rsA` we simply order the n_t triangles of `tri` by decreasing size, and from these choose the first $\lfloor \sigma n_t \rfloor$ for refinement by `refineX`, i.e., we generally use $\sigma =$ `p.nc.sig` as the parameter for the fraction of triangles to refine. The refinement can be either done as RGB if `p.sw.rlong=0`, or by refining only the longest edges of the selected triangles if `p.sw.rlong=1`. RGB is generally better if triangle shapes are crucial, but may result in rather long cascades to avoid hanging nodes (such that σ is only a lower bound for the fraction of triangles actually refined). Refine-long gives more control as *only* the selected triangles are bisected (plus at most one more triangle for each one selected), but may lead to obtuse triangles, and it seems that as for standard FEM very obtuse triangles are more dangerous than very acute triangles. A short computation shows that, e.g., for a right–angled triangle refine–long increases $\delta = h/r_{\mathrm{in}}$ by 45%; however, this can often be repaired by combining refine–long with `retrigX`, see b). See also [She02] for a very useful discussion of mesh quality (in the planar setting, and in 3D).

Conversely, `coarsenX` can be used to coarsen previously refined triangles, again from a list generated by `p.fuha.e2rs`, which should be reset from the one chosen for refinement. For instance, `e2rsAi` selects the $\lfloor \sigma n_t \rfloor$ triangles of *smallest* area, but these have to be from the list of previously refined triangles.

`degcoarsenX` works differently: It aims to collapse short edges to remove acute triangles, defined by $s - r > 2R$, where s, r, R are the semiperimeter, the inradius, and the circumradius for each triangle. For us, this is mainly important for *small* acute triangles (which develop when X bulges in, i.e., at

[6] This turned out to be convenient, i.e.: When plotting from file we very often want to change the look of plots "globally", i.e., without first loading the point and then adapting settings.

necks), and hence before coarsening we compute a size parameter $\varepsilon = |T_{n_c}|$ where we assume the triangles ordered by increasing size and let $n_c = \lfloor \sigma_c n_t \rfloor$, with $\sigma_c =$ `p.nc.sigc` a user chosen fraction (upper bound) of triangles to coarse. Then, only acute triangles of size $|T| \leq \varepsilon$ are refined. Both, `refineX` and `degcoarsenX` can be told to *not* refine/coarsen boundary triangles, which is useful for the case of periodic BCs.

b) We also provide two small modifications of (actually interfaces to) code from [PS04]. In `retrigX.m` we generate a new (Delauney) triangulation of X, keeping intact the surface structure of X. This is in particular useful if X has been obtained from *long* refinement, which typically results in nodes having 8 adjacent triangles (valence 8), while "standard" triangulations (and the output of `retrigX`) have valence 5 and 6, which generally seems to result in more robust continuations. In `moveX` we combine `retrigX` with motion of the points in X due to "truss forces" of the triangulation, aimed at more uniform edge lengths. Due to the similarity of the triangulation truss forces and surface tension, this works best for minimal surfaces (H=0), or otherwise for surfaces with small $|H|$.

c) In `refineX, coarsenX`, and `degcoarsenX`, the last solution u and in particular the u component of the branch tangent τ are interpolated to the new X, but the PDE $G(u, \lambda) = 0$ is *not* directly solved to correct the refinement/coarsening/interpolation error. In simple cases, if any of the above functions is called during continuation, for instance in a "REFinement–User–FUnctions" `refufu`, this is done in the next continuation step; see `refufu` and `refufumaxA` in the demo `spcap1`, §3.1. In more complicated cases, e.g., when more than one of `refineX, coarsenX`, and `degcoarsenX` are called for refinement, it may be advantageous to solve $G(u, \lambda) = 0$ for correction between the different calls; see `refufu` in the demo `vesicles`, §4.1. The same applies in principle to `retrigX` and `moveX`, which keep the number of mesh–points fixed but deliberately also change the (discrete) system $G(u, \lambda) = 0$. However, in our demos we call `retrigX` and `moveX` in direct combination with one of `refineX, coarsenX` or `degcoarsenX`, and hence the error introduced by `retrigX` and `moveX` is taken care of automatically by the subsequent solve. ⌋

Example 2.7 As an example for different meshing and mesh refinement options, in the demo `parabol` we consider the (non–parametric) quarter of a paraboloid

$$P = \{z = z_a(x, y) = x^2/a^2 + y^2/b^2 : (x, y) \in Q = (0, 1)^2\}.$$

The mean curvature of P is $H_P(x, y) = \dfrac{a^2 + b^2 + 4x^2/a^2 + 4y^2/b^2}{a^2 b^2 (1 + 4x^2/a^4 + 4y^2/b^4)^{3/2}}$, and hence we want to solve for X the Dirichlet problem

$$H - H_P = 0, \quad z = z_P \text{ for } (x, y) \in \partial Q \tag{39}$$

We start with a coarse initial mesh ($n_p = 36$ points) in the x–y–plane, mapped to $\mathbb{R}^3$ as $(x, y, z(x, y))$. This is *not* a solution of the discrete problem (39) due to the discretization error of the numerical H (see Fig.4 and Fig.5). Hence we want to solve (39) and use mesh adaptation to improve the approximation, which we measure as $z - z_a$ (in different norms). In Fig. 6(a–c) we first show three options for meshes. In (a), we have a default triangle mesh in the x–y plane, while (b) shows a "criss–cross" mesh obtained from (a) by one uniform "refine–long" step. In contrast to (a), (b) is symmetric (in the x–y plane) wrt reflection in x and y, and such meshes have proven useful for legacy PDE problems in the plane (and in 3D), where it is sometimes crucial to have the mesh reflect symmetries of the PDE, see [Uec21, §4.1.1]. From the `Xcont` point–of–view, a salient feature of (b) is that it alternates valence 4 and valence 8 points in the bulk, which always happens when applying refine-long to a valence–6–triangle mesh. In (c) we have the same points as in (b) but applied `retrigX`, which here converted most of the bulk nodes back to valence 6.

We now use the mesh from (a) and three different iterative "adaption-solve" strategies S_j, $j = 0, 1, 2$ to approximate the known solution $(x, y, z_a(x, y))$ of (39). In S_0 we use M^{full} and refine-long of the $\sigma = 1/2$ largest triangles. S_1 is like S_0 but with $M = M_{\text{Voronoi}}$ instead of M^{full}, and S_2 is S_1 but with `retrigX` after each refinement. In (d,e) we show the errors for S_0 and S_2 after one step (S_1 is quite similar to S_0 here), with naturally refinement in the top half of the initial mesh. The main message is that the main error sits at the valence 4 and 5 nodes, and that refine–long followed by `retrigX` introduces a layer of valence 5 nodes between areas that have/have not been refined. As shown in (f), S_2 is best for the given simple problem, but the overall convergence (in $\|\cdot\|_\infty$) of the three strategies is similar, and we have also have the same rate ≈ -1 in $\|\cdot\|_2$. ⌋

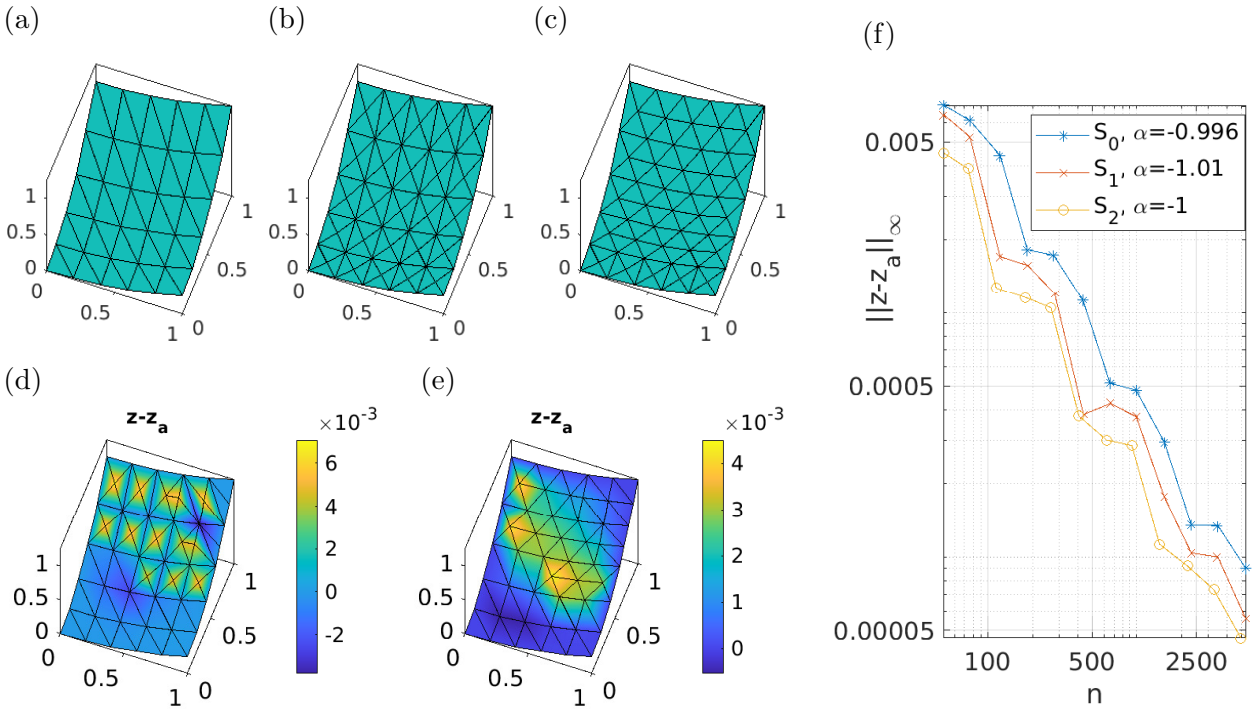

Figure 6: Example 2.7, `geomtut/parabol/cmds1.m`. Initial mesh (a), different mesh–refiments and local effects on error (b)–(e), but similar overall convergence (f).

3 Second order example implementations and results

Our demos are meant to show how to set up different geometric bifurcation problems, in particular with different BCs. They mostly deal with classical minimal or more generally CMC surfaces, for instance the Enneper and Schwarz–P surfaces, and so called nodoids (including physically relevant liquid bridges). Many demos start with CMC surfaces of revolution, and our main interest then are bifurcations breaking the rotational symmetry. The minimal surfaces in §3.2 are motivated by Plateau's problem of soap films spanning a given wire, and in §4 we consider 4th order problems obtained from the Helfrich functional.

All demos come with a number of function files namely (at least, with * a placeholder, usually to be replaced by a short problem name, cf. §2.3): `sG*.m` describing the rhs of the problem; `*init.m` for initialization; `userplot.m` and `getM.m` for technical reasons (downward compatibility). Additionally, in some demos we overload functions from `libs/Xcont`, e.g., `cmcbra.m` for branch output. Finally, there are script files `cmds*.m` with * a number if there is more than one script. In our descriptions of the first demos, we give tables listing the used files and their purpose (starting with the scripts), and

we give a few listings of (parts of) pertinent files to discuss important points. This becomes less for the later more advanced demos, for which we rather put more comments into the m–files themselves.

3.1 Spherical caps

We start with the continuation in volume V of spherical caps over the unit circle γ in the x–y plane, as previewed in Fig. 1(a). It is known [ALP99], [KPP15, §2.6] that no bifurcations occur, and hence this only serves as an introductory toy model. Table 5 gives an overview of the used files, and Listings 1–2 show the initialization `scinit.m`, the rhs `sGsc.m`, and the first script `cmds1.m`. The BCs are $\partial X = \gamma = \{(x, y, 0) \in \mathbb{R}^3 : x^2 + y^2 = 1\}$, which since they hold for the initial unit disk translate into

$$u|_{\partial X} = 0. \tag{40}$$

Remark 3.1 a) We can as well continue directly in H, without any constraints, and starting from the disk again obtain the same branch (see lines 14,15 of `cmds1.m`). Our setup in `cmds1.m` is motivated by applications, where typically the volume is the external parameter. I.e., the setup is a template how to use the volume (`qfV`) or area (`qfA`) constraints, together with the derivatives (`qjacV` and `qjacA`). Note that `p.nc.ilam=[2,1]` for using V as the *primary active* parameter in `cmds1`, and `p.nc.ilam=[3,1]` for using A, while $H =$ `par(1)` is a *secondary* active parameter in both cases.

b) Only the *active* continuation parameters are updated in `p.u`; thus, when continuing only in H, say, then to plot, e.g., A over H we cannot choose `p.plot.bpcmp`=3 (the parameter index of A), but must take `p.plot.bpcmp`=3+2=5. This is because the computed A is put second after the parameters in the output function `cmcbra`, and here we have three parameters (H, V, A). But again, $A =$ `par(3)` is only updated if $3 \in$ `p.nc.ilam`, i.e., if A is an active parameter. ⌋

Table 5: Files in `pde2path/demos/geomtut/spcap1`; the last two are typical examples of (small) local mods of library functions.

`cmds1.m`	continuation in (V, H) and in (A, H), respectively.
`cmds2.m`, `cmds3.m`	tests of different mesh refinement options, and MCF tests.
`getM.m`	standard (Voronoi) mass matrix.
`scinit.m`	Init, data stored in `p.u` (including computed H, A and V), and in `p.X` and `p.tri`.
`sGsc.m, scjac.m`	rhs based on (26), and Jacobian based on (23).
`mcff.m`	mean curvature flow rhs f; problem dependent via choice of `getN`.
`cmcbra.m`	local copy and mod of library function `cmcbra.m` to put error $e(X)$ (42) on branch.
`refufu.m`	local copy and mod (and renaming) of `stanufu.m` to do adaptive mesh refinement based on $e(X)$; "switched on" by setting `p.fuha.ufu=@refufu`.
`coarsufu.m`	similar to `refufu.m`, used for mesh coarsening of decreasing caps;

```
1 function p=scinit(nx,par) % spherical cap, init
  p=stanparam(); p=stanparamX(p);  % set stanparam, adapt to X; then reset some
  p.fuha.sG=@sGsc; p.fuha.sGjac=@scjac; p.sw.spcalc=0; p.sw.bifcheck=0;
  pde=diskpdeo2(1,nx,round(nx/2)); % disk preimage discretization, pde-object
  % not stored, only p.DBC, p.tri and p.X (generated below) used subsequently
6 p.np=pde.grid.nPoints; p.nu=p.np; p.nt=pde.grid.nElements;  % store dimensions
  p.sol.xi=1/p.nu; p.n0=p.np; % u-vs-lam weight, initial mesh size (for coarsening)
  p.nc.neq=1; p.sw.jac=0; % here, for simplicity, numerical Jacs
  po=pde.grid.p; x=po(1,:)'; y=po(2,:)'; u=0*ones(p.np,1); % set ICs
  p.u=[u; par]; p.X=[x,y,0*x]; p.tri=pde.grid.t(1:3,:)'; % store initial X and tri
11 p.DIR=pde.grid.e(1:2,:)'; p.idx=unique(p.DIR(:)); % edges, and points for BCs
  p=oosetfemops(p); p.plot.auxdict={'H','V','A'}; % dummy oosetfemops!
  p.u(p.nu+2)=getV(p,p.u); p.u(p.nu+3)=getA(p,p.u); % get initial vol & area
  p.nc.lammax=200; p.nc.dsmax=3; p.nc.dlammax=3; p.file.smod=10;
```

Listing 1: `spcap1/scinit.m`; the pde-object `pde` in line 4 is generated in a legacy `pde2path` setup but only used to generate the initial `p.X`, with initial triangulation stored in `p.tri` (line 10).

During init, we call `pde=diskpdeo2` to generate a temporary FEM object from which we extract the initial mesh to generate the initial `p.X` and store `p.tri` as the triangulation. Additionally we extract `p.DBC` as the (Dirichlet) boundary *edge* index vectors, and `p.idx` as the boundary *point* indices. This is in principle redundant, but it makes the setup of the DBCs in `sGsc` shorter.

```
function r=sGsc(p,u) % spherical cap PDE (more generally: CMC with DBCs)
par=u(p.nu+1:end); H0=par(1); u=u(1:p.np); % split into u and parameters
N0=getN(p,p.X); X=p.X+u.*N0; N=getN(p,X); % normal, new X, new normal
M=getM(p,X); LB=cotmatrix(X,p.tri);  % mass matrix and Laplace-Beltrami
r=-0.5*dot(LB*X,N,2)+M*(H0*ones(p.np,1)); % rhs-PDE, i.e., -H(X)+H0=0
r(p.idx)=u(p.idx);    % Dirichlet BCs
```

```
p2pglob.tsw=1; p2pglob.vi=[20,40]; p2pglob.edc='k'; % plotting controls
%% init; pars will be overwritten in scinit
nx=12; h0=0; v0=0; a0=0; par=[h0; v0; a0]; % initial pars
p=scinit(nx,par); p=setfn(p,'cap1'); p.sol.ds=0.1;
p.sw.jac=0;  % numerical (0) or functional (1) jacs for G, speed no problem
p.sw.qjac=1; % numerical (0, too slow), hence functional (1) derivative for q
p.nc.ilam=[2 1]; p.nc.nq=1; p.fuha.qf=@qfV; p.fuha.qfder=@qjacV; % cont in V,
p.plot.bpcmp=1; p.nc.usrlam=[2 4]; % cmp for branch-plot, vals for forced output
p=cont(p,5); % go
%% alternate cont and mesh refinement based on triangle areas
p=loadp('cap1','pt5','cap1r'); p.sw.nobdref=0; p.sw.rlong=1; p.file.smod=2;
sig=0.2; for i=1:10; p=refineX(p,sig); p=cont(p,5); end
%% just cont in H (no constraints)
p=loadp('cap1','pt0','Hcont');p.nc.nq=0;p.nc.ilam=1;p.sol.ds=-0.1;p.plot.bpcmp=5;
sig=0.2; for i=1:4; p=cont(p,5); p=refineX(p,sig); end % alternate ref. and cont
```

Listing 2: `spcap1/sGsc.m`, and start of `cmds1.m` (omitting plotting).

In `cmds1.m` we then continue the initial disk (with $V = 0$ and $A = \pi$) in V. For this we switch on the constraint $V(u) = V$ via `p.nc.nq=1`, `p.fuha.qf=@qfV; p.fuha.qfder=@qjacV`, with the `Xcont` library functions `qfV` and `qjacV`, and set `p.nc.ilam=[2,1]`, cf. Remark 3.1. For mesh adaptation we use the triangle areas on X as selector, and `refineX` also updates `p.DBC` and `p.idx`, leading to Fig. 1(a), where we use repeated mesh refinement every 5th step. This way we can accurately continue to arbitrary large V, i.e., arbitrary large "cap radius" R, where $H = 1/R$ asymptotes to $H = 0$. In the second part of `cmds1.m` we continue in A and hence set `p.nc.ilam=[3,1]; p.fuha.qf=@qfA;` and `p.fuha.qfder=@qjacA`. This yields exactly the same branch as the continuation in V, and all this works very robustly and fast.

Remark 3.2 The numerical Jacobians of G (for `p.sw.jac=0` in line 5 of Listing 2) are sufficiently fast to not play a role for the speed of the continuation, at least for $n_p < 2000$, say, because MATLAB's `numjac` can efficiently exploit the known sparsity (structure) of $\partial_u G$, given by the sparsity structure of the Laplacian K, or equivalently, by the sparsity structure of the (full, not Voronoi) mass matrix M. On the other hand, if q implements some integral constraints, e.g., area or volume, then $\partial_u q(u) \in \mathbb{R}^{n_q \times n_p}$ is dense, and numerical derivatives for $\partial_u q$ are a serious bottleneck. For illustration, in `cmds1.m` we use the commands `jaccheck` and `qjaccheck`, which are rather important for "debugging" when numerical Jacobians become too slow. Both return relative errors between functional and numerical Jacobians, and as a rule of thump, in $\partial_u G$ relative errors $\leq 10^{-3}$ should be achieved, and do not affect the continuation or bifurcation results, and for $\partial_u q$ even somewhat larger relative errors are usually no problem. ⌋

In `cmds2.m` we test different options for mesh adaptation, see Listing 3 and Fig. 7.[7] The black line `capr1` in (a), starting from `cap1/pt10`, corresponds to adaptation each 15th step, with "refinement

[7] Fig. 7(a,b) shows essentially verbatim output from `plotbra` in `cmds2.m`, where the dots and numbers indicate the continuation step, subsequently used also in the sample plots as in (c). This also holds for all subsequent plots, and the only "manual adjustments" are the occasional repositioning of the numbers at the arrows by drag and drop, as this is not automatically optimized.

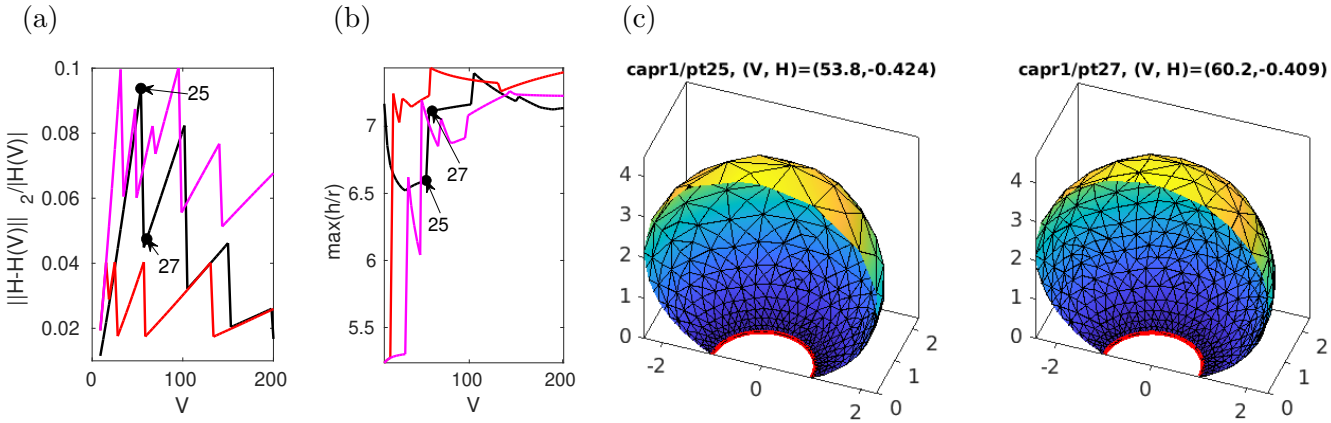

Figure 7: Results from `spcap1/cmds2.m`. (a) Error $e(X) := \|H - H(V)\|_2/|H(V)|$ for refinement each 15th step (`capr1`, black) (starting at step 10), when $e(X) >$ `p.nc.errtol` $= 0.05$, using `p.fuha.ufu=@refufu` (`capr3`, red), and when $\max(A) > 0.3$ using `p.fuha.ufu=@refufumaxA` with $\sigma = 0.3$ (`capr4`, magenta). At $V = 200$, $n_p = 1452$ on `capr1`, $n_p = 1486$ on `capr3`, and $n_p = 636$ on `capr4`. (b) Mesh distortion $\delta_{\text{mesh}} = \max(h/r)$ (edge-length over in–radius). (c) Illustration of meshes before/after refinement at pt25; plots cropped at $y = 0$ for better visibility of the meshes, and the boundary at $z = 0$ marked in red.

factor" $\sigma = 0.3$ (fraction of triangles marked for refinement). As we choose `p.sw.rlong=1` we only bisect the longest edge of a selected triangle, and the actual fraction of refined triangles is between σ and 2σ.[8] For `capr3` (red) we refine when the "error" $e(X)$ exceeds `p.nc.errbound` $= 0.04$, where $e(X)$ is also used for plotting and defined as follows: For given V we compute the (exact) $H(V)$ of the associated (exact) spherical cap $C(V)$ as

$$H(V) = -\frac{\pi^{1/3}(3V + s - \pi^{2/3})(s - 3V)^{1/3}}{s(3V + s)^{1/3}}, \quad \text{where} \quad s = \sqrt{9V^2 + \pi^2}. \tag{41}$$

We then define the "relative L^2 error"

$$e(X) = \|H(X) - H(V)\|_{L^2(X)}/|H(V)|, \tag{42}$$

and put $e(X)$ on the branch in the modified local copy `cmcbra.m` of the standard (library) `cmcbra.m`.[9] $e(X)$ can then be plotted like any other output variable, and, moreover, can be used (without recomputing) in `p.fuha.ufu` (user function), which is called after each successful continuation step. The default (library) setting `p.fuha.ufu=@stanufu` essentially only gives printout, and to switch on the adaptive meshing we rename and modify a local copy as `refufu.m`, and set `p.fuha.ufu=@refufu`. Since $e(X)$ is at position 13 in (our modified) `out=cmcbra(p,u)`, and since `out` is appended to the six values from `bradat`, cf. Remark 2.5a), in `refufu.m` we then simply add the commands

```
if brout(6+13)>p.nc.errbound; p=refineX(p,p.nc.sig); end.
```

Another "natural" alternative is to refine when

$$a_{\max} = \max(a_1, \ldots, a_{nt}) > \texttt{p.maxA}, \tag{43}$$

i.e., when the maximum area of the `nt` triangles exceeds a chosen bound. This is *not* an error estimator

[8] For `p.sw.rlong=0` (RGB refinement with possibly longer cascades to avoid hanging nodes) σ is only a lower bound.

[9] We also put $\|H(X) - H(V)\|_\infty/|H(V)|$ and $z_{\max}(V) - z_{\max}$ on the branch, where $z_{\max}(V)$ is the height of $C(V)$ and $z_{\max}$ the numerical height; these can then also be plotted via `plotbra`, and/or chosen as error indicators, but the L^2 error seems most natural. Also note that $e(X)$ is normalized by $|H(V)|$ (which decays in V), but not by A (which increases with V).

in any sense (as a plane can be discretized by arbitrary large triangles), but an ad hoc criterion, with typically an ad hoc choice of `p.maxA`, which could be correlated to H. It is implemented in `refufumaxA.m` which, if $\max A >$ `p.maxA`, calls `refineX` with `e2rsmaxA` to select all triangles with $A > (1-\sigma)$`p.maxA`. With `p.maxA` $= 0.3$ and $\sigma = 0.2$ this yields the magenta line in Fig.7(a).

The samples in Fig.7(c) illustrate a refinement step on the black branch, yielding a "reasonable" mesh also at large V. However, this naturally depends on the choice of steps between refinements (and on the refinement fraction `sig` and continuation stepsize `ds`). For the red line in Fig. 7(a), the refinement when the error $e(X)$ exceeds the chosen bound `p.nc.errbound` is more genuinely adaptive, and this similarly holds for `capr4` based on (43), see also `cmds2.m` for various further plots. (b) shows that the long–refinement generally yields a (mild) increase of the mesh distortion δ_{mesh}, but overall the mesh–quality stays very good.

```
%% tests of mesh-refinement, preparation: set rlong=1 and dsmax,dlammax
p=loadp('cap1','pt10'); p.sw.rlong=1;  p.nc.dsmax=4; p.nc.dlammax=4; p0=p;
%% alternate refine and cont, here ref.each 15th step; single steps for saving
p=p0; p=setfn(p,'capr1'); sig=0.3; nsteps=13;
for i=1:5; p=refineX(p,sig); p=cont(p,1); p=cont(p,1); p=cont(p,nsteps); end
%% refine when error exceeds errbound, using refufu
p=p0; p=setfn(p,'capr3'); p.fuha.ufu=@refufu; p.nc.errbound=0.04; p=cont(p,100);
%% refine when max A exceeds p.maxA, using refufumaxA
p=p0; p=setfn(p,'capr3'); p.fuha.ufu=@refufumaxA; p.fuha.e2rs=@e2rsmaxA;
p.maxA=0.3; p=cont(p,100);
%% error plots; error appended at end of cmcbra, component c=13
lab=[25 27]; c=13; mclf(8); plotbra('capr1','pt71',8,c,'lab',lab,'fp',11);
plotbra('capr3',8,c,'lab',[],'cl','r','fp',11);
```

Listing 3: Selection from `spcap1/cmds2.m`, refinement each 15th step, $e(X)$–dependent refinement via setting `p.fuha.ufu=@refufu` and `p.nc.errbound=0.04`, and refinement based on (43).

In `cmds3.m` and Fig. 8 we *decrease* V from $V \approx 150$ (running the branch `capr1` from Fig. 7 backwards), and test the MCF from a spherical cap at $V \approx 15$. For both, because the shrinking of the caps gives mesh distortions, the main issue is that we now need to alternate continuation/flow and mesh–*coarsening*. For the continuation we give two options: similar to the refinement for increasing V in Fig. 7, we either coarsen after a fixed number of steps (black branch), or when $\delta_{\text{mesh}} > 8$ (magenta branch). Both here work efficiently only until $V \approx 35$, after which new parameters for the coarsening should be chosen. For the MCF in (d) we similarly coarsen after a given number of time steps. With this we can flow back to the disk, more or less reached at $t = 3$, but the last plot in (d) shows that along the way we have strongly distorted meshes, which are somewhat repaired in the coarsening steps, and the final distortion with $\delta_{\text{mesh}} \approx 30$ is not small but OK.

```
%% go branch backwards, cont-coarsening loop
p=loadp('capr1','pt56','capr1b'); p.sol.ds=-3.2; p=resetc(p);p.fuha.e2rs=@e2rsAi;
p.file.smod=5; p0=p;  sig=0.5; for i=1:8; p=coarsenX(p,sig); p=cont(p,5);   end
%% coarsen via coarsufu
p=p0;p=setfn(p,'capr1d');p.fuha.ufu=@coarsufu; p.nc.delbound=8; p=cont(p,40);

%% MCF, with initial large V; to handle meshing, alternate flow and coarsening
% this may require trial and error to balance dt, flow-length nf, and
% coarsening sigc.     First some graphics settings, then load and prepare:
p2pglob.cut=0; p2pglob.vi=[30,40]; p2pglob.cm='spring'; p2pglob.tsw=4;
p=loadp('cap1r','pt15','mcf'); sigc=0.1;  dt=0.0005; nf=500; nplot=100;
p.sw.nobdcoarsen=0; p.t=0; plotHK(p); figure(1); title('t=0');  % prepare MCF
p.fuha.flowf=@mcff; t=0; ts=[]; p.fuha.e2rs=@e2rsAi;
%% the MCF/coarsening loop; repeat this cell as desired
for i=1:4; [p.X,t,ts]=geomflow(p,t,ts,dt,nf,nplot); p=coarsenX(p,sigc); end
```

Listing 4: Selection from `spcap1/cmds3.m`; decreasing V by going backwards, and MCF; both need to be combined with coarsening. Omission between lines 5 and 14 deal with plotting, and further experiments are at the end of `cmds3.m`.

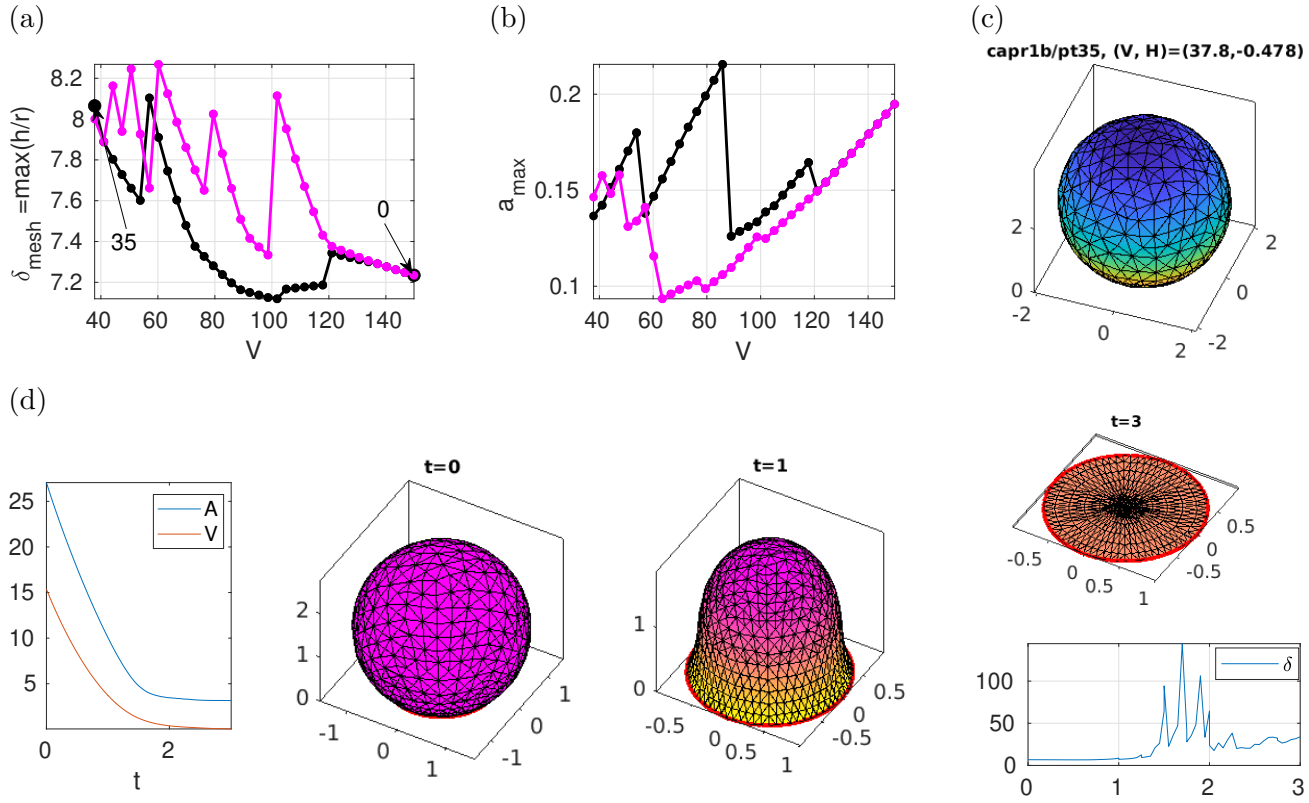

Figure 8: Results from `spcap1/cmds3.m`. (a)-(c) continuation backwards in V from $V{\approx}150$ (n_p=1452); coarsening each 5th step (`capr1b`, black, n_p=644 at V=40) vs coarsening when $\delta_{\text{mesh}} > 8$ (magenta, n_p=650 at V=40)). (d) MCF from the spherical cap at $V{\approx}15$. time series of A and V, sample plots, and time series of δ_{mesh} (last plot). Coarsening at times $t = 0.25j$, altogether from $n_p = 773$ at $t = 0$ to $n_p = 450$ at $t = 3$.

Remark 3.3 The performance of the MCF as in Fig. 8, based on our simple explicit Euler stepping, depends on the choice of flow parameters, i.e., step size `dt`, number n_{f} of steps before coarsening, and coarsening factor σ. With too weak coarsening (large n_{f}, or small σ), triangles may degenerate. Too aggressive coarsening (large σ) may lead to wrong identification of boundary edges. Altogether, at this point we must recommend trial and error. ⌋

3.2 Some minimal surfaces

Plateau's problem consists in finding soap films X spanning a (Jordan) curve (a wire) γ in $\mathbb{R}^3$, and minimizing area A. Mathematically, we seek a *minimal* surface X, i.e., $H(X) \equiv 0$, with $\partial X = \gamma$. Such problems have a long history, and already Plateau discussed non–uniqueness and bifurcation issues, called "limits of stability" in [Pla73].

A classical example for which a bifurcation is known is Enneper's surface, see §3.2.3. However, in the demo `bdcurve` we first start with other BCs, meant to illustrate options (and failures) for prescribing boundary values in our numerical setup. We introduce parameters $\alpha \in \mathbb{R}$ and $k \in \mathbb{N}$ (angular wave number) and a switch `p.bcsw`, and consider BCs of the form

$$u|_{\partial X} = 0, \quad \text{(for } \texttt{p.bcsw=0}\text{)}, \tag{44}$$

$$X_3|_{\partial X} = \alpha \sin(k\phi), \quad \phi = \arctan(y/x) \quad \text{(for } \texttt{p.bcsw=1}\text{)}, \tag{45}$$

$$\partial X = \gamma(\cdot; \alpha, k) \quad \text{(for } \texttt{p.bcsw=2}\text{)}, \tag{46}$$

where γ in (46) is a prescribed boundary curve, depending on parameters α, k. Specifically, in §3.2.2

we choose

$$\gamma(\phi;\alpha,k)=\begin{pmatrix}\beta\cos(\phi)\\ \beta\sin\phi\\ \alpha\cos(k\phi)\end{pmatrix},\quad \phi\in[0,2\pi),\quad \beta=\sqrt{1-\alpha^2\cos^2(k\phi)}. \tag{47}$$

For (45), ∂X is not uniquely determined by the parameter α (and fixed k), and this illustrates how our scheme (4) can fail, and that the condition $Y\in\mathcal{N}_C$ cannot be dropped in Lemma 2.1. Relatedly, for (45) the continuation can genuinely depend on the continuation stepsize `ds`, as different predictors give different BCs (45). In other words, the problem is under–determined and the continuation algorithm itself "chooses" the BCs. Thus, (45) is a cautionary example, though it produces interesting minimal surfaces. On the other hand, (46) is a genuine DBC with unique continuation, which however requires a modification of the "standard" `updX.m`, and careful mesh handling. Together, (45) and (46) are meant to illustrate options.

The condition on β in (47) yields that $\|\gamma\|_2=1$, i.e., that γ lies on the unit sphere, for $\alpha\in[0,1]$. Moreover, the projection of γ into the x–y plane is injective, and this is useful since we then can extract ϕ from ∂X. In §3.2.3 we treat a variant of (47), associated to the Enneper surface, where the discretization of γ requires a further trick. On the other hand, γ from (47) becomes singular at $\phi=j\pi/k$ as $\alpha\to 1$, which is useful to test mesh–handling. Thus, §3.2.2 and §3.2.3 are quite related, but illustrate different effects.

Table 6 shows the used files, Listing 5 shows `sGbdcurve`, and Listing 6 the "new" (compared to `spcap1/`) files needed to run the BCs (46). The other files are essentially as in `spcap1/`, except that we now have altogether five parameters (H,V,A,α,k), and that we use the additional parameter `p.bcsw`.

Table 6: Files in `pde2path/demos/geomtut/bdcurve`.

`cmds1a.m`	continuation in α (and H) for (45), see Fig.9; MCF tests in `cmds1b`.
`cmds2.m`	continuation in α for (46), see Fig. 10.
`bdcurveinit.m`	Initialization, very similar to `scinit`.
`sGbdcurve.m`	very similar so `sGsc`, except for the BCs.
`updX.m`	mod of standard `updX`; for `p.bcsw=2` setting the boundary curve.
`bcX.m`	user function to give γ, here implementing (47).

```
function r=sGbdcurve(p,u)  % PDE rhs, discrete mean curvature
if p.bcsw==2; Xbc=bcX(p,u); p.X(p.idx,:)=Xbc; end % set Bdcurve
par=u(p.nu+1:end); H0=par(1); N=getN(p,p.X); X=p.X+u(1:p.np).*N;
M=getM(p,X); LB=cotmatrix(X,p.tri); N=getN(p,X);
r=-0.5*dot(LB*X,N,2)+M*(H0*ones(p.np,1)); % PDE-rhs, i.e., H-H0=0
switch p.bcsw   % BCs
    case 1; % X_3=al*sin(k*phi)
        al=par(4); k=par(5); phi=angle(X(p.idx,1)+1i*X(p.idx,2));
        r(p.idx)=X(p.idx,3)-al*sin(k*phi);
    otherwise; r(p.idx)=u(p.idx);  % \pa X=\ga  (boundary curve)
end
```

Listing 5: `bdcurve/sGbdcurve.m`, with BCs depending on `p.bcsw`.

```
function Xbc=bcX(p,u) % set BCs; called in sGbdcurv, with u=0 on bdry there
par=u(p.nu+1:end); al=par(4); k=par(5); phi=angle(p.X(p.idx,1)+1i*p.X(p.idx,2));
b=sqrt(1-(al*cos(k*phi).^2));  Xbc=[b.*cos(phi), b.*sin(phi), al*cos(k*phi)];
```

```
function [p,u]=updX(p,u)  % local mod of updX.m, with update of BCs.
if p.bcsw==2; Xbc=bcX(p,u); p.X(p.idx,:)=Xbc; end
N=getN(p,p.X); np=p.nu/p.nc.neq; p.up=u; p.X=p.X+u(1:np).*N; u(1:np)=0;
```

Listing 6: `bcX.m` and `updX.m` from `bdcurve/`, needed to run with the BCs (46).

3.2.1 Prescribing one component of X at the boundary

In `cmds1a.m` (Listing 7) we continue (45) in α, starting with $\alpha = 0$ at the flat disk, and first with angular wave number $k = 2$. Some results are shown in Fig. 9.

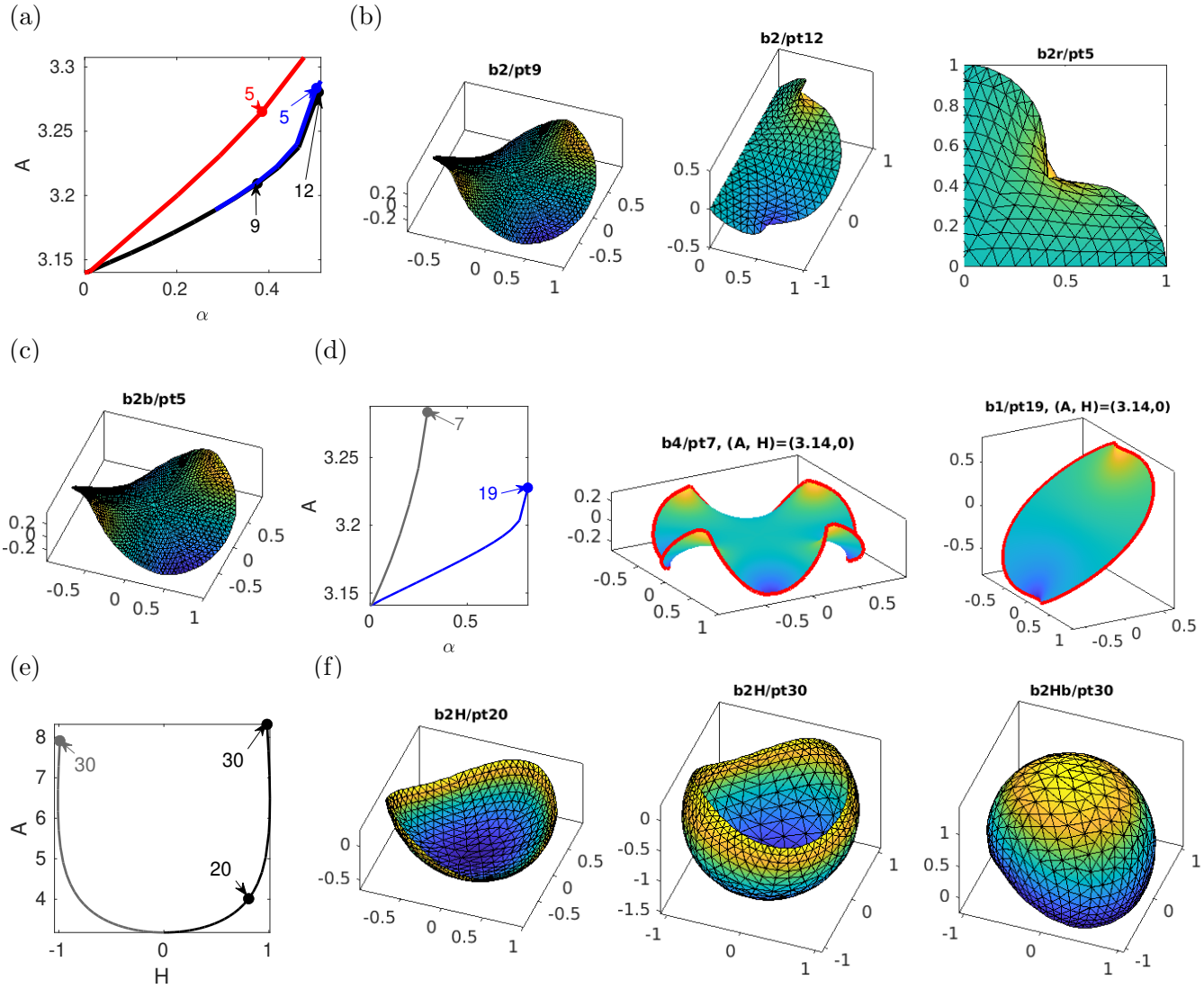

Figure 9: Results from `bdcurve/cmds1a.m`. (a) Continuation in α with BCs (40), $k = 2$: black and red branches with n_p=945 mesh points and fixed `ds=0.05` (black) and `ds=0.1` (red), illustrating the step–length dependence of the continuation; blue branch `b2r` via refinement at boundary. Samples in (b) and (c), partly with cropping. (d) Like (a,b) but on finer meshes, with k=1 and k=4, and with marking ∂X in red. All of (a–d) are *minimal* surfaces, i.e., $H \equiv 0$. (e,f) Switching back to continuation in H at `b2/pt6`.

```
%% cont in alpha, BCs X_3-alpha*sin k\phi=0; MUST fail when X gets 'vertical'
nx=15; h0=0; v0=0; a0=0; al=0; k=2; par=[h0; v0; a0; al; k];
p=bdcurveinit(nx,par);  p=setfn(p,'b2'); p.nc.ilam=4;  p.bcsw=1; ds=0.05;
p.sol.ds=ds; p.nc.dsmax=ds; p.nc.lammax=1.5; p.nc.lammin=-1.5;  p=cont(p,13);
```

Listing 7: Start of `bdcurve/cmds1a.m`, running BCs (45).

As we increase α, the surface lifts at $\phi = \pi/4$ and $\pi = 5\pi/4$ according to $X_3 = \alpha \sin(2\phi)$, and sinks at $\phi = 3\pi/4$ and $\phi = 7\pi/4$. Near $\alpha = 0.5$ (`b2/pt12`), X becomes vertical at these angles, and hence our scheme (4) can no longer continue to fulfill the BCs. To better resolve the boundary we use some mesh–refinement *only at the boundary*. For this we choose `p.fuha.e2rs=@e2rsbdry` at `b2/pt6` and obtain the blue branch (with a sample top view as last plot in (b)), which however naturally runs into the same continuation failure at $\alpha \approx 0.5$. Although this was on quite coarse meshes, none of

this changes on finer meshes, and hence this mainly serves as an example of necessary failure of the algorithm (4), and as an example of mesh refinement with `e2rsbdry`.

The red branch in (a) together with sample (c) shows that here the branches are continuation stepsize `ds` dependent. In (d) we choose finer meshes and wave numbers $k = 1$ (blue branch `b1`) and $k = 4$ (`b4`, grey), and get analogous results up to continuation failure. In (e,f) we switch back to continuation in (H, A) from `b1/pt6`, in both directions of positive (black branch) and negative (grey branch) H. As α is now fixed again, ∂X stays fixed even with the BCs (45). The branches are `ds`–independent again, and H asymptotes to nonzero $\pm H_\infty$ as $A \to \infty$. In `cmds1b.m` we run MCF (not shown) from selected solutions from (f), where again we need to undo the refinement which happened, e.g., during the continuation H from `b2Hb/pt0` to `pt30`, and thus we alternate between `geomflow` and `coarsenX` as in `spcap1/cmds2.m` and Fig.8.

3.2.2 A Plateau problem

In `cmds2.m` we choose the BCs (46) with γ from (47). We again continue in α, for $k = 2, 3$, starting at $\alpha = 0$ with the unit disk. The basic idea (Listing 6) to implement (46), (47) is to

$$\text{set } \partial X = \gamma \text{ in } \texttt{updX}, \text{ and } u|_{\partial X} = 0 \text{ in } \texttt{sGbdcurv}. \tag{48}$$

```
%% genuine bdcurve, first with k=2, 5 initial steps
nx=15; al=0; h0=0; v0=0; a0=0; k=2; par=[h0; v0; a0; al; k];
p=bdcurveinit(nx,par); p=setfn(p,'d2'); p.nc.ilam=4; p.bcsw=2; p=cont(p,5);
%% some boundary refinement and coarsening, trial and error to choose sig
p=loadp('d2','pt5');   % reload point (easier for trial and error)
sigr=0.1; sigc=0.1; p=refineX(p,sigr); p=cont(p,2); p=degcoarsenX(p,sigc);
%% continuation alternating with moveX, and refine and coarsen, parameters:
nis=15; ncs=1; % #inner steps (before ref/coars), #cont-steps (before more)
dt=0.1; nit=5; % stepsize and iterations in moveX
for i=1:3;     % outer loop,
  for j=1:nis;  % inner loop, alternate moveX and cont
      p=moveX(p,dt,nit); pplot(p,20); p=cont(p,ncs);
  end
  p=refineX(p,sigr); p=degcoarsenX(p,sigc); % refine and coarsen
end
```

Listing 8: First 15 lines from `bdcurve/cmds2.m`, using the BCs (46).

Figure 10 shows some results from `cmds2.m`. The crucial points are that as we increase α (in particular beyond $\alpha = 0.2$, say) we

- move mesh points via `moveX(p,dt,it)` after `ncs` continuation steps (here `ncs=1`);
- after `nis=15` "inner" steps refine X (introduce new points), here near the boundary, *and* coarsen X, here via `degcoarsenX(p,sigc)`, to remove "bad" triangles.

The parameter `dt` in `moveX` is the Euler step size to balance the "truss forces" [PS04] (`nit` gives the number of iterations), while `sigc` in `degcoarsenX` has a similar meaning as in `refineX` and `coarsenX`, i.e., giving the fraction of triangles to coarsen.[10] Again, the parameters `ncs, nis, sigr` and `sigc` are generally highly problem dependent and it may require (educated) trial and error to find good values. In summary, Fig. 10 shows that with a good combination of `moveX, refineX` and `degcoarsenX` we can continue rather complicated minimal surfaces X (X with complicated boundary curve γ) with reasonable meshes.[11]

[10] In more detail, `degcoarsenX` can also be called as `p=degcoarsenX(p,sigc,iter)`, where `iter` (default 5) gives the number of internal iterations, or as `p=degcoarsenX(p,sigc,iter,keepbd)` where `keepbd=1` (default 0) means that boundary triangles are kept, which is mainly needed for periodic BCs, see §3.4.

[11] As already said in Rem. 2.6b), due to the analogy between the truss forces and surface tension (constant in minimal surfaces) `moveX` works particularly well for minimal X.

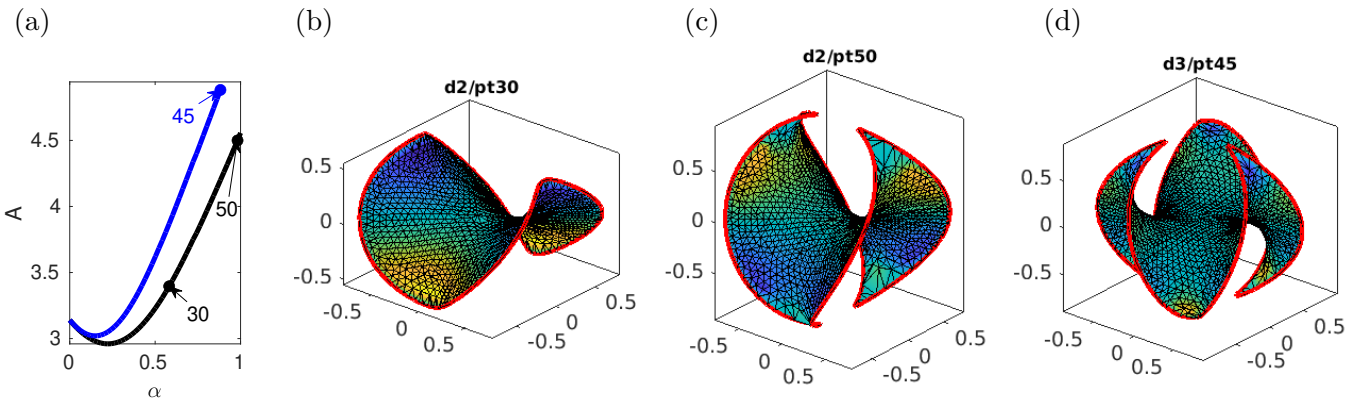

Figure 10: Results from `cmds2.m` for BCs (46) with k=2 (black branch `d2`) and k=3 (blue branch `d3`). Along the way in, e.g., `d2` we do 3 refinements and coarsenings, and the total number of mesh points only increases mildly from n_p=945 to n_p=1177. These are really two different (k=1 vs k=2) continuation problems, out of inifinitely many ($k\in\mathbb{N}$), and hence the two branches in (a) are from different problems and are both stable.

3.2.3 Bifurcation from the Enneper surface

The Enneper surface is a classical minimal surface. Bounded parts of it can be parameterized by[12]

$$X_E = X_E(r,\vartheta) = \begin{pmatrix} r\cos(\vartheta) - \frac{r^3}{3}\cos(3\vartheta) \\ -r\sin(\vartheta) - \frac{r^3}{3}\sin(3\vartheta) \\ r^2\cos(2\vartheta) \end{pmatrix}, \quad (r,\vartheta)\in D_\alpha = [0,\alpha)\times[0,2\pi), \tag{49}$$

see Fig.11. We start by reviewing some basic facts, see [BT84] and the references therein. For $\alpha \le 1/\sqrt{3}$, the boundary curve

$$\gamma(\vartheta;\alpha) = \left(\alpha\cos(\vartheta) - \tfrac{\alpha^3}{3}\cos(3\vartheta), -\alpha\sin(\vartheta) - \tfrac{\alpha^3}{3}\sin(3\vartheta), \alpha^2\cos(2\vartheta)\right), \quad \vartheta\in[0,2\pi) \tag{50}$$

has a convex projection to the x–y–plane, and for $1/\sqrt{3} < \alpha \le 1$ the projection is still injective. This yields uniqueness (of the minimal surface spanning γ) for $0 < \alpha \le 1$ (see [Ruc81] for $\alpha\in(1/\sqrt{3},1]$). For $\alpha > 1$ uniqueness of X_E fails, i.e., at $\alpha = 1$ we have a (pitchfork, by symmetry) bifurcation of different minimal surfaces spanning γ_α [Nit76]. This has been analyzed in detail in [BT84] as a two–parameter bifurcation problem, showing a so called cusp catastrophe.[13]

In the demo `enneper` we simply choose α as a continuation/bifurcation parameter for

$$H(X) = 0, \quad \partial X = \gamma_\alpha, \tag{51}$$

and get the pitchfork bifurcation at $\alpha = 1$. The used files `bcX.m, cmds1.m, cmds2.m, enninit.m, sGenn.m, updX.m` are very similar to those from the demo `bdcurve`, but we also include a Jacobian `sGjacenn.m`, a function `mytitle.m` for customized titles, and `thinterpol.m`, discussed next.

The problem (51) is "easy" in the sense that we have the explicit parametrization (49) which we can use at any α, but like in §3.2.2 it does require care with the meshing, and compared to (47) it requires an additional trick to update ϑ on ∂X after mesh adaption (at the boundary): Since we cannot in general extract ϑ from (50) from the projection to the x–y plane (which is not injective for $\alpha > 1$), we keep a field $\vartheta =$ `p.th` associated to `p.idx` (the indices of ∂X) in the given discretization. Then, if `p.X1` is obtained from refining `p.X` with new mesh–points `p.nX` on ∂X, then we need to update the $\vartheta =$ `p.th` values of `p.nX`. This is done in `p=thinterpol(p,idxold,thold)` by

[12] see also Remark 3.9 for the Enneper–Weierstrass representation

[13] See, e.g., [Uec21, Example 1.30] and the references therein for comments on cusps (and other catastrophes).

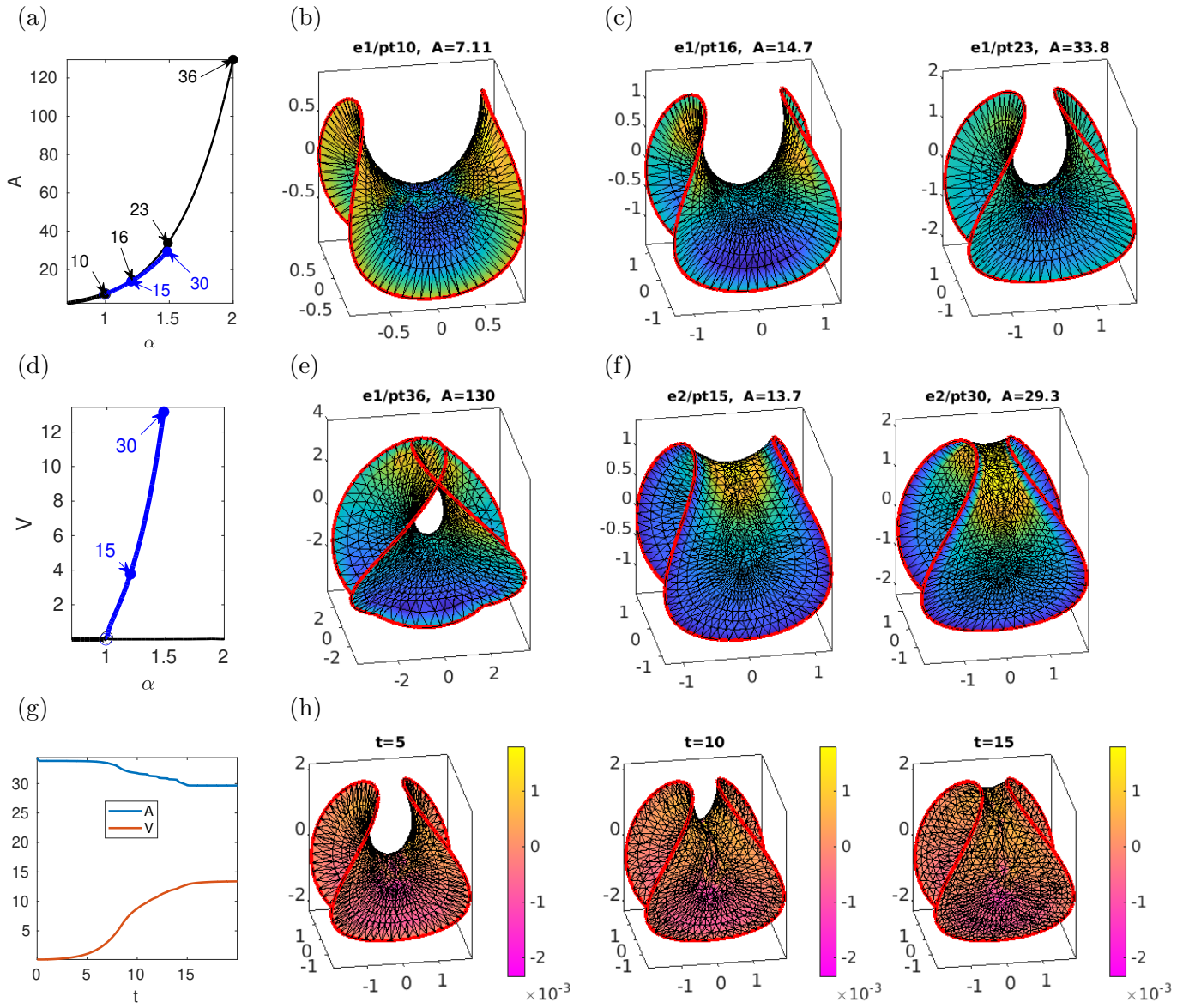

Figure 11: Bifurcation from the Enneper surface X_E, A over α (a), and V over α (d). At $\alpha = 1$ (`e1/pt10` in (b)), the branch `e1b` (blue) with smaller A bifurcates from `e1` (black), samples in (b,c) and (e,f). (g,h) MCF from perturbation of `e1/pt23` to `e2/pt30`, samples showing H.

- finding the (old–point) neighbors of the new points on ∂X;
- linear interpolation of the neighbors' ϑ values to the new points.

This is a question of indexing, and we refer to the source of `thinterpol.m` for comments. A refinement step thus takes the form

```
idold=p.idx;  thold=p.th;  p=refineX(p,sigr);  p=thinterpol(p,idold,thold);
```

see `cmds1.m` which produces Fig. 11. The other files in `enneper/` are very much like in `bdcurve/`.

At $\alpha = 1$ we find a supercritical pitchfork bifurcation from X_E, branch `e1` (black), to a branch `e2` (blue) which breaks the $(x, y, z) \mapsto (-y, x, -z)$ symmetry of X_E (rotation by $\pi/2$ around the z axis and mirroring at the $z = 0$ plane). The solutions "move up" (or down) in the middle, which decreases A compared to X_E, cf. (c) vs (f). (d) illustrates that the (algebraic) volume V of X_E is always zero. The numerical continuation of `e1` to large α is no problem, using suitable mesh–adaption, even as $\gamma(\cdot;\alpha)$ self–intersects for $\alpha > \sqrt{3}$, because the associated parts of X_E do not "see" each other, cf. (e) for an example. The continuation of `e2` to larger α is more difficult, and fails for $\alpha > 1.5$, as for instance shortly after `e1b/pt30` we can no longer automatically adapt the mesh near the top.

However, physically the change of stability at the symmetry breaking pitchfork at $\alpha = 1$ is most interesting. Using suitable combinations of `geomflow`, `refineX`, `degcoarsenX` and `moveX` we can use MCF to converge for $\alpha > 1$ and $t \to \infty$ to `e2`, from a variety of ICs, for instance from perturbations of `e1`, see Fig. 11(g,h), and `enneperflow.avi` in [MU24]. After convergence we can then again continue the steady state, see `cmds2.m`.

3.3 Liquid bridges and nodoids

Weightless liquid bridges are CMC surfaces with prescribed boundary usually consisting of two parallel circles wlog centered on the z-axis at a fixed distance l and parallel to the x–y plane. Additionally there is a volume constraint, which makes the problem different from Plateau's problem. See for instance [SAR97] and the references therein for physics background and results (experimental, numerical, and semi-analytical).

We consider liquid bridges between two fixed circles C_1 and C_2 of

$$\text{radius } r = r^* = 1, \text{ parallel to the } x\text{–}y \text{ axis and centered at } z = \pm l = \pm 1/2. \tag{52}$$

A trivial solution X_0 is the cylinder, with $H = 1/2$, volume $V = 2\pi l$ and area $A = 4\pi r l$ (without the top and bottom disks). Further explicit solutions are known in the class of surfaces of revolution, for instance nodoids. We first review some theory for nodoids with DBCs, and then continue basic liquid bridges (embedded nodoids), with bifurcations to non axial branches, see Figures 12 and 13. In Figure 14 we then start directly with nodoids with one "inner loop". Nodoids with "periodic" BCs are studied in [MP02], and numerically in §3.4, where we also comment on the theory for these.

3.3.1 Nodoid theory

In [KPP17], a family of nodoids $\mathcal{N}(r, R)$ is parameterized by the neck (smallest) radius r and the buckle (largest) radius R. Let $l \in \mathbb{R}$ and $C_1, C_2 \subset \mathbb{R}^3$ be two circles of radius r^* centered at heights $z = \pm l$ and parallel to the x–y plane. With the two parameters $a, H \in \mathbb{R}$ the nodoids are parameterized by the nodary curve

$$(x, z) : [-t_0, t_0] \to \mathbb{R}^2, \quad t \mapsto \big(x(t), z(t)\big) = \left(\tfrac{\cos t + \sqrt{\cos^2 t + a}}{2|H|}, \ \tfrac{1}{2|H|} \int_0^t \tfrac{\cos\tau + \sqrt{\cos^2\tau + a}}{\sqrt{\cos^2\tau + a}} \cos\tau \, \mathrm{d}\tau \right), \tag{53}$$

which is then rotated around the z axis, i.e.,

$$\mathcal{N}_{t_0} : M \to \mathbb{R}^3, \qquad (t, \theta) \mapsto \big(x(t)\cos\theta, x(t)\sin\theta, z(t)\big), \tag{54}$$

where $M = [-t_0, t_0] \times [0, 2\pi)$. Thus, in terms of §2.1 these nodoids are immersions of cylinders. While (53) only gives nodoids with an even number of self intersections (or none), shifting t_0 also gives odd numbers of self intersections. From the immersion $\mathcal{N}_{t_0}$, we can determine geometric quantities by evaluating the parametrization at the endpoints. For example the height and the radius are given by

$$2l = \frac{1}{|H|} \int_0^{t_0} \frac{\cos t + \sqrt{\cos^2 t + a}}{\sqrt{\cos^2 t + a}} \cos t \, \mathrm{d}t, \quad r^* = \frac{\cos t_0 + \sqrt{\cos^2 t_0 + a}}{2|H|}, \tag{55}$$

and the buckle radius (at $t = 0$) is $R = \dfrac{1 + \sqrt{1 + a}}{2|H|}$. Implicitly, the equations in (55) define $a(t_0)$, hence also the mean curvature H, and thus t_0 parameterizes a family of nodoids $t_0 \mapsto \mathcal{N}_{t_0}$. Conversely, given r, l in (52), the implicit equation

$$\frac{l}{2r} \left(\cos t_0 + \sqrt{\cos^2 t_0 + a} \right) - \left(\sin t_0 + \int_0^{t_0} \frac{\cos^2\tau}{\sqrt{\cos^2\tau + a}} \, \mathrm{d}\tau \right) = 0 \tag{56}$$

defines all possible combinations of a and t_0 satisfying the boundary condition, which we exploit to relate our numerics to results from [KPP17], see Remark 3.6.

In order to detect bifurcations from the family (54), we search for Jacobi fields vanishing on the boundary, cf. (24). The unit normal vector (field) of $\mathcal{N}_{t_0}$ is

$$N = \big(\cos t \cos\theta, \cos t \sin\theta, \sin t\big), \quad t \in [-t_0, t_0),\ \vartheta \in [0, 2\pi),$$

and for every fixed vector $\vec{x} \in \mathbb{R}^3$, the function $\langle \vec{x}, N\rangle$ is a solution to (23). So the task is to find $\vec{x}$ and t_0 such that the Dirichlet BCs are fulfilled. The components of N have zeros if the nodoid meets the boundary horizontally (parallel to the x–y plane), which happens at $t_0 = \frac{\pi}{2} + n\pi$, or vertically, which happens at $t_0 = n\pi$ for $n \in \mathbb{N}$. Choosing the unit basis $(e_i)_{i=1,2,3}$, we have in the horizontal case that $\langle e_i, N\rangle\,|_{\partial\mathcal{N}_{t_0}} = 0$ for $i = 1, 2$, and in the vertical case $\langle e_3, N\rangle\,|_{\partial\mathcal{N}_{t_0}} = 0$.

Lemma 3.4 *[KPP17, Lemma 3.4 and Proposition 3.6] Consider the one parameter family $\mathcal{N}_{t_0}$. If for some $t_0 \in \mathbb{R}_+$ the normal vector at $\partial\mathcal{N}_{t_0}$ is*

1. $N = \Big(0, 0, \nu(x)\Big)$, *then $L = \partial_u H(u)$ has a double zero eigenvalue.*
2. $N = \Big(\nu_1(x), \nu_2(x), 0\Big)$ *then $L = \partial_u H(u)$ has a simple zero eigenvalue.*

The immersions are isolated degenerate, i.e., there exists an $\varepsilon > 0$ such that $(\mathcal{N}_t)_{t\in[t_0-\varepsilon,t_0+\varepsilon]}$ has a jump in the Morse index. In 1. this occurs for $t_0 = \frac{\pi}{2} + k\pi$, and in 2. for $t_0 = k\pi$, for every $k \in \mathbb{N}$.

Now general bifurcation results (see the discussion after Lemma 2.2) yield the existence of bifurcation points at the horizontal and vertical cases presented in Lemma 3.4.

Theorem 3.5 *[KPP17, Propositions 3.5 and 3.6] In cases 1. and 2. in Lemma 3.4 we have bifurcation points for the continuation in H. Moreover,*

1. *if $\psi = \langle e_i, N\rangle \in \ker L$ for $i = 1, 2$, then the bifurcating branch breaks the axial symmetry;*
2. *if $\psi = \langle e_3, N\rangle \in \ker L$, then the bifurcating branch breaks the $z \mapsto -z$ symmetry.*

3.3.2 Nodoid continuation with fixed boundaries

Nodoids with DBCs at the (fixed) top and bottom circles are treated in the demo `nodDBC`. Table 7 lists the pertinent files. We treat two cases:

- Short embedded nodoids (liquid bridges) in `cmds1.m`, starting from the cylinder (eventually continued to self-intersecting nodoids).
- Long nodoids (with self–intersections from the start) in `cmds2.m`.

For solutions without axial symmetry we additionally need to set a rotational phase condition (PC): If X is a solution to (3), so is $R_\phi X$, where ϕ is the angle in the x–y plane, and

$$R_\phi \vec{x} = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \vec{x}. \tag{57}$$

Thus, if $\partial_\phi (R_\phi X)|_{\phi=0} = \frac{1}{x^2+y^2}\,(-y\partial_x X + x\partial_y X) \in \mathbb{R}^3$ is non–zero, then it gives a non–trivial kernel of L, which makes continuation unreliable and bifurcation detection impossible. See, e.g., [Uec21, §3.5] for further discussion of such continuous symmetries. Here, to remove the kernel we use the PC

$$q(u) := \int_X \langle \partial_\phi X_0, X_0 + uN_0\rangle \,\mathrm{d}S = \int_X \langle \partial_\phi X_0, N_0\rangle\, u \,\mathrm{d}S =: \int_X \mathrm{d}\phi\ u \,\mathrm{d}S \stackrel{!}{=} 0, \tag{58}$$

Table 7: Files in `pde2path/demos/geomtut/nodDBC`;

`cmds1.m`	continuation in (A, H) of "short" nodoids, starting from a cylinder. First yielding classical liquid bridges, but eventually turning into self intersecting nodoids, requiring restarts. See Figs. 12,13, and `cmds1plot.m` for the plotting.
`cmds1A2t.m`	Relating the numerical BPs from `cmds1.m` to Theorem 3.5.
`cmds2.m`	Continuation in (A, H) of "long" nodoids with one inner loop, see Figs. 14,15.
`bdmov1.m, bdmov2.m`	scripts to make movies of Fig. 12 and Fig. 14, see also [MU24].
`nodinit.m, nodinitl.m`	Initialization of "short" and "long" nodoids
`sGnodD.m, sGnodDjac.m`	rhs with DBCs, and Jacobian.
`qfArot.m`	area and rotational constraints, see `qjacArot` for derivative.
`getN.m`	overload of `getN` to flip N.
`coarsufu.m`	mod of `stanufu.m` for adaptive coarsening, see Fig. 15.

where X_0 is from the last step, with normal N_0, where ϕ is the angle in the x–y plane, and hence $\partial_\phi X = -X_2\nabla_{X_1}X + X_1\nabla_{X_2}X$, where ∇_{X_j} are the components of the surface gradient, cf. (13). On the discrete level we thus obtain the linear function

$$q(u) = (\mathrm{d}\phi)^T u, \text{ with derivative } \partial_u q = (\mathrm{d}\phi)^T, \tag{59}$$

$\mathrm{d}\phi = \langle -X_2\nabla_{X_1}X + X_1\nabla_{X_2}X, N\rangle$, node–wise, i.e., $\nabla_{X_j}X$ is interpolated to the nodes via `c2P`, with Voronoi weights. We then add $s_{\rm rot}q(u)$ to E from (2) with Lagrange multiplier $s_{\rm rot}$, and thus modify the PDE to $G(u) := H(u) - H_0 + s_{\rm rot}\mathrm{d}\phi \stackrel{!}{=} 0$. This removes the ϕ–rotations of non-axisymmetric X from the kernel of $\partial_u G(u)$, and, moreover, $|s_{\rm rot}| < 10^{-8}$ for all the continuations below.

Since the (algebraic) volume V of self–intersecting nodoids is not intuitive, here we use continuation in area A and H. Thus, we start with the constraint `qfA` with derivative `qjacA`. For non–axisymmetric branches we set up `qfArot` and its derivative, where we put (58) as a second component of `qfA`, and similarly for the derivatives, and when we bifurcate to a non-axisymmetric branch, we set `p.nc.nq=2` (2 constraints, area and rotational phase) and `p.fuha.qf=@qfArot`.

3.3.3 Short nodoids

Listing 9 shows how we initialize by either a cylinder (`icsw=0`) or parametrization (54). Here, `pde.grid` is a 2D rectangular FEM mesh, of which we use the second component as $\phi \in [-\pi, \pi]$. Lines 16–20 implement (53) and (54) with *twice* the line $\phi = \pm\pi$ where the rotation around the z–axis closes. To obtain a mesh without duplicate points from this, in the last line of Listing 9 we use `clean_mesh` from the `gptoolbox`.

```
h=par(1); a=par(4); po=pde.grid.p; x=po(1,:)'; phi=po(2,:)';
if icsw==0; p.X=[a.*cos(phi), a.*sin(phi),x]; % just cylinder
else % KPP17 parametrization of nodoids
  x1=(cos(x)+sqrt(a+cos(x).^2))/(2*abs(h)); x2=zeros(size(x)); xl=size(x,1);
  for i=1:xl
    y=@(t) 1./(2*abs(h)).*(cos(t)+sqrt(cos(t).^2+a)).*cos(t)./sqrt(cos(t).^2+a);
    x2(i)=integral(y,0,x(i));
  end
  p.X=[x1.*cos(phi), x1.*sin(phi), x2]; % initial X
end
par(7)=max(p.X(:,3))-min(p.X(:,3)); % needed in getV
[p.X,p.tri]=clean_mesh(p.X,p.tri,'SelfIntersections','ignore');
```

Listing 9: From `nodDBC/nodinit.m`; setting initial `p.X` as a cylinder (`icsw==0`) or via the parametrization (53) and (54) with (x=t) and subsequent removal of duplicate points.

```
%% 1st BP, double, use gentau to choose bif direction
aux.besw=0; aux.m=2; p1=qswibra('N','bpt1',aux);
p=gentau(p1,[1 0],'N1'); p.sol.ds=0.125; p.nc.tol=1e-5; p.sw.bifcheck=0;
```

```
p=cont(p,2); % 2 steps without PC, and with bifcheck=0,
p.nc.nq=2; p.nc.ilam=[3 1 6]; p.fuha.qf=@qfArot; p.fuha.qfder=@qjacArot;
p.sw.bifcheck=1; p.nc.tol=1e-8;  p.sw.jac=0; p=cont(p,20);
```

Listing 10: From nodDBC/`cmds1.m`; branch switching at double BP, and continuation with rotational PC.

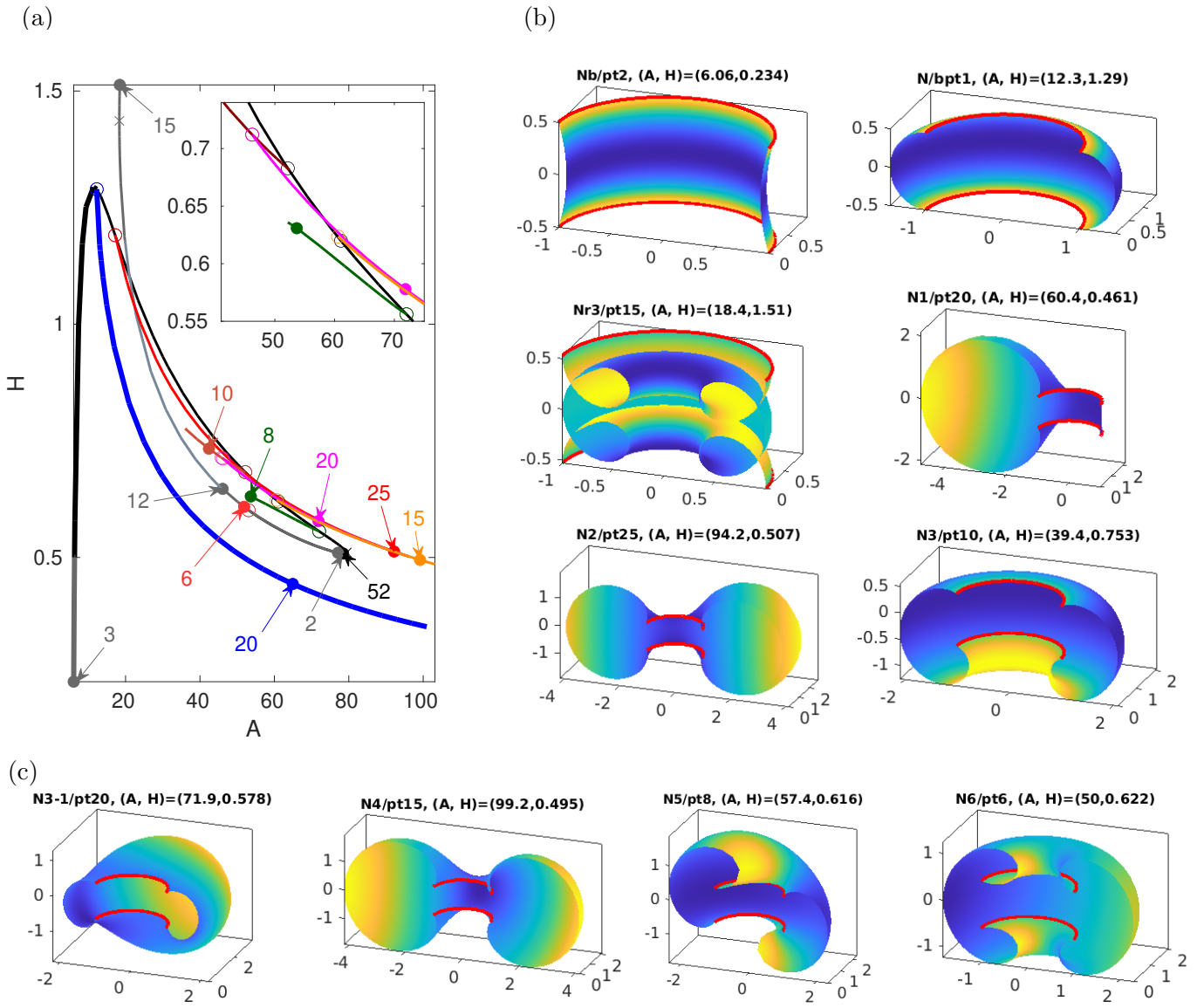

Figure 12: Bifurcation diagram of (mostly) embedded nodoids (a), with samples in (b,c) cut open at the x–z plane ($y = 0$). Branches `N` (black), `Nb` (grey), `N1` (blue), `N2` (red), `N4` (orange), `N5` (green), `N6` (light blue), `N3-1` (magenta), and `Nr1`, `Nr2` and `Nr3` ("restarts" of `N`, grey). See text for details, and Fig. 13 for plots of `N/pt52`, `Nr1/pt2`, and `Nr2/pt12`.

Figure 12 shows results from `cmds1.m` (see also the movie `nodDBCs.avi` from [MU24] to go step by step through the bifurcation diagram). We start at the cylinder and first continue to larger A (black branch `N`). The first BP at $(A, H) \approx (12.24, 1.29)$ is double with angular wave number $m = 1$. We simply select one of the kernel vectors to bifurcate, and do two steps without PC (blue branch `N1`, lines 11-13 of Listing 10). Then we switch on the rotational PC in line 14 and continue further. As predicted, BP1 occurs when X meets the lower and upper boundary circles horizontally, and the stability changes from `N` to `N1`.[14] The second BP yields the $m = 2$ branch `N2` (red). These results fully

[14] `N` up to `BP1`, `Nb`, and `N1` are the only stable (in the sense of VPMCF) branches in Fig. 12, and hence physically most relevant; the further branches we compute are all unstable, and hence of rather mathematical than physical interest.

agree with those from [Bru18]. The branch `Nb` (grey, with `pt3`) is the continuation of `N` to smaller A (and V), where the cylinder curves inward.

The third BP on `N` is simple with $z \mapsto -z$ symmetry breaking, yielding branch `N3` (brown). On `N3` there are secondary bifurcations, and following the first we obtain `N3-1` (magenta). The 4th BP on `N` again has $m = 2$ but is different from the 2nd BP on `N` as the nodoid has already "curved in" at the boundary circles, which is inherited by the bifurcating branch `N4` (orange). The 5th BP on `N` yields a skewed $m = 2$ nodoid `N5` (green).[15] After the fold, the mesh in `N` becomes bad at the necks, see `N/pt52` in Fig. 13. Thus, for accurate continuation we use (54) to remesh, see `Nr1/pt2` and Remark 3.6(a) and Fig. 13(a–c), yielding the branch `Nr1` (grey) in Fig. 12(a). `Nr1/pt12` in Fig. 13 shows that as after a number of steps the nodoid bulges further in, the mesh at the neck deteriorates again, and so we remesh again to `Nr2` (light grey). The nodoid then self–intersects at $(A, H) \approx (22.9, 1.05)$, and at `Nr2/pt10` we do the next restart to `Nr3`. Using such remeshing we can continue the branch `N` (as `Nr1, Nr2, Nr3, ...`) to many loops and self–intersections, with many further BPs as predicted in Lemma 3.4. In any case, although by branch switching from `Nr1/bpt1` instead of from `N/bpt6` we use a somewhat adapted mesh to compute branch `N6` (red), we only compute a rather short segment of `N6` because on `N6` we quickly run into bad meshes again. See also §3.3.4 for further comments/experiments on the meshing of nodoids. In Fig. 13(d) we illustrate the correspondence of our numerical results for the continuation in A to Theorem 3.5, see Remark 3.6(b).

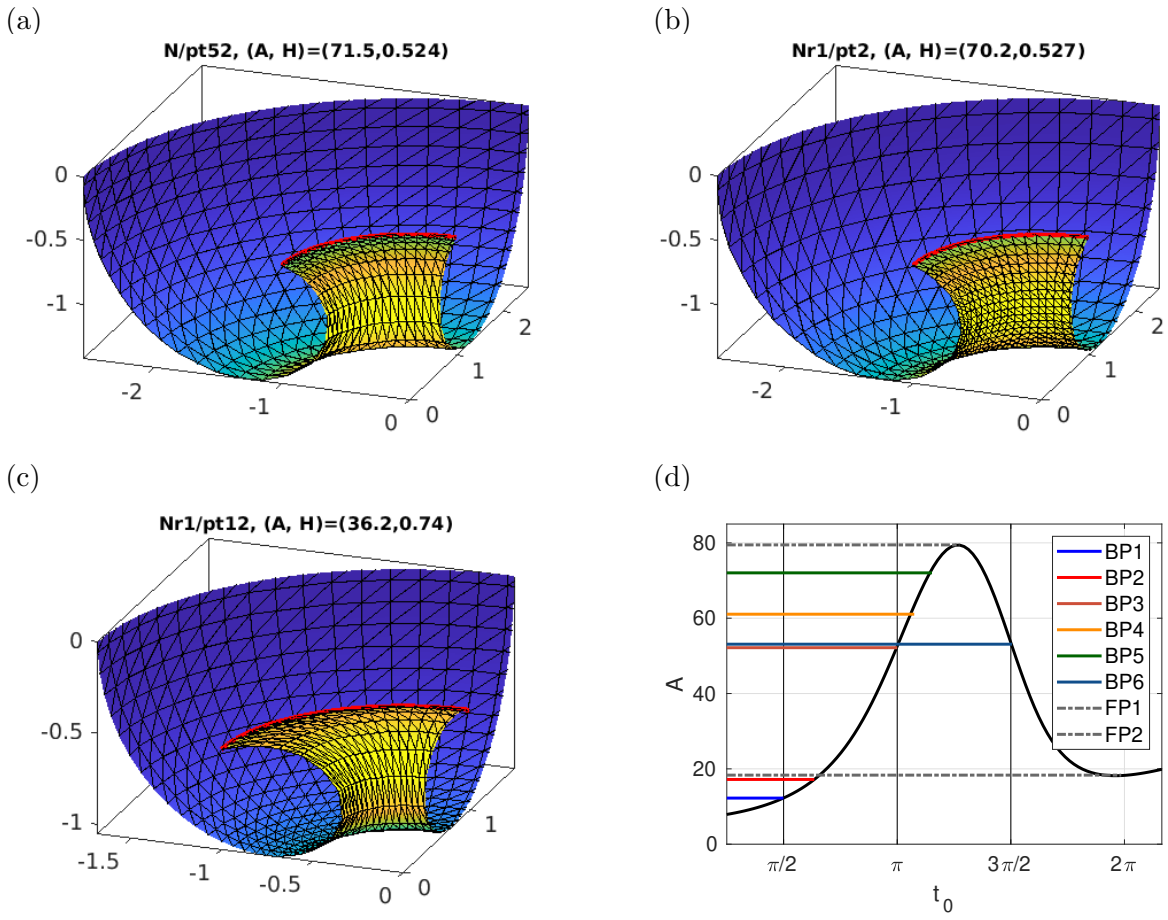

Figure 13: Continuation of Fig.12; (a–c) (1/8th of) solutions on `N` before and after remeshing. (d) Comparison to analytical results, see Rem. 3.6(b).

Remark 3.6 a) For axi– and Z_2 symmetric nodoids, we can easily extract $a{=}(2HR{-}1)^2 - 1$ from our numerical data, with R the radius on the $z{=}0$ plane. We can then numerically solve the second

[15] BP5 is an example of a BP qualitatively predicted in [KPP17, Prop.3.9] at large t_0.

equation in (55), i.e., $1 = r^* = \dfrac{\cos t_0 + \sqrt{\cos^2 t_0 + a}}{2|H|}$ for t_0, and use this for restarts with a new mesh, for instance from `N/pt52` to `Nr1/pt1` in Fig. 13.

b) Similarly, given $r^* = 1$ and $l = 0.5$, we can solve (56) for a and t_0 in a continuation process, see `cmds1A2t.m`. Then computing $A = A(a, t_0)$ gives the black curve in Fig. 13(d), and intersecting the A values of our numerical BPs gives the t_0 values for BP1, BP3 and BP6 as predicted, and explains the folds FP1 and FP2. In summary, the BPs on `N`, their multiplicities, and their relation to Theorem 3.5 (if applicable) are

BP number	BP1	BP2	BP3	BP4	BP5	BP6
multiplicity	2	2	1	2	2	2
Theorem 3.5	1.	NA	2.	NA	NA	1.
t_0	$\pi/2$	1.995	π	3.377	3.622	$3\pi/2$

(60)

where NA means not applicable, and where for BP1, BP3 and BP6 we give the exact values, with as indicated in Fig. 12(c) very good agreement of the numerics.[16] ⌋

3.3.4 Long nodoids

In `nodDBC/cmds2a.m` and Fig. 14 (see also `nodDBCs.avi` from [MU24]) we consider "long" nodoids with self–intersections.[17] As a slightly more explicit alternative to (54), in `nodinitl.m` we now parameterize an initial axisymmetric nodoid $\tilde{\mathcal{N}}_{r,R}$ following [Mla02] by

$$X : [-\pi/2, \pi/2] \times [0, 2\pi] \to \mathbb{R}^3, \qquad (x, \varphi) \mapsto \begin{pmatrix} \frac{r}{\delta(x,k)} \cos(\varphi) \\ \frac{r}{\delta(x,k)} \sin(\varphi) \\ RE(x,k) - rF(x,k) - Rk^2 \frac{\sin(x)\cos(x)}{\delta(x,k)} \end{pmatrix}, \qquad (61)$$

where r and R are the neck and the buckle radius, F and E are the elliptic integrals of the first and second kind, $k = \sqrt{(R^2 - r^2)/R^2}$, and $\delta(x, k) = \sqrt{1 - k^2 \sin(x)}$. It turns out that here we again need to be careful with the meshes, and besides adaptive mesh refinement we also use suitable initial meshes. We discretize the box $[-\pi/2, \pi/2] \times [0, 2\pi]$ (pre-image in (61)) by Chebychev nodes in x and equidistant nodes in y. This is implemented in a slight modification of `stanpdeo2D.m` in the current directory, and adapted to the parametrization (61), which "contracts" the mesh for the loop in the middle.

As we continue the axisymmetric branch `lNA` (black) to larger A, the inner loop "contracts and moves out", cf. `lNA/pt2` vs `lNA/pt20` in Fig. 14(b). Along the way we find several BPs, the first yielding a branch `lNA1` (blue) with broken $z \mapsto -z$ symmetry. The next three BPs yield branches with angular wave numbers m=2, m=1, and m=3. As we continue these branches to larger A, the mesh quality deteriorates due to very acute triangles where the inner loop strongly contracts. This suggests coarsening by removing degenerate triangles, which we exemplarily discuss in Fig. 15, see also the end of `cmds2.m`. The red line in (a) shows the mesh–distortion along `lNA2`, and (b) shows a zoom of `lNA2/pt10`; the very acute triangles on the inner loop ($\delta_{\text{mesh}} \approx 400$ at `pt10`) lead to stepsize reduction and eventual continuation failure. The brown line in (a) and the samples in (c,d) show results from the `degcoarsenX–cont` loop

```
for i=1:6; p=degcoarsenX(p,sigc,nit,keepbd); p=cont(p,4); end;
```

with `sigc`=0.5, `it`=6, `keepbd`=1 (cf. footnote 10) starting at `lNA2/pt3`. The distortion stays smaller (with $\delta_{\text{mesh}} \approx 50$ actually at the boundary), and the continuation runs faster and more robustly (larger

[16] This also holds for further BPs and folds, but we refrain from plotting these in the already cluttered BD in Fig. 12.
[17] All of the branches from Fig. 14 are unstable, i.e., Footnote 14 applies more strongly.

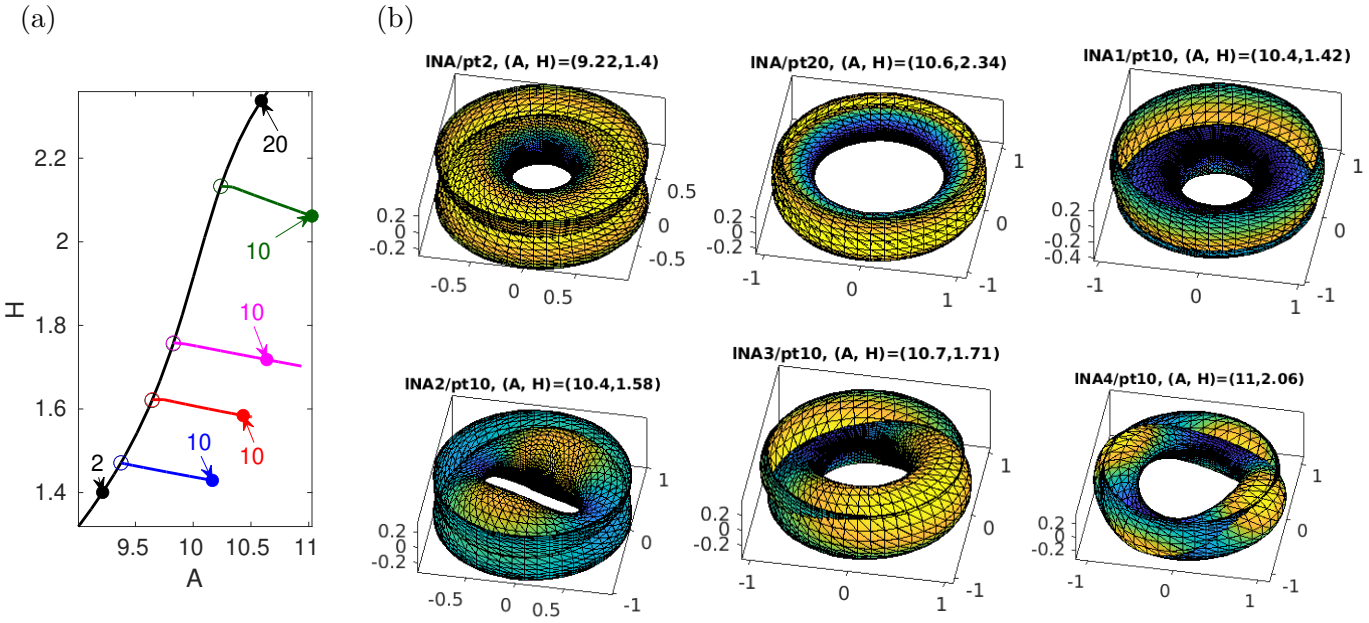

Figure 14: (a) Bifurcation diagram of self–intersecting nodoids; branch `lNA` (black) starts near $(A, H) = (9, 1.3)$ and shows four BPs to `lNA1` (blue, broken z–symmetry), `lNA2` (red, $m = 2$), `lN3A` (magenta, $m = 1$), and `lNA4` (green, $m = 4$). Samples in (b).

stepsizes feasible) than on the original mesh. The magenta line is from setting `p.fuha.ufu=@coarsufu` which adaptively coarsens (cf. `refufu.m` for refinement in Fig. 7) when δ_{mesh} exceeds 100. Both here yield quite similar results, and naturally, similar use of `degcoarsenX` is also useful for the other nodoids from Fig. 14, and for those from Fig. 12, in addition to the very specific remeshing used there, which is only possible because of the explicit formulas. Nevertheless, we remark again that the parameters for `degcoarsenX` need trial and error for robustness and efficiency.

3.4 Nodoids with pBCs in z

In [MP02], bifurcations of axisymmetric to non–axisymmetric nodoids are studied with the period (the "height") along the axis of revolution (wlog the z–axis) as the continuation/bifurcation parameter. This uses a different parametrization of the nodoids than (54) or (61), which we do not review here, as we shall again use (54) for the initialization. For fixed $H = 1$, [MP02] proves that there is a $r_0 > 0$ such that for neck radii $r > r_0$ ($r < r_0$) there are (are not) bifurcations from nodoids, and gives detailed asymptotics of bifurcation points in a regime ($\tau \to -\infty$ in [MP02]) which corresponds to $(R - r)/R \to 0$ with outer radius R, see below. In particular, the 2nd variation of the area functional around a given nodoid $\mathcal{N}_\tau$ is analyzed with $z \in \mathbb{R}$, i.e., for the full non–compact nodoid, not just for one period cell. This proceeds by Bloch wave analysis, and first establishes the band structure of the spectrum. Using a parametrization similar to (61), a detailed analysis of the second variation of the area functional, and ultimately two different numerical methods, [Ros05] shows that $r_0 = 1/2$, and the first bifurcation (i.e., at r_0) leads to non–axisymmetric nodoids with angular wave number $m = 2$ and same periodicity in z, i.e., Bloch wave number $\alpha = 0$ in [MP02].

Here we also consider periodic (in z) nodoids with fixed $H = 1$ using the height δ as continuation/bifurcation parameter. We recover the primary bifurcation at $r = r_0 = 1/2$ from [Ros05], and further bifurcations, see Figs. 16 and 17.

Remark 3.7 Similar to §3.3.2 we distinguish between "short" and "long" nodoids. Here, this merely corresponds to computing on one respectively two period cells in z, and the main distinction is as follows: All 1–periodic solutions are naturally n–periodic for any $n \in \mathbb{N}$. With respect to bifurcations, the 1–cell computations then correspond to Bloch wave numbers $\alpha = 0$ in [MP02]. For $n \geq 2$ periods cells we obtain further discrete Bloch wave numbers, e.g., additionally $\alpha = \pi$ for $n = 2$. This then

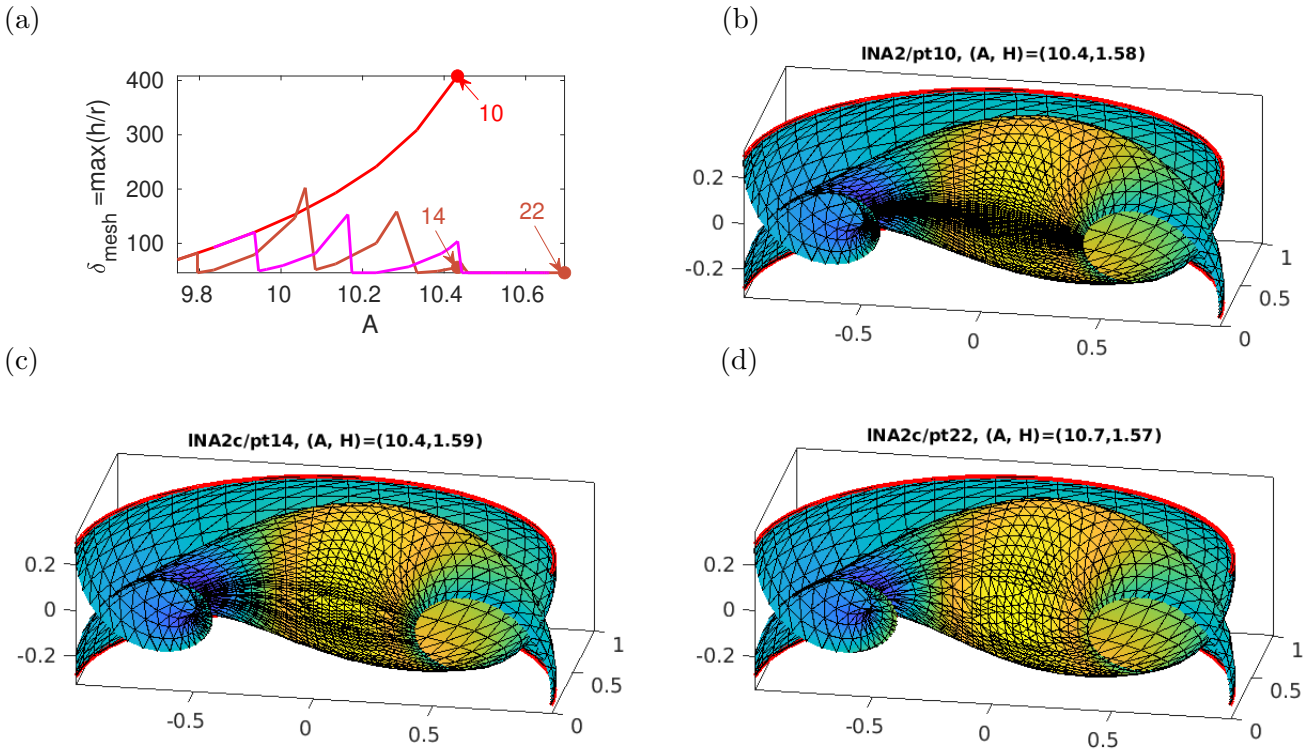

Figure 15: Results from the end of `cmds2.m`, example of a `degcoarsenX–cont` loop (lNA2c, brown), and adaptive coarsening (lNA2cc, magenta) for `lNA2`. (a) mesh quality $\delta_{\text{mesh}} = \max(h/r)$ over A, original `lNA2` in red. (b) original `lNA2/pt10` (cut open), n_p=3430; (c,d) samples from `lNA2c` with $n_p = 2744$ and $n_p = 2347$.

allows bifurcations which simultaneously break the S^1 and the Z_2 symmetry of the symmetric nodoid, and this is illustrated in Fig. 17, which only gives a basic impression of the extremely rich bifurcation picture to be expected when the computational domain is expanded further in z. To avoid clutter we refrain from putting the cases $n = 1$ and $n = 2$ in one figure. ⌋

Numerically, to set up "periodic boundary conditions in z", we proceed similar to the `pde2path` setup for periodic boundary conditions on fixed (preimage) domains, see [Uec21, §4.3]. The basic idea is to identify points on ∂X at $z = \pm\delta$. Thus, before the main step $X_0 \mapsto X_0 + uN_0$ for all our computations, we transfer the values of u from $\{X_3 = -\delta\}$ to $\{X_3 = \delta\}$ via a suitable "fill" matrix `p.mat.fill`, which has to be generated at initialization and regenerated after mesh–adaptation. The essential command is `box2per`, which calls `getPerOp` to create `p.mat.fill` (and `p.mat.drop` which is used to drop redundant variables), and which rearranges u by dropping the (redundant) nodal values at points which are filled by periodicity.

Similar to §3.3.2 we need a rotational PC for non–axisymmetric branches, but here for all computations we additionally need translational PCs in x, y and z directions, i.e. $S_i\vec{x} = \vec{x} + e_i$. These translations act infinitesimally in the tangent bundle as $S_iX_0 = \nabla_iX_0$, and hence the pertinent PCs are

$$q_i(u) = \langle \nabla_i X_0, X_0 + uN_0\rangle = \langle \nabla_i X_0, N_0\rangle\, u, \quad i = 1, 2, 3, \tag{62}$$

with derivatives $\partial_u q_i(u) = \langle \nabla_i X_0, N_0\rangle$. Like (59), they are implemented node–wise, and their derivatives are added to G with Lagrange multipliers s_x, s_y, s_z. Table 8 comments on the files used, and Listings 11–14 show the main new issues from the otherwise typical function and script files.

```
% pdeo etc only needed once, so here make dummy q, later discarded
q=p; q.pdeo=pde; q.pdeo.grid.p=q.X'; q.pdeo.grid.t=p.tri;
```

Table 8: Files in `pde2path/demos/geomtut/nodpBC`; similar to `nodDBC`, and we mainly comment on the differences.

`cmds1.m`	script for Fig. 16 (single period cell); experiments with mesh-adaptions in `cmds1b.m`.
`cmds2.m`	script for Fig. 17 (double period cell).
`lnodpBCmov.m`	script to make a movie of Fig. 17, see also [MU24].
`cmds2b.m`	experiments with trying to solve quadratic(cubic) bifurcation equations at BP2, and checking mode crossing at BP2 by varying H_0.
`nodbuckinit.m`	initialization, using parametrization (54), extracts initial height δ, and sets the "fill" and "drop" matrices for the pBCs.
`sGnodpBC.m`	rhs with pBCs; here using numerical Jacobians.
`qf.m, qjac.m`	3 translational constraints (for S^1 symmetric branches), and derivative.
`qfrot.m`	`qf.m` extended by rotational phase condition, derivative in `qjacrot.m`.
`getN.m`	flips N, and additionally forces horizontal normals at bottom and top.
`getM.m`	mod of standard `getM` with subsequent `filltrafo` for reduction to active nodes.
`getMf.m`	renaming of standard `getM` to get M for the full X (no dropping of per. boundaries).
`getA.m`	small mod of standard `getA` to work on full X.
`mytitle.m`	helper function for plots, called in `pplot` since `p2pglob.tsw=5`.

```
p=box2per(q,3); % generate (initial) p.mat.fill and p.mat.drop
```

Listing 11: Initializing pBCs in `nodpBC/nodbuckinit.m` via `box2per`, which automatically generates the pertinent `drop` and `fill` matrices to deal with the pBCs in direction 3 (= z–direction).

```
function r=sGnodpBC(p,u)  % nodoids with pBCs
par=u(p.nu+1:end); H0=par(1); del=par(6); % height
sx=par(7); sy=par(8); sz=par(9); sphi=par(10);% Lag.for translations & rotation
u=u(1:p.nu); uf=p.mat.fill*u; N0=getN(p,p.X); X=p.X+uf.*N0; % fill, then as usual
```

Listing 12: Start of `nodpBC/sGnodpBC.m`, l4 fills u, and afterwards we proceed as before.

```
function q=qf(p,u) % constraints: 3 translations
u=u(1:p.nu); N0=getN(p,p.X); grX=grad(p.X,p.tri);
grXx=grX(1:p.nt,:); grXy=grX(p.nt+1:2*p.nt,:); grXz=grX(2*p.nt+1:3*p.nt,:);
I=c2P(p.X,p.tri); % I interpolates from triangle centers to points
grXx=I*grXx; grXy=I*grXy; grXz=I*grXz; % grXx, grXy, grXz now act on nodal u
dx=grXx*p.X; dx=dot(dx,N0,2); % x-translations
dy=grXy*p.X; dy=dot(dy,N0,2); % y-translations
dz=grXz*p.X; dz=dot(dz,N0,2); % z-translations
qx=(p.mat.fill'*dx)'*u; qy=(p.mat.fill'*dy)'*u; qz=(p.mat.fill'*dz)'*u;
```

Listing 13: `nodpBC/qf.m`, implementing the three translational PCs; the gradient matrices are computed on the full `p.X`, and their periodic parts are dropped for their actions on `u(1:p.nu)` (line 8). For non axi–symmetric nodoids we extend `qf` in `qfrot` by a rotational PC as an additional 4th line.

```
p2pglob.tsw=3; p2pglob.pbctol=1e-3; % weak tol to identify boundaries
V0=0; A0=0; h0=1; a=2.4; r=0.2; del=1; sx=0; sy=0; sz=0; sphi=0;
par=[h0; V0; A0; a; r; del; sx; sy; sz; sphi]; % del=height, 10=rot
lx=pi; ly=pi; sym=0; p=[]; ny=70; nx=30; %start with rather coarse mesh
p=nodbuckinit(p,lx,ly,nx,ny,par,sym); p=setfn(p,'bN'); pplot(p); p.sw.Xcont=1;
%% 1st step
p.nc.ilam=[6 7 8 9]; p.nc.nq=3; % 3 translational constraints
p.fuha.qf=@qf; p.fuha.qfder=@qjac; p.sw.qjac=1; p.sol.ds=-0.01; p=cont(p,1);
%% refine initial mesh (twice), in particular at boundary
sig=0.2; p=loadpp('bN','pt1'); p.sw.rlong=1; p.sw.nobdref=0;
p=refineX(p,sig); sig=0.25; p=refineX(p,sig); p=retrigX(p);
```

Listing 14: Start of `nodpBC/cmds1.m`. Initialization, first step, and initial mesh refinements.

Fig. 16 shows some results from `cmds1.m`. For robustness (essentially due to the strong contractions at the inner loops later in the branches) it turns out to be useful to initialize with a rather coarse

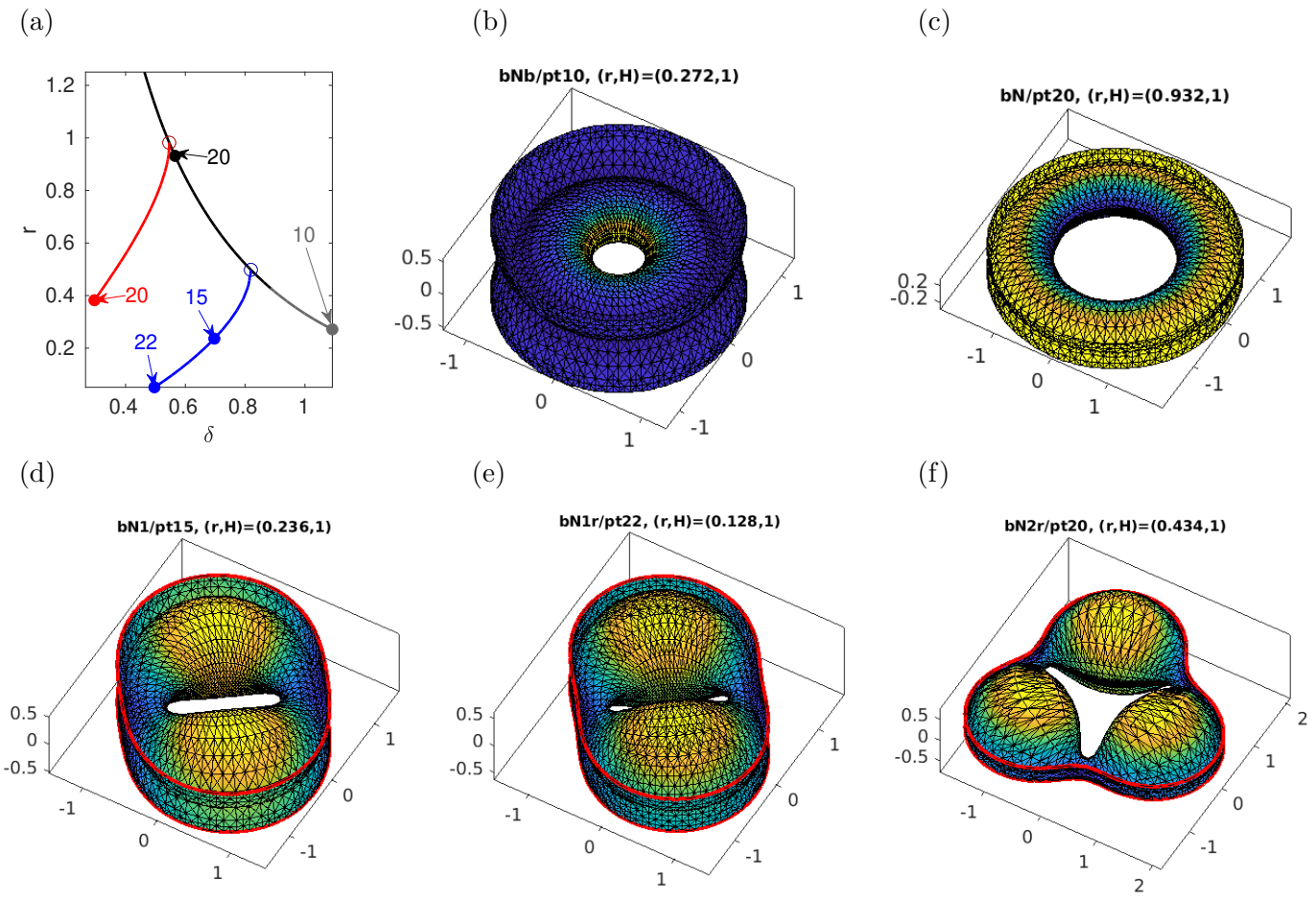

Figure 16: (a) Bifurcation diagram of nodoids parametrized by height δ, fixed $H = 1$. The axisymmetric branch `bN` (black) starts near $\delta = 0.88$ via (54), and in direction of decreasing δ shows a sequence of BPs to nodoids with broken S^1 symmetry, here `bN1` (blue, $m = 2$) and `bN2` (red, $m = 3$). Samples in (b–f), with `bN1r` and `bN2r` after some refinement.

mesh and after 1 or 2 steps refine by area. As we then decrease δ from the initial $\delta \approx 0.88$, we find the first BP at $\delta \approx 0.82$ and with $r = 0.5$, corroborating [Ros05], to the angular wave number $m = 2$ branch `bN1`. Using suitable mesh refinement along the way we can continue `bN1` to small δ, where in particular we have multiple self–intersections; first, the inner loops extend the "height" δ for $\delta < \delta_0 \approx 0.78$, and second the inner loops intersect in the plane $z = 0$ for $\delta < \delta_1 \approx 0.43$ (not shown), making the inner radius $r = 0$ (or rather undefined). The branch `bN2` from the next BP at $\delta \approx 0.54$ has $m = 3$, and otherwise behaves like the $m = 2$ branch. All these branches are rather strongly unstable, with $\mathrm{ind}(X) > 4$, and Footnotes 14 and 17 again apply.

As indicated in Remark 3.7, the branching behavior of the periodic nodoids very much depends on which period cell in z we prescribe, with Fig. 16 corresponding to one cell. To illustrate the richness that can be expected for larger cells, in `cmds2.m` and Fig. 17 we consider twice the minimal cell, see also `nodpBCl.avi` from [MU24]. This yields the same primary nodoid branch, and as a subset of bifurcations the bifurcations from Fig. 16, with two stacked copies of the solutions from Fig. 16 along all these branches. Additionally we have a new BP2 around $\delta = 1.44$, with small eigenvalues $\mu_{1,2} \approx 0.0003$ and $\mu_{3,4} \approx 0.004$, and the next eigenvalues are $\mu_5 \approx -0.67$ (simple) and $\mu_{6,7} \approx 0.87$. The (approximate) kernel vectors associated to $\mu_{1,3}$ are ϕ_1, ϕ_3 given in Fig. 17(d,e), and additionally we have $\phi_2 = R_{\pi/2}\phi_1$ and $\phi_4 = R_{\pi/2}\phi_1$ (rotation around the z axis). From the 4 small eigenvalues separated from the rest of the spectrum we might guess that BP2 is fourfold, which should have important consequences for the branching behavior at BP2, in the sense of "mixed modes", see [Uec21, §2.5.4]. However, ϕ_1 and ϕ_3 do not seem related by any symmetry, and, moreover, using `qswibra` and `cswibra` (see `cmds2b.m`) to search for solutions of the algebraic bifurcation equations at

BP2 only yields the "pure" modes ϕ_1 and ϕ_3 (and their rotations). Thus we conclude that BP2 is *not fourfold*, but corresponds to two double BPs close together.[18]

To further corroborate this, in `cmds2b.m` we compute the "`lN`" branches for $H_0 = 0.8$ and $H_0 = 1.1$. In both cases we find a similar spectral picture at the pertinent BPs as at BP2 for $H_0 = 1$ (4 small eigenvalues, well separated from the rest of the spectrum, with kernel vectors similar to Fig. 17(d,e)), but the two pertinent pairs are themselves more clearly separated, and ϕ_1, ϕ_3 flip order between $H_0 = 0.8$ and $H_0 = 1.1$. If BP2 at $H_0 = 1$ was fourfold, then we would expect this to be due to symmetry, and hence to also hold for $H_0 \neq 1$. That this is not the case suggests that near $H_0 = 1$ we rather have a "mode crossing" at BP2, and moreover do not expect bifurcating branches of "mixed modes".

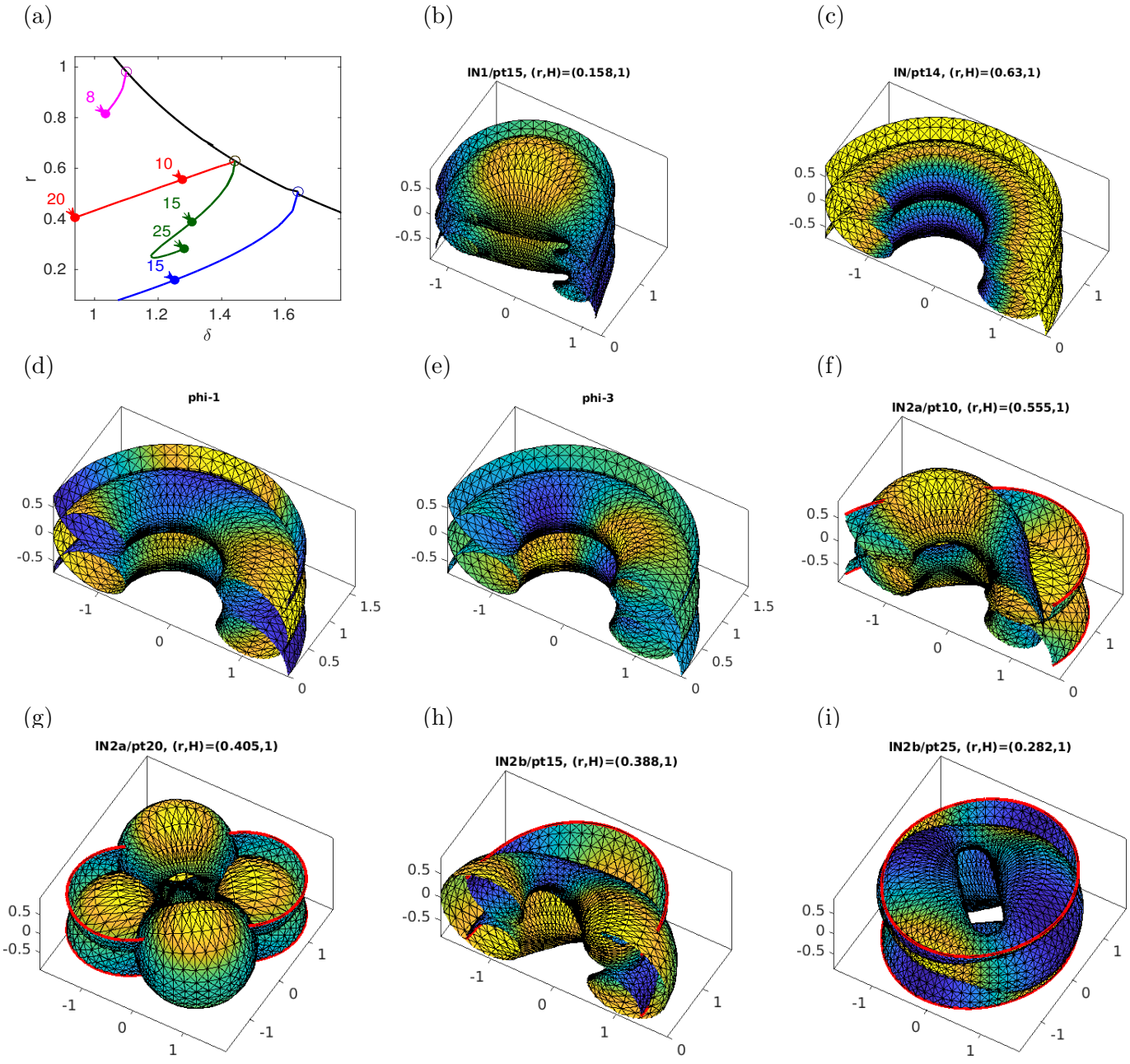

Figure 17: (a) Branching from the S^1 nodoid from Fig. 16 on twice the minimal cell, see text for details.

Therefore, in `cmds2.m` we simply do a branch switching in direction of the modes ϕ_1 (see Listing

[18] See also §3.5.2, where we discuss the opposite effect in more detail: There, an analytically double (due to obvious symmetry) zero eigenvalue is split up, with a larger split there than between $\mu_{1,2}$ and $\mu_{3,4}$ here.

15) and ϕ_3 separately, and obtain the branches `lN2a` (red) and `lN2b` (green).[19] In detail, we do an initial step in direction ϕ_1 (resp. ϕ_3), and then switch on the rotational PC as before. The branch `lN2a` continues to small δ without problems. The branch `lN2b` folds back near $\delta = 1.18$, and after `pt25` continuation fails, due to mesh degeneration in the inner bends. This can be fixed to some extent by careful mesh adaptation (yielding later continuation failure), but we do not elaborate on this here as Fig. 17 is mostly intended as an illustration of the rich bifurcation behavior over larger period cells.

```
%% 2nd BP possibly double, but algebraic bifurcation equations (ABEs) only yield
% pure modes (see cmds2b). Hence, here switch off ABEs for speed.
% For small ds, two separate BPs may also be detected; but treating them
% via the qswibra-trick should always work
aux.besw=0; % switch off ABE computations, just compute kernel
aux.mu2=0.01; aux.m=4; aux.ali=[]; p1=qswibra('lN','bpt2',aux);
%% go on 1st
p=gentau(p1,1,'lN2a',2); p.sol.ds=0.1; p.nc.tol=1e-4; p.sw.bifcheck=0;
p=cont(p,2); % two steps without rotational PC; then switch on and cont further
p.nc.nq=4; p.nc.ilam=[6 7 8 9 10]; p.fuha.qf=@qfrot; p.fuha.qfder=@qjacrot;
p.file.count=p.file.count+1; p.nc.tol=1e-4; p.sol.ds=-0.01; p=cont(p,15);
```

Listing 15: Selection from `nodpBC/cmds2.m`. Branch switching from BP2 via `qswibra` and `gentau`.

3.5 Triply periodic surfaces

Triply periodic surfaces (TPS) are CMC surfaces in $\mathbb{R}^3$ which are periodic wrt three independent (often but not always orthogonal) directions. Triply periodic *minimal* surfaces (TPMS) (this implicitly also means embedded, sometimes abbreviated as TPEMS) have been studied since H.A. Schwarz in the 19th century, and have found renewed interest partly due to the discovery of new TPMS by A. Schoen in the 1970ies, and due to important (partly speculative) applications of TPMS (and their non–zero H TPS companions) in crystallography, mechanics and biology. See for instance [AHLL88] and [STFH06], and [Bra23] for a long list of TPMS.

From the PDE point of view, TPS solve (3) with periodic BCs on a bounding box. Some families of TPMS were studied as bifurcation problems in [KPS18], using a cell length (period) in one direction as continuation/bifurcation parameter, and combined with numerical results from [ES18]. Much of the theory of TPMS is based on Enneper–Weierstraß representations. See Remark 3.9, where we relate some of our numerical results for the Schwarz P surface family to results from [KPS18] obtained via Enneper–Weierstraß representations. A way to *approximate* TPS is as zeros of Fourier expansions of the form

$$F(\vec{r}) = \sum_{k\in\mathbb{Z}^3, |k|\le N} F(k)\cos(2\pi k\vec{r} - \alpha(\vec{r})).$$

A simple first order approximation of the Schwarz P surface (cf. Fig.1(b)) is

$$\text{Schwarz P surface } \approx \{(x,y,z)\in\mathbb{R}^3 : \cos(x)+\cos(y)+\cos(z)=0\}, \tag{63}$$

Better approximations with some higher order terms are known, also for many other "standard" TPS, see, e.g., [GBMK01] for a quantitative evaluation of such approximations. In the demo `TPS` we focus on the Schwarz P family, and some CMC companions.[20]

[19] This trick can also be summarized as follows: We do not localize each of the close–together BPs near BP2, which would require very small (arclength) stepsizes `ds`, and possibly many bisections. Instead, we just approximately localize some BP* near BP2, subsequently compute the (approximate) kernel at BP* using `qswibra`, and then select a kernel vector and try branch–switching in that direction. This "usually" works (in particular it works here), and is in particular useful if many BPs (including BPs of higher multiplicity) are close together. See also §3.5.2 for application of this trick

Table 9: Selected files in `TPMS/`, others (`getA.m, qf.m, qjac.m, updX.m`) are rather standard, with minor mods to account for the "filling" of the periodic boundaries.

`cmds1.m`	Schwarz P family, relation to Weierstrass representation in `cmdsaux.m`, Figs. 18 and 19.
`cmds2.m`	CMC companions of Schwarz P, Fig. 20
`Pinit.m`	Initialization, based on (63) and `distmesh`.
`getN.m`	mod of standard `getN`, applying corrections at the boundaries of X, see Remark 3.8.
`getperOpX.m`	here we also fill X; see also `Xfillmat.m`.

3.5.1 The Schwarz P minimal surface (family)

In `TPS/cmds1.m` we study continuation (and bifurcation) of the Schwarz P surface in the period δ in z–direction, focusing on one period cell, i.e., the box

$$B_\delta := [-\pi,\pi)^2 \times [-\delta/2,\delta/2). \tag{64}$$

To get an initial (approximate) X on $B_{2\pi}$, we use (63) and the mesh generator `distmesh` [PS04], on one eighth of $B_{2\pi}$, which we then mirror to $B_{2\pi}$. The continuation in δ proceeds similar to §3.4, by first scaling $X = S_\delta$`p.X` to period δ in z and then setting $X = X + uN$ and solving for (u,δ). Subsequently, the same scaling is applied in `updX` to set the new `p.X`. As in §3.4 we have translational invariance in x, y and z, and hence exactly the same PCs, implemented in `qf.m`, with derivatives in `qjac.m`.

Somewhat differently from §3.4 we now also "fill" X by taking the ∂X values from the left/bottom/front of the box to the right/top/back of the box. While u is stilled filled via $u =$ `p.mat.fill`$*u$, for filling X we compute matrices `p.Xfillx, p.Xfilly, p.Xfillz` (in `Pinit.m`, via `Xfillmat`, which calls `getPerOpX`) similar to `p.mat.fill`, but with -1 (instead of 1) where we want to transfer X values from one side of the box to the opposite side (assuming symmetry wrt the origin). Finally, it turns out that the continuation is slightly more robust if in `getN` we correct N at the boundaries to lie *in* the boundaries of B_δ, see Remark 3.8.

Figure 18 shows some results from `cmdsP.m`. Decreasing δ from 2π (`P/pt1` in (b) at $\delta = 6.2732$), X gets squashed in z direction, and at $\delta = \delta_1 \approx 5.9146$ we find a D_4 symmetry breaking pitchfork bifurcation (with the two directions corresponding to interchanging the x and y axis wrt shrinking and expansion) to a branch `P1`, which then extends to large δ. On the other hand, increasing δ from 2π (branch `pB`, grey), we find a fold on the P branch at $\delta = \delta_f \approx 6.408$. Both δ values agree well with results from [KPS18] based on the Enneper–Weierstrass representation, summarized in Fig. 18(h), see Remark 3.9.

Remark 3.8 a) The results from Fig. 18 can also be obtained by choosing "Neumann" BCs on ∂B_δ. However, for other TPMS we need the pBCs. For instance, we can also continue the H surface family on a suitable (almost minimal) rectangular box, where however solutions fulfill pBCs but not Neumann BCs. Due to the necessary larger period cell, and due to branch points of higher multiplicity, the numerics for the H family are more elaborate, and these results will be presented elsewhere.

b) In fact, in the local copy `TPS/getN.m` we apply a trick and zero out N_1 at $x = \pm\pi$, N_2 at $y = \pm\pi$ and N_3 at $z = \pm\delta/2$. Thus, N is forced to always lie in the cube's faces, yielding a "combination of NBCs and pBCs" in the sense that the trick forces X to meet the cube's faces orthogonally, while the pBCs keep X on opposite faces together. However, the trick is for convenience as without it we get the same branches but in a less robust way, i.e., requiring finer discretizations and smaller continuation stepsizes. ⌋

in a different setting, and, e.g., [Uec21, §10.1] for further discussion.

[20] The approximation (63), and higher order corrections, also arise from solving the amplitude equations for a Turing bifurcation on a simple cubic (SC) lattice, where hence the Schwarz P surface, or, depending on volume fractions a CMC companion of Schwarz P, occurs as the phase separator between "hot" and "cold" phases. See, e.g., [CK97] and [Uec21, §8.1,8.2], and similarly [WBD97] for the occurrence of Scherk's surface in 3D Turing patterns.

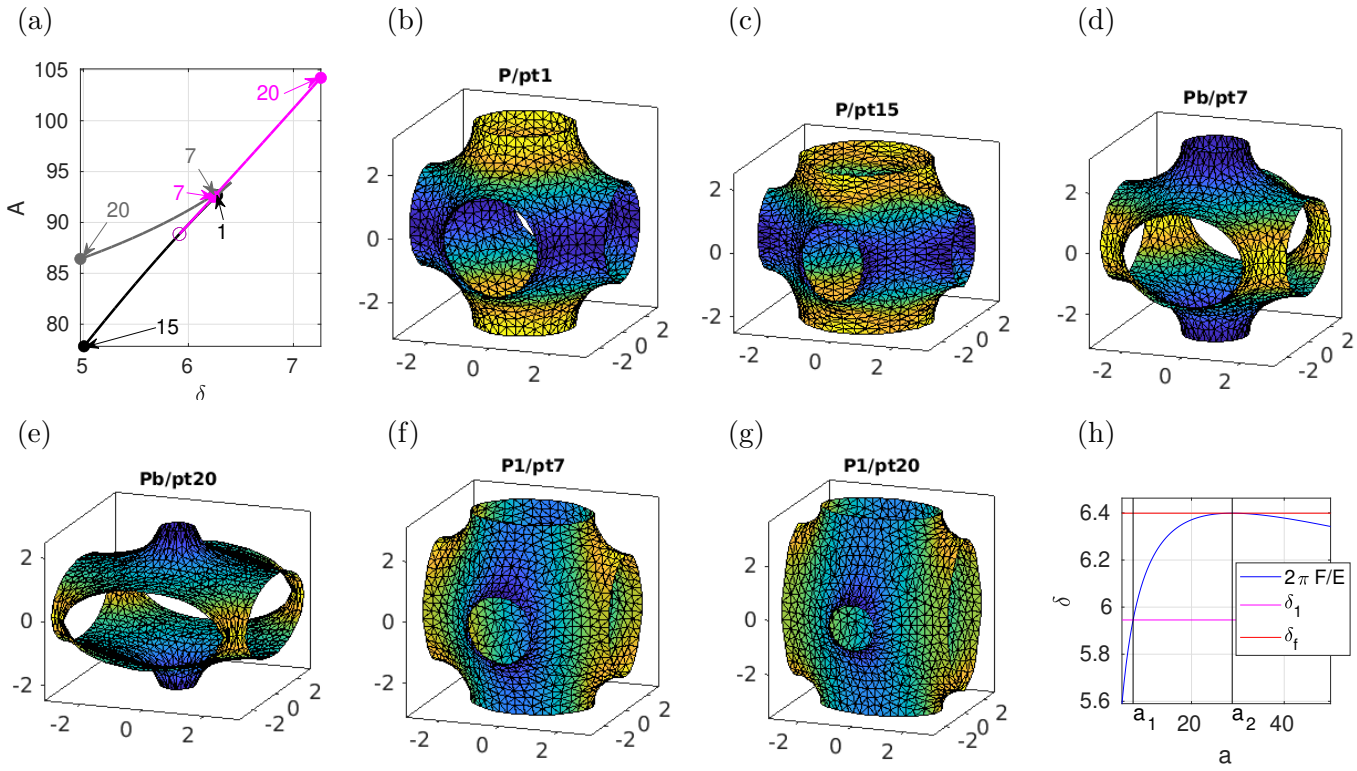

Figure 18: (a) Bifurcations in the Schwarz P family, black (P) and grey (Pb) branch; bifurcating magenta branch (P1) breaks D_4 symmetry. Samples in (b–g). Comparison with [KPS18] in (h), cf. Remark 3.9.

Remark 3.9 The Enneper–Weierstrass representation of a minimal surface is

$$\big(x, y, z\big) = \mathrm{Re}\left[\mathrm{e}^{\mathrm{i}\vartheta}\int_{p_0}^{p}(1-z^2,\, \mathrm{i}(1+z^2),\, 2z)R(z)\,\mathrm{d}z\right], \tag{65}$$

$p_0, p \in \mathcal{M}$ with $\mathcal{M}$ a Riemannian surface, where ϑ is called Bonnet angle, and $R : \mathcal{M} \to \mathbb{C}$ is called Weierstraß function. The Enneper surface E from §3.2.3 is given by the data $\mathcal{M} = D_\alpha$ (disk of radius α) and $R(z) \equiv 1$. For TPMS, R is a meromorphic function, and $\mathcal{M}$ consists of sheets connected at branch points given by poles of R. See, e.g., [Oss14, §8] for a very readable introduction to Weierstrass data and the connection of minimal surfaces and holomorphic functions. [Hof90] for a basic discussion of the Weierstrass data of TPMS, [Ros92] for identifying the Riemannian surface $\mathcal{M}$ for the Schwarz P surface with $S^2 \times S^2$ by stereographic projection, where S^2 is the unit sphere, and [FW14] for further examples for construction of TPMS from Weierstraß data.

Following [KPS18], we consider $\mathcal{M}$ a double cover of $\mathbb{C}$, and, for $a \in (2, \infty)$, let

$$R(z) = 1/\sqrt{z^8 + az^4 + 1}, \tag{66}$$

where the Schwarz P surface with period cell $[-\pi, \pi)^3$ is obtained for $\vartheta = 0$ and $a = 14$.[21] See also [GK00] for the explicit computation of a fundamental patch of Schwarz P based on (65) and (66) with $a = 14$ and a small planar preimage $\subset \mathbb{C}$.

In [KPS18], a is taken as a bifurcation parameter along the Schwarz P family with the *periods* for

[21] For $\vartheta = \pi/2$ we obtain the Schwarz D family, and for $\vartheta \approx 0.9073$ Schoen's gyroid, as two further TPMS. Moreover, since these have the same Jacobians as Schwarz P, all bifurcation results from Schwarz P carry over to Schwarz D and the gyroid, but these appear to be much more difficult to treat in our numerical setting.

Schwarz P given by [KPS18, §7.3]

$$E = 2\int_0^1 \frac{1-t^2}{\sqrt{t^8+at^4+1}}\,\mathrm{d}t + 4\int_0^1 \frac{\mathrm{d}t}{\sqrt{16t^4-16t^2+2+a}} \quad \text{(periods in } x \text{ and } y\text{)}, \tag{67}$$

$$F = 8\int_0^1 \frac{t}{\sqrt{t^8+at^4+1}}\,\mathrm{d}t \quad \text{(period in } z\text{)}, \tag{68}$$

up to homotheties (uniform scaling in all directions). We have $\delta = 2\pi F/E$ for our δ, and evaluating E, F numerically (or as elliptic integrals) and plotting $\delta(a) := 2\pi F/E$ as a function of a we get the blue curve in Fig. 18(h), which corresponds to [KPS18, Fig.13]. In particular, $\delta(a)$ has a maximum at $a = a_2 \approx 28.778$, and $\delta(a_2) = \delta_f$ agrees with our fold position in Fig. 18(a). On the other hand, with suitable mesh adaptation the branch P1 continues to at least $\delta = 10$. Next, [KPS18] based on [ES18] gives a bifurcation from the P family at $a = a_1 \approx 7.4028$, and again we find excellent agreement $\delta(a_1) = \delta_1$ with our BP at δ_1. ⌋

Remark 3.10 a) The fact that P does not extend to "large" δ (but folds back) has also been explained geometrically in [Hof90], without computation of the fold position.

b) The stability of Schwarz P (and hence also Schwarz D) on a minimal period cell and wrt *volume preserving* variations is shown in [Ros92]. However, "larger pieces" of P, e.g., P on $[-\pi,\pi)^2\times[-2\pi,2\pi)$ are *always* unstable, even wrt volume preserving variations. See also [Bra96, §8] for a useful discussion, and illustrations. Numerically, in Fig. 18 we find: $\mathrm{ind}(X) = 2$ except on the segment $\mathcal{S}$ of P (and Pb) between the fold and the BP at δ_1, where $\mathrm{ind}(X) = 1$. However, the most (and on $\mathcal{S}$ only) unstable eigenvector has a sign, see Fig. 19, and hence the solutions on $\mathcal{S}$ are stable wrt volume preserving variations. ⌋

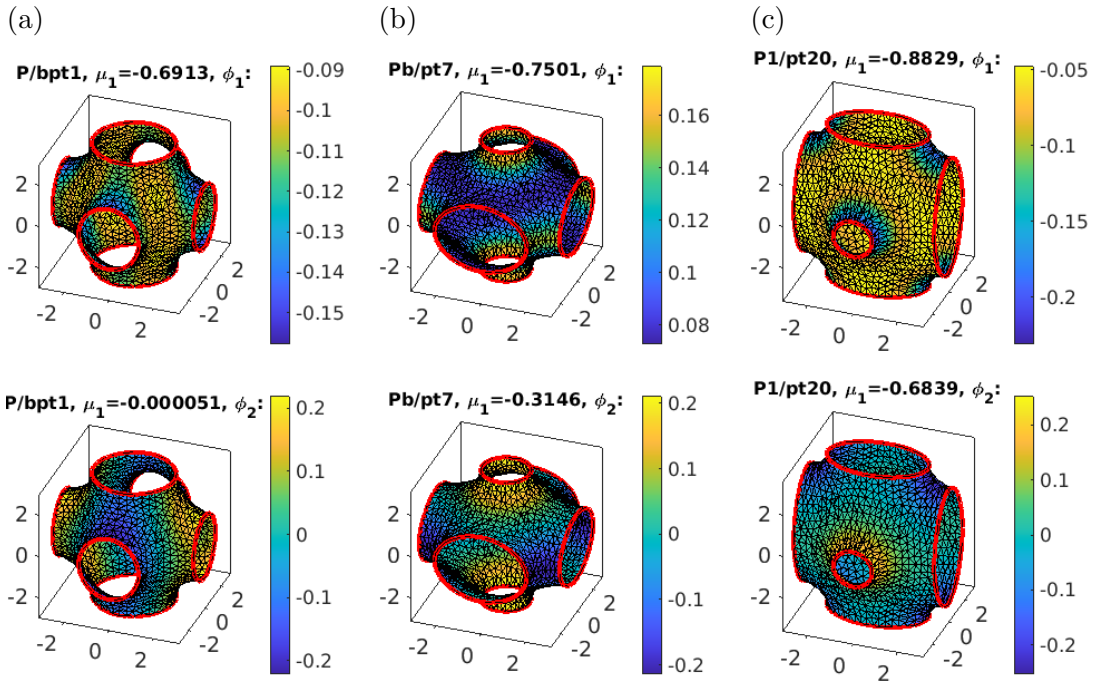

Figure 19: Selected eigenvectors at points as indicated, cf. Remark 3.9b). Top: the most unstable direction, which does not change sign. Bottom: the second eigenvector; in (a) this approximately spans the kernel.

3.5.2 CMC companions of Schwarz P

In `TPS/cmds2.m` we compute some CMC companions of the Schwarz P surface, see Fig. 20, where all we have to do is set `ilam=[1,5,6,7]` as H sits at position 1 in the parameter vector (and the

translational Lagrange multipliers at `[5,6,7]`). Continuing first to smaller H (black branch `PH`), X (the volume enclosed by X and the boundaries of the cube) "shrinks" and we find a BP at $H \approx -0.1$. In the other direction (grey branch `PHb`), X (the volume enclosed by X) "expands", with a BP at $H \approx 0.1$. The continuation of both these branches fails at $H \approx -0.3$ and $H \approx 0.3$ (respectively), though they can be continued slightly further with careful mesh adaptation.

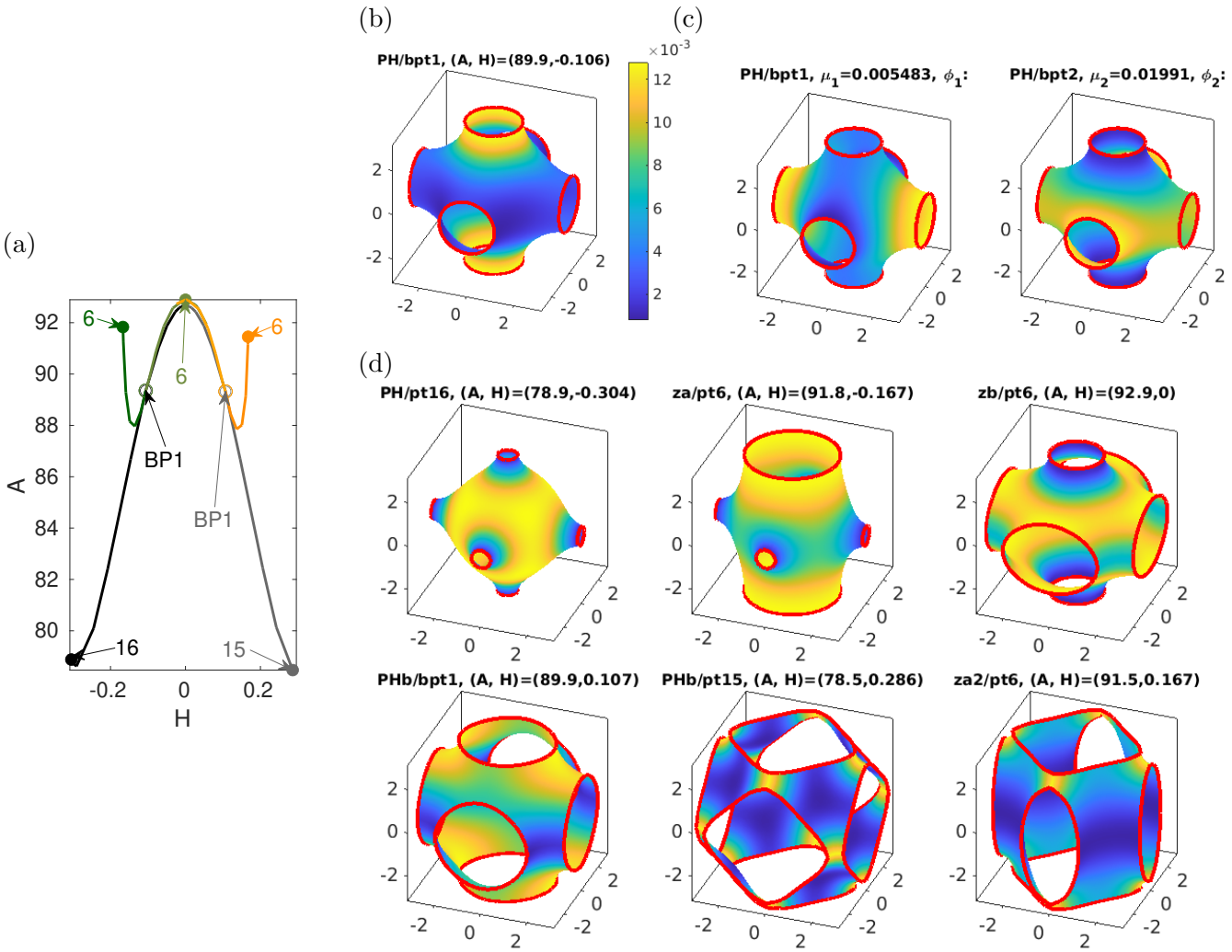

Figure 20: Some results from `TPS/cmds2.m`. Continuation of Schwarz P in H at fixed $\delta = 2\pi$. BD in (a): `PH` (black), `PHb` (grey), `za` (dark green) and `zb` (lighter green), and `za2` and `zb2` (orange), but altogether these are just two different branches. BP1 on `PH` and approximate kernel vectors in (b,c), and further samples in (d). See text for details.

Our main purpose here is to show how symmetry considerations and some tricks can help to avoid numerical pitfalls. By symmetry, the BP `PH/bpt1` (and similarly `PHb/bpt1`) must be double, although the smallest (in modulus) eigenvalues reported at `PH/bpt1` are $\mu_1 \approx 0.005$ and $\mu_2 = 0.02$.[22] See Fig. 20(b) for `PH/bpt1`, and (c) for the (approximate) kernel vectors ϕ_1, ϕ_2. In fact, the plot in (b) (stronger correction along the z–axis) shows that at least the last step in the localization of `PH/bpt1` violated the S_4 symmetry of the (now fixed) cube, which explains the rather significant splitting of the in principle double eigenvalue $\mu_1 = 0$. Clearly, we expect $\phi_{1,2}$ to approximate two bifurcation directions, with D_4 symmetry along the x axis (ϕ_1) and y axis (ϕ_2). By symmetry we then must have at least one more bifurcating branch, with D_4 symmetry along the z axis. To find this bifurcation direction, we can use `qswibra` with numerical derivation and solution of the algebraic bifurcation

[22] Additionally, there is a simple negative eigenvalue $\mu_0 \approx -0.7$, and the next two eigenvalues are $\mu_{3,4} \approx 0.5$, i.e., $\mu_{1,2}$ are well separated from the rest of the spectrum.

equation (ABE) [Uec21, §3.2.2]. However, this is expensive and not always reliable. Here, the three bifurcation directions (oriented along x, along y, and along z) *are* returned, but we have to relax the tolerance `isotol` for identifying solutions of the ABE as isolated. Alternatively, cf. also Footnote 19, we can use `qswibra` with `aux.besw=0` (bifurcation equation switch= 0) to let `qswibra` just compute and plot the (approximate) kernel ϕ_1, ϕ_2. This lets us guess to approximate the third direction as $\phi_3 = 0.2\phi_1 + \phi_2$. This turns out to be sufficiently accurate and gives the transcritical branch(es) `za` (dark green) and `zb` (other direction, lighter green).

On `za`, the continuation fails after `pt6`. `zb/pt6` is at $H = 0$ and corresponds to `Pb/pt7` from Fig. 18. Subsequently, `zb` continues to `PHb/bpt1`, and is indeed identical to the branch(es) `za2` (and `zb2`), transcritically bifurcating there. In particular, `PHb/bpt1` is again double, and we can compute the three branches oriented along x, y or z as above (see `cmds2.m`). `zb2` (light orange) then continues back to `PH/bpt1`, while `zb2` fails after `pt6` (last sample in (d)). The continuation failures of `za` and `za2` after `pt6` are due to poor meshes as the different boundaries of X come close to each other, like after `PH/pt16` and `PHb/pt15`, and it seems difficult to automatically adapt these meshes.

4 Fourth order biomembranes

The (dimensionless) Helfrich (or spontaneous curvature (SC)) functional [Hel73] is

$$E = \int_X (H - c_0)^2 + bK \,\mathrm{d}S, \tag{69}$$

where $c_0 \in \mathbb{R}$ is called spontaneous curvature, and $b \in \mathbb{R}$ is called saddle–splay modulus. The motivation of (69) are the shapes of closed vesicles with a lipid bilayer membrane, for instance red blood cells (RBCs), for which E is to be minimized under the constraints of fixed area $A(X) - A_0 = 0$ and enclosed volume $V(X) - V_0 = 0$. This motivated much work, e.g., [SBL90, Sei97, NT03, VDM08, OYT14], aiming to understand the various shapes of RBCs[23], mostly in the axisymmetric case. Applying our algorithms to closed vesicles (without a priori enforcing any symmetry) we recover many of the results from the above references. See also [LWM08, KIPM$^+$20] for further biological and mechanical background, [JQJZC98] for non–axisymmetric shapes (under different constraints), and [FVKG22] for the related problem of 1D radial wrinkling of arteries, with an additional restoring force due to the surrounding tissue, and a very rich bifurcation structure.

The Lagrangian for (69) is

$$F = \int_X (H - c_0)^2 + bK \,\mathrm{d}S + \lambda_1(A - A_0) + \lambda_2(V - V_0), \tag{70}$$

where λ_1 (corresponding to a surface tension [Lip14]) and λ_2 (corresponding to a pressure difference between outside and inside) are Lagrange multipliers for area and volume constraints. For closed X, the term $b\int_X K \,\mathrm{d}S$ in (69) can be dropped due to the Gauß–Bonnet theorem, cf. Footnote 1, as $\int_X K \,\mathrm{d}S = 2\pi\chi(X)$ is a topological constant, and the Euler-Lagrange equation for normal variations $X = X_0 + uN$ is

$$\Delta H + 2H(H^2 - K) + 2c_0K - 2c_0^2H - 2\lambda_1 H - \lambda_2 = 0. \tag{71}$$

If X is not closed, then often one or both of the constraints $A - A_0 = 0$ and $V - V_0 = 0$ is (are) dropped, and the associated Lagrange multipliers $\lambda_{1,2}$ are treated as external parameters, often with

[23] or, more down to earth, lipid bilayer membrane vesicles which develop upon injection of lipids into water, and which for instance can also organize into tubes; see also [SL95, §8] for a discussion of additional structures (networks of spectrin tetramers) on the membrane of RBCs

$\lambda_2 = 0$. If in the Gauss–Bonnet formula

$$\int_X K \,\mathrm{d}S = 2\pi\chi(X) - \int_{\partial X} \kappa_g \,\mathrm{d}s \tag{72}$$

we assume $\gamma = \partial X$ to be parameterized by arclength, then the geodesic curvature κ_g is the projection of the curvature vector $\gamma''(\vec{x})$ onto the tangent plane $T_{\vec{x}}(X)$, see, e.g., [Tap16, §4.3]. If we restrict to normal variations $\psi = uN$ which fix the boundary, i.e.,

$$u|_{\partial X} = 0, \tag{73}$$

then

$$\partial_\psi L = \int_X (\Delta H + 2H(H^2 - K) + 2c_0 K - 2Hc_0^2 - 2\lambda_1 H)u \,\mathrm{d}S + \int_{\partial X} (H - c_0 + b\kappa_n)\partial_n u \,\mathrm{d}s,$$

where $\kappa_n = \langle \gamma'', N\rangle$ is the normal curvature of $\gamma = \partial X$, i.e., the projection of the curvature vector onto the normal plane, see, e.g., [PP22] and the references therein. Thus we again obtain (71) (with $\lambda_2 = 0$), and additionally to (73) we can consider either of

$$\partial_n u = 0 \text{ on } \partial X \text{ (clamped BCs, or Neumann BCs)}, \tag{74}$$

$$H - c_0 + b\kappa_n = 0 \text{ on } \partial X \text{ (stress free BCs)}. \tag{75}$$

In the case of (74) we have $\int_{\partial X} \kappa_g \,\mathrm{d}s = 0$ in (72), and hence $\int bK \,\mathrm{d}S$ again becomes constant and can be dropped from (69).

Remark 4.1 a) With N the inner normal, the stability for (71) refers to the Helfrich flow (see, e.g., [KN06] for the existence theory near spheres)

$$\left\langle \dot{X}, N \right\rangle = -(\Delta H + 2H(H^2 - K) + 2c_0 K - 2c_0^2 H - 2\lambda_1 H), \tag{76}$$

with BCs (73) and (74) or (75) for non–closed vesicles.

b) Biological vesicles can undergo topological transitions which are important for their biological function, e.g., fission of a small bud from the vesicle, or fusion of two vesicles. We cannot capture such transitions in our setup of steady state continuation. Some examples of splitting in DNS for a phase field model are given in, e.g., [DLW06]; see also [BGBC22] and the references therein for a state of the art discussion of phase field modeling of vesicles.

c) In a certain continuum limit, and with different interpretations of the Lagrange multipliers $\lambda_{1,2}$, (71) can also be derived as the shape equation for carbon nanostructures, see [MDHV13].

d) Besides the Helfrich functional (69), a number of related models exist, for instance the so–called bilayer–coupling (BiC) model [SZ89],

$$E = \int_X H^2 \,\mathrm{d}S, \quad F = E + \mu_1(A - A_0) + \mu_2(V - V_0) + \mu_3(M - M_0), \tag{77}$$

where $M = \int_X H \,\mathrm{d}S$ is the integrated mean curvature, M_0 is an external parameter, $\mu_{1,2}$ are the Lagrange multipliers for the area and volume constraints, and μ_3 is the Lagrange multiplier for the constraint of fixed area difference between the outer and inner lipid monolayer, expressed via Taylor expansion around a virtual middle layer as $q = M - M_0 = 0$. By identifying $\mu_1 = \lambda + c_0^2$, $\mu_2 = \lambda_2$ and $\mu_3 = -c_0$ this yields the same shape equation (71) as (70), but the additional constraint $M - M_0 = 0$ drastically changes the phase diagram of minimizers for (77) compared to those for (69). In particular, for (77) non–axisymmetric minimizers of spherical topology are known to exist, but not for (69).

Another model is the so called area difference elasticity (ADE) model, where the area difference is not a hard constraint but added as an energy penalization, i.e.,

$$E = \int_X H^2 \,\mathrm{d}S + \frac{\alpha}{2}(M - M_0)^2, \quad F = E + \mu_1(A - A_0) + \mu_2(V - V_0), \tag{78}$$

$\alpha > 0$. This again allows stable non–axisymmetric minimizers which moreover compare well to some experimental results; see [WDS96, DEK+97] and [Sei99], including a discussion of the shortcomings of the SC model and the relations between the SC, BiC and ADE models.

Additionally, there are mechanochemical models which couple bending energies $E = \int_X (H - c_0)^2 \,\mathrm{d}S$ with a scalar morphogen on the surface which aggregates in regions of high mean curvature and which in turn increases c_0 [MMCRH13], or with for instance Brusselator type reaction–diffusion systems on the surface, where at least one species again increases c_0 [TN20]. Most of these models are not gradient systems and somewhat phenomenological, but easily lead to stable non–axisymmetric vesicle shapes, and also to persistent wave–like behavior. However, to the best of our knowledge the (numerical) study of these models so far was restricted to DNS. See also [ES13, BGN15] and the references therein for FEM discretizations of the dynamics of a variety of models, including the SC, the BiC and the ADE models, and, moreover considering the dynamics of vesicles in a fluid. ⌋

From the variety of models related to (69), here we opt for the 'classical' Helfrich SC model, while results including non–axisymmetric minimizers for the BiC model, and some bifurcation study for [MMCRH13] type models will be presented elsewhere. In §4.1 we present some basic results for closed topologically spherical vesicles, and §4.2 deals with biomembrane caps (topological disks) with stress–free BCs (75), while "bio-cylinders" with clamped BCs are relegated to App. B.

4.1 Closed Vesicles of spherical topology

We start with closed topologically spherical vesicles. Following [NT03] we set $\lambda_1 = -\tilde{\lambda}_1/2$ where $\tilde{\lambda}_1$ is the Lagrange multiplier for the area constraint, and write the shape equations (71) as

$$\Delta H + 2H(H^2 - K) + 2c_0 K - 2c_0^2 H + \lambda_1 H - \lambda_2 = 0, \tag{79a}$$

together with the volume and area constraints

$$q_1(X) = V(X) - V_0 = 0 \quad \text{and} \quad q_2(X) = A(X) - A_0 = 0. \tag{79b}$$

The bending energy $E = \int_X (H - c_0)^2 \,\mathrm{d}S$ is scaling $X \mapsto \gamma X$ invariant, and hence a useful dimensionless quantity to characterize solutions of (79) is the reduced volume

$$v = V/V_0, \tag{80}$$

where for given $A = 4\pi R_0^2$ (hence $R_0 = \sqrt{A/4\pi}$), $V_0 = 4\pi R_0/3$ is the volume of the equivalent sphere. At $v = 1$, the sphere is the only solution, and for decreasing v we may expect more and more solutions of various shapes.

A short review of previous work. The unit sphere is a solution of (79) if (for N the inner normal and hence $H \equiv 1$),

$$\lambda_1 = -2c_0 + 2c_0^2 + \lambda_2. \tag{81}$$

By [NT03, Thm3.1], bifurcations from the sphere occur at

$$\lambda_1 = n(n+1) - 4c_0 + 2c_0^2, \quad \lambda_2 = n(n+1) - 2c_0, \tag{82}$$

$n \geq 2$, with kernels of dimension $2n+1$ spanned by the spherical harmonics Y_{nm}, $m = -n, \dots, n$. Already from [Pet83] it is known that branches originating from spherical harmonics with $n \geq 3$ are *never* stable, at least near $v = 1$, while some of the branches bifurcating at the first BP with $n = 2$ contain stable solutions, again see also [NT03]. The bifurcation (unique branch modulo symmetry, see below) is trancritical (in λ_1), with one direction yielding *oblates* (oblate ellipsoids, disk like, turning into biconcave RBC shapes, see below for sample plots and more specific classifications), and the other direction yielding *prolates* (prolate ellipsoids). In particular, these are axisymmetric shapes.

An extensive largely numerical study of axisymmetric vesicles is given in [SBL90], including phase and energy diagrams, where our use of c_0 differs from the one in [SBL90] by a factor 1/2, i.e., our $c_0 = \frac{1}{2} c_{0,[\text{SBL90}]}$. In a nutshell, the results in [SBL90, Fig.8,10, 11,13,17] are:

- In the v–c_0 phase diagram [SBL90, Fig.10], there is a line $(0, 1] \ni v \mapsto c_0(v)$ decreasing in v and with $c_0(1) = -5/6$ such that near $v = 1$, either oblates (for spontaneous curvature $c_0 < c_0(v)$) or prolates ($c_0 > c_0(v)$) have minimal E.
- For decreasing v, the prolates lose stability to pears, and the oblates lose stability to stomatocytes. These transitions are discontinuous, i.e., occur via subcritical bifurcations, where the bifurcating branches (pears from prolates, stomatocytes from oblates) gain stability after one (or more) fold(s).
- Some regions in the v–c_0 phase diagram remain unstudied, but in particular for $v > 0.5$ and moderate $|c_0|$, say, there is strong evidence that all local minimizers of spherical topology of E are axially symmetric.

4.1.1 Our setup

Given the above results, here we mostly focus on the first BP ($n = 2$ in (82)) and axisymmetric solutions, and only compute a few secondary bifurcations from the axisymmetric branches and some bifurcations from the sphere with $n = 3$. In this, we fix three values of c_0, namely $c_0 = 0, c_0 = -1$, and $c_0 = 1.4$, then first continue in λ_1 along the spherical branch to prepare branch switching at the respective BPs from (82), and after branch–switching to non–spherical solutions continue in v, see Remark 4.2. The BDs are then plotted as E over v, and agree with [SBL90, Fig.8, 11, 13, 17] for the axisymmetric branches in the v–ranges we can reach. Additionally, our stability information is wrt general normal variations, not just axisymmetric ones.

Before we embark on this program we briefly comment on the numerical challenges and solutions to these. The basic setup again consists in setting $X = X_0 + uN_0$ (with here N the inner normal), and then writing (71) as a system of two second order equations for $u = (u_1, u_2)$, namely

$$G(u) := \begin{pmatrix} Lu_2 + Mf(u_1, u_2) \\ Mu_2 - H \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{83}$$

respectively $M_d \dot{u} = -G(u)$ for the Helfrich *flow* (76) with the dynamical mass matrix

$$M_d = \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix}, \tag{84}$$

and where as before M is the (Voronoi) mass matrix, L is the cotangent Laplacian, and

$$f(u_1, u_2) = 2u_2(u_2^2 - K) - 2\lambda_1 u_2 + 2c_0 K - 2c_0^2 u_2.$$

The mean curvature $H = H(u_1)$ is computed as $H = \frac{1}{2} \langle LX, N \rangle$, and the Gaussian curvature $K =$

$K(u_1)$ is obtained from `discrete_curvatures`, cf. (31). The reason for the reformulation of (79a) as two second order equations (83) is that this way we can easily implement the *two* BCs (73) and (74) or (75) when required.

For the closed vesicles, i.e., without any BCs, we always need the three linear translational PCs

$$q_i(X) := \int_{X_0} \langle u, N_i \rangle \, \mathrm{d}X \stackrel{!}{=} 0, \quad i = 1, 2, 3, \tag{85}$$

cf. (62), where N_i is the ith component of the (here inner) normal N of X_0. For (non–spherical) surfaces of revolution (axisymmetric branches), we need two rotational PCs (omitting the axis of revolution). For this, let $\vec{l_1} = (l_1, l_2, l_3)^T$ with $\|\vec{l_1}\| = 1$ be the rotational axis, which we find as

$$\vec{l_1} = X_{i_0} / \|X_{i_0}\| \tag{86}$$

with either $i_0 = \mathrm{argmax}_i \|X_i\|$ (prolates) or $i_0 = \mathrm{argmin}_i \|X_i\|$ (oblates), and take $\vec{l_1}, \vec{l_2}, \vec{l_3}$ with

$$\tilde{l_2} = \begin{pmatrix} -l_2 \\ l_1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ -l_3 \\ l_2 \end{pmatrix} + \begin{pmatrix} -l_3 \\ 0 \\ l_1 \end{pmatrix}, \quad \vec{l_2} = \frac{\tilde{l_2}}{\|\tilde{l_2}\|}, \text{ and } \vec{l_3} = \vec{l_1} \times \vec{l_2},$$

as an orthonormal basis of $\mathbb{R}^3$. Then the normal variations of rotations around $\vec{l_2}$ and $\vec{l_3}$ are spanned by $\left\{ \left\langle \vec{l_2} \times X, N \right\rangle, \left\langle \vec{l_3} \times X, N \right\rangle \right\}$, and the natural rotational PCs are

$$q_{3+i}(u) := \int_X \left\langle \vec{l}_{i+1} \times X, N \right\rangle u \, \mathrm{d}S \stackrel{!}{=} 0, \quad i = 1, 2. \tag{87}$$

For non–axisymmetric X we additionally use the third rotational PC $q_6 := \displaystyle\int_X \left\langle \vec{l_1} \times X, N \right\rangle u \, \mathrm{d}S \stackrel{!}{=} 0$, and we add $\eta_i \partial_u q_i(u)$ to the first component of G from (83), with Lagrange multipliers η_i. See Table 10 for a summary. Technically, after branch–switching from the sphere we first do two steps without rotational PCs. For axisymmetric solution branches we then detect the rotational axis via (86) and switch on the two rotational PCs around $\vec{l_2}, \vec{l_3}$. After a secondary bifurcation to a non–axisymmetric branch, or for primary bifurcations to non–axisymmetric branches from the trivial branch (present for $n \geq 3$ in (82)) we switch on the third rotational PC.

Table 10: Contraints and active parameters for different branch types; the parameters s_* and r_* are the Lagrange multipliers for the translational and rotational constraints, and stay $\mathcal{O}(10^{-6})$ during all continuations.

type	active parameters	constraints
trivial (sphere)	$\lambda_1, \lambda_2, s_x, s_y, s_z$	area A and volume V, 3 translational PCs
axisymmetric	$v, \lambda_1, \lambda_2, s_x, s_y, s_z, r_1, r_2$	A and V, 3 translational and 2 rotational PCs
non–axisymmetric	$v, \lambda_1, \lambda_2, s_x, s_y, s_z, r_1, r_2, r_3$	A and V, 3 translational and 3 rotational PCs

Remark 4.2 The eigenvalues for the linearization around a steady state are computed from the extended system

$$\begin{pmatrix} M_d & 0 \\ 0 & M_q \end{pmatrix} \partial_t V = - \begin{pmatrix} G_u(U) & G_w(U) \\ q_u(U) & q_w(U) \end{pmatrix} V, \tag{88}$$

where $U{=}(u,w)$ is the steady state including the active parameters w but without the primary active (genuine continuation) parameter, where $V{=}(v,z)$ is the considered perturbation, and where M_d and M_q are the pertinent dynamical mass matrices. For (83) we have $M_d = \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix}$ from the rewriting as a 2nd order system, and $M_q{=}0 \in \mathbb{R}^{6\times 6}$ (axisymmetric case) or $M_q{=}0 \in \mathbb{R}^{7\times 7}$ (non–axisymmetric case). Importantly, to obtain the correct stability information we cannot use one of the Lagrange multipliers as primary active parameter, because the Lagrange multipliers are in general not fixed for the flow, and hence we use V for the continuation of the nontrivial branches. ⌋

The initial discretization of the sphere is obtained by standard subdivision and projection (like in Fig.4) with $n_p = 2562$ nodes and $n_t = 5120$ triangles. Many of the interesting solutions show narrow necks, and hence adaptive mesh–refinement and coarsening will play a vital role. See Table 11 for a list of the pertinent files and further comments. Suitable choices of the parameters of `refufu` allow the resolution and robust continuation of rather challenging solutions. Nevertheless, here we restrict to $n_p \le n_{\max} = 6000$ nodes in the mesh, and remark that of course for axisymmetric solutions a 1D setting as in [SBL90] is more efficient and allows yet finer meshes (for the generatrix) and hence the resolution of narrower necks.

Table 11: Selected files from `geomtut/vesicles`; `sG.m` and `sphinit.m` as usual, some convenience functions (to shorten scripts) at bottom; see also §4.1.3 for files used for vesicle *flow*.

`cmds0.m`	continuation of the basic nontrivial branches in V_0 for $c_0 = 0$, see Fig. 21, plotting in `cmds0plot.m`.
`cmdsm2.m`	$c_0 = -1$, plotting in `cmdsm2plot.m`, see Fig. 22
`cmds2.m`	$c_0 = 1.4$, plotting in `cmds2plot.m`, see Fig. 23
`hvesbra.m`	local copy and mod of library function `cmcbra.m` to also put, e.g., the bending energy E and the reduced volume v onto the branch for later plotting.
`refufu.m`	local copy and mod (and renaming) of `stanufu.m` for mesh–adaptation; "switched on" by setting `p.fuha.ufu=@refufu`. Proceeds in two steps: 1 (in a loop, until $\delta_{\text{mesh}} <$ `p.nc.delbound` (user set, typically 10), where in almost all cases only iteration is needed): If $\delta_{\text{mesh}} >$ `p.nc.delbound` (user set, typically 10) then call `degcoarsenX` to remove triangles with poor mesh–quality. Solve (83) for u and update X. Refine the mesh according to `e2rsshape1`, call `retrigX`, Solve (83) for u and update X. 2: refine all triangles of area larger than `p.nc.Ab` (the base triangle size of the initial discretization of the sphere, or user set). Solve (83) for u and update X.
`qAV.m`	area–, mass–, and three translational constraints (85), derivative in `qAVder.m`
`qAV2rot`	`qAV.m` augmented by 2 rotational constraints, derivative in `qAV2rotder.m`
`qAVfrot`	`qAV.m` augmented by all three rotational constraints, derivative in `qAVfrotder.m`
`rotax.m`	find rotational axis for axi-symmetric state, see also `showaxis.m`
`stan2rot.m`	switch on 2 rotational PCs for axi-symmetric state; calls `rotax.m` (convenience function). `stanfullrot.m` switches on all three rotational PCs

4.1.2 Results

• $c_0 = 0$. Fig.21(a) shows a basic BD for $c_0 = 0$. As already said, we always start with the unit sphere at some $\lambda_1 < 6 - 4c_0 + 2c_0^2$, cf. (82), and initially continue to larger λ_1 to obtain the BPs from the sphere, although we know them explicitly from (82). This gives the black trivial branch in the 3rd plot in Fig. 21(a). However, for the nontrivial branches we use v as the primary parameter (see Remark 4.2), and get the BD in the first plot in (a), with a zoom in the second.

For the BPs in (82) we have $\lambda_1 = 6$ for $n = 2$ and $\lambda_1 = 12$ for $n = 3$; at the first BP $\lambda_1 = 6$ we have a 5 dimensional kernel of spherical harmonics, but modulo rotations the only bifurcating branch we find bifurcates transcritically in λ_1 and consists of prolates (orange, stable) in one direction

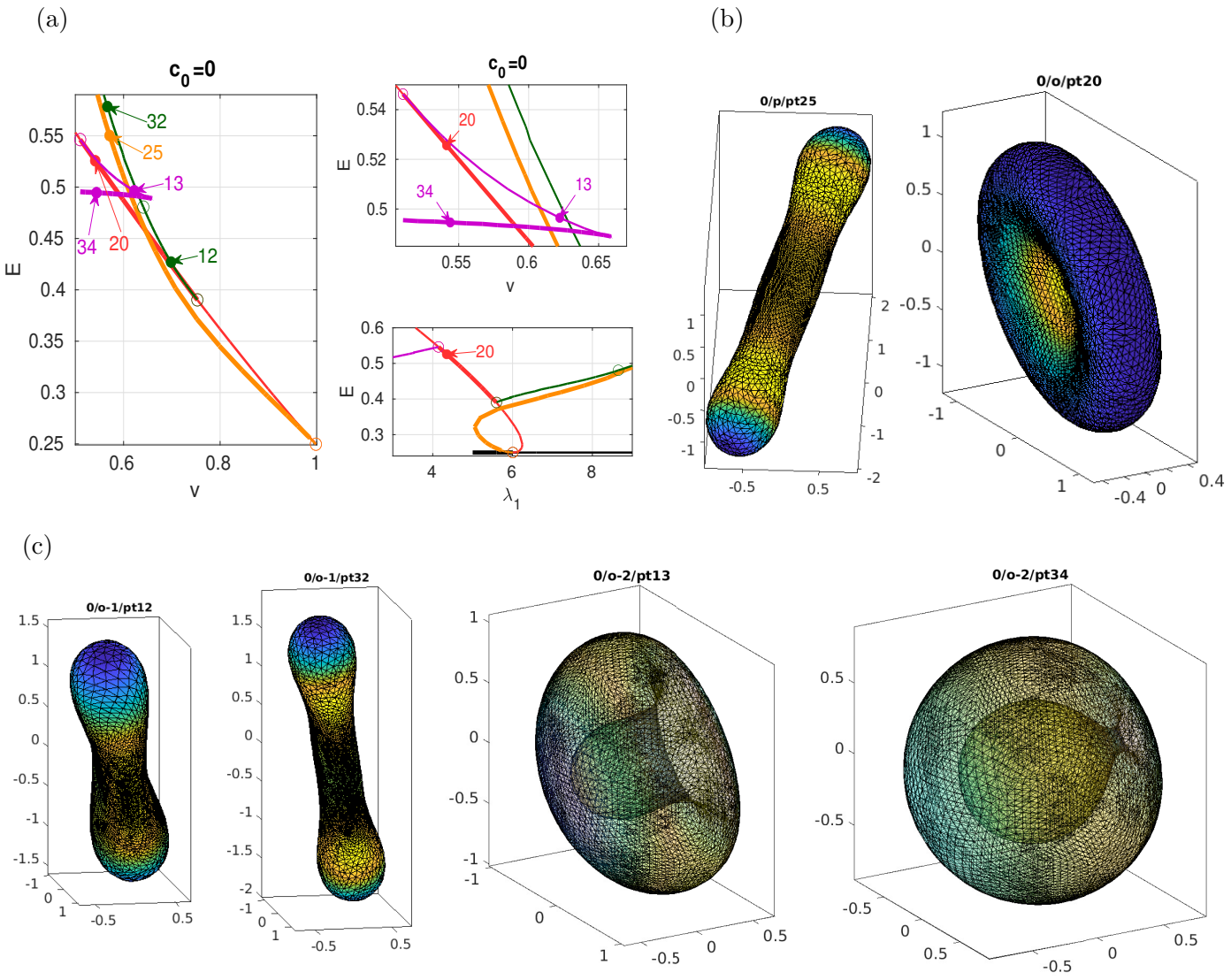

Figure 21: $c_0 = 0$. (a) BDs, E over v, with zoom, and E over λ_1; prolates (orange), oblates (red), and two secondary branches o-1 (green), and o-2 (stomatocytes, violet). (b) samples of one prolate and one oblate. (c) samples along secondary branches bifurcating from oblates.

and oblates (red, initially unstable) in the other direction, see the samples in (b), with the oblate of the typical RBC shape. The oblates gain stability at $v \approx 0.76$ where the green (non–axisymmetric) branch bifurcates, on which solutions first look like elongated RBCs (pt15) and then become similar to prolates (pt42). The oblates (RBCs) lose stability at $v \approx 0.51$ (and shorty after become non–physical due to self intersections) to a subcritical branch of stomatocytes, which stabilizes in a fold at $v \approx 0.66$, after which solutions at low v take the shape of two spheres, the inner one called an inverted sphere, connected by a narrow neck [SBL90]. See the last sample in (c), after which we cannot continue the branch further without refining to more than $n_{\max} = 6000$ nodes. Importantly, the stomatocytes have lower E than the oblates for $v \le v_{os} \approx 0.57$, and this corresponds to the discontinuous transition from oblates to stomatocytes in [SBL90, Fig.10].

• $c_0 = -1$. For $c_0 = -1 < -5/6$ the first BP is at $\lambda_1 = 12$, and the stabilities of prolates and oblates near $v = 1$ flips, i.e., the oblates are now stable near $v = 1$, and the prolates unstable, and remain so for all $v \in (0, 1)$. See Fig. 22, corresponding to [SBL90, Fig.17]. Near $v = v_s \approx 0.7$ stomatocytes bifurcate subcritically from the oblates. According to [SBL90, Fig.17] these stomatocytes stabilize in a fold near $v = 0.95$, but in our numerics we can only reach $v \approx 0.92$. The red oblates become unphysical near $v = 0.55$ due to self intersections, but the branch folds back near $v = 0.5$ and turns

into discocytes with two invaginations, see o/pt70. Again according to [SBL90, Fig.17] the branch stabilizes in a fold near $v \approx 0.85$, but again we cannot continue it sufficiently far and stop at `pt70`.

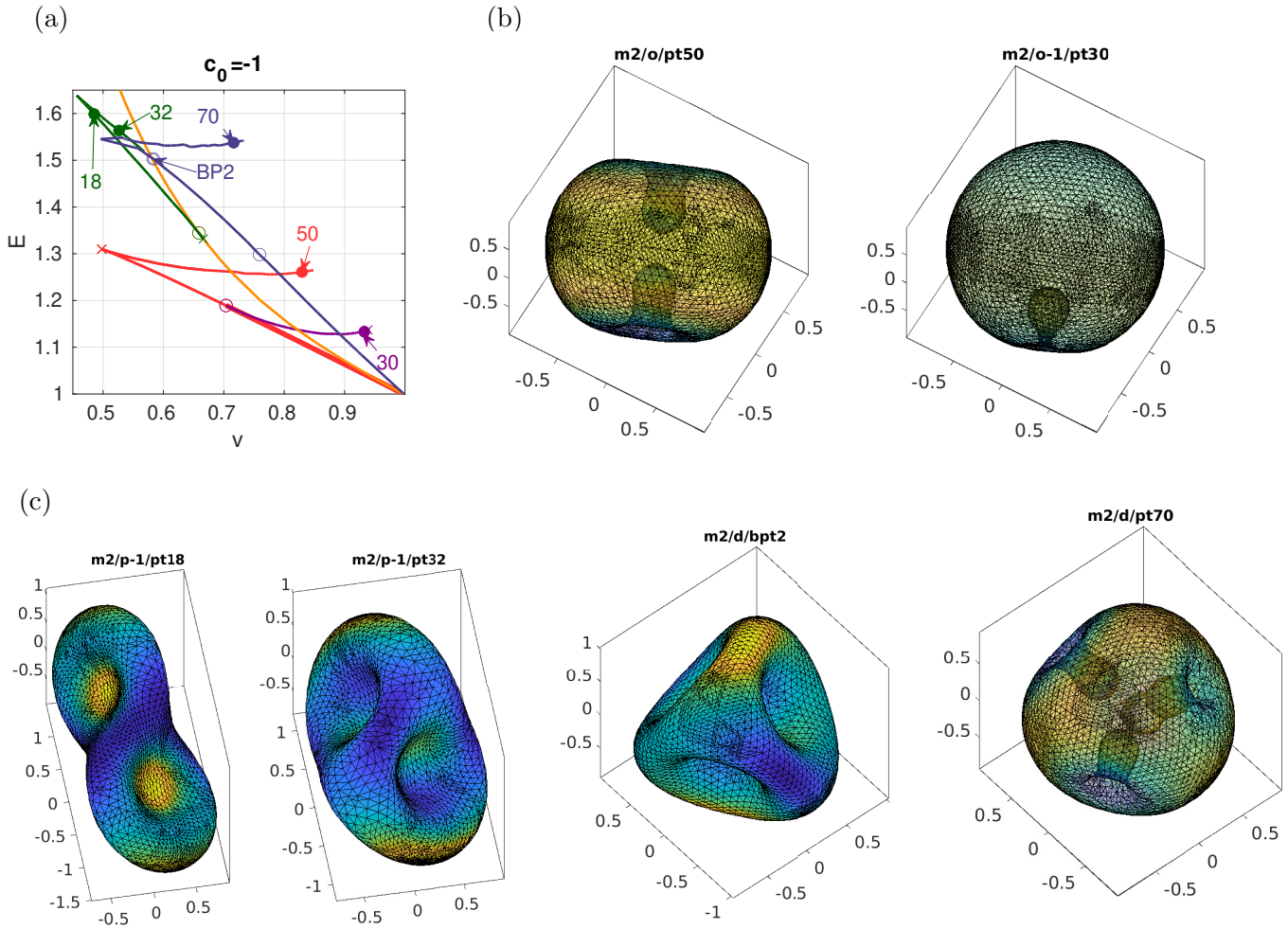

Figure 22: $c_0 = -1$; oblates (red) which turn into discocytes with two stomas; stomatocytes o-1 (violet); prolates (orange), and secondary branch p-1 (green) which connects to (a rotation of) diamonds d (blue) from BP2; the last sample shows the 4 stomas on d.

Instead, in Fig.22 we additionlly show two branches bifurcating from the 2nd BP at $\lambda_1 = 20$, and a number of non-axisymmetric branches. The branch c (magenta) from BP2 is axisymmetric with a roughly conical shape near $v = 1$, which elongates for smaller v, with several 2ndary bifurcations. Another branch d from BP2 has tetrahedral symmetry, with four invaginations, and again with 2ndary bifurcations. Moreover, we show the green branch which bifurcates from the first BP on the orange prolates branch. This also folds back, and connects to d at $v \approx 0.55$, see panel (c). While all branches shown in (a) are unstable, except the oblates for $v > v_s$, we believe that our selection gives a useful first impression of the extremely rich bifurcation structure, in particular showing that and how 2ndary bifurcations from the $n = 2$ primary branches may connect to $n \geq 3$ branches.

• $c_0 = 1.4$**.** According to [SBL90, Fig.10], for $c_0 > c_p \approx 1$ and decreasing v, prolates lose stability to pears. This is illustrated in Fig.23 for $c_0 = 1.4$, together with some secondary bifurcations from the oblates to D3 and D4 "starfish vesicles" o-1 and o-2, and a tertiary bifurcation to o-1-1, which can be thought of as an analogue of pears.

In detail, at $v{=}v_p{\approx}0.73$ the green pear branch bifurcates subcritically from the prolates, gains stability in a fold near $v{=}0.88$, and shows lower E than the prolates for $v{<}v_{pp}{\approx}0.8$, which hence gives the discontinuous transition from prolates to pears in [SBL90, Fig.10]. Again, the D_k branches

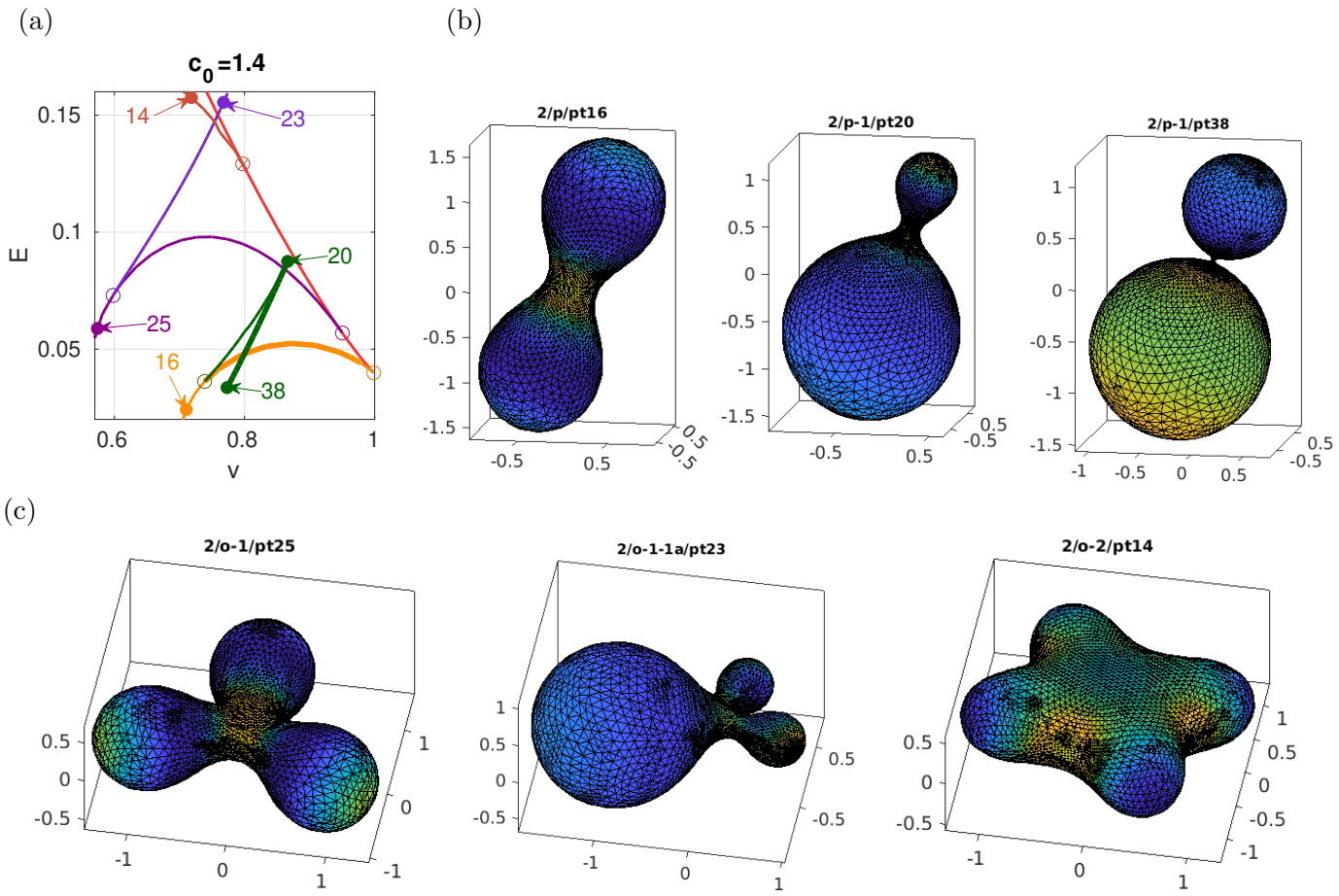

Figure 23: $c_0 = 1.4$. (a) BD, prolates (p, orange), with pears (p-1, green); oblates (red), with secondary bifurcations to D3 (o-1, violet) and D4 (o-2, brown), and tertiary bifurcation (o-1-1a, lilac).

with samples in (b) only contain unstable solutions for the SC model. However, some solutions of this type are in fact stable in the BiC model (77), which yields the same Euler–Lagrange equation (71) as (69), but with a different energy E in (77), and with different constraints, which change the stability properties. Therefore we find it useful to show the branches o-1 and o-2 and associated sample solutions in Fig.23, but a detailed bifurcation analysis of the BiC model will be given elsewhere.

4.1.3 Intermezzo: Numerical Helfrich flow

We explain a setup for implicit geometric flow in `imdnsX.m`, and present some results for Helfrich flow. Essentially, for $G(u) = (G_1(u), G_2(u))$ from (83), temporally writing $G_1(X) := G_1(X_0 + u_1 N)$ instead of $G_1(u)$ and similarly for the n_q constraints q, we discretize the flow $M\dot{X} = -G_1(X)N$ with mass matrix M and together with $q(X) = 0 \in \mathbb{R}^{n_q}$ as follows. With temporal stepsize h, $t_{n+1} = t_n + h$, and $X_n = X(t_n)$, the lhs is approximated as

$$M\dot{X} \approx \frac{1}{h} M_n (X_{n+1} - X_n) = \frac{1}{h} M_n (X_n + u_{n+1} N_n - X_n) = \left(\frac{1}{h} M_n u_{n+1} \right) N_n,$$

and we approximate the rhs as $-G_1(u_{n+1})N_n$. With a slight abuse of notation writing $U = (u, w)$ and $u = (u_1, H)$, where H is the mean curvature of $X = X_0 + uN$, canceling N_n and collecting we obtain

$$\mathcal{G}(U_{n+1}) := \begin{pmatrix} \frac{1}{h} M_{d,n} u_{n+1} + G(u_{n+1}, w_{n+1}) \\ q(U_{n+1}) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \mathbb{R}^{n_u \times n_q}, \tag{89}$$

where M_d is the dynamical system mass matrix (i.e., $M_d = \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix}$, cf. (84)), and $U = (u, w)$ consists of the PDE unknowns and the active parameters *without a primary active parameter* typically used in continuation. Thus, (89) is an implicit time–marching scheme, which as usually we solve via Newton loops. The Jacobian of $\mathcal{G}$ reads

$$\partial_U \mathcal{G} = \begin{pmatrix} \frac{1}{h} M_d + \partial_u G & \partial_w G \\ \partial_u q & \partial_w q \end{pmatrix}, \ M_d \text{ at } t_n, \text{ and } G, q \text{ at } t_{n+1}, \tag{90}$$

and to evaluate (89) and (90) we can fall back on the data structures and functions already used for continuation.

Table 12 lists the pertinent functions. The user interface is `p=imdnsX(p,nt,pmod,omod,smod,nc)`, where `pmod,omod,smod` are controls for plotting/output/saving (e.g., plotting each pmodth step), `nt` is the number of steps, and where the behavior of `imdnsX` is further controlled by fields in `p.nc`, e.g., `p.nc.h` (current stepsize), and `p.nc.hmin, p.nc.hmax` (min and max stepsize). Starting from some X_0 (typically near a steady state, i.e., a result of `cont`), `imdnsX` can be called repeatedly until, e.g., convergence to a (new) steady state is achieved. Importantly, like `geomflow`, see §3.2.3, `imdnsX` usually must be combined with mesh–adaption after a few time-steps, for which we provide a basic function `p=dnsmeshada(p)`. During DNS, data is written each `omod`th step into the time–series `ts` via `imdnsofu`, and for postprocessing there are `Xflplot` and `Xflmov`.

Internally, `imdnsX` calls `nloopdnsX` to solve (89) to accuracy `p.nc.tol` with at most `p.nc.imax` Newton steps, and this gives some rudimentary time–stepping control: subject to `p.nc.hmin` $\leq$ `p.nc.h` $\leq$ `p.nc.hmax`, if $\|\mathcal{G}\|_\infty >$ `p.nc.tol` after `p.nc.imax` steps, then `p.nc.h` is halved; if $\|\mathcal{G}\|_\infty >$ `p.nc.tol` after `iter<p.nc.imax` iterations, then `p.nc.h` is doubled. However, altogether `imdnsX` is mainly aimed at flowing from some IC to a stable steady state, and mesh adaptions along the way may distort the time scale for the convergence, but once we approach a steady state transient errors are automatically repaired.

Table 12: Functions for DNS via (89), and postprocessing

function	remarks
`p=imdnsX(p,nt,pmod,omod,smod)`	main interface for the implicit DNS scheme.
`p=dnsmeshada(p)`	template for mesh adaptation to be alternated with `imdnsX`.
`out=imdnsofu(p)`	output at each `omod`th time-step. Default $(t, \|u\|,$ `p.fuha.outfu`$)$.
`nloopdnsX`	solving (89)
`Xflplot, Xflmov`	plot DNS results, and generate movie from DNS.

The script `geomtut/vesicles/cmdsflow.m` gives a number of examples, where we start near some steady states from the continuations in Fig.21–Fig.23. For all flows we switch off all phase conditions and zero out the PC Lagrange multipliers $s_x, \ldots, r_3$, as they are only needed for robust steady state continuation, while the positional and rotational invariance of the vesicles play no role for the flow. Thus in all flows we set `p.nc.nq=2` and `p.nc.ilam=[6 2 3]`, where the primary continuation parameter is arbitrary as it is fixed, and `p.fuha.qf=@qAVfl` which implements $(A - A_0, V - V_0) = (0, 0)$. The results can be summarized as follows, see `cmdsflow.m` for details and comments, Fig.24 for a typical example, and [Uec24] for movies of vesicle flows:

1. If we start with an X_0 which is not "too deformed", and choose suitable flow parameters `tol, hmin, hmax`, and suitable alternating between `imdnsX` and `dnsmeshada`, then the flow takes us to a stable steady state X_s.
2. Starting near strongly deformed but stable X_s (e.g., stomatocytes or pears with narrow necks), the flow takes us back to the starting X_s.
3. Once the flow has converged to a (new) steady state X_{new}, this can be used as a starting point for continuation again.

4. Starting near (unstable) strongly deformed X_s (e.g., X_s an unstable stomatocyte or pear), it may be difficult to handle the flow of the mesh and the flow may eventually fail.
5. Unless 4 happens, all our examples confirm the (in)stability of steady states as given in Figs.21–23, and A and V are conserved to at least 6 digits.

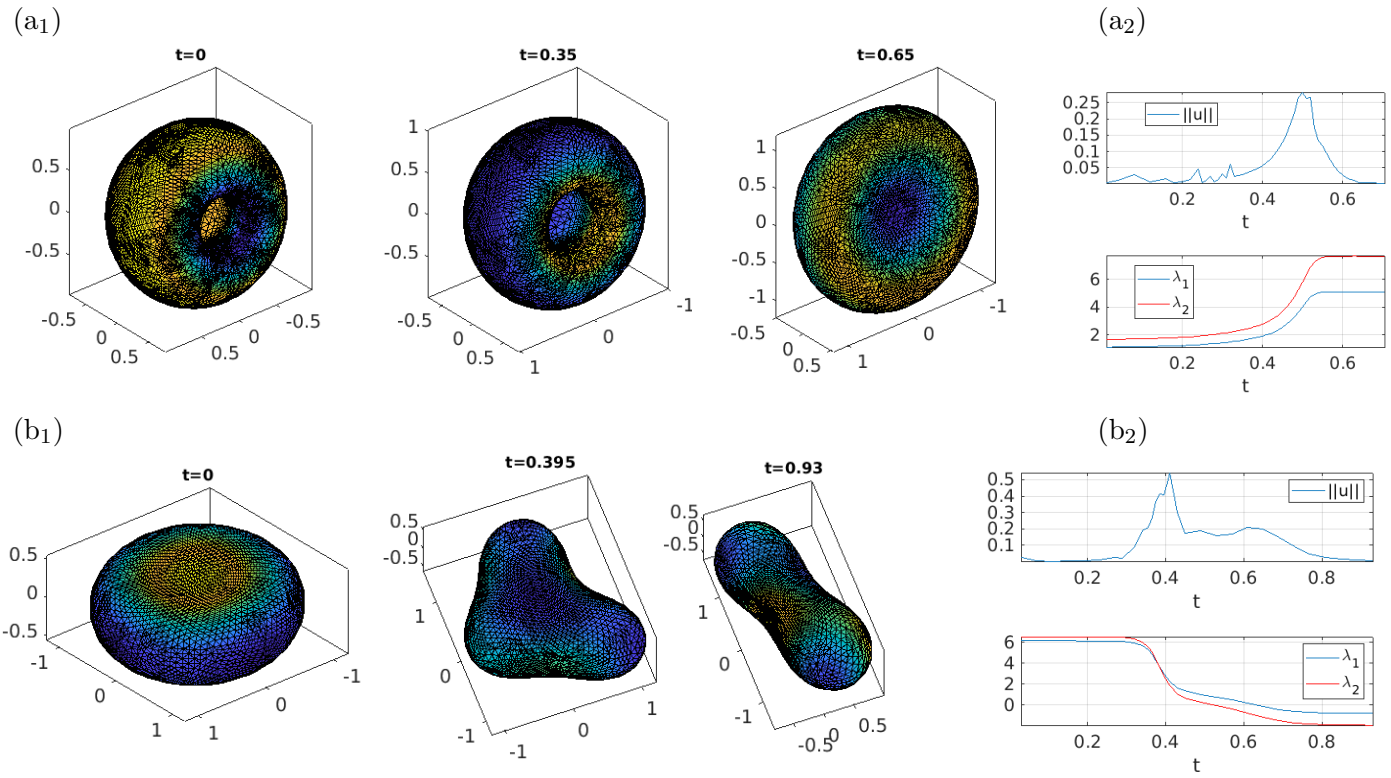

Figure 24: Examples of Helfrich flows. (a_1) $c_0 = 0$, snapshots for flow from perturbation of unstable stomatocyte (near Fig.21, 0/o-2/pt13) to oblate; (a_2 behaviors of $\|\dot{X}\|$ and $\lambda_{1,2}$. (b) Analogous for $c_0 = 1$ (cf. Fig23); flow from (unstable) oblate to prolate.

4.2 Biocaps

In the demo `biocaps` we fix the boundary circle $\partial X = \{x^2 + y^2 = \alpha^2\}$ in the x–y plane. Thus,

$$\Delta H + 2H(H^2 - K) + 2c_0 K - 2c_0^2 H - 2\lambda_1 H = 0 \text{ on } X, \text{ and} \tag{91a}$$

$$u = 0 \text{ and } H - c_0 + b\kappa_n = 0 \text{ on } \partial X. \tag{91b}$$

In our experiments we fix $\alpha = 1$ and $\lambda_1 = 1/4$, and first $c_0 = 1/2$ and vary b, and want to start with the upper unit hemisphere. Then $H = K = 1$ (choosing the inner normal for the hemisphere) and $\kappa_n = 1$ and hence (91b) requires $b = -1/2$.

Remark 4.3 a) For X not closed it is an open problem for what parameters, and boundaries and BCs, the minimization of L from (70) is a well–posed problem. In [Nit93], the following conditions on c_0, b and λ_1 are posed for L with $\lambda_2 = 0$ to be definite in the sense that $L \geq C_0$ for some $C_0 > -\infty$ for all connected orientable surfaces X of regularity C^2 with or without boundary:

$$\text{(i) } \lambda_1 \geq 0, \qquad \text{(ii) } -1 \leq b \leq 0, \qquad \text{and (iii) } -bc_0^2 \leq \lambda_1(1+b). \tag{92}$$

This proceeds as in [Nit91] by scaling properties of E for various surfaces composed of planes (of area A), cylinders (of lengths l and radius r_c), and (hemi)spheres (of radius r_S), and considering the

asymptotics of L as $A, l \to \infty$ and/or $r_c, r_s \to 0$. For instance, the condition (92)(i) arises most naturally by considering X to contain a plane with $A \to \infty$, which for $\lambda_1 < 0$ gives $L \to -\infty$.

On the other hand, in the physics literature no restrictions on $c_0, b \in \mathbb{R}$ are given, and in a given problem a fixed ∂X and the BCs (74) or (75) may make L definite for much larger ranges than given in (92). In our experiments below we do take parameters to rather extreme values, e.g., $b = -4$ in Fig. 26, where we find interesting solutions.

b) For $\partial X \neq \emptyset$, and in particular for the cases of caps considered below, we are not aware of analytic bifurcation results, except [PP22] which presents some results for caps in a slightly different setting. ⌋

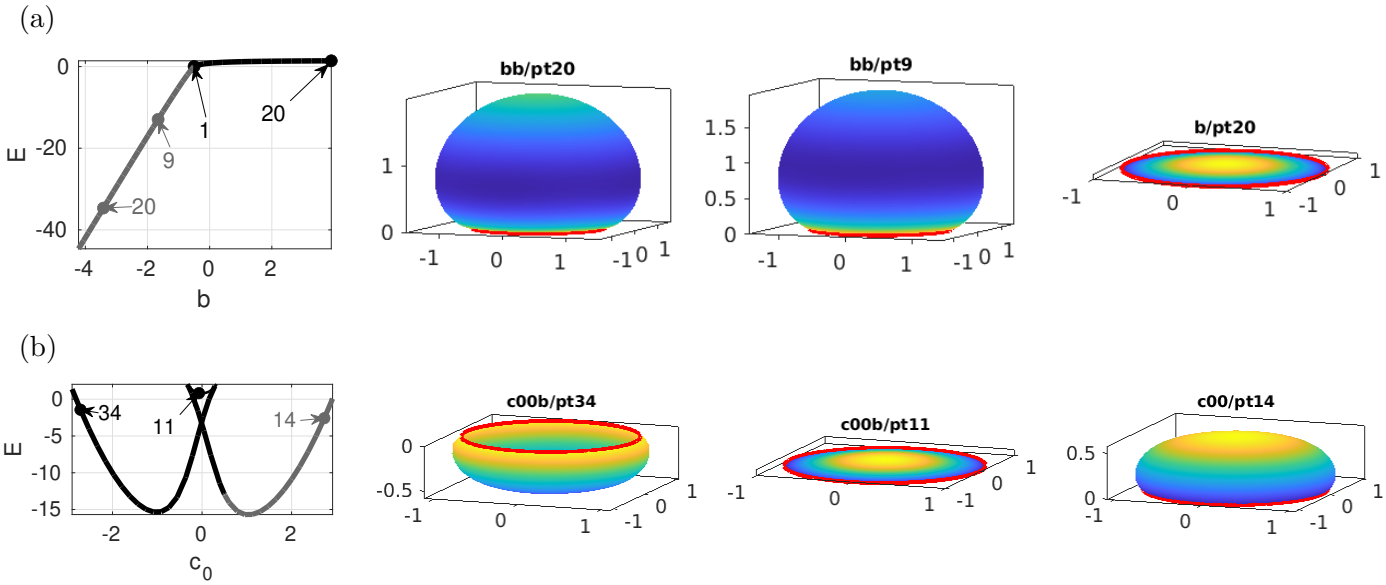

Figure 25: Initial results for (71),(75) from `biocaps/cmds1.m`. In (a) we continue in b starting at `b/pt1` from the unit hemisphere with $(\alpha, \lambda_1, c_0, b) = (1, 1/4, 1/2, -1/2)$, to increasing b (branch `b`, black) and decreasing b (branch `bb`, grey). On `b` we go to the flat disk (last sample), while on `bb` the hemisphere bulges out. This is mainly intended for later continuation in c_0, and in (b) we do so starting from `bb/pt9` at $b \approx -1.66$. This gives the double well shape for E, with a short unstable segment between the two folds at $c_0 \approx \pm 0.33$. See Fig. 26 for the cases of $b \approx -3.4$ (`bb/pt20`) and $b \approx -4$.

Fig.25(a) shows the continuation of the initial hemisphere in b. This is mainly intended for subsequent continuation in c_0 at negative b, and (b) shows the case of $b \approx -1.66$. The problem is symmetric under $(c_0, X_3) \mapsto -(c_0, X_3)$, and in particular at $c_0 = 0$ we have the flat disk as an exact solution (for any b). See `c00b/pt11` for a nearby solution with $c_0 \approx -0.07$, which lies between two folds with exchange of stability. This unstable part will feature interesting bifurcations to non–axisymmetric branches at more negative b, see Fig. 26, while the remainder(s) of the axisymmetric branches are all stable, with the samples `c00b/pt34` and `c00/pt14` in 25(b) showing the typical behavior at strongly negative or positive c_0, respectively.

Remark 4.4 The only non–standard file in `geomtut/biocaps` is `bdint.m`, used to evaluate $\int_{\partial X} \kappa_g \, ds$ by a trapezoidal rule, see (c) below. The main numerical challenges and used tricks for (91) are:

a) For the initial hemisphere we again use a subdivision and projection algorithm, followed by one mesh–refinement at the boundary, as a good resolution near ∂X turns out helpful later. The initial mesh then has `np=2245` nodes, which later is refined to `np` > 6000. The mesh quality in all our solutions stays quite good, i.e., $\delta_{\text{mesh}} < 20$ for all solutions, and mostly $\delta_{\text{mesh}} < 10$.

b) The boundary $\gamma = \partial X$ is parameterized by arclength as $\gamma(\phi) = \alpha(\cos(\phi/\alpha), \sin(\phi/\alpha), 0)$. Then $\kappa = \gamma'' = -\gamma/\alpha$ and the normal curvature on ∂X reads $\kappa_n = -\frac{1}{\alpha}\langle N, X\rangle$, which is used to implement the BCs (91b).

c) The "integral" `sum(K)` over the discrete Gaussian curvature `K` always evaluates to $2\pi\chi(X)$, cf. Footnote 1. Thus we once more use Gauss-Bonnet $\int_X K\,\mathrm{d}S = 2\pi\chi(X) - \int_{\partial X} \kappa_g\,\mathrm{d}s$ to compute the energy E, where $\kappa_g = \mathrm{sign}(N_3)\frac{1}{\alpha}\|N \times \gamma\|$. ⌋

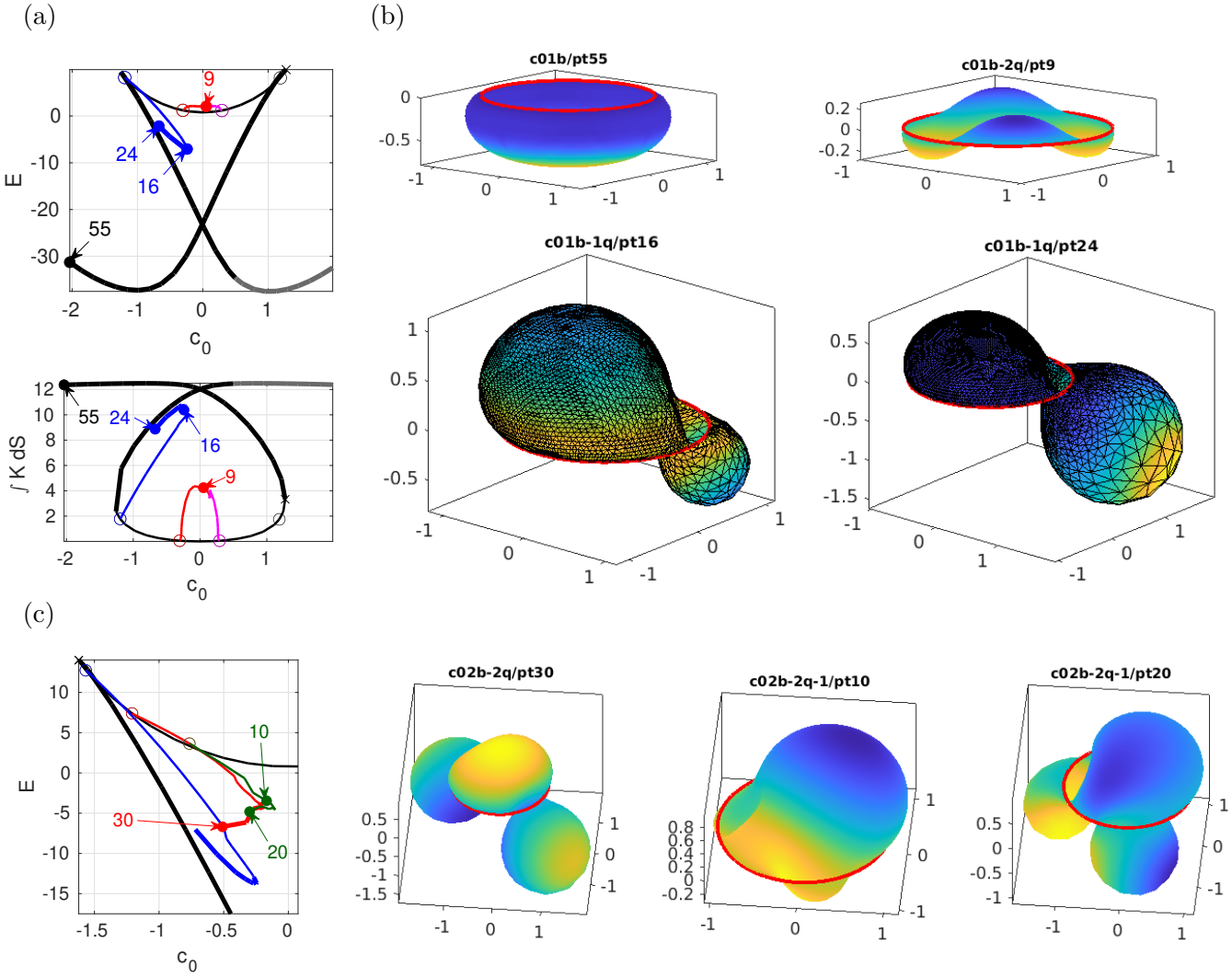

Figure 26: Continuation of Fig. 25. (a,b) Continuation in c_0 from `bb/pt20`, $(\alpha, \lambda_1, b) = (1, 0.25, -3.4)$, starting from $c_0 = 0.5$, branches `c01b` (black, to decreasing c_0) and `c01` (grey, to increasing c_0). There are two BPs on the unstable part of `c01b` for $c_0 < 0$, and the symmetric BPs for $c_0 > 0$. The blue branch `c01b-1q` has azimuthal wave number $m = 1$ and is stable after its fold. The red branch `c01b-2q` has $m = 2$ and connects to the symmetric BP at $c_0 > 0$. The 2nd plot in (a) shows where the part $b\int K\,\mathrm{d}S$ of E becomes dominant, taking into account the rather large $|b|$. (c) Similarly starting at `bb/pt24` with $b \approx -4$; zoom of BD near upper left fold of the branch `c02b` (black) similar to `c01b` from (a). The blue branch is qualitatively as in (a), but now the $m = 2$ branch `c02b-2q` (red) also folds back giving stable solutions, and there is a secondary BP on it, giving the green branch `c02b-2q-1`.

In Fig. 26 we repeat the continuation in c_0 from Fig. 25(b) at more negative b, namely $b \approx -3.4$ in (a) and $b \approx -4$ in (b). For lower b, the unstable part of the c_0 continuation expands, and we find two (or more, for even lower b) BPs between the left fold and $c_0 = 0$, with azimuthal wave numbers $m = 1$

and $m = 2$. As before, these bifurcations are double by S^1 symmetry, and to continue the bifurcating branches we set the usual rotational PC after two steps. The blue $m = 1$ branch then behaves similarly in (a) and (b), i.e., it becomes stable after a fold at $c_0 \approx -0.2$ ($b = -3.4$) resp. $c_0 \approx -0.27$ ($b = -4$). However, the $m = 2$ branch behaves differently: For $b = -3.4$ it connects to the symmetric BP at $c_0 > 0$. For $b = -4$, the red branch `c02b-2q` first shows a secondary BP to a branch (`c02b-2q-1`, green) with broken $\mathbb{Z}_2$ symmetry, and then shows a fold at $c_0 \approx -0.21$ where it becomes stable. The branch `c02b-2q-1` also shows a fold, at $c_0 \approx -0.11$, after which however one unstable eigenvalue remains, i.e., $\mathrm{ind}(X) = 1$ at, e.g., `c02b-2q-1/pt20` (last sample in (c)). The somewhat non–smooth shape of the red and green branches is due to repeated and heavy mesh refinement, to, e.g., $n_p = 5560$ at `c02b-2/pt30`.

In Fig. 27 we continue some $m = 1$ solutions from the blue branch in Fig. 26(b) in b and in λ_1, with fixed c_0. The unstable black solutions with $c_0 \approx -0.35$ in (a) continue to $b \approx -1.8$, where the branch bifurcates from the axisymmetric branch. This is in contrast to Fig. 25 where for the continuation in b at $c_0 = 1/2$ no bifurcations from the axisymmetric branch were found. The blue branch in Fig. 27(a) with $c_0 \approx -0.24$ shows a fold at $b \approx -3.11$. From these and similar experiments, $m = 1$ solutions do not seem to exist for b somewhat larger than -2, and no stable ones for b larger than -3. In Fig. 27(b), the black unstable branch continues to large λ_1, but the stable blue branch again shows a fold, at moderate $\lambda_1 \approx 0.44$. This indicates that solutions of this type also do not continue to "large" λ_1. Nevertheless, while the physical (biological) significance of the parameter regimes and solutions in Figs. 25–27 certainly needs to be discussed, mathematically we have obtained some stable non–axisymmetric solutions.

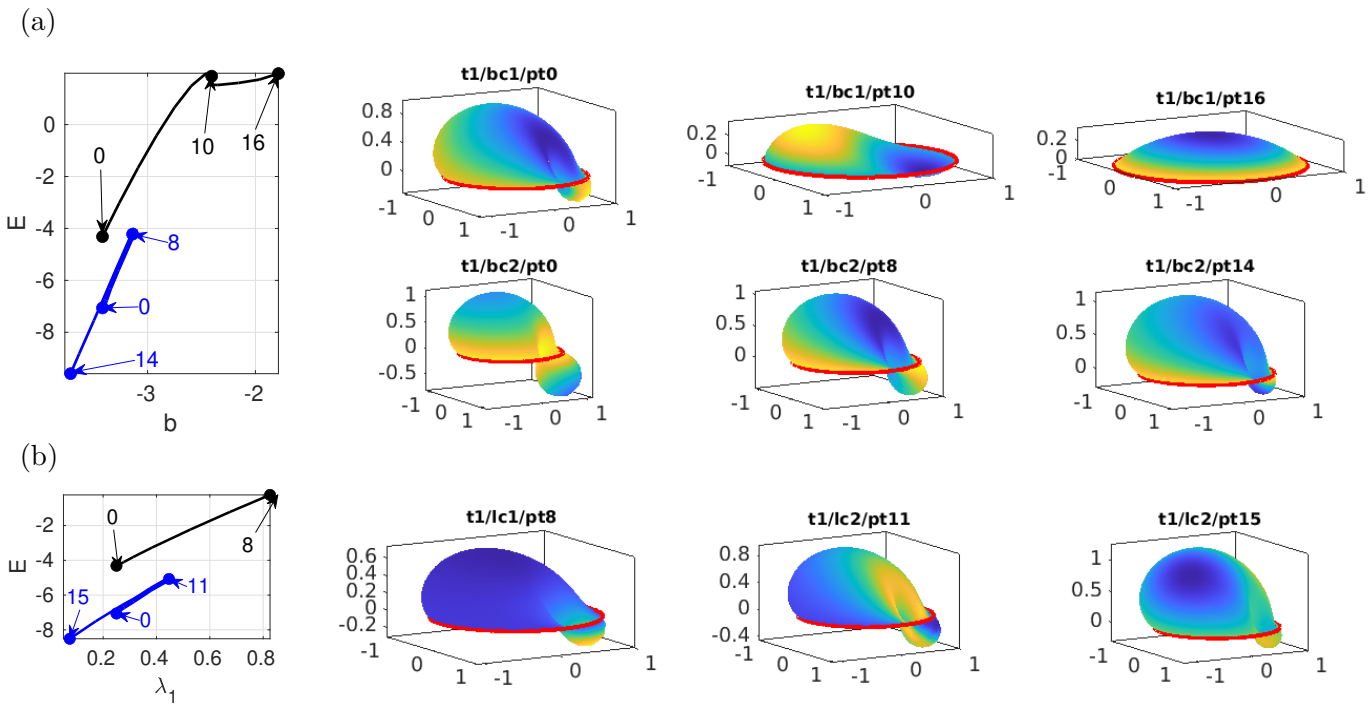

Figure 27: `cmds2.m`, experiments with continuation of $m = 1$ solutions from Fig. 26 in b (a), and in λ_1 (b). The starting points in (a) are from $c \approx -0.35$ for the black branch `bc1`, and from $c \approx -0.24$ for the blue branch `bc2` (this is the same solution as `c01b-1q/pt16`). The continuation in λ_1 in (b) has the same starting points and the same colors. The unstable branch (now with fixed $(c_0, b) = (-0.35, -3.4)$) continues to large λ_1, but the blue branch with fixed $(c_0, b) = (-0.24, -3.4)$ has a fold at $\lambda_1 \approx 0.44$.

Figure 28 shows some results from `biocaps/cmdsflow.m` for the numerical flow for (91), based on `imdnsX`, cf.§4.1.3, with the main difference to the flow for closed vesicles is that we now have no constraints. The flows behave as predicted from the BD Fig.26, and once the amplitude of the flow has dropped below 10^{-3} we can use the obtained state as an initial guess for continuation.

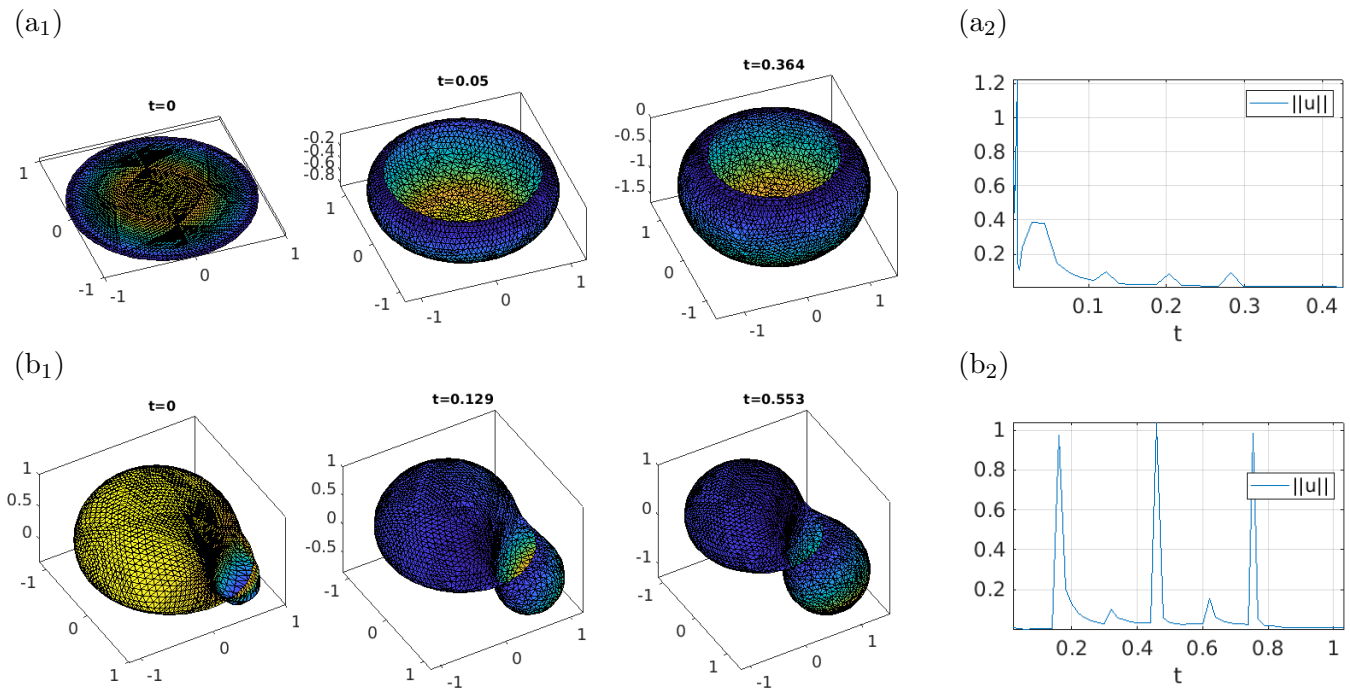

Figure 28: Flows starting from perturbations of (a) the flat disk at $c_0 = -0.5$, and (b) `c01b-1q/pt10` with $c_0 = -0.316$ (unstable). For both we need repeated mesh-adaptation to obtain convergence to the larger amplitude solution on `c00b` in (a), and to the stable solution on `c01b-1q` after the fold. At the end of the flows, both solutions are sufficiently close to the respective steady state to start continuation. See `biocaps/cmdsflow.m` for details.

5 Summary and outlook

We explained the basic setup of the `pde2path` extension library `Xcont` for continuation of 2D submanifolds X (surfaces) of $\mathbb{R}^3$, and gave a number of examples. These were partly introductory, e.g. the spherical caps in §3.1, partly classical, e.g., Enneper's minimal surface in §3.2.3, the nodoids in §3.3.2 and §3.4, the TPS in §3.5, and the closed vesicles in §4.1, and partly rather specific such as the Plateau problem in §3.2.2 or the 4th order Helfrich type caps in §4.2 (and cylinders in §B. Besides [Bru18], and to some extent [Bra96], there seem to be few numerical continuation and bifurcation experiments for such geometric problems for 2D surfaces, i.e., without imposing some axial symmetry, and we are not aware of a general software for such tasks.

The basic setup for all our problems (except those of 4th order) is similar: We consider CMC surfaces, which mainly differ wrt constraints and/or boundary conditions. Along the way we explained a number of techniques/tricks which we expect to be crucial in many applications. A major problem for continuation (over longer parameter regimes) is the mesh handling as X changes and hence the mesh distorts. We explained how this (often) can be abated via moving of mesh points (`moveX`), refinement (`refineX` which sometimes should be combined with re–triangulation by `retrigX`) and coarsening (`coarsenX`), and coarsening of degenerate triangles (`degcoarsenX`), although the choice of the parameters controlling these functions often requires some trial and error. In any case, X bulging out is usually harmless, but bulging in (the development of necks) is more challenging.

This is a first step. With the demos we hope to give a pool of applications which users can use as templates for their own problems, and we are curious what other applications users will consider, and of course are happy to help if problems occur. As indicated above, our own further research, to be presented elsewhere, includes:

- Further classical minimal surfaces (and CMC companions) such as Schwarz H and Scherk surfaces (surface families);
- Alternate models for closed vesicles, which in contrast to the SC model from §4.1 show non–

axisymmetric minimizers;

- Coupling of membrane curvature and morphogen dynamics or reaction–diffusion equations as in, e.g., [MMCRH13, TN20].

A Spheres, hemispheres, VPMCF, and an alternative setup

A.1 Spheres

The demo `spheres`, containing only the (somewhat minimally necessary) files `sphereinit.m`, `sG.m`, `getM.m`, and `cmds1.m`, is mostly meant to illustrate volume preserving mean curvature flow (VPMCF) near spheres, see Fig. 29.[24] We refer to `sphereinit.m` and `cmds1.m` for comments (and to `geomflow.m` and `vpmcff.m` from `libs/Xcont` for the VPMCF) and here only note:

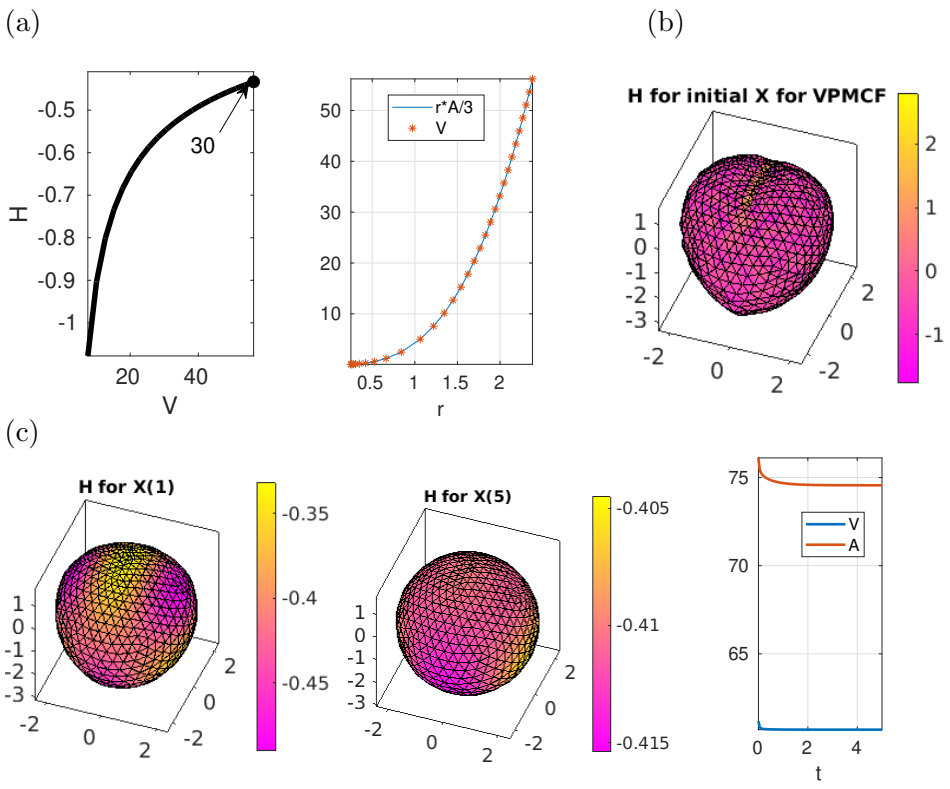

Figure 29: Results from `sphere/cmds1.m`. (a) Continuation of a sphere in V, with comparison of A and V. (b) An IC for VPMCF from a perturbation of `S3/pt30`, with (c) solutions at $t = 1$ and $t = 5$, and a time–series for V, A; V is conserved to within 0.5%.

1. The comparison between $rA/3$ and V in Fig. 29(a) shows a very small error which indicates that the solutions are good approximations of spheres.
2. For convex closed initial X (meaning that $X = \partial\Omega$ for a convex domain $\Omega \subset \mathbb{R}^3$), the VPMCF converges to a sphere. See also, e.g., [ES98] for theoretical background. This also holds for "slightly" non–convex initial X as in Fig. 29(b).[25]
3. Our (explicit Euler) implementation of the VPMCF does not conserve V exactly, but with "reasonable accuracy", i.e.: Even for quite "non–spherical" initial $X(0)$, the error $|1 - V_\infty/V_0| < 0.01$, where $V_\infty = \lim_{t\to\infty} V(t) < V_0$ in all our tests, i.e., V_∞ is always slightly smaller than V_0.[26]

[24] Additionally, the demo contains `convtest.m` for convergence tests, see Fig. 4.

[25] In detail, $X(0)$ here is obtained as $X(0) = S_{r_0} + 0.4(\sin(\vartheta)(|x| - r_0) + \xi)N$, where ϑ is the azimuth, and $\xi = 0.2(\texttt{rand} - 0.5)$ with `rand` $\in [0, 1]$ a MATLAB random variable on each node.

[26] This "volume–accuracy" of `geomflow` for VPMCF depends weakly on the Euler–stepsize `dt`, and more strongly on the fineness of the discretization of X. This can be checked by changing `sw` in `spheres/cmds1.m` for initializing X.

4. During continuation, the position of `X` is not fixed, i.e., we have the threefold (in the discrete setting approximate) translational invariance in x, y, z, and hence always a three-dimensional (approximate) kernel. This could be removed by suitable translational PCs (see the demo `hemispheres`). However, the approximate kernel is not a problem for the continuation here since in the Newton loops the right hand sides are orthogonal to this kernel. As here we are mostly interested in the VPMCF, for which the translational invariances are irrelevant, we refrain from these PCs to keep the demo sleek and simple.

A.2 Hemispheres

In the demo `hemispheres` we continue in volume V hemispheres X sitting orthogonally on the $z = 0$–plane, i.e.,

$$\partial_r X_3 = 0 \text{ where } r = \sqrt{x^2 + y^2}, \tag{93}$$

and test VPMCF for perturbations of such hemispheres, see Fig. 30. Additionally we use the PCs

$$\int_X N_1 \, \mathrm{d}S = \int_X N_2 \, \mathrm{d}S = 0, \quad X = X_0 + uN_0, \tag{94}$$

to fix the translational invariance.

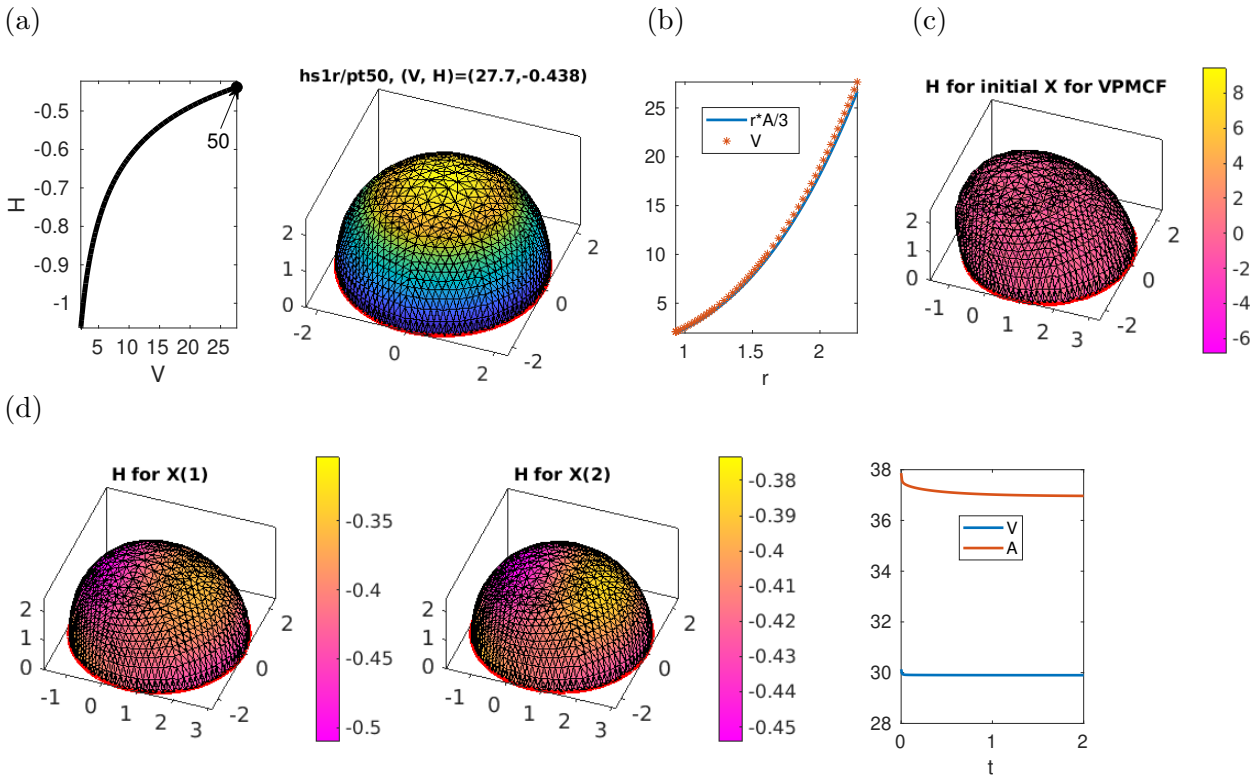

Figure 30: Results from `hemisphere/cmds1.m`. (a) Continuation of hemisphere in V, with sample at end. (b) Comparison of A and V for (a). (c) A perturbation of `hsr1/pt50` as IC for VPMCF. (d) Solutions at $t = 1$ and $t = 2$, and time–series for V, A from (c,d); V conserved to within 0.8%.

Compared to the spheres in §A.1 this requires a few more files, listed in Table 13. The PCs (94) (and u–derivatives) are implemented in `qf2, qjac2`, and the rhs $G(u) = H - H_0$ in `sGhs` is augmented to $\widetilde{G}(u) = H - H_0 + s_x N_1 + s_y N_2$ with multipliers s_x, s_y. These stay $\mathcal{O}(10^{-6})$ during continuation, and the only effect of the construction is that the 2D kernel of translational invariance of $\partial_u G$ is removed

Table 13: Files in `pde2path/demos/geomtut/hemispheres`.

`cmds1.m`	continuation in V, and VPMCF flow test.
`hsinit.m, sGhs.m`	init and rhs
`qf2.m, qjac2.m`	PC (94), and derivative
`getN.m`	mod of getN, correction at $z = 0$
`diskpdeo2.m`	mod of (`pde2path`–)default `diskpdeo2.m` to have a finer mesh at $r = 1$.

from the linearization of the extended system $(\widetilde{G}, q)$. See the end of `cmds1.m` for further comments.[27]

The BCs (93) allow motion of ∂X in the "support–plane" $z = 0$, and are implemented as

$$\texttt{r(idx)=grXz*(X(idx,1).\^{}2+X(idx,2).\^{}2)} \stackrel{!}{=} 0)$$

in `sGhs`, where as usual `idx` are the boundary indices, and `grXz` is the z–derivative (operator) on X, as before obtained from `grX=grad(X,p.tri)` and interpolation to the nodes. This forces the x, y–coordinates of the points on X directly above the $z = 0$ layer to also fulfill $x^2 + y^2 = 1$, i.e., we obtain a "cylindrical socket" for the hemispheres. To mitigate this effect, in `hsinit` we initialize with a somewhat specialized mesh over the unit disk D with higher density towards ∂D, which is then mapped to the unit hemisphere via $z = \sqrt{1 - x^2 - y^2}$. Nevertheless, after continuation to larger V, which is combined with some mesh–adaptation by area, we obtain a mismatch between $rA/3$ and V, see Fig. 30(a,b), and compare Fig. 29(b).

In Fig. 30(c,d) we give an example of VPMCF from a perturbation of `hs1r/pt50`, here of the form $X|_{t=0} = X + 0.4(\cos(\vartheta)(\max(z) - z + 0.1(\text{rand} - 0.5))N$, where ϑ is the angle in the $z = 0$ plane. We *do not use any BCs* in the rhs `vpmcff.m`. Instead, we use the correction

```
N(p.idx,3)=0; N=normalizerow(N);
```
(95)

in `getN.m`, to let N lie in the x–y–plane, cf. Remark 3.8(b) for a similar trick. Without (95), the *continuation* of hemispheres works as before, but the solutions of the VPMCF start to (slowly) lift off the $z = 0$ plane, and then (quickly) evolve towards a planar X such that also $V \to 0$. With (95), the solutions flow back to hemispheres, and V is again conserved up to 0.5%, and "after convergence" (e.g., from $X|_{t=2}$) we can start continuation again, showing consistency.

A.3 Spherical caps via 2D finite elements

For the sake of completeness and possible generalization, we show how the spherical caps with DBCs can be treated in a classical FEM setting. Let $\Omega \subset \mathbb{R}^2$ be a bounded domain and X_0 be a surface with parametrization $\phi_0 : \Omega \to \mathbb{R}^3$, and as before define a new surface via $X = X_0 + u\, N_0$, $u : \Omega \to \mathbb{R}$. Then (1) reads

$$G(u, H, V) = \begin{pmatrix} H(X) - H \\ V(X) - V \end{pmatrix} = 0, \text{ with } u|_{\partial\Omega} = 0. \tag{96}$$

This is a quasilinear elliptic equation for $u : \Omega \to \mathbb{R}$, and after solving (96) we can again update $X_0 = X_0 + uN_0$ and repeat. To discretize (96) we now use the FEM in Ω. The main issue is how to compute H (and A and V and similar quantities) without the `gptoolbox`, and Table 14 lists the pertinent files.

[27] In `cmds1.m` we also monitor the "positions" (x_0, y_0) of X defined as $x_0 = \int_X X_1 \,\mathrm{d}S$, $y_0 = \int_X X_2 \,\mathrm{d}S$. These behave as expected, namely: very small drifts for continuation without PCs, but no drifts for continuation with the PCs (94).

Table 14: Overview of files in `geomtut/spcap2`.

`cmds1`	Main script; initialization, continuation, plotting.
`spcapinit`	Initialization, rather standard, except for `p.X=[x,y,0*x]` for the initialization of X.
`oosetfemops`	Setting mass matrix M, and first order differentiation matrices which are used to compute H in `sG` (via `getmeancurv`).
`sG, qV`	rhs and volume constraint.
`getN`	compute normal vector.
`getA, getV`	compute $A(X)$ and $V(X)$, overload of `Xcont` functions.
`getmeancurv`	H from (11), see also `get1ff, get2ff` for 1st and 2nd fundamental forms.
`e2rs`	ElementToRefineSelector function, based on triangle areas.
`cmcbra, pplot`	like `Xcont/cmcbra` and `Xcont/pplot`, but overloaded since `p.tri` not present here.
`getGupde`	overload of library function to deal with larger bandwidth.

To implement $H(X)$, $X = X_0 + uN$, we here directly use the definition

$$H = \frac{1}{2}\frac{h_{11}g_{22} - 2h_{12}g_{12} + h_{22}g_{11}}{g_{11}g_{22} - g_{12}^2}$$

of the mean curvature based on the fundamental forms of X. This is brute force and in particular neither confirms to a weak (FEM) formulation nor allows simple Jacobians, see Remark A.1.

Remark A.1 a) In `getN` and `get1ff` we compute the *nodal* values for X_x and X_y (via weighted averages of the adjacent triangles), and the corresponding nodal values of N and g_{ij}; this is needed here as X_x and X_y appear nonlinearly in N, and similarly g_{ij} appear nonlinearly in H. On the other hand, the second derivatives h_{ij} only appear linearly in H, and hence we take the second derivatives in `get2ff` using the *element* differentiation matrices `Kx` and `Ky`. Thus, `H` in `r=-2*H+p.mat.M*(H0*ones(p.nu,1))` in `sG` is (an approximation of) the element wise mean curvature, and hence `H0` must also be multiplied by the mass matrix `p.mat.M`.

b) Since G as implemented uses products of differentiation matrices, the associated Jacobian $\partial_u G$ has more bandwidth than before, i.e., the sparsity pattern S of $\partial_u G$ is that of M^2, rather than that of M, and thus we use `S = M^2 > 0` in `getGupde`. ⌋

```
function p=spcapinit(nx,par) % spherical cap, init, legacy setup
p=stanparam(); p.sw.spcalc=0; p.sw.bifcheck=0; % set stanparam, overwrite some
pde=diskpdeo2(1,nx,round(nx/2)); % disk preimage discretization
p.pdeo=pde; p.np=pde.grid.nPoints; p.nu=p.np; p.nt=pde.grid.nElements;
p.sol.xi=1/p.nu; p.nc.neq=1; p.sw.sfem=-1; % store dimensions
p.fuha.outfu=@cmcbra; p.fuha.e2rs=@e2rs; % branch data, refinement selector
p.sw.Xcont=1; p.plot.pstyle=-1; % call userplot for plotting
po=getpte(p); x=po(1,:)'; y=po(2,:)'; u=0*ones(p.np,1); % initial soln
p.u=[u; par]; p.X=[x,y,0*x]; % set IC, including X
p=oosetfemops(p); % here constant mass and stifness matrices (until mesh changes)
p.plot.auxdict={'H','V','alpha','A'}; p.u(p.nu+4)=getA(p,p.u); % initial area
```

```
function p=oosetfemops(p) % legacy setting, precompute FEM matrices
gr=p.pdeo.grid; fem=p.pdeo.fem; [~,p.mat.M,~]=fem.assema(gr,1,1,1); % mass
E=center2PointMatrix(gr); % to map elem differentiation matrices to nodal ones
p.mat.p2c=point2CenterMatrix(gr); % to interpolate from nodes to elem centers
[Dx,Dy]=fem.gradientMatrices(gr); p.mat.Dx=E*Dx; p.mat.Dy=E*Dy;
p.mat.Kx=fem.convection(gr,[1;0]); p.mat.Ky=fem.convection(gr,[0;1]);
p.idx=unique(p.pdeo.grid.e(1:2,:)); % store bdry-indizes for DBCs
```

```
function r=sG(p,u)  % PDE rhs
par=u(p.nu+1:end); H0=par(1); u=u(1:p.nu); al=par(3); % split in par and PDE u
N0=getN(p,p.X); X=p.X+u.*N0; H=getmeancurv(p,X); % mean curv.based on FEM
r=-2*H+p.mat.M*(H0*ones(p.nu,1)); r(p.idx)=u(p.idx)-al; % residual, and DBCs
```

```
function H=getmeancurv(p,X) % mean curv based on 1st and 2nd fundamental form
[E,F,G]=get1ff(p,X); [L,M,N]=get2ff(p,X); H=0.25*(L.*G-2*M.*F+N.*E)./(E.*G-F.^2);

function V=getV(p,u)
u=u(1:p.nu); N0=getN(p,p.X); X=p.X+u.*N0;
N=cross(p.mat.Dx*X,p.mat.Dy*X,2); % normal at X, NOT normalized
V=sum(p.mat.M*(dot(X,N,2)))/3;
```

Listing 16: `spcapinit, oosetfemops, sG, getmeancurv` and `getV` from `spcap1`. See sources for, e.g., `get1ff`, `get2ff`, and the Element2RefineSelector function `e2rs` used for mesh adaptation.

Despite the caveats in Remark A.1, we can now set up a simple script (Listing 17) and produce a continuation diagram and sample plots fully analogous to §3.1. For mesh–refinement, to select triangles to refine we use the areas on X, see `e2rs`, but otherwise the adaptive mesh refinement works as usual in the legacy setting of `pde2path` via `oomeshada`.

```
%% cmc spherical caps, init; pars will be overwritten in spcapinit
nx=10; al=0; h0=0; v0=0; a0=0; par=[h0; v0; al; a0]; % initial pars
p=spcapinit(nx,par); plotsol(p); p=setfn(p,'cap1'); p.sol.ds=0.01;
p.nc.dsmax=0.2; p.nc.usrlam=[2 4];  p.plot.bpcmp=1;
p.nc.ilam=[2 1]; p.nc.nq=1; p.fuha.qf=@qV; p.fuha.qfder=@qVjac; % cont V, free H
p.sw.jac=0; p.sw.qjac=1; % using numerical Jacs for G, and approximate q_u
p=cont(p,10); % go
%% example of mesh-adaption; loading soln useful for testing parameters
% such as ngen (number of adaption loops) and sig (frac of triangles to refine)
p=loadp('cap1','pt10','cap1r'); p.nc.ngen=1; p.nc.sig=0.2; p=oomeshada(p);
p=cont(p,20); % continue refined solution
```

Listing 17: Short script `cmds1.m` from `geomtut/spcap`.

As already said, the main advantage of this setup is simplicity in the sense that the function `getmeancurv` (based on `get1ff` and `get2ff`) is a direct translation of the differential geometric definition. Moreover, we can work with fixed preassembled differentiation matrices (independent of X) as long as the mesh (in Ω) is fixed. The main disadvantage is that this implementation of (11) is not a weak form but mixes FEM and FD differentiation matrices.

B Biocylinders with clamped BCs

In the demo `biocyl` we consider the Helfrich SC functional (69) for a cylindrical topology along the x–axis with BCs (74). The equation and BCs thus are

$$\Delta H + 2H(H^2 - K) + 2c_0 K - 2c_0^2 H - 2\lambda_1 H = 0, \tag{97a}$$

$$X_2^2 + X_3^2|_{X_1=\pm 1} = \alpha^2, \quad \text{and} \quad \partial_x X_2|_{X_1=\pm 1} = \partial_x X_3|_{X_1=\pm 1} = 0. \tag{97b}$$

For $c_0 = 0$ we have the explicit family

$$X_{\mathrm{cyl},\alpha} = (x, \alpha\cos\phi, \alpha\sin\phi), \quad x \in [-1,1], \ \phi \in [0, 2\pi), \ \text{with } \lambda_1 = \frac{1}{4\alpha^2}, \tag{98}$$

of Helfrich cylinders, and [DDG21] proves various existence results for axisymmetric solutions near (98) and in other regimes in the α–λ_1 plane. See also [VDM08] for other shapes with cylindrical topology, which fit into our setting by prescribing other contours at $x = \pm 1$ instead of the circles of radius α in (97b).

The external parameters for (97a) are (α, λ_1, c_0), and continuation in any of these yields interesting results. We first continue in c_0 at fixed $\alpha = 1$, since we believe this is the numerically most challenging case due to the exact solution (99) below. Subsequently we continue in α, and again in c_0 at different α, and finally in λ_1, always starting from the Helfrich cylinder (98).

Table 15: Scripts in `pde2path/demos/geomtut/biocyl`.

`cmds1.m`	Continuation of $X_{\mathrm{cyl},\alpha}$ in c_0, $(\alpha,\lambda_1)=(1,1/4)$, Figs. 31 and 32
`cmds2.m`	Continuation of $X_{\mathrm{cyl},\alpha}$ in α, $(c_0,\lambda_1)=(1/4,1)$, Fig. 33.
`cmds3.m`	Continuation of "big" and "small" cylinders $X_{\mathrm{cyl},\alpha}$ in c_0, i.e., $\alpha=1.4$ and $\alpha=0.6$, Fig. 34.
`cmds4.m`	Continuation of $X_{\mathrm{cyl},\alpha}$ like solutions in λ_1, $(\alpha,c_0)=(1,0.7)$, Fig. 36.
`cmds4b.m`	Like `cmds4.m` but with $(\alpha,c_0)=(1,0)$, Figs. 35 and 37.

Table 15 lists the command scripts in `biocyl/`. Besides `cylinit.m`, `sG.m`, and `getM.m` (see Remark 4.1), we then additionally have the PC `qfrot.m` and its derivative `qfrotder.m`, and the scripts `bdmov1.m` and `bdmov2.m` to produce movies of the BDs in Fig. 36. In `hcylbra.m` we put $A, V, E, \lambda_1 A$ and the mesh–quality data on the branch. In `cylinit.m` we set `p.sw.Xcont=2`, such that the colors in solution plots mainly give some visual structure to X, but do not in general give the continuation direction uN, cf. the remarks after (36). The PC multiplier $|s_{\mathrm{rot}}|<10^{-4}$ on all the branches, and in all cases the final u plotted is of order 10^{-6} or smaller.

B.1 Continuation in the spontaneous curvature

In `cmds1.m` we fix $\alpha=1$, and start with the Helfrich cylinder (98), hence $\lambda_1=1/4$. We choose a rather coarse uniform initial mesh of `np`=1770 points, and first continue to $c_0<0$, yielding the branch `c0b` in Fig. 31. The solutions contract in the middle and via two folds produce a bistability region around $c_0=-6.25$, but otherwise no bifurcations. As the neck thins, we refine the mesh based on `e2rsshape1`, cf. (**??**), with `p.sw.rlong=1` and combined with `retrigX` to avoid obtuse triangles, cf. Remark 2.6. The 2nd plot in (a) shows the distortion δ_{mesh}, with refinements at `pt25`, `pt35` and `pt40`, and the second row in (b) shows rather strong zooms into the necks, illustrating the refinement step from `pt35` to `pt36`, and the reasonable mesh in the neck at point `pt45`.

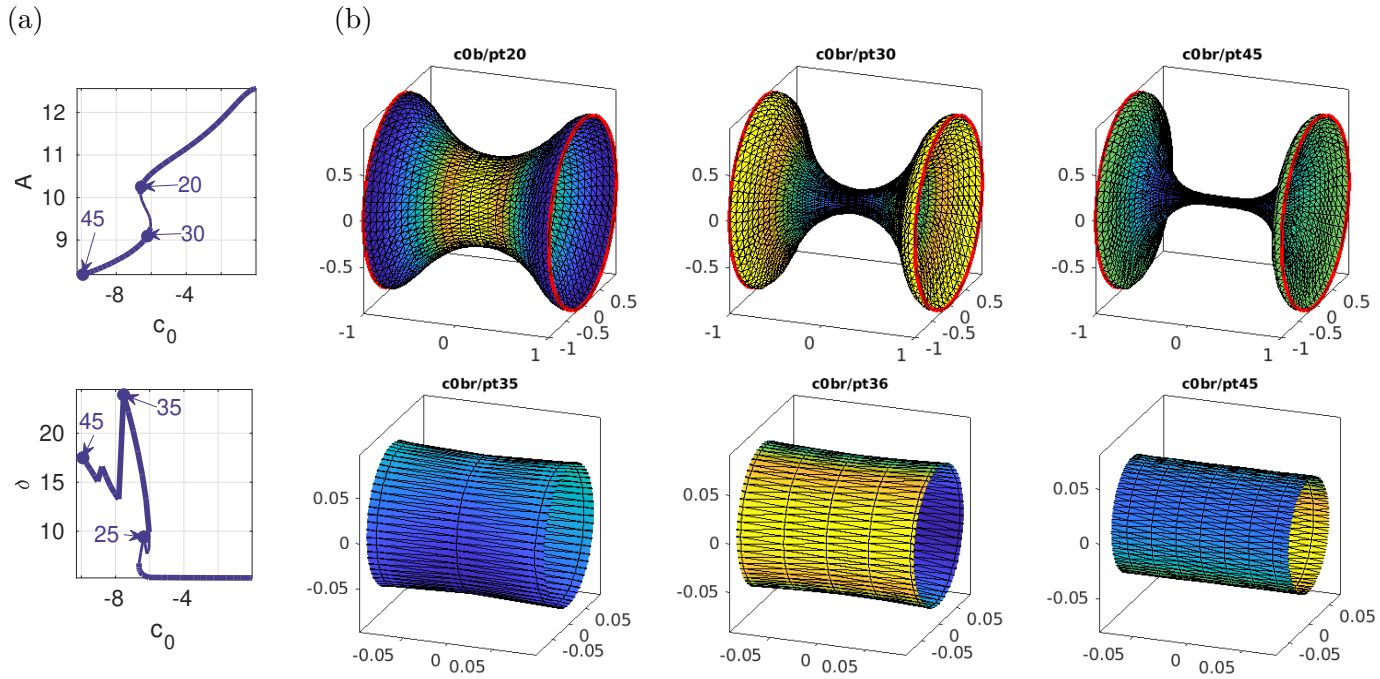

Figure 31: Results for (**??**) from `biocyl/cmds1.m`. $(\alpha,\varepsilon)=(1,1/4)$ fixed, continuation to $c_0<0$, with a bistability region near $c_0=-6.25$. Initially uniform mesh of $n_p=1770$ points, and the 2nd plot in (a) indicates that with suitable refinement (at `pt25` and `pt35`, to $n_p=3561$) the meshes stay good. Samples in (b), with zooms of the necks to illustrate the meshes.

In Fig. 32 we continue to $c_0>0$. This is initially easy, but after a first fold near $c_0\approx 1.3$ and then decreasing c_0 below $c_0\approx 0.15$ it becomes difficult to maintain mesh distortion $\delta<100$. We give the

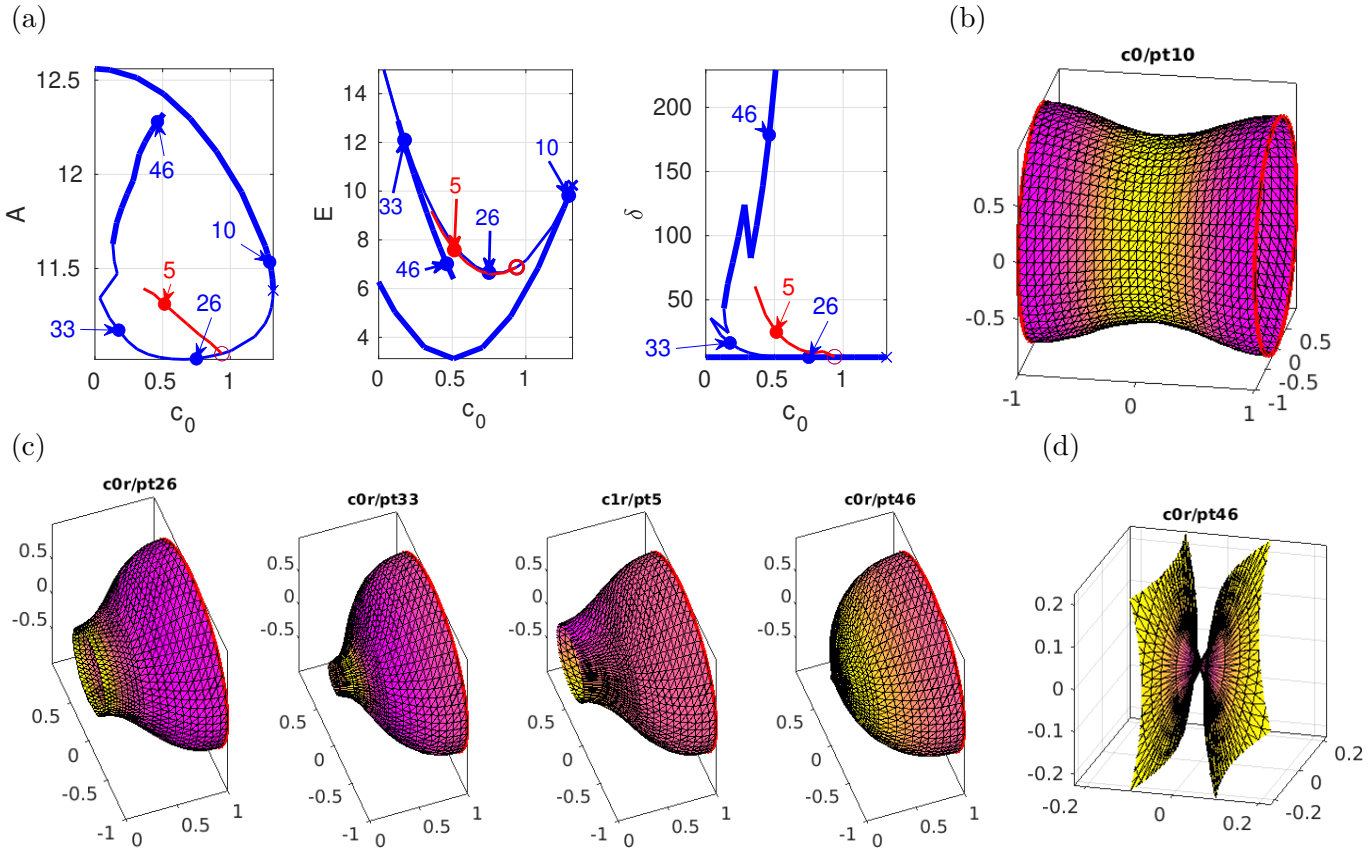

Figure 32: Further results from `biocyl/cmds1.m`, $(\alpha, \varepsilon) = (1, 1/4)$, continuation of Helfrich cylinder to $c_0 > 0$ (blue branch `c0`, `c0r` after mesh refinement), with bifurcation to `c1r` (red). (a) BDs of area A, energy E, and mesh distortion δ_{mesh}. (b) A full sample at `c0/pt10`, different colormap (yellow>pink) for better visibility of mesh. (c) further samples, cut at $x = 0$. (d) zoom into solution close to $X_{2\text{HS}}$. On `c0`, results after `pt33` become somewhat mesh–dependent, but our experiments suggest that the branch connects to $X_{2\text{HS}}$ at $c_0 = 1/2$ (with `c0r/pt46` shortly before that connection).

full result of our continuation but remark that the behavior on `c0` (`c0r` after refinement, to $n_p = 3130$ at `pt46`) after `pt30`, say, becomes mesh dependent. First, however, at $c_0 \approx 0.94$, we find a BP to a non–axisymmetric branch (`c1`, red), 3rd sample in (c). We then get a second fold at $c_0 \approx 0.08 \pm \eta$, with $\eta \in (-0.02, 0.02)$ dependent on the mesh. Relatedly, the further continuation becomes somewhat non–smooth and quantitatively depends on the mesh, but qualitatively we get similar behavior for different meshes, namely: The neck becomes thin and short, and the solutions seem to approximate a solution

$$X_{2\text{HS}} \text{ consisting of two hemispheres with radius 1, centered at } (x, y, z) = (\pm 1, 0, 0), \tag{99}$$

and hence touching at $(0, 0, 0)$, see `c0r/pt46`, at $c_0 = 0.47$. $X_{2\text{HS}}$ is an exact solution of (97) with $\lambda_1 = 1/4$ at $c_0 = 1/2$, and given our various experiments with mesh refinement we believe that the branch `c0r` connects to $X_{2\text{HS}}$. However, the third plot in (a), and the samples in (c) (at $c_0 \approx 0.47$, shortly before the supposed connection to $X_{2\text{HS}}$) show how the mesh quality seriously degrades as we approach the supposed connection.

Also, we have, e.g., $\text{ind}(X) = 3$ at `pt33` (with one unstable direction from the fold at $c_0 \approx 1.3$, and two from the double BP at $c_0 \approx 0.94$), while the "almost–$X_{2\text{HS}}$"–solutions near `pt46` have $\text{ind}(X) = 0$, i.e., are stable. Hence, since $\text{ind}(X)$ should decrease by one at the (supposed) fold between `pt33` and `pt46`, we expect another double BP between `pt33` and `pt46`. However, the behavior of the branch is quantitatively mesh–dependent also near the left fold, and while the changes in $\text{ind}(X)$ are detected, the bisection loops to localize the fold and/or the BPs do not converge. Thus, altogether the behavior

after `pt33` *is conjectured*. On the other hand, for $c_0 \in (0.3, 0.5)$, say, and in particular near `pt46`, the solutions like `pt46` stay stable under mesh refinement. In any case, a pinch–off (topological change) which should occur after `pt46` as we approach $X_{2\text{HS}}$ cannot be resolved with our methods, but probably requires the use of some phase field method, see, e.g., [DLW06].[28]

The results for `c0r` up to `pt33` are mesh independent, including the bifurcation of the non-axisymmetric branch `c1` at `c0/bpt1`. The mesh then also degrades on the non–axisymmetric branch (`c1`, and in a similar way like on the branch `c0` but at larger c_0), and since next we focus on non–axisymmetric branches in a slightly different setting we do not further pursue `c1` here.

B.2 Intermezzo: Other radii

In `cmds2.m` we continue $X_{\text{cyl},\alpha}$ in α, with fixed $(c_0, \lambda_1) = (0, 1/4)$, see Fig. 33. This shows no bifurcations. Numerically, the case $\alpha \searrow 0$ is challenging due to boundary layers developing at $x = \pm 1$, see [Gru12]. Alternating continuation and mesh adaptation based on `e2rsshape1` we can reliably continue to $\alpha = 0.05$ and slightly further.

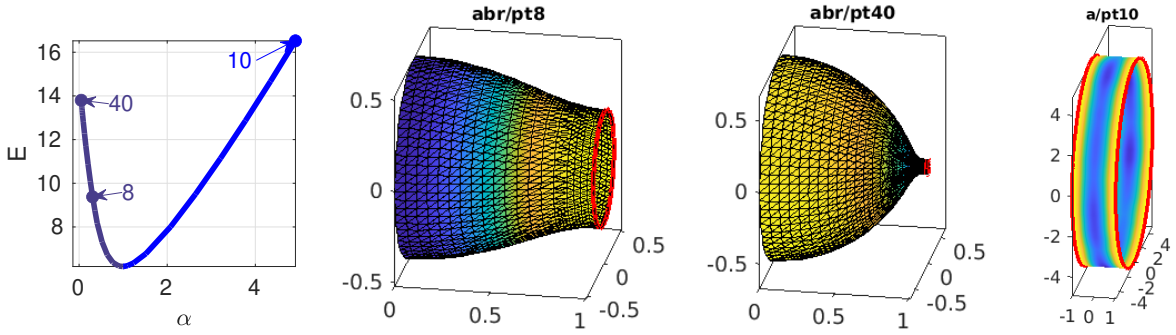

Figure 33: Results for (97) from `cmds2.m`, continuation in α.

In `cmds3.m` and Fig. 34 we repeat the continuation in c_0 from Fig. 31 and Fig. 32 for different starting cylinders $X_{\text{cyl},\alpha}$, namely $\alpha = 1.4$ in Fig. 34(a) and $\alpha = 0.6$ in (b–d). The main differences to the case $\alpha = 1$ are as follows: For both, $\alpha = 1.4$ and $\alpha = 0.6$, continuing to negative c_0 we no longer have the two folds and associated bistability range as in Fig. 31, and the branches seem to extend to arbitrary large negative c_0. We refrain from plotting this for $\alpha = 1.4$ in (a), and for $\alpha = 0.6$ in (b) we mainly remark that large negative c_0 yields very long and thin necks, which can be handled with adaptive mesh refinement as in Fig. 31.

Continuing to positive c_0, for $\alpha = 1.4$ in (a), the main difference to Fig. 32 is that the branch `c0r` only has one fold (at $c_0 \approx 2.3$) and then continues to negative c_0, see `b/c0r/pt51`. The difference to $\alpha = 1$ is that a solution like $X_{2\text{HS}}$ no longer exists, i.e., two hemispheres of radius $\alpha = 1.4$ (or any $\alpha \neq 1$) cannot be near a (genus 1) cylinder type solution. Additionally, the larger radius makes the continuation of the red non–axisymmetric branch `b/c1qr` easier wrt to mesh handling, see the last sample in (a). For $\alpha = 0.6$ in (c,d), the smaller radius gives a "pearling" behavior after the fold on the blue branch, and it seems likely that the branch `s/c0r` continues to a solution of type hemisphere–sphere–hemisphere, which would give similar problems as discussed for $X_{2\text{HS}}$ in Fig. 32. Here we stop the continuation after `pt30` (third sample in (d)), as further continuation requires excessive mesh adaptation and the solutions are unstable anyway. The pearling shape is also inherited by the non-axisymmetric branch `s/c1` (last sample in (d)).

[28] For Neumann BCs (74), hemispheres are highly degenerate in the following sense: Setting up (97) over a *single* circle at wlog at $X_1 = 1$, and wlog with $(\alpha, \lambda_1, c_0) = (1, 1/4, 1/2)$ the single unit hemisphere is again an exact solution, but we are not able to continue this in λ_1 or c_0, i.e., it seems isolated. This is different for BCs (75), see §**??**.

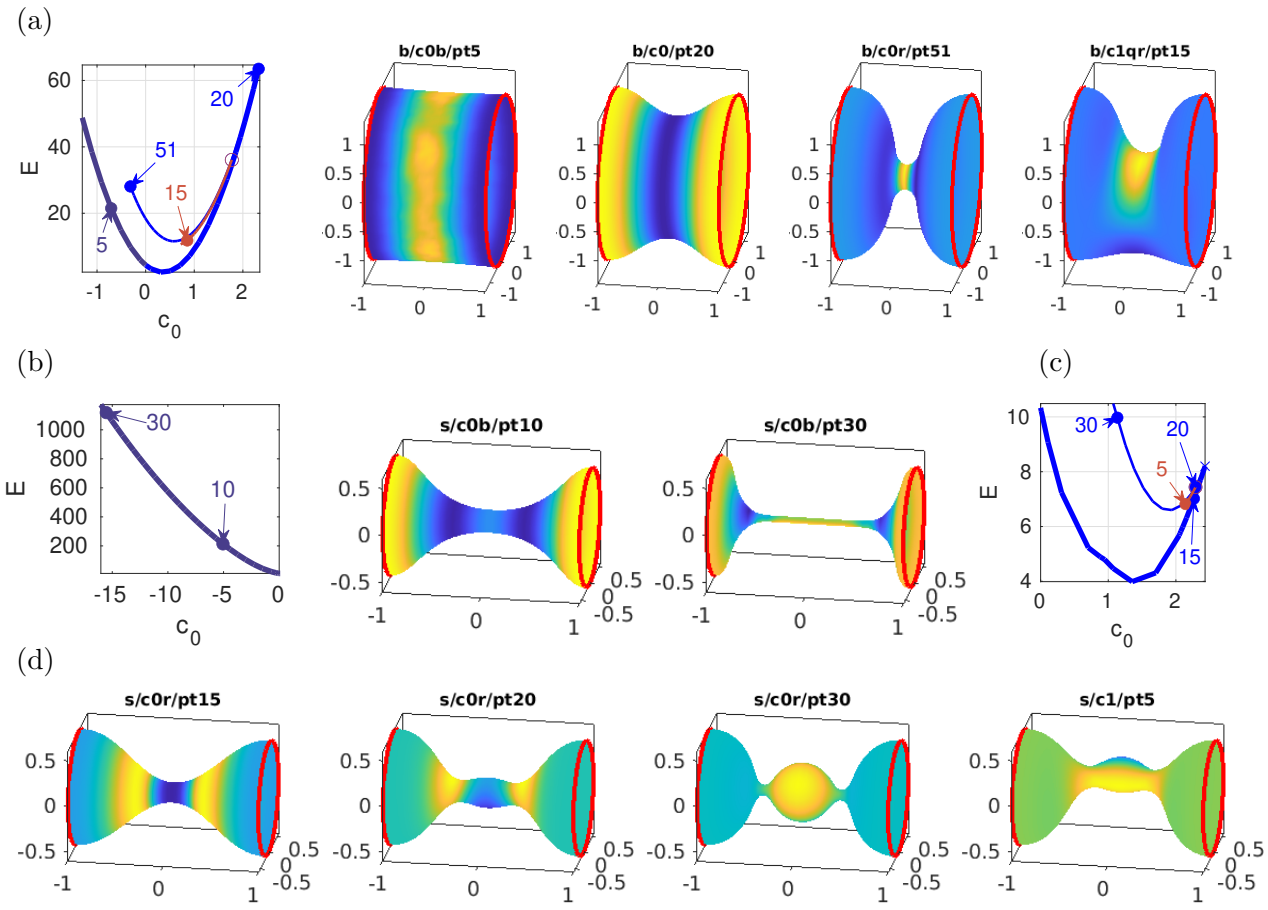

Figure 34: `cmds3.m`, continuation in c_0 for $(\alpha, \lambda_1) = (1.4, 1/5.6)$ in (a), and $(\alpha, \lambda_1) = (0.6, 2.4)$ in (b–d).

B.3 Continuation in surface tension

In `cmds4.m` we return to fixed $\alpha = 1$ and continue in λ_1, starting at $\lambda_1 = 0.25$ and $c_0 = 0.7$ corresponding to Fig 32, and in `cmds4b.m` we repeat this for $c_0 = 0$, starting from the genuine Helfrich cylinder $X_{\mathrm{cyl},\alpha}$. In both cases, for increasing λ_1 the solutions initially only slightly change shape, but at larger λ_1 (near $\lambda_1 = 8$ for $c_0 = 0$, and near $\lambda_1 = 4$ for $c_0 = 0.7$) we get an S–shaped bistability region due to two consecutive folds, similar to the case of $c_0 \approx -6$ in Fig. 31 (with fixed $\lambda_1 = 1/4$).[29] However, we find no bifurcations to non–axisymmetric branches. The case of decreasing λ_1 is more interesting, and in Fig. 35 we only show the basic result for $c_0 = 0$ and large $\lambda_1 > 0$ for completeness.

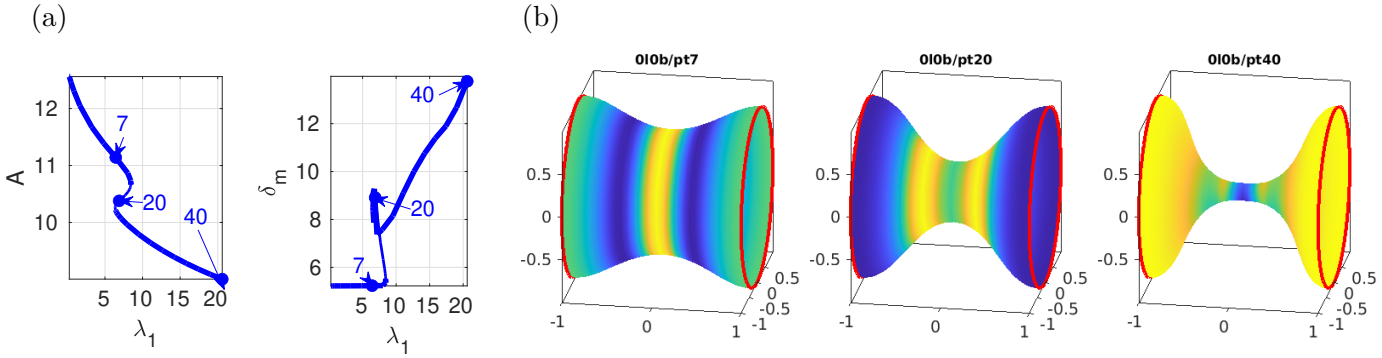

Figure 35: `cmds4b.m`, continuation of $X_{\mathrm{cyl},\alpha}$ in λ_1, $(c_0, \alpha) = (0, 1)$.

[29] Such folds were already found in [Doe17, §6.3] for $c_0 = 0$ in the 1D axisymmetric setting.

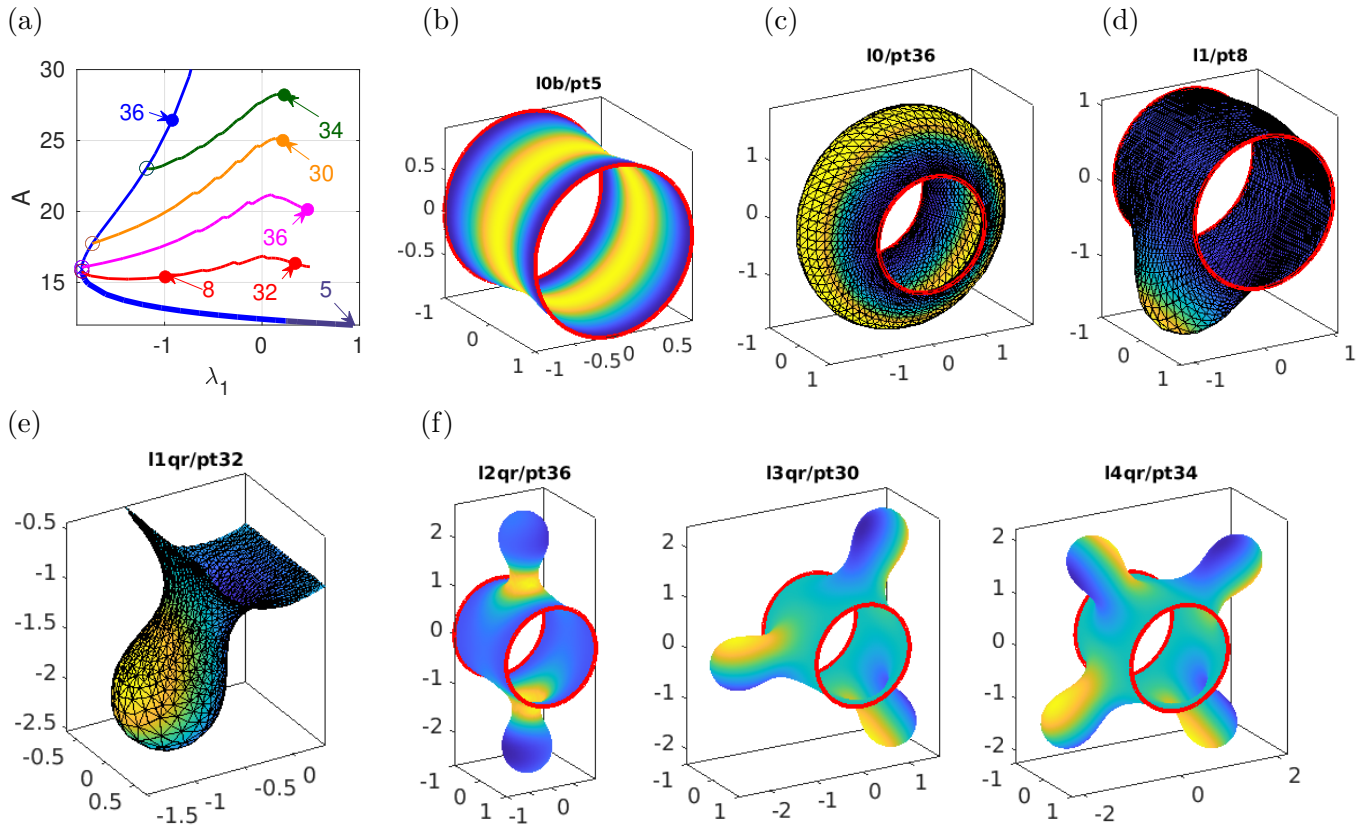

Figure 36: `cmds4.m`, continuation in λ_1 at $c_0 = 0.7$. In (a): axisymmetric branch `l0b` (dark blue, sample in (b)), `l0` (blue, sample in (c))), and four bifurcating branches, `l1` (red, refined to `l1qr`, samples in (d) and (e), with zoom), `l2` (magenta), `l3` (orange), `l4` (green), samples in (f). The kinks of `l1`–`l4` are due to adapative mesh refinement every 5th step (after pt10), from `np` $= 2700$ to `np` ≈ 3600 at the end of each branch.

Even if $\lambda_1 < 0$ is in general unphysical, see Remark **??**(b), continuation to $\lambda_1 < 0$ yields interesting results, in particular for $c_0 > 0$. For both, $c_0 = 0.7$ and $c_0 = 0$, for $\lambda_1 < 0$ the solutions start to bulge out in the "middle" (near $x = 0$), and there are folds around $\lambda_1 = -2$ after which we obtain pronounced "tire shapes". Moreover, we obtain bifurcations to D_m–symmetric branches, with increasing angular wave numbers $m = 1, 2, 3, 4, \ldots$. For $c_0 = 0.7$, the bifurcating branches return to $\lambda_1 > 0$, such that in Fig. 36(a–f) we obtain four new solutions at $\lambda_1 = 0.25$. For $c_0 = 0$, the D_m symmetric branches initially behave similarly, but do *not* reach $\lambda_1 > 0$ and instead asymptote to $\lambda_1 = 0_-$, with arbitrary large A, see Fig. 37 and again Remark **??**(b). Therefore we put $c_0 = 0.7$ first in Fig.36. Nevertheless, for both $c_0 = 0$ and $c_0 = 0.7$, the BCs do not allow "large portions of planes" as solutions and thus yield the folds of the blue branches. Moreover, E stays bounded below (see Fig. 38), and also for $c_0 = 0$ in Fig. 37 and Fig. 38(b) it seems that $\lambda_1 A \to 0$ as $\lambda_1 \nearrow 0$, i.e., that the area grows slower than $|1/\lambda_1|$.

The main issue in both, `cmds4.m` and `cmds4b.m` (which is essentially a copy of `cmds4.m`, with a different start (at $c_0 = 0$) of the blue branch and hence directory names starting with 0), is the mesh adaptation for the strong budding of the D_m symmetric branches. In detail, to compute the branch, e.g., `l1` ($m = 1$, sample in Fig. 36(d)) with refinement (sample in (e)) we proceed as follows. After branch switching at `l0/bpt1` (double, by rotational symmetry, but again we just choose the first of the two kernel vectors) we do a few steps without rotational PC, which we then switch on. Subsequently, we continue with mesh adaptation based on area every five continuation steps. This works very robustly, also on the branches with $m = 2, 3, 4$ and similarly for $c_0 = 0$ in Fig. 37, and allows continuation to large buds. This shows again that bulging *out* (i.e., expanding) is typically not problematic wrt meshes, though at mesh–refinement we obtain the kinks on the $m = 1, 2, 3, 4$ branches in Fig. 36(a). To have smoother branches we would need more frequent but less strong refinement, but this is clearly not critical.

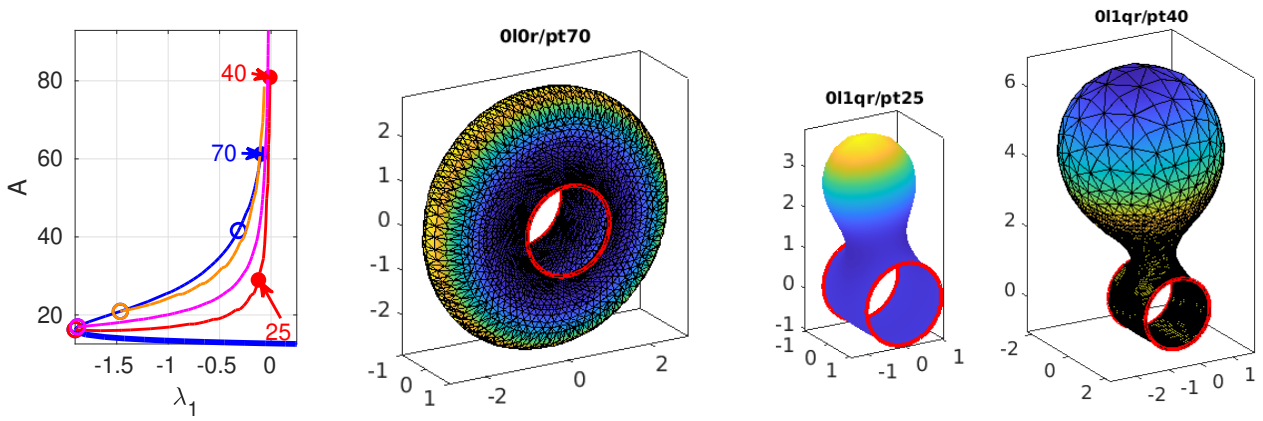

Figure 37: `cmds4b.m`, continuation in λ_1 at $c_0 = 0$; color code as in Fig.36, and, e.g., `np`=4400 at `0l1qr/pt40`.

The branches in Fig. 36 can be continued further by alternating `cont` and `refineX`, but this requires some fine–tuning of the `cont`–`refineX` loop parameters, and eventually the continuation of the branches `l*qr` fails again due to bad triangles in the necks. Thus, to keep the script `cmds4.m` simple we stop at the given points. Finally we note that the stabilities in Fig. 36 and Fig. 37 are as expected, namely $\mathrm{ind}(X) = m$ on the D_m branch(es).

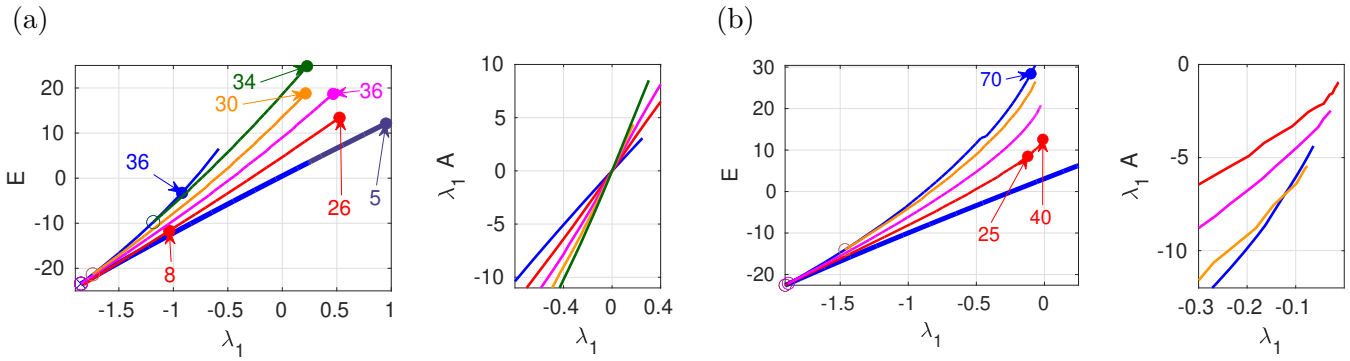

Figure 38: Energies E and the part $\lambda_1 A$ of E along the branches of Fig. 36; (a) $c_0 = 0.7$, (b) $c_0 = 0$.

References

[AHLL88] S Andersson, S. T. Hyde, K Larsson, and S. Lidin. Minimal surfaces and structures: from inorganic and metal crystals to cell membranes and biopolymers. *Chemical Reviews*, 88(1):221–242, 1988.

[ALP99] L. J. Alías, R. López, and B. Palmer. Stable constant mean curvature surfaces with circular boundary. *Proc. Amer. Math. Soc.*, 127(4):1195–1200, 1999.

[BF67] R. Bowen and St. Fisk. Generations of triangulations of the sphere. *Mathematics of Computation*, 21:250–252, 1967.

[BGBC22] M. Bottacchiari, M. Gallo, M. Bussoletti, and C. Casciola. Activation energy and force fields during topological transitions of fluid lipid vesicles. *Commun. Phys.*, 5:12, 2022.

[BGN15] J. Barrett, H. Garcke, and R. Nürnberg. Numerical computations of the dynamics of fluidic membranes and vesicles. *PRE*, 92:052704, 2015.

[BGN20] J. Barrett, H. Garcke, and R. Nürnberg. Parametric finite element approximations of curvature-driven interface evolutions. In *Geometric partial differential equations. Part I*, volume 21 of *Handb. Numer. Anal.*, pages 275–423. Elsevier, Amsterdam, 2020.

[BGNZ22] Weizhu Bao, H. Garcke, R. Nürnberg, and Quan Zhao. Volume-preserving parametric finite element methods for axisymmetric geometric evolution equations. *J. Comput. Phys.*, 460:Paper No. 111180, 23, 2022.

[BNP10] A. Bonito, R. H. Nochetto, and M. S. Pauletti. Geometrically consistent mesh modification. *SIAM J. Numer. Anal.*, 48(5):1877–1899, 2010.

[Bol11] M. Bollhöfer. ILUPACK V2.4, `www.icm.tu-bs.de/~bolle/ilupack/`, 2011.

[Bra96] K. Brakke. The surface evolver and the stability of liquid surfaces. *Philos. Trans. Roy. Soc. London Ser. A*, 354(1715):2143–2157, 1996.

[Bra23] K. Brakke. Triply Periodic Minimal Surfaces, 2023. `http://facstaff.susqu.edu/brakke/evolver/examples/periodic/periodic.html`.

[Bru18] N. D. Brubaker. A continuation method for computing constant mean curvature surfaces with boundary. *SIAM J. Sci. Comput.*, 40(4):A2568–A2583, 2018.

[BT84] M. J. Beeson and A. J. Tromba. The cusp catastrophe of Thom in the bifurcation of minimal surfaces. *Manuscripta Math.*, 46(1-3):273–308, 1984.

[CK97] T. K. Callahan and E. Knobloch. Symmetry-breaking bifurcations on cubic lattices. *Nonlinearity*, 10:1179–1216, 1997.

[CR71] M. G. Crandall and P. H. Rabinowitz. Bifurcation from simple eigenvalues. *J. Functional Analysis*, 8:321–340, 1971.

[DCF+97] E. Doedel, A. R. Champneys, T. F. Fairgrieve, Y. A. Kuznetsov, B. Sandstede, and X. Wang. AUTO: Continuation and bifurcation software for ordinary differential equations (with HomCont). `http://indy.cs.concordia.ca/auto/`, 1997.

[DDG21] K. Deckelnick, M. Doemeland, and H-C. Grunau. Boundary value problems for a special Helfrich functional for surfaces of revolution: existence and asymptotic behaviour. *Calc. Var. Partial Differential Equations*, 60:32, 2021.

[DE13] G. Dziuk and Ch. M. Elliott. Finite element methods for surface PDEs. *Acta Numer.*, 22:289–396, 2013.

[DEK+97] H.-G. Döbereiner, E. Evans, M. Kraus, U. Seifert, and M. Wortis. Mapping vesicle shapes into phase diagrams: A comparison of experiment and theory. *PRE*, 55(4):4485–4474, 1997.

[Des04] M. Deserno. Notes on differential geometry, `https://www.cmu.edu/biolphys/deserno/pdf/diff_geom.pdf`, 2004.

[DLW06] Qiang Du, Chun Liu, and Xiaoqiang Wang. Simulating the deformation of vesicle membranes under elastic bending energy in three dimensions. *J. Comput. Phys.*, 212(2):757–777, 2006.

[Doe17] M. Doemeland. Axialsymmetrische Minimierer des Helfrich-Funktionals, Master Thesis, available at `http://www-ian.math.uni-magdeburg.de/home/grunau/papers/Doemeland_Master.pdf`, 2017.

[DS13] H. Dankowicz and F. Schilder. *Recipes for continuation*, volume 11 of *Comp. Sc. & Eng.* SIAM, Philadelphia, PA, 2013.

[dWDR+23] H. de Witt, T. Dohnal, J.D.M. Rademacher, H. Uecker, and D. Wetzel. pde2path - Quickstart guide and reference card, 2023. Available at [Uec24].

[ES98] J. Escher and G. Simonett. The volume preserving mean curvature flow near spheres. *Proc. Amer. Math. Soc.*, 126(9):2789–2796, 1998.

[ES13] C. M. Elliott and B. Stinner. Computation of two-phase biomembranes with phase dependent material parameters using surface finite elements. *Commun. Comput. Phys.*, 13(2):325–360, 2013.

[ES18] Norio Ejiri and Toshihiro Shoda. The Morse index of a triply periodic minimal surface. *Differential Geometry and its Applications*, 58:177–201, 2018.

[FS21] S. A. Funken and A. Schmidt. A coarsening algorithm on adaptive red-green-blue refined meshes. *Numer. Algorithms*, 87(3):1147–1176, 2021.

[FVKG22] B. Foster, N. Verschueren, E. Knobloch, and L. Gordillo. Pressure-driven wrinkling of soft inner-lined tubes. *New J. Phys.*, 24(January):Paper No. 013026, 15, 2022.

[FW14] S. Fujimori and M. Weber. A contruction method for triply periodic minimal surfaces. In *Proceedings of the 16th OCU*, pages 79–90. OCAMI Studies, 2014.

[GBMK01] P. J. F. Gandy, S. Bardhan, A. L. Mackay, and J. Klinowski. Nodal surface approximations to the P,G,D and I-WP triply periodic minimal surfaces. *Chemical Physics Letters*, 336:187–195, 2001.

[GK00] P. J. F. Gandy and J. Klinowski. Exact computation of the triply periodic Schwarz P minimal surface. *Chemical Physics Letters*, 322:579–586, 2000.

[Gru12] H.-C. Grunau. The asymptotic shape of a boundary layer of symmetric Willmore surfaces of revolution. In *Inequalities and Applications '10. Dedicated to the memory of Wolfgang Walter. Selected papers of the 2nd conference on inequalities and applications, Hajdúszoboszló, Hungary, September 19–25, 2010*, pages 19–29. Basel: Birkhäuser, 2012.

[GS02] M. Golubitsky and I. Stewart. *The symmetry perspective*. Birkhäuser, Basel, 2002.

[Har13] D. Hartley. Motion by volume preserving mean curvature flow near cylinders. *Comm. Anal. Geom.*, 21(5):873–889, 2013.

[Hel73] W. Helfrich. Elastic properties of lipid bilayers: Theory and possible experiments. *Zeitschrift für Naturforschung*, 28:693, 1973.

[Hof90] D. A. Hoffman. Some basic facts, old and new, about triply periodic embedded minimal surfaces. *J. Physique*, 51:197–208, 1990. Intern. Workshop on Geometry and Interfaces (Aussois, 1990).

[Hoy06] R.B. Hoyle. *Pattern formation*. Cambridge University Press., 2006.

[Jac13] A. Jacobson. Algorithms and interfaces for real- time deformation of 2D and 3D shapes, `https://doi.org/10.3929/ethz-a-00979066`, 2013.

[Jac24] A. Jacobson. gptoolbox, `https://github.com/alecjacobson/gptoolbox`, 2024.

[JQJZC98] Yan Jie, Liu Quanhui, Liu Jixing, and Ou-Yang Zhong-Can. Numerical observation of nonaxisymmetric vesicles in fluid membranes. *Phys. Rev. E*, 58:4:4730–4736, 1998.

[Kie12] H. Kielhöfer. *Bifurcation theory*. Springer, New York, second edition, 2012. An introduction with applications to partial differential equations.

[KIPM+20] V. Kralj-Iglc, G. Pocsfalvi, L. Mesarec, V. Šuštar, H. Hägerstrand, and A. Iglic. Minimizing isotropic and deviatoric membrane energy–an unifying formation mechanism of different cellular membrane nanovesicle types. *PLOS ONE*, 2020.

[KN06] Y. Kohsaka and T. Nagasawa. On the existence of solutions of the Helfrich flow and its center manifold near spheres. *Differential Integral Equations*, 19(2):121–142, 2006.

[KPP15] M. Koiso, B. Palmer, and P. Piccione. Bifurcation and symmetry breaking of nodoids with fixed boundary. *Adv. Calc. Var.*, 8(4):337–370, 2015.

[KPP17] M. Koiso, B. Palmer, and P. Piccione. Stability and bifurcation for surfaces with constant mean curvature. *Journal of the Mathematical Society of Japan*, 69(4):1519 – 1554, 2017.

[KPS18] M. Koiso, P. Piccione, and T. Shoda. On bifurcation and local rigidity of triply periodic minimal surfaces in $\mathbb{R}^3$. *Ann. Inst. Fourier (Grenoble)*, 68(6):2743–2778, 2018.

[Lip14] R. Lipowsky. Coupling of bending and stretching deformations in vesicle membranes. *Advances in colloid and interface science*, 208:14–24, 2014.

[Lóp13] R. López. *Constant Mean Curvature Surfaces with Boundary.* Springer, 2013.

[LWM08] G. Lim, M. Wortis, and R. Mukhopadhyay. Red blood cell shapes and shape transformations: Newtonian mechanics of a composite membrane. In *Soft Matter: Lipid Bilayers and Red Blood Cells, Volume 4*, pages 83–254, 2008.

[Man11] C. Mantegazza. *Lecture notes on mean curvature flow.* Birkhäuser/Springer Basel, 2011.

[MDHV13] I. M Mladenov, P. A. Djondjorov, M. T. Hadzhilarova, and V. M. .Vassilev. Equilibrium configurations of lipid bilayer membranes and carbon nanostructures. *Comm. Theor. Phys.*, 59(2):213–228, 2013.

[MDSB03] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, Math. Vis., pages 35–57. Springer, Berlin, 2003.

[Mla02] I. M. Mladenov. Delaunay surfaces revisited. *C. R. Acad. Bulgare Sci.*, 55(5):19–24, 2002.

[MMCRH13] M. Mercker, A. Marciniak-Czochra, T. Richter, and D. Hartmann. Modeling and computing of deformation dynamics of inhomogeneous biological surfaces. *SIAM J. Appl. Math.*, 73(5):1768–1792, 2013.

[MP02] R. Mazzeo and Fr. Pacard. Bifurcating nodoids. In *Topology and geometry: commemorating SISTAG*, volume 314 of *Contemp. Math.*, pages 169–186. AMS, Providence, RI, 2002.

[MU24] A. Meiners and H. Uecker. Supplementary information for *Differential geometric bifurcation problems in* `pde2path`, `www.staff.uni-oldenburg.de/hannes.uecker/pde2path/tuts`, 2024.

[Nit76] J. C. C. Nitsche. Non-uniqueness for Plateau's problem. A bifurcation process. *Ann. Acad. Sci. Fenn. Ser. A I Math.*, 2, 1976.

[Nit91] J. C. C. Nitsche. Periodic surfaces which are extremal for energy functionals containing curvature functions. IMA Preprint Series #785, 1991.

[Nit93] J. C. C. Nitsche. Boundary value problems for variational integrals involving surface curvatures. *Quart. Appl. Math.*, 51(2):363–387, 1993.

[NT03] T. Nagasawa and I. Takagi. Bifurcating critical points of bending energy under constraints related to the shape of red blood cells. *Calc. Var. PDEs*, 16(1):63–111, 2003.

[Oss14] R. Osserman. *A Survey of Minimal Surfaces.* Dover reprint of the 1968 edition, 2014.

[OYT14] Z. C. Ou-Yang and Z. C. Tu. Overview of the study of complex shapes of fluid membranes, the Helfrich model and new applications. In *Proceedings of the conference in honour of the 90th birthday of Freeman Dyson*, pages 277–287. World Sci. Publ., Hackensack, NJ, 2014.

[Pet83] Mark A. Peterson. An instability of the red blood cell shape. *Journal of Applied Physics*, 57:1739–1742, 1983.

[Pla73] J. Plateau. *Experimental and theoretical statics of liquids subject to molecular forces only*, translated by K. Brakke, `facstaff.susqu.edu/brakke/plateaubook/plateaubook.html`, 1873.

[PP22] B. Palmer and Á. Pámpano. The Euler-Helfrich functional. *Calc. Var. Partial Differential Equations*, 61(3):Paper No. 79, 28, 2022.

[PS04] P. Persson and G. Strang. A simple mesh generator in matlab. *SIAM Review*, 46(2):329–345, 2004.

[Ros92] M. Ross. Schwarz' P and D surfaces are stable. *Differential Geom. Appl.*, 2(2):179–195, 1992.

[Ros05] W. Rossman. The first bifurcation point for Delaunay nodoids. *Experiment. Math.*, 14(3):331–342, 2005.

[Ros08] A. Ros. Stability of minimal and constant mean curvature surfaces with free boundary. *Matematica Contemporanea*, 35:221–240, 2008.

[RU19] J.D.M. Rademacher and H. Uecker. The OOPDE setting of pde2path – a tutorial via some Allen-Cahn models, 2019. Available at [Uec24].

[Ruc81] H. Ruchert. A uniqueness result for Enneper's minimal surface. *Indiana Univ. Math. J.*, 30(3):427–431, 1981.

[SAR97] L. Slobozhanin, J. Alexander, and A. Resnick. Bifurcation of the equilibrium states of a weightless liquid bridge. *Phys. Fluids*, 9(7):1893–1905, 1997.

[SBL90] U. Seifert, K. Berndl, and R. Lipowsky. Shape transformations of vesicles: Phase diagram. *PRA*, 44:1182–1202, 1990.

[Sch22] A. Schmidt. ameshcoars, `https://github.com/aschmidtuulm/ameshcoars`, 2022.

[Sei97] U. Seifert. Configurations of fluid membranes and vesicles. *Advances in Physics*, 46(1):13–137, 1997.

[Sei99] U. Seifert. Giant vesicles: A theoretical perspective. In P. L. Luisi and P. Walde, editors, *Perspectives in Supramolecular Chemistry: Giant Vesicle*, volume 6, pages 71–91. Wiley, 1999.

[She02] J. R. Shewchuk. What is a good linear element? Interpolation, conditioning, and quality measures. In *Eleventh International Meshing Roundtable (Ithaca, New York)*, pages 115–126, 2002.

[SL95] U. Seifert and R. Lipowsky. Morphology of Vesicles. In R.Lipowsky and E. Sackmann, editors, *Handbook of Biological Physics*, volume 1, pages 403–463. Elsevier, 1995.

[STFH06] G. E. Schröder-Turk, A. Fogden, and S. T. Hyde. Bicontinuous geometries and molecular self-assembly: comparison of local curvature and global packing variations in genus-three cubic, tetragonal and rhombohedral surfaces. *Europ.Phys.J.B - Condensed Matter and Complex Systems*, 54(4):509–524, 2006.

[SZ89] S. Svetina and B. Zeks. Membrane bending energy and shape determination of phospholipid vesicles and red blood cells. *Eur.Biophys J,*, 17(2):101–111, 1989.

[Tap16] K. Tapp. *Differential geometry of curves and surfaces*. Springer, [Cham], 2016.

[TN20] Naoki Tamemoto and Hiroshi Noguchi. Pattern formation in reaction–diffusion system on membrane with mechanochemical feedback. *Scientific reports*, 10(19582), 2020.

[Uec21] H. Uecker. *Numerical continuation and bifurcation in Nonlinear PDEs*. SIAM, Philadelphia, PA, 2021.

[Uec24] H. Uecker. `www.staff.uni-oldenburg.de/hannes.uecker/pde2path`, 2024.

[UY17] Masaaki Umehara and Kotaro Yamada. *Differential geometry of curves and surfaces*. World Scientific, 2017. Translated from the second (2015) Japanese edition by W.Rossman.

[VDM08] V. M. Vassilev, P. A. Djondjorov, and I. M. Mladenov. Cylindrical equilibrium shapes of fluid membranes. *J. Phys. A*, 41(43):435201, 16, 2008.

[Vel20] R. Veltz. BifurcationKit.jl, `https://hal.archives-ouvertes.fr/hal-02902346`, 2020.

[War08] M. Wardetzky. Convergence of the cotangent formula: an overview. In *Discrete differential geometry*, volume 38 of *Oberwolfach Semin.*, pages 275–286. Birkhäuser, Basel, 2008.

[WBD97] A. De Wit, P. Borckmans, and G. Dewel. Twist grain boundaries in 3D lamellar Turing structures. *Proc. Nat. Acad. Sci.*, 94:12765–12768, 1997.

[WDS96] W. Wintz, H.-G. Döbereiner, and U. Seifert. Starfish vesicles. *EPL*, 33(5):403–408, 1996.

[Xu04] Guoliang Xu. Convergent discrete Laplace–Beltrami operators over triangular surfaces. *Computer Aided Geometric Design*, 21:767–784, 2004.

[XX09] Zhiqiang Xu and Guoliang Xu. Discrete schemes for Gaussian curvature and their convergence. *Comput. Math. Appl.*, 57(7):1187–1195, 2009.