Pattern formation with `pde2path` – a tutorial

Hannes Uecker

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

September 12, 2020

Abstract

We explain some `pde2path` setups for pattern formation in 1D, 2D and 3D. A focus is on steady bifurcation points of higher multiplicity, typically due to discrete symmetries, but we also review general concepts of pattern formation and their handling in `pde2path`, including localized patterns and homoclinic snaking, again in 1D, 2D and 3D, based on the demo `sh` (Swift–Hohenberg equation). Next, the demos `schnakpat` (a Schnakenberg reaction–diffusion system) and `chemtax` (a quasilinear RD system with cross–diffusion from chemotaxis) simplify and unify previous results in a simple and concise way, `ch` (Cahn-Hilliard) and `fCH` (functionalized Cahn–Hilliard) deal with mass constraints, `hexex` deals with (multiple) branch points of higher degeneracy in a scalar problem on a hexagonal domain, and `shgc` illustrates some global coupling. The demos `acS, actor,schnakS` and `schnaktor` (the Allen–Cahn and Schnakenberg models on spheres and tori) consider pattern formation on curved surfaces, `cpol` considers a problem of cell polarization described by bulk–surface coupling, and `bruosc` (Brusselator) explains how to augment autonomous systems by a time periodic forcing. Along the way we also comment on the choice of meshes, on time integration, and we give some examples of branch point continuation and Hopf point continuation to approximate stability boundaries.

MSC: 35J47, 35B32, 37M20

Contents

1 Introduction

The `Matlab` bifurcation and continuation package `pde2path` [UWR14, Uec20b] can be used to study solution branches and bifurcations in pattern forming systems (PFS), in particular reaction diffusion systems of the form

$$M_d\partial_t u = D\Delta u + f(u) =: -G(u,\lambda), \quad u = u(x,t) \in \mathbb{R}^N,\ t \geq 0,\ x \in \Omega, \tag{1}$$

where $\Omega \subset \mathbb{R}^d$ is a bounded domain, $d = 1, 2, 3$ (1D, 2D and 3D case, respectively), $D \in \mathbb{R}^{N\times N}$ is a positive (semi-)definite diffusion matrix, $\Delta = \partial^2_{x_1} + \ldots + \partial^2_{x_d}$, where the "reaction part" f is a smooth function, where λ in $G(u,\lambda)$ stands for one or several parameters present, and where (1) can be completed by various kinds of boundary conditions (BCs). Moreover, the dynamical mass matrix $M_d \in \mathbb{R}^{N\times N}$ in (1) may be singular, allowing to also write elliptic–parabolic systems in the form (1), or to rewrite some 4th order problems such as the Swift–Hohenberg equation in the form (1) in a consistent way. See, e.g., [UWR14, §4.2] and [UW14, Uec16, Wet16, BGUY17, ZUFM17, Uec19a] for examples, mostly related to pattern formation and Turing bifurcations [Mur89].

In applications (discrete) symmetries of the domain often enforce higher multiplicities of BPs. For instance, for Turing bifurcations over square domains with Neumann BCs we have "stripes in x_1" and "stripes in x_2" as two kernel vectors, and altogether we obtain three (modulo discrete spatial shifts)

bifurcating branches, namely stripes (twice) and spots as a superposition of stripes. In the following, we always use

$$m = \dim N(G_u(u_0, \lambda_0)) \tag{2}$$

to denote the dimension of the kernel of $G_u(u_0, \lambda_0)$, and call this m the *multiplicity of the BP* (u_0, λ_0).

The (analytically) higher multiplicity $m \geq 2$ of BPs in situations as above can be circumvented by some tricks, which for instance have been used in [UWR14, UW14]. Essentially we can exploit the fact that even on ideal domains, the discretization breaks up multiple BPs, and/or we can strengthen this breakup by slightly distorting the domain. However, besides the lack of elegance, using these tricks has some serious disadvantages: (a) The localization of close together simple BPs (obtained from the breakup of multiple BPs) is quite inefficient. (b) The branching behavior at the (artificially) simple BPs is in general quite different from that at the originally multiple BP. For instance, two simple stripes may hide the spots also present. This then requires further tricks/analytical understanding to relate the numerics to the true analytical situation.

Algorithms for branch switching at steady BPs of higher multiplicity, aimed particularly at pattern formation in $d \geq 2$ space dimensions, have been implemented in `pde2path` since 2018. Here we explain these in a somewhat wider context and review in a tutorial style some general ideas of applying `pde2path` to PFS in 1D, 2D and 3D. To make the tutorial somewhat self-contained, in §2 we briefly review some basics of PFS, in particular those related to amplitude equations and symmetries, using the Swift–Hohenberg (SH) equation as an example problem.

Table 1 lists the demos discussed in this tutorial. In §3 and §4 we explain the `pde2path` demos `sh` and `schnakpat`, which implement the SH equation and the Schnakenberg reaction diffusion system, respectively, over various 1D, 2D and 3D domains, mostly with homogeneous Neumann BCs (NBCs). In particular, the demo `sh` also explains how to use M in (1) to rewrite the 4th order SH equation as a 2–component 2nd order system in a consistent way, and `schnakpat` simplifies and unifies in a concise way many of the results from [UWR14, §4.2] and [UW14]. Additionally, we also consider some periodic BCs (pBCs) to illustrate how to deal with the interplay of discrete and continuous symmetries.

In §5 we collect some shorter demos. These deal for instance with approximation of the Eckhaus instability curve (in the SH equation) via BP continuation, with mass constraints (in the Cahn–Hilliard (CH) problem and a 'functionalized' CH problem), with multiple branch–points of higher order indeterminacy (in a scalar problem on a hexagonal domain), with a quasilinear chemotaxis problem, and with global coupling (again in a SH equation), which requires some customized linear system and eigenvalue solvers.

In §6 we consider pattern formation on curved surfaces. The case of the sphere is in particular interesting from a symmetry point of view, yielding BPs of rather high multiplicity, and again requires to deal with both discrete and continuous symmetries at bifurcation. Moreover, we give an example how to patch together problems living on different domains and only coupled via a common boundary, and we consider a simple model for cell–polarization with a bulk–surface coupling.

While in §3-§6 we restrict to steady patterns, in §7 we give an outlook on oscillatory patterns, including secondary bifurcations of periodic orbits such as period doubling, and explain a trick how to consider time periodic forcing. Along the way we also comment on the choice of meshes in 2D and 3D (§3.7), and on tips and tricks (§3.8, including deflation and time–integration, aka direct numerical simulation (DNS)) how to deal with problems which are characterized by a high multiplicity

Table 1: Subdirectories of `/demos/pftut`.

sh	The Swift–Hohenberg (SH) equation on 1D, 2D and 3D boxes with homogeneous Neumann BCs, 'main' demo directory, §3
shpbc	The SH in 2D with pBCs, §3.6
shEck	SH, 1D, with branch–point continuation to approximate the Eckhaus instability, §5.1
shgc	SH with global coupling, to illustrate customized linear system solvers, §5.6
schnakpat	A Schnakenberg 2-component reaction diffusion model, §4
ch	Cahn-Hilliard (CH) model, to illustrate a mass constraint setup, §5.2
fCH	a 'functionalized' CH model, §5.3
hexex	A scalar problem on a hexagonal domain, with multiple BPs of higher degeneracy, §5.4
chemtax	A quasilinear RD system modeling chemotaxis, §5.5
actor, acS	Allen–Cahn equations on tori and spheres, based on Laplace–Beltrami operators, §6.1 and §6.2, mainly as a preparation for schnakS and schnaktor
schnakS/tor	The Schnakenberg model on spheres and tori, §6.3 and §6.4, also §6.5 for cones.
accyl	An Allen-Cahn eqn on a cylinder with boundary coupling to a Poisson eqn, §6.6
cpol	a model for cell–polarization with bulk–surface coupling, §6.7
bruosc	The Brusselator, with oscillating Turing patterns and period doubling, §7

of solutions. The software `pde2path`, including all the demo directories and a number of further tutorials can be downloaded at [Uec20b].

2 Some theory: pattern formation in the Swift-Hohenberg equation

Consider the (quadratic-cubic) Swift-Hohenberg (SH) equation

$$\partial_t u = -(1+\Delta)^2 u + \lambda u + \nu u^2 - u^3, \quad u = u(x,t) \in \mathbb{R},\ x \in \Omega \subset \mathbb{R}^d, \tag{3}$$

with instability parameter $\lambda \in \mathbb{R}$, second parameter $\nu \in \mathbb{R}$, and BCs $\partial_n u|_{\partial\Omega} = \partial_n(\Delta u)|_{\partial\Omega} = 0$. The original (cubic) SH model [SH77] corresponds to $\nu = 0$, while the case $f(u) = \nu u^3 - u^5$ instead of $f(u) = \nu u^2 - u^3$ is called the cubic-quintic SH equation. Swift–Hohenberg equations of this type are canonical and much studied model problems for pattern formation in dissipative system [CH93, Pis06, SU17]. For later comparison with the numerics, we start with some theory for (3), already using numerical results from the `pde2path` demo directory `sh` for illustration, but conversely no problem specific analytical results (except of symmetries) are used in the numerics.

For us, the main advantage of the SH equation compared to RD systems of type (1), which may show exactly the same type of (Turing) instabilities, is that the SH equation allows much simpler and explicit computation of the amplitude equations on the center manifold at bifurcation from the trivial branch. Additionally, (3) is a gradient system $\partial_t u = -\nabla\mathcal{E}(u)$ wrt the energy

$$\mathcal{E}(u) = \int_\Omega \frac{1}{2}((1+\Delta)u)^2 - \frac{1}{2}\lambda u^2 - F(u)\,\mathrm{d}x, \quad F(u) = \int_0^u f(v)\,\mathrm{d}v, \tag{4}$$

where either $\Omega = \mathbb{R}^d$ or Ω a bounded domain and as above we assume the homogeneous Neumann BCs $\partial_x u|_{\partial\Omega} = \partial_x \Delta u|_{\partial\Omega} = 0$. In particular, local minima of $\mathcal{E}$ are stable stationary solutions of (3), and (3) does not have time–periodic solutions (with finite energy). Moreover, the translational invariance

of $\mathcal{E}$ yields the existence of a spatially conserved quantity for steady solutions, a Hamiltonian, cf., e.g., [ALB+10, Proposition 1]. If for instance we consider the steady problem in a spatial dynamics formulation in 1D, i.e., $U = (u_1, u_2, u_3, u_4) := (u, \partial_x u, \partial_x^2 u, \partial_x^3 u)$ such that

$$\frac{\mathrm{d}}{\mathrm{d}x} U = \big(u_2, u_3, u_4, -2u_2 - (1-\lambda)u_1 + f(u_1)\big)^T,$$

then the Hamiltonian, written as a function of u,

$$H(u) = \partial_x u \partial_x^3 u - \frac{1}{2}(\partial_x^2 u)^2 + (\partial_x u)^2 + \frac{1}{2}(1-\lambda)u^2 - F(u), \quad F(u) = \int_0^u f(v)\,\mathrm{d}v, \tag{5}$$

is conserved, i.e., $\frac{\mathrm{d}}{\mathrm{d}x} H(u(x)) = 0$. While we do not make use of the energy (4), H can be used to discuss the location (in parameter space) of localized patterns, see §3. A similar Hamiltonian also exist in 2D, see, e.g., [ALB+10].

For all $\lambda \in \mathbb{R}$, (3) has the spatially homogeneous state $u^* \equiv 0$ (trivial branch). For $\Omega = \mathbb{R}^d$, the linearization $\partial_t v = -(1+\Delta)^2 v + \lambda v$ at $u^* \equiv 0$ has the solutions $v(x,t) = \mathrm{e}^{\mathrm{i}k\cdot x + \mu(k)t}$, $k \in \mathbb{R}^d$, where

$$\mu(k,\lambda) = -(1-|k|^2)^2 + \lambda, \quad |k|^2 := k_1^2 + \ldots + k_d^2. \tag{6}$$

Thus, $u^* \equiv 0$ is asymptotically stable for $\lambda < 0$, unstable for $\lambda > 0$ with respect to periodic waves with wave vector k with $|k| = k_c = 1$, and in 1D we expect a pitchfork bifurcation of spatially 2π periodic patterns at $\lambda = 0$, if permitted by the domain and the BCs.

Remark 2.1. a) In the following we rather briefly describe the computations of the pertinent amplitude equations. More details can be found in, e.g., [UW19], where we moreover describe the `pde2path` tool `ampsys`, which can be used to *automatically* compute the amplitude equations with minimal user input, in 1D, 2D and 3D, and for SH type of equations and RD systems.
b) Since (3) with $f(u) = \nu u^2 - u^3$ has the equivariance $(u,\nu) \mapsto (-u,-\nu)$ it is sufficient to restrict to $\nu \geq 0$. ⌋

2.1 1D

Over $\mathbb{R}$ we have two bands of unstable wave numbers k around ± 1, i.e.,

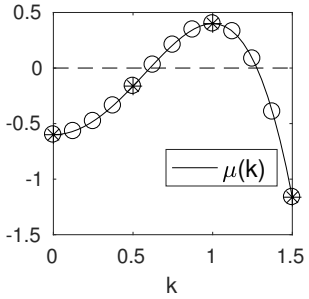
$$\mathcal{K}_u = \left\{ k \in \mathbb{R} : |k| \in \left(\sqrt{1-\sqrt{\lambda}}, \sqrt{1+\sqrt{\lambda}} \right) \right\}. \tag{7}$$

If $\Omega = (-l\pi/2, l\pi/2))$, then the admissible wave numbers are $k \in \frac{1}{2l}\mathbb{N}$, and for large l we have many bifurcation points for small $\lambda > 0$. The first bifurcation at $\lambda_1 = 0$ has $k_1 = 1$, then $k_{2,3} = 1 \pm 1/(2l)$, $k_{4,5} = 1 \pm 1/l, \ldots$, which are usually called sidebands of $k = 1$. See Fig. 1 for how the sidebands are filled for increasing l. Still, generically, BPs are simple. For $l \to \infty$ the center manifold becomes smaller and smaller, and in the limit the bifurcating solutions must be described by the Ginzburg–Landau equation as an amplitude equation, see, e.g., [Mie02] and [SU17, Chapter 10].

For simplicity we first restrict to the primary bifurcation at $\lambda = 0, k = 1$. To compute the amplitude equation on the center manifold we make the ansatz $\lambda = \mu\varepsilon^2$, where $\mu = 1$ or $\mu = -1$, and

$$u(t,x) = \varepsilon A_1(T) e_1 + \varepsilon^2 \left[\frac{1}{2} A_0(T) + A_2(T) e_2 \right] + \text{c.c.} + \text{h.o.t}, \quad e_j = \mathrm{e}^{\mathrm{i}jx}, \tag{8}$$

(a) spectral curve, and admissible k;



(b) modes for $k = 1, 3/4, 5/4$.

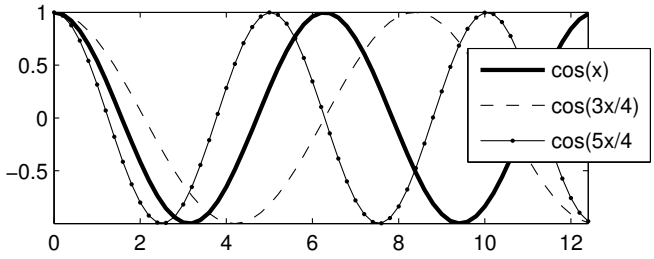


Figure 1: (a) Dispersion relation with admissible k for $\Omega = (0, \pi)$ (*) and $\Omega = (0, 4\pi)$ ($\circ$). (b) First 3 modes for $\Omega = (0, 8\pi)$.

with complex coefficients $A_j = A_j(T)$, which depend on the slow time scale $T = \varepsilon^2 t$. Furthermore, h.o.t denotes higher order terms which are not relevant for the present computation, and c.c. stands for the complex conjugate of the preceding terms, to obtain real valued u. The c.c. of, e.g., $A_1 e_1$ is also conveniently written as $A_{-1}e_{-1}$. Here the BCs enforce $\mathrm{Im}(A_j) = 0$ such that $A_{-j} = A_j$ and for instance $|A_j|^2 = A_j^2$, but for the sake of generality we pretend that the A_j are genuinely complex for $j \neq 0$, which, e.g., is the case for periodic BCs or homogeneous Dirichlet BCs.

Plugging (8) into (3) we first obtain the $\mathcal{O}(\varepsilon^2)$ terms $0 = -A_0 e_0 - 9A_2 e_2 + \nu(2|A_1|^2 e_0 + 2A_2^2 e_2) + \text{c.c.}$, and solving for $A_0 = 2\nu|A_1|^2$ and $A_2 = \frac{2}{9}A_1^2$, and collecting terms at $\mathcal{O}(\varepsilon^3 e_1)$ yields

$$\dot{A}_1 = A_1(\mu - c_1|A_1|^2) \text{ with } c_1 = 3 - \frac{38}{9}\nu^2. \tag{9}$$

Thus, for $\nu^2 < \nu_0^2 := \frac{27}{38}$ ($\nu^2 > \nu_0^2$) we obtain a supercritical (subcritical) pitchfork bifurcation of 2π periodic solutions. In §3 we first verify this numerically, and then focus on the case $\nu > \nu_0$. On large domains the subcritical bifurcation then yields interesting secondary bifurcation to snaking branches of localized patterns, see §3.

2.2 2D

Over $\mathbb{R}^2$, for $\lambda > 0$ we have an annulus $\mathcal{K}_u(\lambda) := \{k \in \mathbb{R}^2 : |k| \in [\sqrt{1-\sqrt{\lambda}}, \sqrt{1+\sqrt{\lambda}}]\}$ of unstable wave vectors. On a bounded box, its side-lengths determine which discrete wave vectors fall into $\mathcal{K}_u(\lambda)$, respectively onto $\partial\mathcal{K}_u(\lambda)$, which in turn determines the sequence of bifurcation points, and in particular the dimension of the kernel.

2.2.1 A square domain.

We first let $\Omega = (-l_1\pi, l_1\pi) \times (-l_2\pi, l_2\pi)$, $l_1, l_2 \in \mathbb{N}/2$, such that $\mu_1 = 0$ at $\lambda = 0$ is double with $k^{(1)} = (1, 0)$, $k^{(2)} = (0, 1)$. The 'natural' associated planforms $u_1 = \cos(x)$ and $u_2 = \cos(y)$ are called stripes. However, any linear combination of these vertical and horizontal stripes are also in the kernel. In particular, combinations of type $u_1 + u_2$ yield spots, and to see what (if any) patterns bifurcate we should compute the amplitude equations. This has for instance been carried out in general form in [Erm91]. These computations can greatly benefit from symmetry considerations, which yield that the reduced system must always be of the form (11) below, and that the only possible bifurcating branches

are stripes and (regular) spots. Finally, numerical kernel computations just yield two (orthogonal) kernel vectors, not knowing a 'natural' base, see Fig. 2 for some examples.

(a) Admissible (k_1, k_2) (b) 'natural' planforms, and numerical eigenvectors at $\lambda = 0$. (c) planforms, and numerical eigenvectors at $\lambda = 1/16^2$.

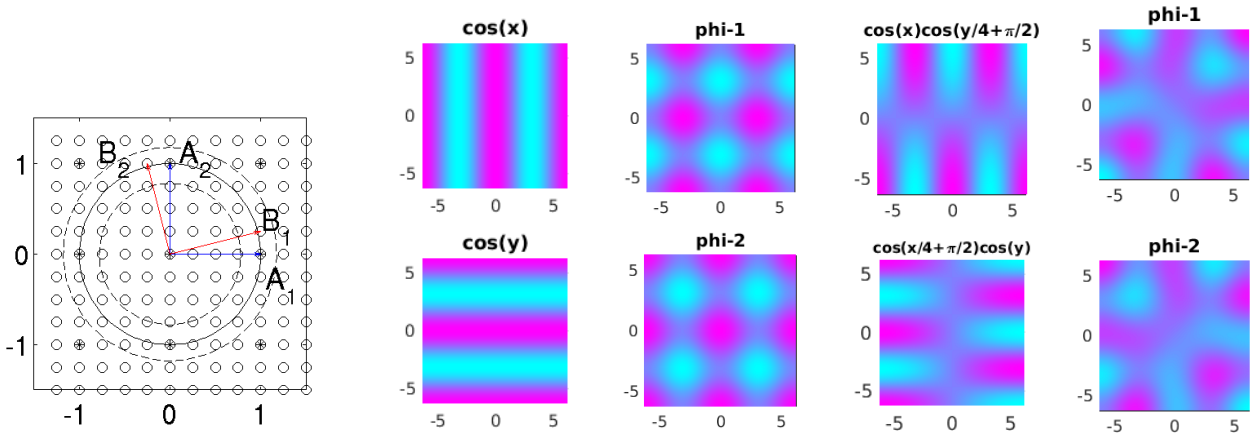


Figure 2: Spectral situation for the linearization of the SH equation around $u \equiv 0$ over square domains. (a) Admissible wave vectors k for $\Omega = (-\pi/2, \pi/2)^2$ (*) and $\Omega = (-2\pi, 2\pi)^2$ ($\circ$), respectively, with Neumann BCs. The amplitudes $A_{1,2}$ and $B_{1,2}$ are used in the amplitude equations below. (b,c) Kernels at the first two bifurcations. In (b), the left column shows the 'natural' planforms of stripes, corresponding to A_1 and A_2, and the right column the kernel vectors obtained numerically. From these the bifurcation directions must be obtained from the solution of the CBE (27) below. Here it is rather easy to see that $\cos(x), \cos(y)$ correspond to $\phi_1 + \phi_2$ and $\phi_2 - \phi_1$. However, this already becomes slightly more difficult at the second BP in (c), where again the left column shows the planforms corresponding to B_1 and B_2.

Here we briefly go through the amplitude equation computations, in a rather ad hoc way, see [GS02, Hoy06] for background on symmetry considerations. We let $\lambda = \mu\varepsilon^2$, $\mu = \pm 1$, and $e_{m,n} = \mathrm{e}^{\mathrm{i}(mx+ny)}$, and make the ansatz

$$u = \varepsilon(A_1 e_{1,0} + A_2 e_{0,1}) + \varepsilon^2(\frac{1}{2}A_0 + A_{1,1}e_{1,1} + A_{-1,1}e_{-1,1} + A_{2,0}e_{2,0} + A_{0,2}e_{0,2}) + \text{c.c.}, \qquad (10)$$

where again c.c. stands for the complex conjugate since we look for real solutions. The BCs enforce that the A_k and A_{k_1,k_2} are all real, and we could as well use $e_1 = \cos(x_1)$ and $e_2 = \cos(x_2)$ and similar for e_{k_1,k_2}, but the (here formally) complex calculus is more general (i.e., also applies to periodic BCs, where $\mathrm{Im}A_k$ may be non zero), and, moreover, is typically more convenient. Similarly, for convenience we use A_1 and A_2 instead of the more consistent notations $A_{1,0}$ and $A_{0,1}$, respectively. Finally, A_k, A_{k_1,k_2} in (10) are again functions of the slow time $T = \varepsilon^2 t$. From (10) we obtain

$$\begin{aligned}
u^2 =& \varepsilon^2\Big[A_1^2 e_{2,0} + A_2^2 e_{0,2} + 2(|A_1|^2 + |A_2|^2)e_0 + 2A_1A_2e_{1,1} + 2A_{-1}A_2e_{-1,1}\Big] \\
&+ 2\varepsilon^3\Big[A_0(A_1e_1 + A_2e_2) + (A_{2,0}A_{-1} + A_{1,1}A_{-2} + A_{1,-1}A_2)e_1 + (A_{0,2}A_{-2} + A_{1,1}A_{-1} + A_{-1,1}A_1)e_2\Big] \\
&+ \text{c.c.} + \text{h.o.t}, \\
u^3 =& 3\varepsilon^3\Big[(|A_1|^2 + 2|A_2|^2)A_1e_1 + (2|A_1|^2 + |A_2|^2)A_2e_2)\Big] + \text{c.c.} + \text{h.o.t}, \\
\partial_t u =& \varepsilon^3(\dot{A}_1e_1 + \dot{A}_2e_2) + \text{c.c.} + \text{h.o.t},
\end{aligned}$$

where h.o.t stands for both, terms of higher order in ε and terms that at $\mathcal{O}(\varepsilon^3)$ fall onto stable wave vectors and are hence irrelevant for the further computations. Collecting terms at $\mathcal{O}(\varepsilon^2)$ and solving for $A_0, A_{2,0}, A_{0,2}, A_{1,1}$ and $A_{-1,1}$ yields

$$A_0 = 2\nu(|A_1|^2+|A_2|^2), \quad A_{2,0} = \frac{\nu}{9}|A_1|^2, \quad A_{0,2} = \frac{\nu}{9}|A_2|^2, \quad A_{1,1} = 2\nu A_1 A_2, \quad A_{-1,1}=2\nu A_{-1} A_2,$$

and the complex conjugate equations for $A_{-2,0}, \ldots, A_{1,-1}$. Now collecting terms at $\mathcal{O}(\varepsilon^3 e_1)$ and $\mathcal{O}(\varepsilon^3 e_2)$ yields the amplitude equations

$$\frac{d}{dT}\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = \begin{pmatrix} A_1(\mu - c_1|A_1|^2 - c_2|A_2|^2) \\ A_2(\mu - c_1|A_2|^2 - c_2|A_1|^2) \end{pmatrix}, \quad c_1 = 3 - \frac{38}{9}\nu^2, \quad c_2 = 6 - 12\nu^2. \tag{11}$$

The amplitude equations (truncated at third order) for the bifurcations from $u \equiv u_0$ on a square with two dimensional kernel *always* take the form (11), see [Erm91, GS02] and [Hoy06, §4.3.1, §5.3], and the specifics of the system condense in the coefficients c_1, c_2.

From (11) we find that the bifurcation problem at $(u, \lambda) = (0, 0)$ on the square is 3-determined except if $c_1 = 0$ or $|c_1| = |c_2|$. Here, a problem is called k–determined if the Taylor expansion up to order k is sufficient to uniquely determine all small solutions, i.e., if any small perturbation of order $k+1$ does not qualitatively change the set of (small) solutions, see [Uec19b] and [Mei00, §6.7] for further discussion. If $c_2 = c_1$, then, returning to real notation $A_{1,2} \in \mathbb{R}$ and wlog assuming that $\mu, c_1 > 0$, (11) has the circle $A_1^2 + A_2^2 = \mu/c_1$ of nontrivial solutions. For $c_2 = -c_1$, we have 'vertical branches' of spots $\mu = 0$ and $(A_1, A_2) = s(1, \pm 1)$, $s \in \mathbb{R}$, and for $c_1 = 0$ we have vertical branches $\mu = 0$ of stripes $(A_1, A_2) = s(0, 1), (A_1, A_2) = s(1, 0), s \in \mathbb{R}$. In all these cases, the bifurcating branches would be determined at fifth order.

Our particular problem (11) at the first BP is thus 3-determined except if $\nu \in \{\nu_1, \nu_2, \nu_3\}$, where

$$\nu_1 := \sqrt{\frac{27}{70}} \ \ (c_1 = c_2 > 0), \quad \nu_2 := \sqrt{\frac{81}{146}} \ \ (c_1 = -c_2 > 0), \quad \nu_3 = \sqrt{\frac{27}{38}} \ \ (c_1 = 0).$$

For $\nu \notin \{\nu_1, \nu_2, \nu_3\}$ we have the nontrivial solutions

$$A_1 = A_2 = \pm\sqrt{\mu/(c_1 + c_2)} \text{ (spots)}, \ A_1 = \pm\sqrt{\mu/c_1}, A_2 = 0 \text{ (or } A_1, A_2 \text{ interchanged, stripes)}, \tag{12}$$

where we assume the right sign of μ for the respective solutions to exist (sub-or supercritically). Moreover, also the stability of the nontrivial solutions can immediately be evaluated, see [Erm91, Theorem], [Hoy06, Fig. 4.10].

Lemma 2.2. *The stripes are stable if* $0 < c_1 < c_2$. *The spots are stable if* $0 < |c_2| < c_1$.

The bifurcation behavior is illustrated in Fig. 3. On the boundaries of the sectors we would need higher order terms in (11) to discuss solutions. However, if we ignore these boundaries, then the two statements in Lemma 2.2 read 'if and only if'. An interesting immediate consequence of this is that (close to bifurcation) spots and stripes are mutually exclusive as stable patterns. For (3), the sectors for c_1, c_2 from (11) as a function of ν are given in (13), and the right column of Fig. 3 confirms the predictions of the amplitude equations via `pde2path`, see §3.

I_ν	$[0, \nu_1)$	(ν_1, ν_2)	(ν_2, ν_3)	(ν_3, ∞)
sector	I	II	III	IV

(13)

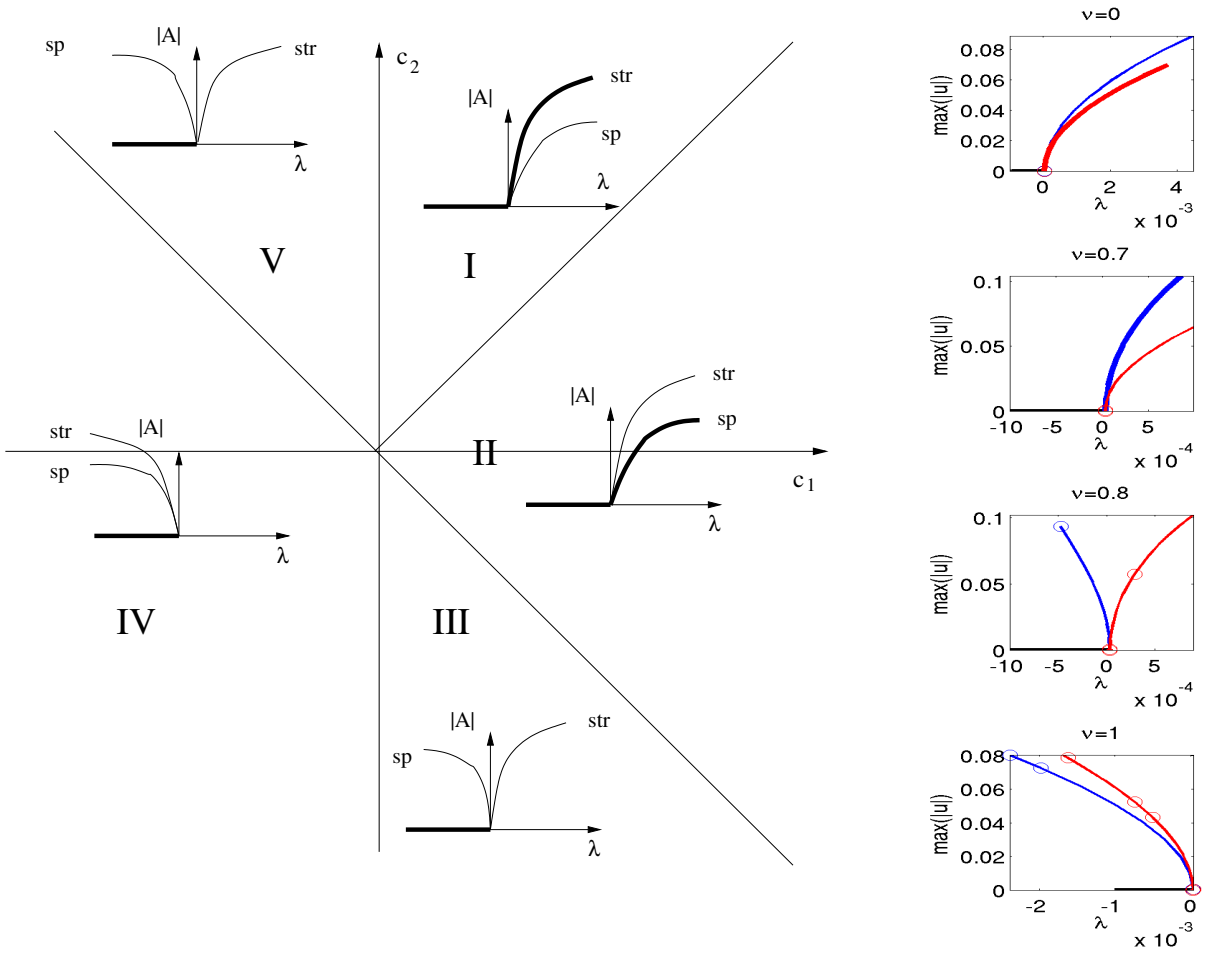


Figure 3: Left: Bifurcation (and stability) diagram for (11) in the c_1-c_2 plane. Right: corresponding numerical bifurcation diagrams for (3) in dependence of ν, stripes=red, spots=blue, thick lines=stable solutions, thin lines=unstable solutions.

Remark 2.3. The second bifurcation point is also double, and Fig. 2(c) illustrates the kernel over $\Omega = (-2\pi, 2\pi)^2$, spanned by, e.g., $e_1 = \cos(x)\cos(y/4 + \pi/2)$ and $e_2 = \cos(x/4 + \pi/2)\cos(y)$. If we make an ansatz $u = \varepsilon(B_1 \mathrm{e}^{\mathrm{i}(x+y/4)} + B_2 \mathrm{e}^{\mathrm{i}(-x/4+y)}) + \text{c.c.} + \text{h.o.t}$, then an analogous computation as above yields the same amplitude equations (11). The nontrivial branches are unstable close to bifurcation as they bifurcate where the trivial branch is already unstable. However, such initially unstable branches may stabilize at larger amplitude, and in fact they do for (3), which is one reason why they may also be interesting. ⌋

2.2.2 A rectangle with a hexagonal dual grid

A special situation occurs for problems with quadratic terms over domains which allow resonant wave vector triads, i.e., critical wave vectors $k^{(1)}, k^{(2)}$ and $k^{(3)}$ such that any $k^{(j)}$ is a linear combination of the other two. As these lie on the circle $|k| = k_c = 1$, the angle between them is $2\pi/3$, and on a rectangular domain this is compatible with the BCs for $\Omega = (-l_1\pi, l_1\pi) \times (-l_2\pi/\sqrt{3}, l_2\pi/\sqrt{3})$, such that $\mu_1 = 0$ at $\lambda = 0$ is double with $k^{(1)} = (1, 0)$, $\phi_1 = \cos(x)$, $k^{(2)} = (-1/2, \sqrt{3}/2)$, and $k^{(3)} = -(1/2, \sqrt{3}/2)$, and, e.g., $\phi_2(x, y) = e_2 + e_3 = \cos(x/2)\cos(\sqrt{3}y/2)$, see Fig. 4.

The ansatz

$$u(x,t) = A_1(t)e_1 + A_2(t)e_2(t) + A_3(t)e_3 + \text{h.o.t} \tag{14}$$

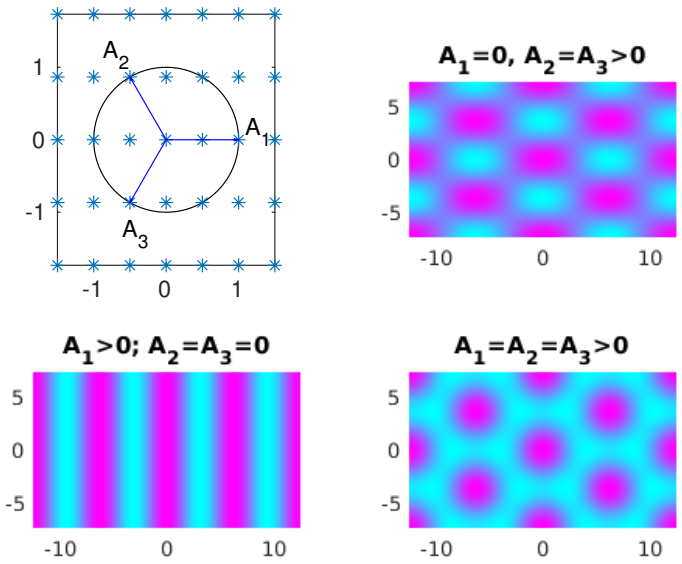


Figure 4: Admissible wave vectors for the domain $\Omega = (-\pi, \pi) \times (-\pi/\sqrt{3}, \pi/\sqrt{3})$, and associated planforms at bifurcation, for clarity plotted over $\Omega = (-4\pi, 4\pi) \times (-4\pi/\sqrt{3}, 4\pi/\sqrt{3})$.

and the pertinent symmetry considerations yield the amplitude equations (in complex notation, although again for Neumann BCs the A_j are real, and, moreover $A_3 = A_2$), truncated at third order,

$$\begin{aligned}
\dot{A}_1 &= \lambda A_1 + \gamma\overline{A_2 A_3} - c_1|A_1|^2 A_1 - c_2(|A_2|^2 + |A_3|^2)A_1,\\
\dot{A}_2 &= \lambda A_2 + \gamma\overline{A_1 A_3} - c_1|A_2|^2 A_3 - c_2(|A_1|^2 + |A_3|^2)A_2,\\
\dot{A}_3 &= \lambda A_3 + \gamma\overline{A_1 A_2} - c_1|A_3|^2 A_3 - c_2(|A_1|^2 + |A_2|^2)A_3.
\end{aligned} \tag{15}$$

The ε–scaling from (10) is omitted in (14) because the derivation of (15) assumes that the quadratic terms in the original system are small (and in particular $\gamma = 0$ if the quadratic terms vanish, i.e., if the original system has the symmetry $u \mapsto -u$). For (3) with $f(u) = \nu u^2 - u^3$ we obtain (recalling that we treat $|\nu|$ as small, and)

$$\gamma = 2\nu + \mathcal{O}(|\lambda\nu|), \quad c_1 = 3 + \mathcal{O}(|\lambda| + \nu^2), \quad c_2 = 6 + \mathcal{O}(|\lambda| + \nu^2). \tag{16}$$

The problem has similar (non-generic) indeterminacies as (11), but is generically 3-determined, and for ν=0 we have we have three bifurcating branches:

$$\begin{aligned}
\text{stripes:} \quad & A_1 = \pm\sqrt{\mu/c_2}, \quad A_2 = A_3 = 0,\\
\text{hexagons:} \quad & A_1 = A_2 = A_3 = A, \quad A = \frac{\gamma}{2(c_1+2c_2)} \pm \sqrt{\frac{\gamma^2}{4(c_1+2c_2)^2} + \frac{\mu}{c_1+2c_2}}\\
\text{patchwork quilt:} \quad & A_1 = 0, \quad A_2 = A_3 = \pm\sqrt{\frac{\mu}{c_1+c_2}} \quad (\textit{only if } \gamma = 0).
\end{aligned}$$

For $\gamma \neq 0$, the hexagons bifurcate transcritically. Those with $A > 0$ ($A < 0$) are also called 'up' ('down') hexagons or 'spots' ('gaps'), and the spots have a fold at $\mu = -\frac{\gamma^2}{2(c_1+2c_2)}$. Analogously, the stripes with $A_1 > 0$ ($A_1 < 0$) could be called 'up' and 'down', but these are related via shifting by π in x. The pq branch only bifurcates from the trivial branch if $\gamma = 0$, and turns into two secondary 'mixed-modes' (or 'rectangle') branches for $\gamma \neq 0$, which connect the stripes and the up hexagons,

and the 'down' stripes and the down hexagons (respectively), namely

$$\text{mixed modes:}\quad A_1 = -\frac{\gamma}{c_1 - c_2},\quad A_2 = A_3 = \pm\sqrt{\frac{1}{c_1 + c_2}(\mu - \frac{\gamma^2 c_1}{(c_1 - c_2)^2})}.$$

The stability of these branches obtained from (15) is as follows. If $\gamma = 0$, then the stripe and pq-branches are stable and the hex branch is unstable. For $\gamma \neq 0$ the stripes are unstable at bifurcation, and become stable at $\mu = \gamma c_1/(c_1 - c_2)^2$. The up hexagons are stable after the fold, until $\mu = \gamma^2(2c_1 + c_2)/(c_1 - c_2)^2$, and the mixed modes for $\gamma \neq 0$ are never stable. These results from the amplitude equations (15) are confirmed and illustrated by `pde2path` numerics over $\Omega = (-2\pi, 2\pi) \times (-2\pi/\sqrt{3}, 2\pi/\sqrt{3})$ in Fig. 7 in §3.4

2.3 3D

In 3D, the situation naturally becomes more complex. We now have a spherical shell $S(\lambda) := \{k \in \mathbb{R}^3 : |k| \in (\sqrt{1-\sqrt{\lambda}}, \sqrt{1+\sqrt{\lambda}}\}$ of unstable wave vectors, and the determination of the branching behavior from the trivial branch is a complicated problem which in general requires lengthy analysis based on results from (symmetry) group theory.

The simplest situation is the so called simple cubic (SC) lattice, spanned by the wave vectors $k_1 = (1, 0, 0), k_2 = (0, 1, 0), k_3 = (0, 0, 1)$, where wlog we focus on the first bifurcation such that $k_c = 1$. This corresponds to a cube of side-lengths $l = 2j\pi$, $j \in \mathbb{N}$, e.g. $\Omega = (-\pi, \pi)^3$, with periodic BCs, while Dirichlet or Neumann BCs as above reduce the problem to smaller solution sets, which we therefore call a sublattice problem. The ansatz for the amplitude equations reads $\lambda = \varepsilon^2\mu, \mu = \pm 1, 0 < \varepsilon \ll 1$, and

$$\begin{aligned} u =& \varepsilon(A_{100} + A_{010} + A_{001}) \\ &+ \varepsilon^2(\frac{1}{2}A_{000} + A_{110} + A_{101} + A_{011} + A_{200} + A_{020} + A_{002} + A_{1-10} + A_{10-1} + A_{01-1}) + \text{c.c.}, \end{aligned} \tag{17}$$

where A_{100}, A_{111} etc are shorthands for $\tilde{A}_{lnm}(T)e_{lnm}(x)$, i.e., coefficient function $\tilde{A}_{lnm}$ and mode $e_{lnm} = \exp(\mathrm{i}(lx_1 + nx_2 + mx_3))$. Then going through the analysis (where again symmetry theory is very helpful), at $\mathcal{O}(\varepsilon^3 e_{100}), \mathcal{O}(\varepsilon^3 e_{010})$, and $\mathcal{O}(\varepsilon^3 e_{001})$ we obtain the amplitude equations

$$\frac{d}{dT}\begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} = \begin{pmatrix} A_1(\mu - c_1|A_1|^2 - c_2(|A_2|^2 + |A_3|^2)) \\ A_2(\mu - c_1|A_2|^2 - c_2(|A_1|^2 + |A_3|^2)) \\ A_3(\mu - c_1|A_3|^2 - c_2(|A_1|^2 + |A_2|^2)) \end{pmatrix}, \tag{18}$$

where again $c_1 = 3 - \frac{38}{9}\nu^2$ and $c_2 = 6 - 12\nu^2$. Naturally, this contains the system (11) as a subsystem with $A_3 = 0$ (or $A_1 = 0$ or $A_2 = 0$), and the stripes and spots of the 2D problem are now classified as *lamellas* $A_1 = \pm\sqrt{\mu/c_1}, A_2{=}A_3{=}0$, and *tubes* $A_1 = A_2 = \pm\sqrt{\mu/(c_1 + c_2)}, A_3 = 0$, respectively. Clearly A_1, A_2, A_3 can be permuted, giving different orientations of the lamellas and tubes. Additionally we have the *rhombs* $A_1 = A_2 = A_3 = \pm\sqrt{\mu/(c_1 + 2c_2)}$, where again depending on c_1, c_2 we assume the right sign of μ.

Moreover, (18) is 3–determined except if $c_1 = 0, |c_1| = |c_2|$, and additionally if $|c_1| = 2|c_2|$. For $c_1 = 0, |c_1| = |c_2|$ we have non-isolated solutions as above with $A_3 = 0$. For $c_1 = 2c_2$ we have the

sphere $A_1^2+A_2^2+A_3^2=\mu/c_1$ of non-isolated solutions, and for $c_1=-2c_2$ we have vertical branches $\mu=0$, $(A_1,A_2,A_3)=s(1,\pm1,\pm1)$, $s\in\mathbb{R}$, of rhombs. The additional exceptional values of ν are $\nu_4=\sqrt{\frac{81}{178}}$ ($c_1=2c_2$) and $\nu_5=\sqrt{\frac{243}{252}}$ ($c_1=-2c_2$). The stabilities of the nontrivial branches on the amplitude equations level can efficiently be computed using symmetry, see, e.g., [CK99, CK99, CK01], which inter alia yields that the tubes are always unstable close to bifurcation, and either the rhombs or the lamellas can be stable, but not both, see also [AGH+05, Theorem 1].

In §3.5 we illustrate some of these results for the SH equation on the cube $\Omega=(-\pi,\pi)^3$, and additionally present results for a so called BCC (body-centered cubic) (sub-)lattice problem on the cube $\Omega=(-\sqrt{2}\pi,\sqrt{2}\pi)^3$.

3 Numerics for the Swift–Hohenberg equation

3.1 Overview

Besides the connection to the analytical results from §2, the SH equation gives an opportunity to show how to rewrite this 4th order (in space) equation as a 2–component 2nd order system in a consistent way. Recall that `pde2path` uses the finite element method (FEM) to convert a system of (2nd order in space) PDEs of the form (1) into a system of ODEs

$$\mathcal{M}\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{u}=-G(\mathrm{u}) \tag{19}$$

for the unknown nodal values $\mathrm{u}\in\mathbb{R}^{n_u}$, where $\mathcal{M}\in\mathbb{R}^{n_u\times n_u}$ denotes the so called mass matrix.[1] Thus, let $(u_1,u_2)=(u,\Delta u)$ to obtain the 2nd order system

$$\begin{pmatrix}1&0\\0&0\end{pmatrix}\partial_t\begin{pmatrix}u_1\\u_2\end{pmatrix}=\begin{pmatrix}-\Delta u_2-2u_2-(1-\lambda)u_1+f(u_1)\\-\Delta u_1+u_2\end{pmatrix}, \tag{20}$$

which immediately translates into the FEM formulation (dropping the notational distinction between the function u and the nodal values u)

$$\mathcal{M}\dot{u}=-(\mathcal{K}u-F(u)), \tag{21}$$
$$\mathcal{M}=\begin{pmatrix}M&0\\0&0\end{pmatrix},\ \mathcal{K}=\begin{pmatrix}0&-K\\K&M\end{pmatrix},\ F(u)=\mathcal{M}\begin{pmatrix}(\lambda-1)u_1-2u_2+f(u_1)\\0\end{pmatrix},$$

where K and M correspond to the scalar stiffness and mass matrices, i.e., $M^{-1}K$ corresponds to $-\Delta$.

Remark 3.1. (a) The formulation (20), resp. (21) with the singular $\mathcal{M}$ on the lhs yields the correct eigenvalues and hence stability information, and, moreover, can be used for time integration via `tint`. See also [Uec19a] for the analogous construction for the Kuramoto-Sivashinsky equation, where it is used to compute Hopf bifurcations and periodic orbits.

(b) Using the same $\mathcal{M}$ on the rhs of (21) as on the lhs is merely for convenience; –we could also implement F in some other way. In particular, we use a 'simplified FEM' setup, where we do not interpolate u from the nodal values to the element centers and then evaluate the nonlinearity and subsequently the pertinent integrals over elements. See [RU19, Remark 1.1] for further comments. ⌋

[1] See, e.g., [RU19, Uec19a] for details on the general classes of systems of PDEs that `pde2path` can treat.

See Table 2 for an overview of files used to implement (21) in 1D, 2D and 3D. In `oosetfemops` we essentially preassemble matrices `p.mat.M` $= \mathcal{M}$ and `p.mat.K` $= \mathcal{K}$, and then set up the rhs in `sG.m` (and the Jacobian in `sGjac`) in a standard way, see Listing 1. Also the init routine is completely standard, and thus below we restrict to brief remarks on the script files.

Table 2: Main scripts and functions in `pftut/sh`.

script/function	purpose, remarks
cmds1d, cmds1dhplot	scripts for 1D, essentially yielding Fig. 5.
cmds2dsq	script for 2D square domain, essentially yielding Fig. 2.
cmds2dhex	script for 2D rectangular domain for hexagons, essentially yielding Fig. 7.
cmds2dhexfro	script for localized hex patterns on a 2D long rectangular domain, see Fig. 8
cmds2dhexfroada	mesh adaptation for localized hex patterns, see Fig. 9
cmds2dhexb	similar to cmds2dhex, but on domain twice as large; meant to illustrate tips and tricks, for problems with 'too many' patterns, essentially yielding Fig. 14.
cmds2dtint	script for patterns from initial guesses and time integration, see Fig. 16.
cmds3dSC	script for simple cube (SC) 3D patterns, see Fig. 10.
cmds3dBCC	script for body centered cube (BCC) 3D patterns, see Fig. 11.
cmds3dBCC	mesh adaptation for BCC 3D patterns
cmdsBCClong	script for localized BCC patterns obtained from initial guesses, see Fig. 17.
cmdsBCClongada	mesh adaptation for localized BCC patterns, see Fig. 18.
shinit	initialization
oosetfemops	set FEM matrices (stiffness K, and two mass matrices M, M0)
sG,nodalf,sGjac	encodes G with 'nonlinearity' in nodalf, and Jacobian
spjac	Jacobian for fold continuation
shbra1d	modification of `stanbra` for putting the Hamiltonian H on the branch
geth	function to compute the Hamiltonian for the spatial dynamics formulation
e2rs	Element2RefineSelection function, used for mesh adaption, here just ad-hoc
hfplot,spl,spplots	convenience functions to plot solutions, Fourier transforms, and planforms

```
function p=oosetfemops(p) % for SH as 2nd order system, hence singular p.mat.M
[K,M,~]=p.pdeo.fem.assema(p.pdeo.grid,1,1,1); % scalar laplacian and mass
p.mat.Dx=makeDx(p); % first order differentiation needed for H
p.mat.K=[[0*K -K];[K M]];   % system stiffness
p.mat.M=[[M 0*M];[0*M 0*M]];  % system mass matrix (here singular)

function f=nodalf(p,u) % SH "nonlinearity" for the 2nd-order system formulation
par=u(p.nu+1:end); lam=par(1); nup=par(2); n=p.nu/2; u1=u(1:n); u2=u(n+1:2*n);
f1=(lam-1)*u1+nup*u1.^2-u1.^3-2*u2; f2=0*u2; % 2nd eqn 0=-lap(u1)+u2 in K
f=[f1;f2];

function r=sG(p,u) % rhs for SH, see nodalf
f=nodalf(p,u); r=p.mat.K*u(1:p.nu)-p.mat.M*f;
```

Listing 1: `oosetfemops.m`, `nodalf.m` and `sG.m` from `pftut/sh`. Here, the 1st component of `nodalf` contains "everything but diffusion", including the linear terms $(\lambda-1)u_1-2u_2$, while the 2nd component of `nodalf` is 0 as we implement the 2nd equation from (20) via $\mathcal{K}$.

3.2 1D

In 1D, the bifurcations of spatially periodic solution branches are simple, and for $\nu > \nu_3 = \sqrt{27/38}$ the primary bifurcation at $\lambda = 0$ is subcritical. An interesting consequence are secondary bifurcations to

steady (approximate) fronts between $u\equiv 0$ and periodic patterns, and to localized patterns, and the "snaking" of the associated branches. This is illustrated in Fig. 5(a1,a2) for (3) over $\Omega=(-10\pi, 10\pi)$.

Snaking branches of localized patterns have attracted much interest in recent years [BK06, Kno08, BKL$^+$09, HK09, ALB$^+$10, KC13, dW19, KUW19]. Since the SH equation has the spatial Hamiltonian (5), i.e., $H(u) = \partial_x u\partial_x^3 u - \frac{1}{2}(\partial_x^2 u)^2 + (\partial_x u)^2 + \frac{1}{2}(1-\lambda)u^2 - F(u)$, $F(u) = \int_0^u f(v)\,\mathrm{d}v$, and since $H(0) = 0$, a front between $u = 0$ and a periodic pattern must connect to a pattern $u_{\rm per} = u_{\rm per}(\lambda)$ with $H(u_{\rm per}(\lambda), \lambda) = 0$. Over $\Omega = \mathbb{R}$ we have a continuum of periodic patterns with wave-numbers near $k = 1$. Over finite domains, the admissible wave numbers of the periodic patterns are discrete, but for localized patterns the 'local' wave numbers in the patterns are free again. Thus, the local wave numbers can and must vary with λ in the snake. For simplicity, i.e., by mirroring the solutions over the right boundary, we also call the (approximate) fronts 'localized patterns'. In Fig. 5(a3) we plot $H(u_{\rm per}(\lambda), \lambda)$ for the first four periodic branches over $\Omega = (-20\pi, 20\pi)$. Comparison with (a1) shows that the wave numbers in the snake should roughly vary between $k = 1 - \frac{1}{80\pi}$ and $k = 1$, which is confirmed by the solution plots and Fourier plots in (b,c). At the left folds in the snake ($\lambda \approx -0.49$), the wave-number is very close to 1, while at the right folds, corresponding to the intersection of the blue branch in (a3) with $H = 0$, it is close to $k = 1 - \frac{1}{80\pi}$. The Fourier plots, however are slightly under-resolved to truly show the shifts of the maxima around $k = 1$ between the left and right folds. Also note that while the 0 mode ($k = 0$) is clearly visible, the second harmonic ($k = 2$) in, e.g., `1D1/pt30` is very small.

Regarding the implementation, the main additional function is `geth` to compute H, which is then put on the branch for plotting in `shbra1d`. Additional to the above results, at the end of `cmds1d` we continue the fold for illustration of fold continuation in a system, cf. [dW17], with a straightforward implementation of `spjac`. Moreover, in §5.6 (demo `gcsh`) we add a global coupling to (3) and illustrate how this modifies the branches in Fig. 5.

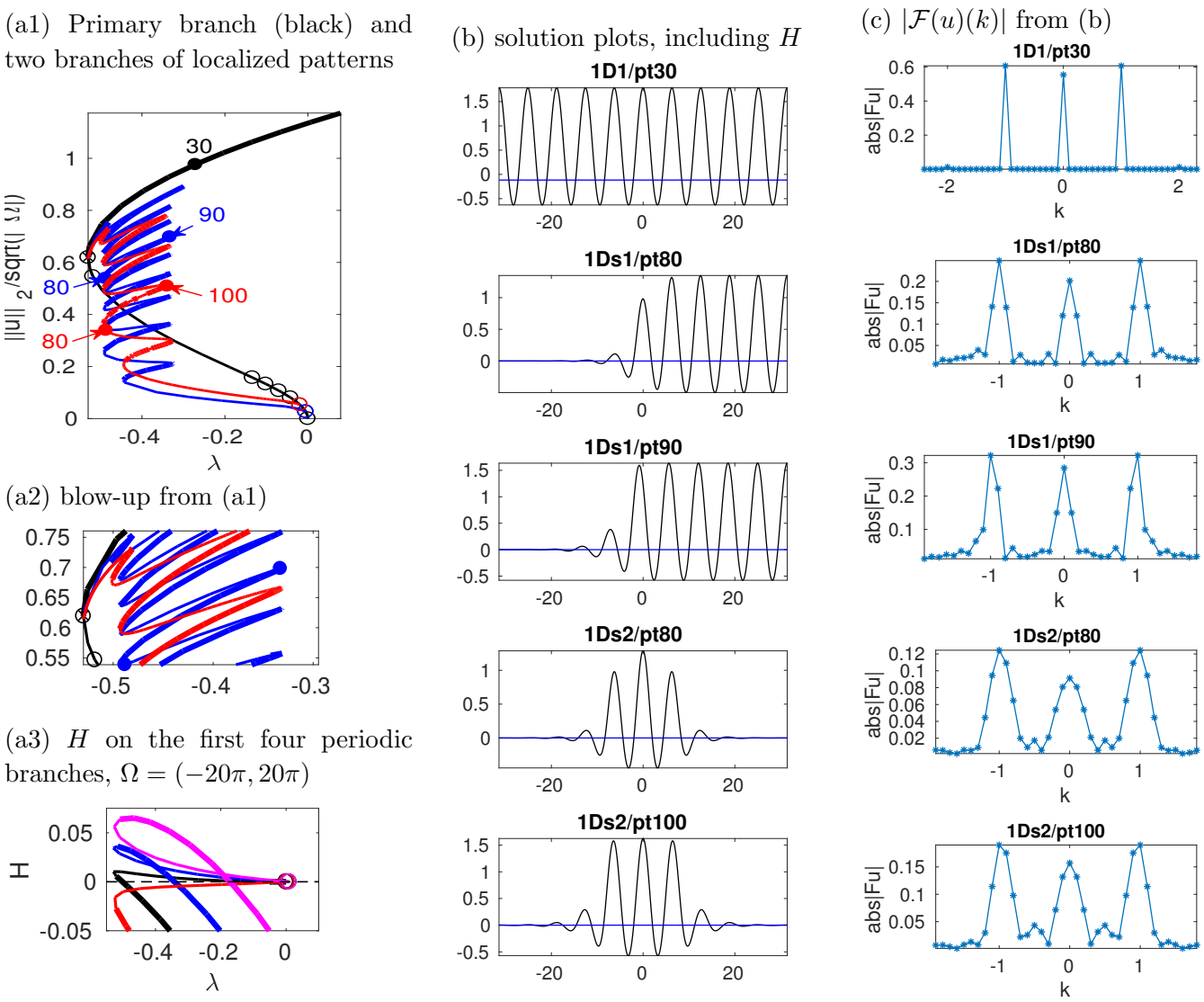


Figure 5: (a1) Subcritical bifurcation of primary periodic patterns (black branch) in the SH equation with $\nu = 2$, $\Omega = (-10\pi, 10\pi)$ and secondary bifurcations of snaking branches of a front (blue, `1Ds1`) and a localized pattern (red `1Ds2`). (a2) blow-up from (a1) showing how the snaking branches reconnect to the primary periodic branch. (a3) The Hamiltonian H on the first four bifurcating branches of periodic patterns on $\Omega = (-20\pi, 20\pi)$, hence corresponding to wave numbers $k = 1$ (black), $k = 1 - \frac{1}{80\pi}$ (blue), $k = 1 + \frac{1}{80\pi}$ (red) and $k = 1 - \frac{1}{40\pi}$ (magenta). (b,c) Solution plots, see main text for further comments.

3.3 Intermezzo: branch switching at BPs of higher multiplicity

As discussed in §2, for 'natural' (i.e. highly symmetric) choices of domains in 2D and 3D, bifurcation points (u_0, λ_0) on homogeneous branches often have a multiplicity

$$m = \dim N(G_u(u_0, \lambda_0)) \geq 2. \tag{22}$$

Here we briefly review the algorithm for branch switching at multiple bifurcation points from [Uec19b]. Let (u_0, λ_0) be a bifurcation point of multiplicity $m \geq 2$, and let

$$N(G_u^0) = \mathrm{span}\{\phi_1, \ldots, \phi_m\},\ N(G_u^{0T}) = \mathrm{span}\{\psi_1, \ldots, \psi_m\},\ \langle \phi_i, \psi_j \rangle = \delta_{ij},\ \text{and } G_\lambda^0 \in R(G_u^0), \tag{23}$$

where $(G_u^0, G_\lambda^0) = (G_u(u_0,\lambda_0), G_\lambda(u_0,\lambda_0))$. Then there exists a unique $\phi_0 \in N(G_u^0)^\perp$ such that $G_u^0\phi_0 + G_\lambda^0 = 0$ and $\langle \phi_0, \psi_j\rangle = 0$, $j = 1,\ldots,m$. The ansatz

$$u'(s_0) = \sum_{j=0}^{m} \alpha_j \phi_j, \quad \alpha_0 = \lambda'(s_0), \tag{24}$$

where $\alpha_j = \langle \phi_j, \dot u(s_0)\rangle, 1 \le j \le m$, and differentiating $G(u(s),\lambda(s)) = 0$ twice and evaluating at s_0 yields the quadratic bifurcation equations (QBE)

$$\begin{aligned}
&B(\alpha_0,\alpha) = 0 \in \mathbb{R}^m, \qquad (25)\\
&B_i(\alpha_0,\alpha) = \sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk}\alpha_j\alpha_k + 2\sum_{j=1}^{m} b_{ij}\alpha_j\alpha_0 + c_i\alpha_0^2, \quad 1 \le i \le m,\\
&a_{ijk} = \left\langle \psi_i, G_{uu}^0[\phi_j,\phi_k]\right\rangle, \quad b_{ij} = \left\langle \psi_i, G_{uu}^0[\phi_0,\phi_j] + G_{u\lambda}^0\phi_j\right\rangle, \quad c_i = \left\langle \psi_i, G_{uu}^0[\phi_0,\phi_0] + 2G_{u\lambda}^0\phi_0 + G_{\lambda\lambda}^0\right\rangle.
\end{aligned}$$

The QBE are quadratic homogeneous in (α_0,α), and hence solutions are only determined up to a factor γ. They are necessary conditions for bifurcating branches. Conversely, each distinct *isolated* zero (α_0,α) gives a distinct solution branch of $G(u,\lambda)$ [KL72]. Here (α_0,α^*) is called isolated if for fixed α_0 and some $\delta > 0$ the only solution in $U_\delta^{\mathbb{R}^m}(\alpha^*)$ is α^*. By the implicit function theorem, a sufficient condition for this is that $J(\alpha) = \partial_\alpha B(\alpha_0,\alpha)$ is non-singular. Without loss of generality we may fix $\alpha_0 = 0$ (if $m = 1$) or $\alpha_0 = 1$, but for scaling reasons (relative to α) it turns out that some small α_0 is more suitable, and our default choice for solving the QBE by a Newton loop in α is $\alpha_0 = 0.001$, and initial guesses for α as all tuples $\alpha \neq 0$ with $\alpha_i \in \{0,\pm 1\}$, $i = 1,\ldots,m$.

For $m = 1$, if (α_0,α_1) is one solution (from the already given branch) of the QBE with $a_{111}\alpha_1 + b_{11}\alpha_0 \neq 0$, then (u_0,λ_0) is a bifurcation point. Moreover, we can solve the QBE explicitly, and this is done in the `pde2path` simple BP branch switching routine `swibra`, see [UWR14, §2], or [Uec19b, Algorithm 2.1].

The case $m \ge 2$ is more difficult, and the QBE (25) may (and typically will) not yield all bifurcating branches, but only those that are 2–determined, see [Uec19b]. Pitchfork bifurcations are at best 3–determined, and in this case we use the ansatz

$$u(s) = u_0 + s\sum_{i=1}^{m} \alpha_i\phi_i + s^2 w, \quad \lambda(s) = \lambda_0 + \beta s^2, \tag{26}$$

with unknowns $\alpha \in \mathbb{R}^m$, $\beta \in \mathbb{R}$ and $w \in N(G_u)^\perp = \mathrm{span}\{\phi_1,\ldots,\phi_m\}^\perp$, i.e. $\langle \psi_i, w\rangle = 0, i = 1,\ldots,m$. Differentiating twice, solving for w at $s = 0$, and differentiating once more and evaluating at $s = 0$ yields the system

$$C(\alpha,\beta) = 0 \in \mathbb{R}^m \tag{27}$$

of m cubic bifurcation equations (CBE), see [Uec19b]. Again wlog we can fix $\beta = \pm 1$ (but numerically use $\beta = \pm\beta_0$ with default choice $\beta_0 = 0.001$), and then each isolated solution α of (27) gives a tangent $(u'(0),\lambda'(0)) = (\sum_{i=1}^m \alpha_i\phi_i, 0)$ to a distinct bifurcating branch. Alternatively, we may choose a small s in (26) and use $(u_0,\lambda_0) + (u(s),\lambda(s))$ as a predictor for the bifurcating branch.

The functions `p0=qswibra(dir,bpt)` (`p0=cswibra(dir,bpt)`) attempt to solve the QBE (CBE) (unless `aux.besw=0`, see below), and store the computed tangents in the fields `p0.mat.qtau` (`/ctau`), re-

spectively, and store the kernel vectors $\phi_1, \ldots, \phi_m$ in `p0.mat.ker`. Additionally, `cswibra` also stores the predictors $s\sum_{i=1}^m \alpha_i\phi_i + s^2 w$ in `p0.mat.pred`. Subsequently we can choose a tangent $\tau=(u'(0), \lambda'(0))$ via `p=seltau(p0,nr, newdir,sw)`, where depending on `sw=2`, `sw=3` and `sw=4` we select vector `nr` from `p0.mat.qtau`, `p0.mat.ctau` or `p0.mat.pred`, respectively. Alternatively, and as a fallback for problems only determined at higher order, we can generate a guess for a tangent to a new branch according to $\tau = \sum_i \gamma_i$`p.mat.ker`(i) via `p=gentau(p0,ga)` where the sum runs from 1 to `length(ga)`. In Algorithm 3.1 we summarize the approach, and in Table 3 we collect auxiliary arguments for fine tuning of `qswibra` and `cswibra`, but otherwise we refer to [Uec19b] for further mathematical comments. Some additional care must be taken if, e.g., the system has continuous symmetries. In this case, `aux.ali` can be used to choose 'active' kernel vectors for setting up and solving the quadratic or cubic bifurcation equations, see §3.6.

Algorithm 3.1: `qswibra, cswibra`, and subsequent `seltau, gentau` for branch-switching at multiple bifurcation points. The arguments `dir, bpt` stand for the `pde2path` setting that the pertinent branch point has filename `fname` in directory `dir`. The function `qcswibra` first calls `qswibra`, then `cswibra`. If `q(c)swibra` is called at a BP with 1D kernel ($m = 1$), then it directly calls `swibra`. Use `q(c)swibra(dir,fname,aux)` to pass optional arguments `aux` listed in Table 3 to `qswibra/cswibra`.

1. Call `p0=qswibra(dir,fname)` to find nontrivial solutions of the QBE (25) and to store these in `p.mat.qtau`. Additionally, store a base of the kernel of G_u in `p.mat.ker`.

2a. If 1 yields nontrivial solutions of the QBE: use `p=seltau(p0,nr,newdir,2)` to choose tangent `nr` as a predictor to the new branch, to be stored in `newdir`.

2b. Use `cont` to continue the new branch, return to 2a to follow more branches.

3. Subsequently/alternatively (if the absence of transcritical branches is known) to 1,2, call `p0=cswibra(dir,fname)` to find nontrivial solutions of the CBE (27). The tangents are then stored in `p.mat.ctau`, and 'effective' predictors (u, λ) are computed from (26) with $s =$ `ds`, normalized and stored in `p.mat.pred`.

4. Proceed as in 2, i.e.: Call `p=seltau(p0,nr,newdir,3)` for choosing tangent `nr` as predictor, or `p=seltau(p0,nr,newdir,4)` to choose the quadratic predictor `nr`. Afterwards call `cont`.

5. For (possible) branches additional to those found in 1.–4.: use `p=gentau(p0,v,newdir)` to generate guesses for tangents to new branches according to $\tau = \sum_i v(i)$`p.mat.ker`(i), where the sum runs from 1 to `length(v)`. Afterwards call `cont`.

6. If `cont` fails after branch-switching, try, e.g., different `ds` (for the quadratic predictor `p=seltau(p0,nr,newdir,4)` this theoretically requires a new call to cswibra).

3.4 Patterns in 2D

In 2D we can use the same basic setup (`oosetfemops` and `sG`) as in 1D, but now need to deal with the multiplicity $m \geq 2$ of BPs over domains that (by symmetry) generate higher dimensional kernels. As indicated in Algorithm 3.1 the idea is to find all bifurcating branches via numerical solution of the associated QBE and CBE by Newton loops for different fixed α_0, β, with different initial guesses α. For this, the two key `pde2path` functions `p0=qswibra(dir,pt,aux)` and `p0=cswibra(dir,pt,aux)` can and sometimes must be fine tuned via the auxiliary argument `aux`, which can have the fields from Table 3.

Table 3: Entries in the auxiliary argument `aux` of `qswibra` and `cswibra`. We assume the call `p0=qswibra(dir,pt,aux)` such first a BP `p0` is loaded from `dir/pt`. Since we do not save data stored in `p.mat` to disk, in this case we first need to recompute the kernel `p0.mat.ker`, and essentially to avoid this we also allow the call `p0=qswibra(p0,aux)`. Similar for `cswibra`. See [Uec19b] for the (scaling) purposes of α_0 and β_0, and further comments.

field	purpose, remarks
soltol	tolerance (default 10^{-10}) to solve the QBE/CBE, i.e., $\|F(\alpha)\| <$ `soltol`.
isotol	tolerance (default 0.1) to classify solutions α of the QBE/CBE as isolated if $\|\det \partial_\alpha F(\alpha)\| >$ `isotol` $\max \|\partial_\alpha F(\alpha)\|$.
mu2	to override p.nc.mu2 where an eigenvalue μ is considered to be zero if $\|\mu\| <$ `mu2`.
m	to explicitly give the dimension of the kernel, instead of $m = \sharp\{\mu : \|\mu\| <$ `mu2`$\}$
al0v	to override α_0=0.001 for `qswibra`; can be a vector α_0=$(\alpha_0(1), \ldots, \alpha_0(j))$
bet0	to override β_0=0.001 for `cswibra`.
alc	to override the initial guesses for the Newton loop for α.
ral	use random initial guesses for α if `ral=1`
ds	to override the steplength selection `p.nc.dsmax/10`; also used for computing the quadratic predictor (u, λ) from (26).
besw	if `besw=0`, then q(c)swibra only compute and store the kernel vectors; useful if subsequently only these are used as (approximate) predictors via `gentau`. Default `bews=1`.
ali	active list of kernel vectors; useful in case of continuous symmetries, see §3.6. Default `ali=[]`, which uses all kernel vectors ϕ_j, $j = 1, \ldots, m$.
hasker	for a subsequent call to `qswibra` or `cswibra` in the syntax, e.g., `p0=cswibra(p0,aux)`; if hasker=1, then the kernel is taken from `p0.mat.ker` and not recomputed; useful for experimenting with parameters, and for instance used in `qcswibra`.
keeplss	if 1, then the linear system solver `p0.fuha.lss` is used for solving the linear systems inside `cswibra`. Otherwise, and as default setting, `lsslu` is used.

3.4.1 A square domain

We proceed by example and consider in Listing 2 the script `cmds2dsq.m` used to generate the branch plots in Fig. 3. Here we know a priori that all bifurcations are pitchforks, and hence can restrict to `cswibra`.

```
%% init and zero-branch
lx=2*pi; nx=round(8*lx); ly=lx; ndim=2; lam=-0.001; nu=0; par=[lam; nu];
p=shinit(p,nx,lx,ly,ndim,par); p=setfn(p,'2D0'); huclean(p); p=cont(p,10);
%% nu=0, hence, sp (spots) supercrit & unstable, st (stripes) supercrit & stable
p0=cswibra('2D0','bpt1'); % cswibra, then reset some parameters
p0.nc.dsmax=0.5; p0.nc.dsmin=0.1; p0.nc.lammax=1; p0.sol.ds=0.1;
%% inspecting the results of cswibra yields tau1=spots, tau3=stripes, hence
% select these and call cont. We just use the tangent; for quadr pred, use
% p=seltau(p0,1,'2D1-1aSp',4); % but that seems to make no difference
p=seltau(p0,1,'2D1-1aSp'); p=pmcont(p,10);
p=seltau(p0,3,'2D1-1aSt'); p=pmcont(p,10);
%% nu=nu_1; indeterminate case, cswibra finds non-isolated solns
p0.u(p0.nu+2)=sqrt(27/70); aux=[]; aux.hasker=1; % subseq. call, reuse kernel
p0=cswibra(p0,aux);
%% nu=0.7; a_1>0, a_2<0, a1>-a2, Sp supercrit & stable, St=supercrit & unstable
p0.u(p0.nu+2)=0.7;  p0=cswibra(p0,aux); p0.sol.ds=0.01;
```

```
p=seltau(p0,1,'2D1-1bSp'); p=cont(p,5); p=seltau(p0,3,'2D1-1bSt'); p=cont(p,5);
```

Listing 2: (Selection from) `sh/cmds2dsq.m`. Script for the 2D SH equation on the square domain $\Omega = (-2\pi, 2\pi)^2$, which yields a double branch points at $\lambda = 0$. `cswibra` in line 5 finds 4 bifurcation directions, falling into the two classes of spots and stripes. Thus, in lines 10,11 we select one spot and one stripe branch and continue these. In line 14 we have a subsequent call (hence set `aux.hasker=1`) to `cswibra` for the indetermiante case $\nu = \nu_1$, cf. (13), where `cswibra` correctly finds non-isolated solutions. The remainder of `cmds2dsq` deals with the other cases for ν (see Fig. 3), plotting, and the 2nd bifurcation point, cf. Remark 2.3.

3.4.2 D_4 symmetry on a non–square domain

The double multiplicity of the BPs in Fig. 2 and of many further BPs follows from Ω being a square domain and the Neumann BCs. However, the generic form of the amplitude equations (11) (in the case of a double BP) follows from the D_4 equivariance. Here we briefly discuss an example of the SH equation on a domain which is not a square (and hence we have no explicit kernel vectors and the multiplicity of BPs is not clear a priori) but which is D_4 invariant (such that in case of double BPs the amplitude equations still have the form (11)). Essentially, we apply results from [Cra91] which explain why some double BPs break up while other stay double, and what primary bifurcations occur. We consider (3) on the perturbed Ω_δ, where we perturb the edges $\Gamma_1 = \{(t,0)\}, \Gamma_2 = \{(2\pi, t)\}, \Gamma_3 = \{(2\pi - t, 2\pi)\}, \Gamma_4 = \{(0, 2\pi - t)\}$ where $t \in [0, 2\pi)$, to

$$\Gamma_1^\delta = \{(t, \delta\sin(\frac{t}{2}))\}, \Gamma_2^\delta = \{(2\pi - \delta\sin(\frac{t}{2}), t)\}, \Gamma_3^\delta = \{(2\pi - t, 2\pi - \sin(\frac{t}{2}))\}, \Gamma_4^\delta = \{(\delta\sin(\frac{t}{2}), 2\pi - t)\},$$

see Fig. 6(d) for the shape of Ω_δ. In Fig. 6(a) we recall the 'pure modes'

$$\phi_{m,n}(x,y) = \cos(mx)\cos(ny),\ n, m \in \mathbb{N}/2, \tag{28}$$

at the first two branch points $\lambda = 0$ and $\lambda = 1/16$ on the (unperturbed) square $\Omega = (0, 2\pi)^2$ with Neumann BCs, where we take the horizontal reflection $\gamma_1 : (x,y) \mapsto (\pi - x, y)$ and the diagonal reflection $\gamma_2 : (x,y) \mapsto (y,x)$ as the generators of D_4. Then $\gamma_1\phi_{m,n} = (-1)^{2m}\phi_{m,n}$ and $\gamma_2\phi_{m,n} = \phi_{m,n}$. In [Cra91] it is explained that the 'mixed' modes $\phi^\pm_{m,n} = \phi_{m,n} \pm \phi_{n,m}$, see Fig. 6(b), are more natural basis functions of the kernels. These fulfill

$$\gamma_2\phi^\pm_{m,n} = \pm\phi^\pm_{m,n}, \text{ and} \tag{29}$$

$$\gamma_1\phi^\pm_{m,n} = (-1)^{2m} \times \begin{cases} \phi^\pm_{m,n} & 2(m+n) \text{ even (parity)}, \\ \phi^\mp_{m,n} & 2(m+n) \text{ odd (parity)}. \end{cases} \tag{30}$$

Using these symmetries, the effect of the perturbation $\delta\sin(t/2)$ of the square is further analyzed in [Cra91]. For odd parity, the perturbation analysis yields that the double eigenvalues stay double, with the simultaneous bifurcation of branches with pure and mixed mode symmetry. However, the even parity modes break into two BPs where only branches of mixed mode symmetry bifurcate, while the previously primary pure modes now become secondary bifurcation. Precisely this is illustrated in Fig. 6(c,d) for the first two BPs, for $\delta = 0.35$. Additionally, the mixed mode branch `mm1` now becomes transcritical as the hidden symmetry of a shift by half a period in x or y is lost, while all other primary branches must remain pitchforks due to the presence of their alternative orientations via γ_2.

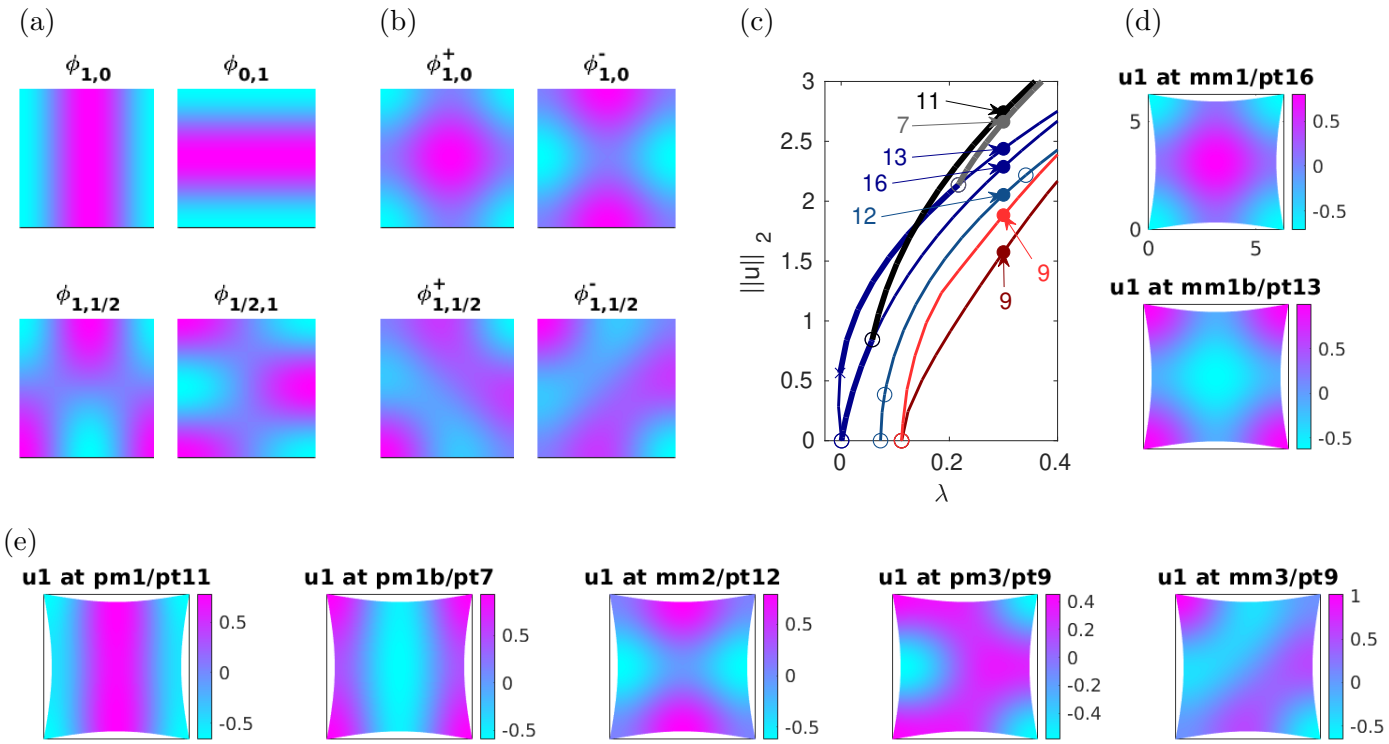


Figure 6: The SH equation on a distorted square with D_4 symmetry. (a) Pure modes at the first two BPs on the square $\Omega = (0, 2\pi)^2$. (b) Associated mixed modes. (c,d) BD and sample solutions over $\Omega_{0.35}$ with $\nu = 0.5$, from `cmds2dpsq`.

Concerning the implementation we refer to the script `cmds2dpsq.m`, and to `shinitpsq.m` for the initialization of the perturbed square, which uses `freegeompdeo.m` to contruct the perturbed square PDE object.

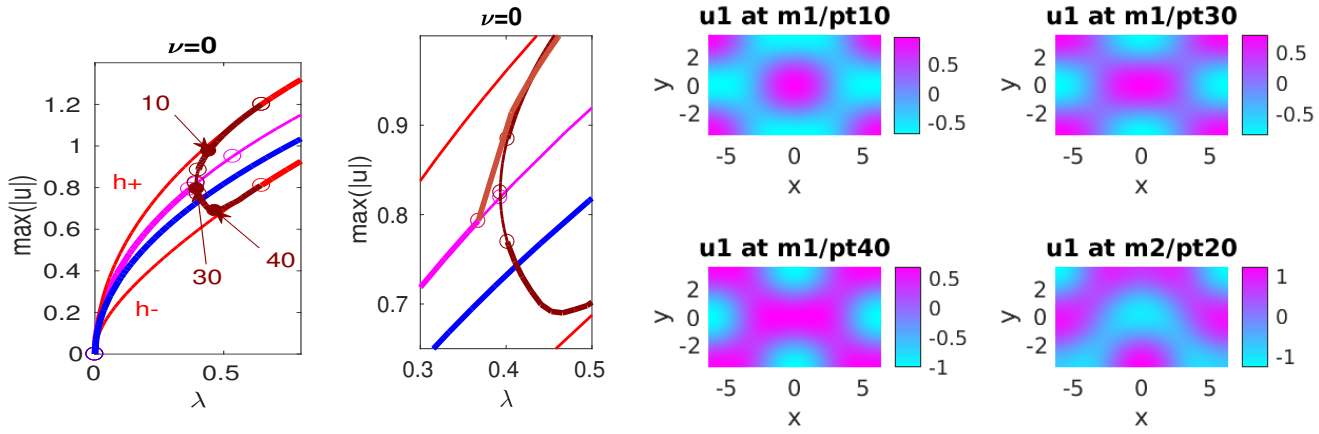
3.4.3 Hexagonal symmetry

The script `cmds2dhex.m` considers (3) over Ω=$(-l_x, l_x)\times(-l_y, l_y)$, l_x=2π, l_y=$2\pi/\sqrt{3}$. This small domain is 4 times the minimal (rectangular) domain $\Omega = (0, l_x) \times (0, l_y)$ allowing hexagon patterns. In Fig. 7(a) we have $\nu = 0$ and hence at $\lambda = 0$ have pitchforks of stripes (stable), pq (stable) and hexagons (unstable). We continue these patterns to rather large amplitude and find secondary bifurcations. The branch m2 (light brown) bifurcates at the loss of stability of the pq branch and gives an example of a (stable) pattern different from those on the minimal domain.

For $\nu = 1.3$ in (b), the hexagons bifurcate transcritically and the up-hexagons become stable after the fold. The stripes are unstable at their (subcritical) pitchfork bifurcation but become stable at larger amplitude, $\lambda = \lambda_1 \approx -0.01$, while the up hexagons become unstable again at $\lambda = \lambda_2 \approx 1.21$. These two points are connected by a mixed mode branch which we call beans. Similarly, the down-hexagons become stable at $\lambda = \lambda_3 \approx 0.46$, and the branch bifurcating there connects to (shifted) up hexagons at $\lambda \approx 1.8$. Even on this relatively small domain there are many additional bifurcation on the hexagons and stripes branches. At the end of `cmds2dhex` we plot, just for illustration, the energies $\mathcal{E}(u)$ associated to some branches from Fig. 7. Moreover, we remark that:

- For $\nu = 0$ we obtain all primary branches from `cswibra`. For $\nu = 1.3$ we naturally use `qswibra` to obtain the hexagons, and then, for convenience, `gentau` to generate the stripes.
- It is again crucial to use `pmcont` to continue the patterned branches; simply using `cont` results

(a) $\nu = 0$ (cubic case), str and pq stable at bifurcation, hex unstable; BD, zoom, and example solutions



(b) $\nu = 1.3$, BD and example solutions

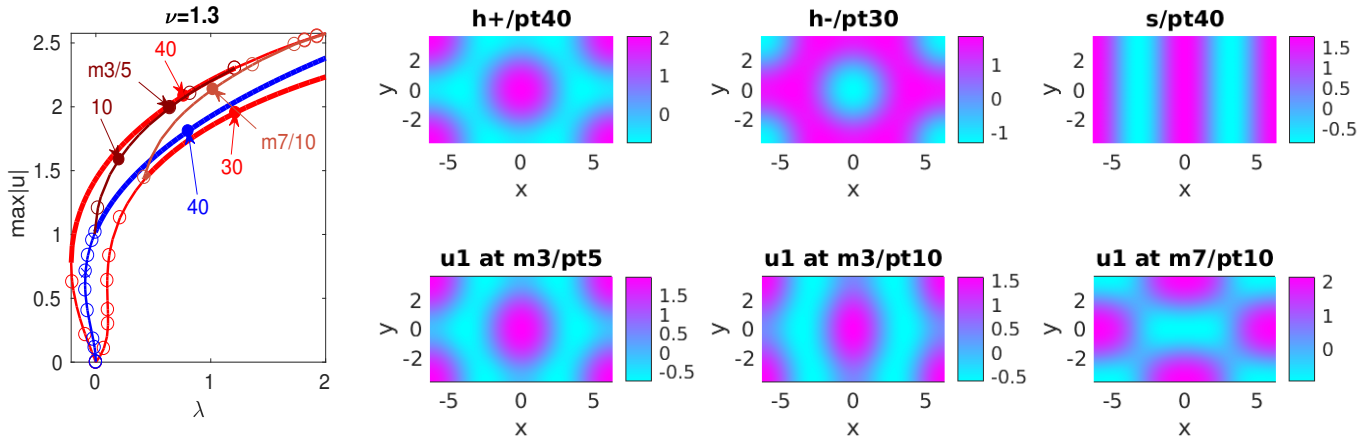


Figure 7: Example results from `cmds2dhex.m`. Bifurcation diagrams and example plots SH over a small rectangular domain permitting hex solutions. For $\nu = 0$ in (a), hex (unstable), str and pq (stable) bifurcate in supercritical pitchforks. The (up and down) hex become stable at $\lambda = \lambda_1 \approx 0.645$, and these points are connected by a mixed mode branch m1, which which passes the pq branch near $\lambda = 0.39$. The pq patterns become unstable $\lambda = \lambda_2 \approx 0.37$, where a stable branch m2 (light brown) bifurcates. For $\nu = 1.3$ in (b) the hex are transcritical, and we consider secondary bifurcations and mixed mode branches at larger amplitude. See text for details.

in various uncontrolled branch-jumpings. Just one illustrative example is given at the end of `cmds2dhex.m`.

- As the secondary bifurcations are generically simple, for their detection we use `p.sw.bifcheck=1`, and a sufficiently small stepsize `ds` to not miss bifurcation points via too large steps.

On larger domains, there naturally are more patterns, and both, the avoidance of branch jumping and the bifurcation detection become more difficult problems. See §3.8.

3.4.4 Planar fronts between hexagons and zero

Localized patterns similar to Fig. 5 can also occur in 2D and 3D. Moreover, while in 1D we basically have localized patterns of 'stripes' connected to $u \equiv 0$, in 2D we can have heteroclinic connections and heteroclinic cycles between various patterns and $u \equiv 0$, or between different patterns, e.g., between stripes and hexagons. This is discussed in more detail in §3.8, in §4, and in, e.g., [UW14], and here

we restrict to planar fronts between hexagons and $u \equiv 0$, and 1D-localized hexagon-patches. Figure 8 illustrates the basic idea. Over sufficiently large (long) domains, there are secondary bifurcations after the primary subcritical bifurcation to (here) hexagons, and these lead to snaking branches of localized hexagons, which (over bounded domains) eventually reconnect to the primary hexagon branch. Listing 3 gives the main commands.

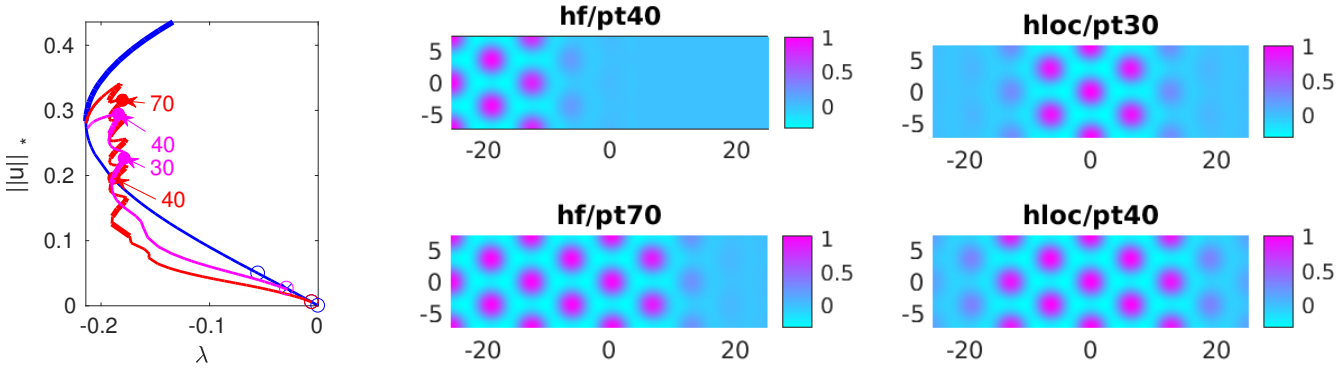


Figure 8: Results from `cmds2dhexfro` for the SH over a long rectangular domain, $\nu = 1.3$. Primary branch of (up) hexagons (blue), and snaking branches of fronts (red, `hf`) between hexagons and $u = 0$ and of localized hex (magenta, `hloc`).

```
%% init and zero-branch
lx=8*pi; nx=round(5*lx);ly=4*pi/sqrt(3); lam=-0.001; nu=1.3; par=[lam; nu];
ndim=2; p=shinit(p,nx,lx,ly,ndim,par); p=setfn(p,'2Dhex8'); p.sol.ds=0.005;
p.nc.dsmin=0.005; p.sol.dsmax=0.05; p.sw.bifcheck=2; p=cont(p,5);
%% hex from qswibra
p0=qswibra('2Dhex8','bpt1'); p0.sw.bifcheck=1;
p=seltau(p0,1,'2Dh8',2); p.pm.resfac=1e-4; p.sol.ds=-0.01; p.nc.dsmin=0.01;
p.u(p.nu+2)=1.3; p.nc.mu2=0.005; p.nc.neig=40; p=pmcont(p,10);
p.sw.bifcheck=0; % switch off bif-detec for further steps for speed
tic; p=pmcont(p,50); toc
%% 2ndary bif to hex-front (lower tol and switch off bifcheck for speed)
p=swibra('2Dh8','bpt1','2DH8f',0.05); p.sw.bifcheck=0; p.nc.tol=1e-6;
p.nc.dsmin=0.005; p.nc.dsmax=0.05; p.nc.ntot=1000; p.pm.mst=8; p=pmcont(p,100);
```

Listing 3: (Selection from) `sh/cmds2dhexfro.m`. Main part of the script to compute a snaking branch of a front between hexagons and $u = 0$. The remainder of the script deals with the snake of localized hexagons and plotting.

In Fig. 9, we use `trullekrul` [Uec19c] to first coarsen hf/pt40 (with $n_p = 4650$) to a coarser mesh, and then continue with mesh adaptation (see also Remark 3.3) each 5th step. Here, a crucial point is to use a rather low `p.trop.Llow` $= 0.1$, because otherwise the flat part of the solution (at $x > 0$) will become 'too coarse'. This is reasonable for a fixed solution, but then the snake fails to grow further hexagons into the flat area, and continuation fails via branch jumping or non-convergence. As usual, the precise form of the adaptation strongly depends on the `trullekrul` parameter choices [Uec19c], and we refer to `cmds2dhexfroada` for details and comments.

```
%% refinement of hex2zero front by trullekrul and subsequent cont with amod=5
p=loadp('2DH8f','pt60','hfr'); p.nc.tol=1e-8; p=resetc(p); p.np
op=troptions2D(); % load default trullerup-options, then overload some
op.innerit=1; op.etafu=@etafua2D; op.verbose=2; op.ppar=2; op.Lup=2;
op.Llow=0.075; % small LLow important to avoid too coarse meshes
p.trop=op; ops=op; % put options in p, then modify to coarsening options
```

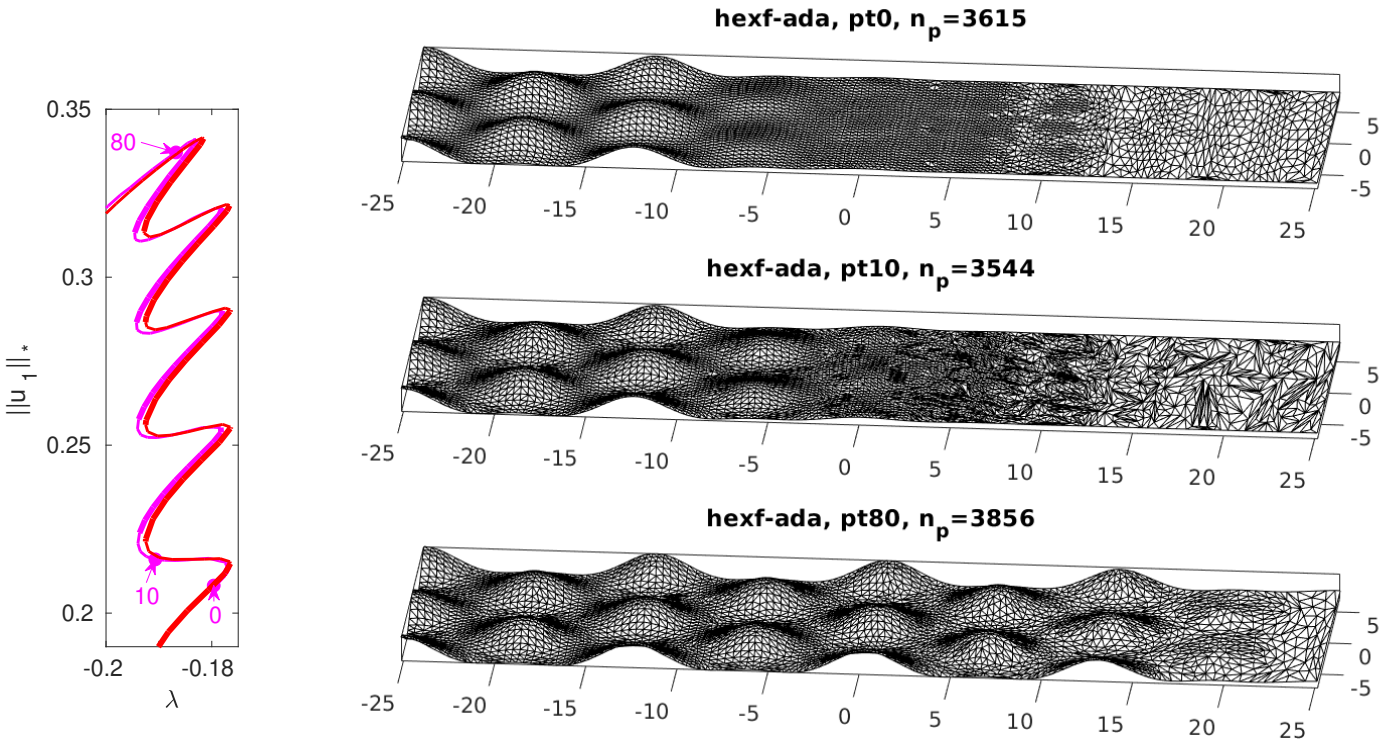


Figure 9: Results from `cmds2dhexfroada`. We coarsen hf/pt40 (with $n_p = 4650$) from Fig. 8 to $n_p = 3615$, then continue with `amod=5` (mesh–adaptation each 5th step), and, importantly, `p.trop.Llow` $= 0.1$, because otherwise the mesh in flat parts of solutions becomes too coarse. On the left, the red curve is from the fixed mesh with $n_p = 4650$, and the magenta curve from adaptation.

```
op.npb=3000; op.sw=5; op.innerit=3; op.crmax=2; p.trcop=op; p.sw.trul=1;
p.sw.ips=2; % interpolation by 'nearest', otherwise bad behaviour at boundary
% call pure coarsening, then reset options
p.trop=p.trcop; p=oomeshada(p,'ngen',3); p.trop=ops; stansavefu(p);
%% meshada each 5th step, switch off pure coarsen(not needed/effective)
p=loadp('hfr','pt0','hfada2'); huclean(p); p.nc.tol=1e-6; p.trcop.crmax=0;
p.nc.amod=5; p.nc.dsmax=0.01; p=resetc(p); stansavefu(p);
p.sw.foldcheck=0; p.trop.innerit=2; p.nc.ngen=2; p=cont(p,101);
```

Listing 4: (Selection from) `sh/cmdshexfroada.m`. The continuation with mesh adaptation is set up in lines 12-14; extra coarsening switched off via trcop.crmax=0.

3.5 Two cubes as models for the SC and BCC lattices

As indicated in §2.3, the bifurcations of Turing patterns in 3D are in general rather complicated. Numerical studies have essentially been restricted to obtaining patterns from time integration, aka direct numerical simulation (DNS), see, e.g., [HSO07, LVE09]. Here we first restrict to a simple 3D analog of §2.2.1, namely $\Omega = (-\pi, \pi)^3$, again with homogeneous Neumann BCs $\partial_n u = \partial_n \Delta u = 0$ on $\partial\Omega$. At the first bifurcation point $\lambda = 0$ we then have a three dimensional kernel $N(G_u) = \mathrm{span}\{\cos(x), \cos(y), \cos(z)\}$, i.e., the wave vectors $k^{(1)} = (1, 0, 0)$, $k^{(2)} = (0, 1, 0)$, and $k^{(3)} = (0, 0, 1)$ generate a so called simple cubic lattice. By symmetry, i.e., from the amplitude equations (18), we also know that all bifurcations at $\lambda = 0$ (and in fact at all subsequent bifurcation points) must be pitchforks, and thus we directly use `cswibra` to obtain bifurcation directions.

Figure 10 shows some results from `cmds3dSC.m`. In (a) we give the three numerically obtained kernel

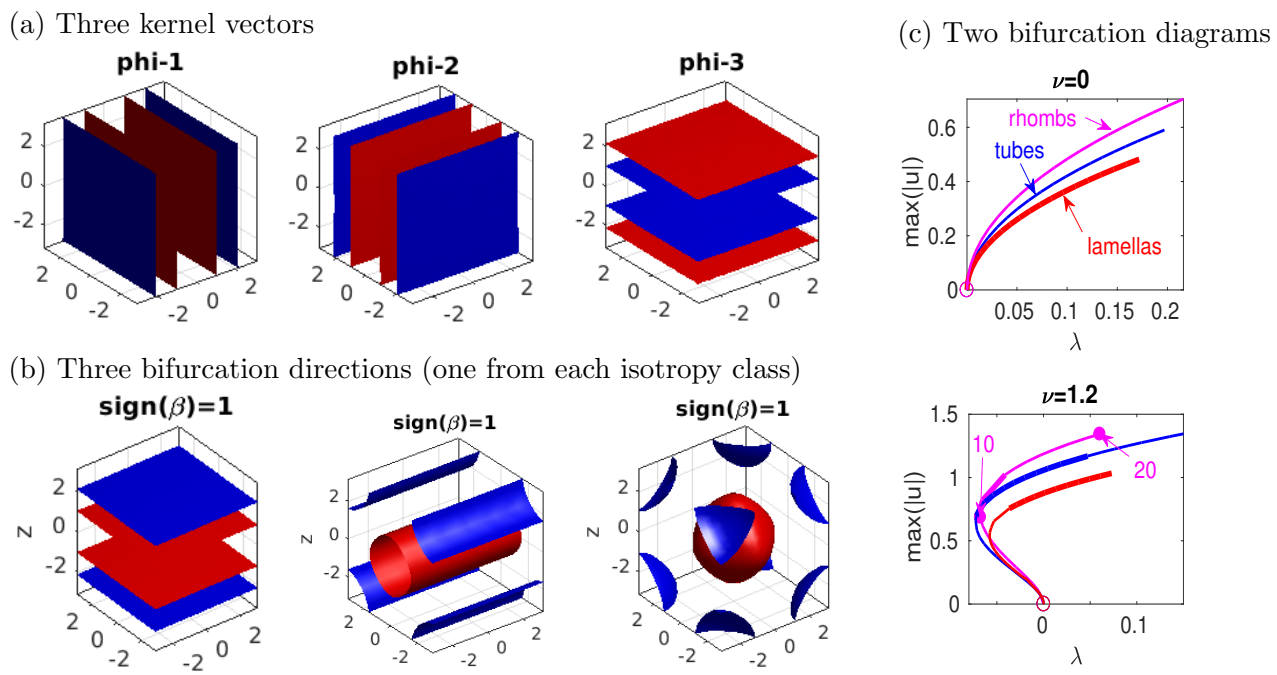


Figure 10: Selected results from demo `sh/cmds3dcube`. Primary bifurcations at $\lambda = 0$ in (3) over the cube $\Omega = (-\pi, \pi)^3$ with homogeneous Neumann BCs, pseudo criss-cross mesh of $n_p = 6006$ points and $n_t = 33000$ tetrahedral elements, see §3.7 for comments on the meshing. (a) Isosurface plot of u for the numerical kernel vectors, where blue and red correspond to $m_{\text{blue}} = \frac{3}{4}m_0 + \frac{1}{4}m_1$, $m_{\text{red}} = \frac{1}{4}m_0 + \frac{3}{4}m_1$, respectively, with $m_0 = \min u$, $m_1 = \max u$. (b) Three (of 8) bifurcation directions obtained from `cswibra`, with $\alpha = (0.002, 0.005, 0.914), (-0.03, 0.52, -0.53)$ and $(1.27, 1.29, -1.3)$, respectively. The other five are obtained from symmetry, i.e., rotation and/or translation. `sign`$(\beta) =$ `1` refers to the case $\nu = 0$. (c) Bifurcation diagrams, $\nu = 0$, and $\nu = 1.2$, stable branches as thicker lines. The shown branches follow the planforms predicted at bifurcation.

vectors ϕ_1, ϕ_2, ϕ_3, given by three clean lamellas. This, however, strongly depends on the chosen mesh, see §3.7 for further remarks, and Fig. 11 for a less clean example, and in general it is not obvious how to compose the pertinent three (modulo symmetries) bifurcation directions from ϕ_1, ϕ_2, ϕ_3. Calling `cswibra` yields ten bifurcation directions, of which we plot three, one of each isotropy subgroup, i.e., τ_1, τ_2, τ_3 in (b). (c) shows the BDs for $\nu = 0$ (all branches supercritical, with the lamellas stable), and $\nu = 1.2$ (all branches subcritical). In the latter case, the tubes become stable shortly after their fold, but later become unstable again, while the lamellas become and stay stable at large amplitude. However, even over this 'minimal' domain there are many secondary bifurcations, and stable large amplitude solutions without simple symmetries.

In Fig. 11 we consider solutions on a cube allowing BCC (body-centered cubic) branches. The BCC lattice corresponds to n=6 wave vectors

$$k_1{=}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\\0\end{pmatrix},\ k_2{=}\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\1\end{pmatrix},\ k_3{=}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\1\end{pmatrix},\ k_4{=}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\\0\end{pmatrix},\ k_5{=}\frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\-1\end{pmatrix},\ k_6{=}\frac{1}{\sqrt{2}}\begin{pmatrix}-1\\0\\1\end{pmatrix}, \quad (31)$$

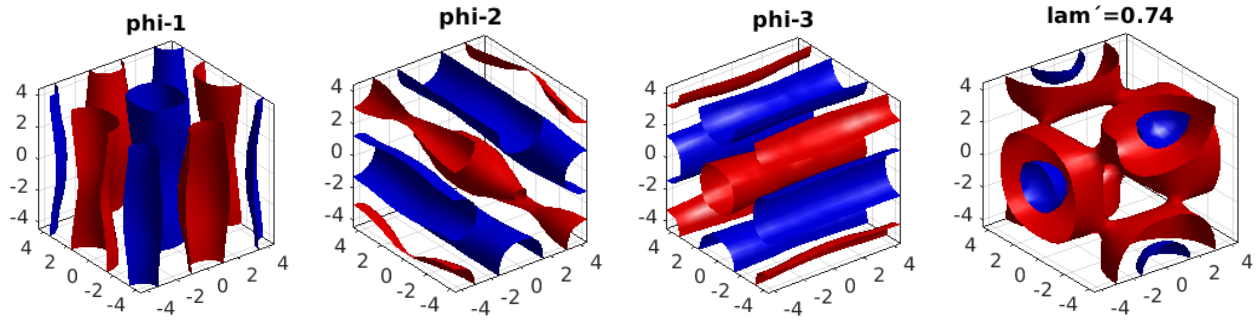
leading to a six–dimensional amplitude systems for amplitudes $A_1, \ldots, A_6$, including quadratic terms due to resonant triads such as $k_1 - k_2 = -k_6$. This amplitude system has a variety of solution branches, see [CK97, CK99], but on a cube with side-length $\sqrt{2}l\pi$, $l \in \mathbb{N}$ and homogeneous Neumann BCs, only

few of the solutions of the amplitude system can be realized, namely:

- Tubes, or more precisely square prisms, corresponding to $0 \neq A_1{=}A_4 \in \mathbb{R}$, $A_2, A_3, A_5, A_6{=}0$, i.e., $u \sim \cos((x+y)/\sqrt{2}) + \cos((x-y)/\sqrt{2})$, and of course other orientations and spatial shifts.
- Balls, or, more precisely, BCCs, $A_1{=}\ldots{=}A_6 \in \mathbb{R}$, which correspond to equal amplitude superpositions of tubes.

The three kernel vectors at the primary bifurcation are computed as distorted tubes. The BCC branch bifurcates transcritically, and bifurcation directions are found by `qswibra`. We call the branch to the left (right) hot (cold) balls as they have maxima (minima) in the centers. The hot balls become stable at the fold, while cold balls and tubes are always unstable.

(a) three kernel vectors on the BCC (sub)lattice, and bifurcation direction for a BCC



(b) BD of BCCs and square-prisms on BCC lattice, and example solutions

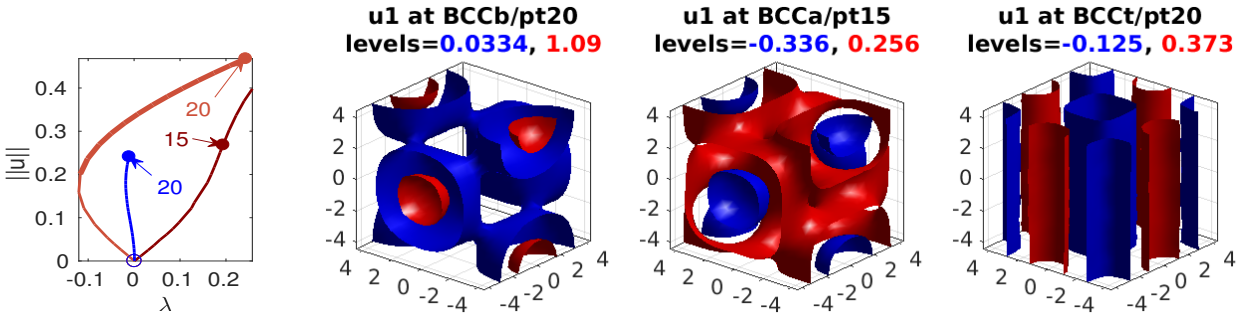


Figure 11: (3) on a "BCC lattice cube" $\Omega = (-\sqrt{2}\pi, \sqrt{2}\pi)^3$, $\nu = 1$. (a) kernel vectors (distorted square prisms) and bifurcation direction for a BCC obtained from `qswibra`. (b) BD, 'hot' and 'cold' BCCs and square prism example solutions. $n_p = 10351$, $n_t = 57024$ tetrahedra, see `cmds3dBCC.m` for details.

Figures 10 and 11 are just intended as first illustrations of 3D pattern formation with `pde2path`. For instance, by extending the domain from Fig. 11 in one direction we can now produce snaking branches of 'localized hot balls', see §3.8. There, however, we rather focus on 'problems with too many solutions', which occur on large (2D and) 3D domains, and generate fronts between the hot balls and zero, and other localized solutions, via 'educated guesses'.

3.6 Periodic domains: Discrete and continuous symmetries, demo `shpbc`

If the system $G(u)$ has continuous symmetries described by a Lie group Γ, i.e., $G(\gamma u) = \gamma G(u)$ for all $\gamma \in \Gamma$, then solutions of $G(u) = 0$ come in (continuous) group orbits, i.e., if u is a solution, then so is $u_\gamma := \gamma u$ for all $\gamma \in \Gamma$. In particular, if the group orbit is nontrivial, i.e., $\gamma u \neq u$ for $\mathrm{Id} \neq \gamma \in H$ where H is a subgroup of Γ, then $\partial_\gamma u|_{\gamma=\mathrm{Id}}$ is in the kernel of $G_u(u)$. Thus G_u always has a zero eigenvalue, and a robust continuation of $G(u) = 0$ requires to remove the symmetry. A natural and

practical selection of a (locally) unique element in the group orbit $\{\gamma u : \gamma \in H\}$ goes by adding a constraint that requires the predictor u from a solution $u_{\rm old}$ to lie transverse to the group orbit of $u_{\rm old}$. In a Hilbert space this is naturally an orthogonality relation, the so-called 'phase condition' (PC)

$$\langle \partial_g u_{\rm old}, u - u_{\rm old} \rangle = 0. \tag{32}$$

See [RU17] for further discussion, and various examples with continuous symmetries and suitable PCs.

Moreover, in case of BPs of higher (discrete) multiplicity with additional continuous symmetries Γ, we must also remove the symmetries Γ for branch switching with `q(c)swibra` because otherwise the predictors for bifurcating branches cannot be isolated. For this, the user can pass the auxiliary list `aux.ali` of 'active' (for the branch switching) kernel vectors to `q(c)swibra`, which typically can be identified after a first inspection of the kernel vectors using `aux.besw=0`. Here we illustrate this procedure with a simple example, namely the SH equation (3) on $\Omega = (-2\pi, 2\pi)^2$, with periodic BCs (pBCs)

$$\begin{array}{llll} \partial_x^j u(2\pi, y) = \partial_x^j u(-2\pi, y), & \partial_x^j \Delta u(2\pi, y) = \partial_x^j \Delta u(-2\pi, y), & j = 0, 1, & \text{(pBCs in } x\text{)}, \\ \partial_y^j u(x, 2\pi) = \partial_y^j u(x, -2\pi), & \partial_y^j \Delta u(x, 2\pi) = \partial_y^j \Delta u(x, -2\pi), & j = 0, 1, & \text{(pBCs in } y\text{)}, \end{array} \tag{33}$$

for all $x, y \in (-2\pi, 2\pi)$, where the pBCs for Δu naturally arise because we have a fourth order problem, or, equivalently, pBCs for the second component of the vector $(u, \Delta u)$. The SH equation is thus translationally invariant in x and y, i.e., the (continuous) symmetry group is

$$\Gamma = \{g_{(\rho,\sigma)} \in [0, 4\pi)^2\}, \text{ with group action } g_{(\rho,\sigma)} u(x, y) = u(x - \rho, y - \sigma)$$

and the obvious addition modulo 4π of the group elements. The generators associated to $g_{(\rho,0)}|_{\rho=0}$ and $g_{(0,\sigma)}|_{\sigma=0}$ are ∂_x and ∂_y.

The bifurcation points from the trivial branch are now $\lambda = (\lambda_1, \lambda_2, \lambda_3, \ldots) = (0, \frac{1}{16}, \frac{1}{4}, \ldots)$, with kernels spanned by

$$\begin{array}{ll} \lambda_1 : & \phi_1 = \sin(x),\ \phi_2 = \cos(x),\ \phi_3 = \sin(y),\ \phi_4 = \cos(y),\ \text{(4 dimensional kernel)}, \\ \lambda_2 : & \begin{array}{l} \sin(x)\sin(y/2),\ \sin(x)\cos(y/2),\ \cos(x)\sin(y/2),\ \cos(x)\cos(y/2), \\ \sin(x/2)\sin(y),\ \sin(x/2)\cos(y),\ \cos(x/2)\sin(y),\ \cos(x/2), \cos(y), \end{array} \ \text{(8 dimensional)}, \\ \lambda_3 : & \sin(x)\sin(y),\ \sin(x)\cos(y),\ \cos(x)\sin(y),\ \cos(x)\cos(y),\ \text{(4 dimensional)}, \\ \vdots & \end{array} \tag{34}$$

For, e.g., λ_1, clearly ϕ_1 and ϕ_2 are related by $\phi_2 = g_{(\pi,0)}\phi_1$, and $\text{span}\{\phi_1, \phi_2\} = \{\sin(x+\rho) : \rho \in [0, 4\pi)\}$, i.e., the group orbit of all shifts (in x) of ϕ_1, and similar relations hold for the kernels at λ_2, λ_3 and all further λ_j. Equivariant bifurcation theory [GS02, Hoy06] now tells us that to find the bifurcations at, e.g., λ_1, it is sufficient to restrict to the ansatz $u = (\alpha_1\phi_1, \alpha_2\phi_3)$ (one representative of stripes in x and y, respectively), because all other solutions (solution branches) are then related to those obtained from ϕ_1, ϕ_3 via the symmetries Γ.

Thus, given a continuous symmetry, to apply `q(c)swibra` at a bifurcation point (u_0, λ_0) we should restrict to a selection of kernel vectors with just one representative from each group orbit. Practically we use the following algorithm (see Listing 5 for example calls), where w.l.o.g. we use `cswibra` because in our first example we only have pitchfork branches; for transcritical branches the algorithm is the same, with `qswibra` instead of `cswibra` (see for instance §6.2).

1. Call `cswibra` with `aux.besw=0` (only compute kernel), possibly with large `aux.m` (to compute many eigenpairs with μ near 0), and `aux.ali=[]` (do not select kernel vectors). This simply plots the eigenvectors belonging to the m eigenvalues of smallest modulus.
2. Inspect these plots to find the eigenvectors related by Γ, and from each class select only one representative by putting its number into `aux.ali`. Then call `cswibra` again with `aux.besw=1` and `aux.hasker=1`.

Often, the symmetries are easy to spot after step 1, such that `aux.ali` in step 2 can be easily chosen. Additionally, a small `aux.ali` can be chosen deliberately to restrict the subspace for the search of (predictors for) solutions.

Figure 12 shows results for (3) with $\nu = 0$ over $\Omega = (-2\pi, 2\pi)^2$ with pBCs and thus kernels (34). At each BP λ_1, λ_2 and λ_3 we compute just 2 bifurcating branches, i.e., one stripe branch and one spot branch. Regarding the pertinent PCs (32) for the continuation of the nontrivial branches we proceed as follows. For vertical/horizontal stripes, the PCs read

$$\langle \partial_x u_{\text{old}}, u \rangle = 0 \text{ (to fix translations in } x\text{), and} \tag{35}$$

$$\langle \partial_y u_{\text{old}}, u \rangle = 0 \text{ (to fix translations in } y\text{),} \tag{36}$$

respectively. For diagonal stripes both $\gamma_{(\rho,0)}$ and $\gamma_{(0,\sigma)}$ generate the same group orbits such we can use either (35) or (36), while for spots we need both. For the implementation, we therefore assemble differentiation matrices `p.mat.Kx` and `p.mat.Ky` at startup, and set up routines `qfx, qfy`, and `qf` (both), and their derivatives. After branch-switching from $u \equiv 0$ we first do a few (1 to 3) steps without PC. We then switch on the pertinent PC by setting, e.g., `p.nc.nq=1` and `p.fuha.qf=@qfx`, `p.fuha.qfder=@qfxder` for (35), and continue further. The case of translational PC in y or in x and y works analogously, and for convenience these commands are collected in little functions `p=qxon(p), p=qyon(p), p=qxyon(p)`, respectively. See Listing 5, `cmdssq.m` for the main script, and Table 4 for an overview of the files involved.

```
%%  BP1, stripes and spots
aux=[]; aux.m=4; aux.besw=0;
aux.ali=[2 4]; aux.besw=1; % 'active' list, first comment out this line
p0=cswibra(dir,'bpt1',aux); p0.sw.bifcheck=2; p0.nc.tol=1e-6; p0.nc.dsmax=0.11;
%% horizontal stripes, 3 initial steps, then switch on PC in y
p=seltau(p0,1,'sq1s',3); p.sol.ds=0.05; p=cont(p,3); p=qyon(p); p=cont(p,20);
%% spots, need both PCs, and pmcont rather than cont
p=seltau(p0,2,'sq1sp',3); p.sol.ds=0.1; p=cont(p,3); p=qxyon(p); p=pmcont(p,20);
```

Listing 5: `cmdssq.m` (selection). Using `cswibra` with `aux.besw=0` to first inspect the kernel, then with `aux.besw=1` and `aux.ali` $= [1, 2]$ to factor out the continuous symmetries before deriving and solving the CBE (27). Subsequently, we use `seltau` to choose the bifurcation direction, and switch on the needed PCs for continuation (`qyon` for horizontal stripes, and `qxyon` for spots).

Remark 3.2. a) The implementation of periodic BCs is explained in [DU17]. Essentially, we only need to call `p=box2per(p,[1 2])` during initialization (see `shinit.m`), and `filltrafo` after each assembly of a system matrix such as `K` or `M`, see `oosetfemops.m`.

b) A phase condition such as (35) is an additional equation and hence requires to free an additional parameter, and to do so we introduce a dummy 'speed' parameter s_x and add $s_x \partial_x u$ to the rhs of the SH equation. Similarly, for (36) we add $s_y \partial_y u$, such that the augmented problem reads

$$0 = -(1 + \Delta)^2 u + \lambda u + \nu u^2 - u^3 + s_x \partial_x u + s_y \partial_y u. \tag{37}$$

Table 4: Scripts and functions in `pftut/shpbc`.

script/function	purpose, remarks
cmdssq, cmdshex	scripts, with cmdssq yielding Fig. 12.
cmdssq_defl	script to compute further solutions by deflation, see §3.8.2
shinit	initialization, including the call `p=box2per(p,[1,2])` to switch on pBCs
oosetfemops	set FEM matrices, including `filltrafo` to account for the pBCs
sG,nodalf,sGjac	rhs and Jacobian as usual
qf,qfder	phase condition (PC) function (in x and y) and derivative
qfx, qfxder	PC and derivative, only in x, e.g., for 'vertical stripes', see also qfy, qfyder
qxon, qyon, qxyon	convenience functions to switch on PCs in x, in y, or in both, respectively.

(a) BD (b) Solutions from branches from λ_1, λ_2, and λ_3.

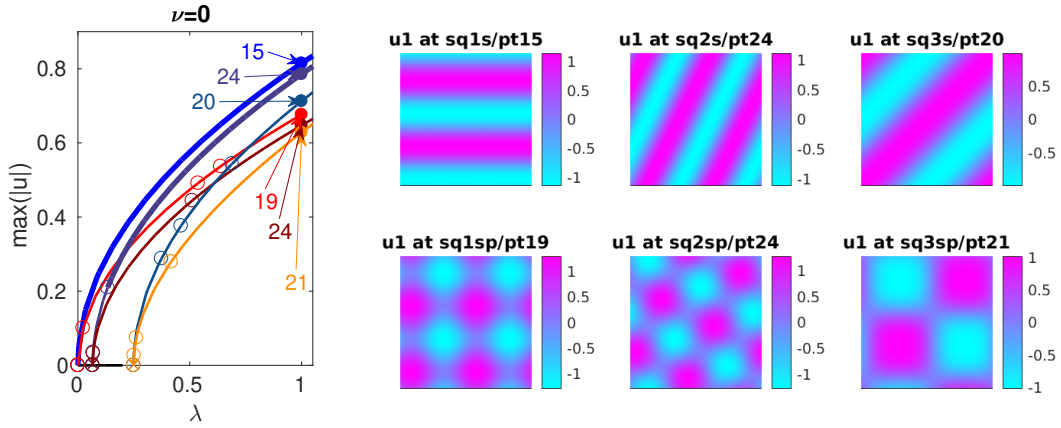


Figure 12: The SH equation (3) over $\Omega = (-2\pi, 2\pi)^2$ with pBCs in x and y, $\nu = 0$. The kernels at the 1st, 2nd and 3rd BPs $u \equiv 0$ and $\lambda = 0, 1/16, 1/4$ are 4,8, and 4–dimensional, respectively. However, modulo the translational symmetries Γ they are only 2,4 and 2–dimensional, respectively. Modulo Γ, at $(0, \lambda_1)$ there bifurcate 3 branches (horizontal and vertical stripes, and spots), and similarly at $(0, \lambda_3)$ (SW to NE and NW to SE stripes, and spots). At $(0, \lambda_2)$ we have 6 bifurcating branches, namely four types of stripes and two types of spots.

In the parameter vector, the new parameters sit at positions 3 for s_x and 4 for s_y, and thus we set `p.nc.ilam=[1 3]` for the x–PC, `p.nc.ilam=[1 4]` for the y–PC, and `p.nc.ilam=[1 3 4]` if both PCs are active. The speeds s_x, s_y are naturally initialized with $(0, 0)$, and they both stay $\mathcal{O}(10^{-6})$ or smaller during all continuations.

c) In `cmdssq_defl.m` we use deflation to compute further branches of solutions for (3) over $\Omega = (-2\pi, 2\pi)^2$ with pBC, see §3.8.2. In `shpbc/cmdshex.m` we treat the related case of a periodic domain allowing hexagons, where essentially for the hexagons we need to use `qswibra` instead of `cswibra`. In §6 we use the ideas explained here to treat related problems for pattern formation on spheres, which naturally lead to large kernels. ⌋

3.7 Remarks on choices of 2D and 3D meshes

The default meshing of rectangles in 2D proceeds via Delauney triangulation of a regular rectangular grid. As a consequence, these meshes have no reflection or rotational (discrete, by $\pi/2$) symmetry, see Fig. 13(a) for a sketch. However, if we have many solutions (solution branches) "close together", and (multiple) bifurcations distinguished by their symmetry, then it is desirable to have meshes as symmetric as possible. So called *criss-cross meshes*, which using the `pdetoolbox` can for instance be

generated by calling `refinemesh(..,'longest')` on a (default) `poimesh`, have a D_4 (rotations by $\pi/2$ and reflections) symmetry (locally if the domain does not have D_4), which also on further uniform refinement stays intact, see Fig. 13(b) for an example. If the `pdetoolbox` is not available, a simple but efficient method to obtain similar meshes, which we call *pseudo criss-cross* is as follows. We start with a regular rectangular grid, and then add the rectangle midpoints to the grid. A subsequent Delauney triangulation then yields meshes of type (c), which are at least reflection symmetric.

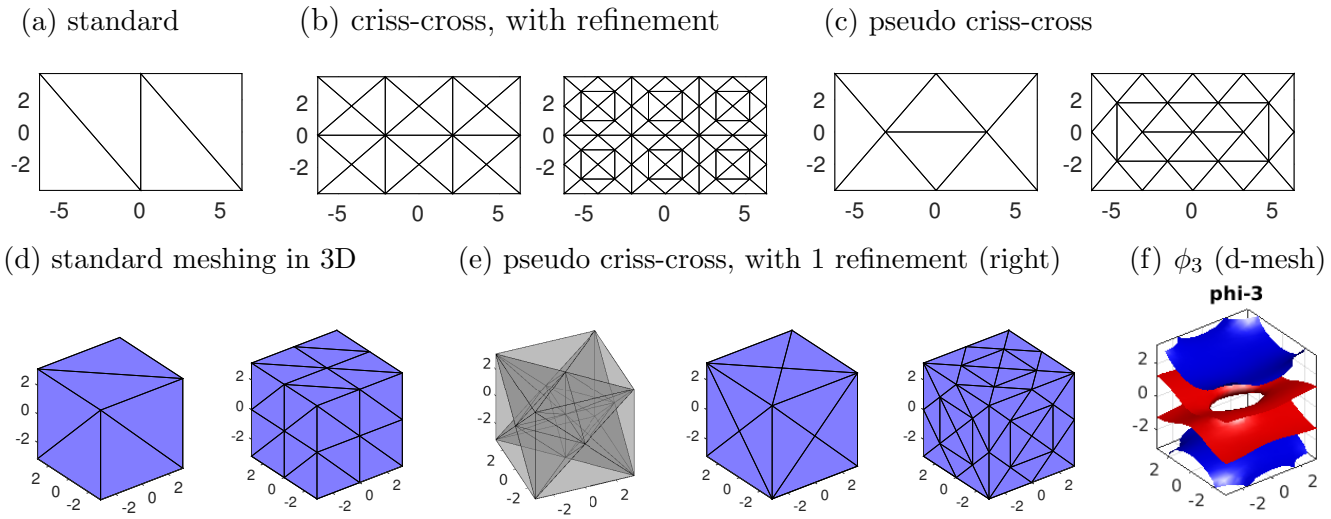


Figure 13: (a-c) 2D meshes on $\Omega = (-l_x, l_x) \times (-l_y, l_y)$, $l_x = 2\pi, l_y = 2\pi/\sqrt{3}$, starting from a meshgrid with $n_x = 3, n_y = 2$. (a) Standard meshing ($n_p = 6$) in 2D destroys reflection and rotation symmetry. (b) criss-cross via "refine-longest", $n_p = 18$, and with one additional default refinement, $n_p = 59$. This has (locally) the full D_4 symmetry (reflections and discrete rotations). (c) pseudo criss-cross, where rectangle centers are added to the grid before meshing, $n_p = 8$, and $n_p = 28$ after 1 (red) refinement. This also always has the reflection symmetries but in general no discrete rotational symmetry. (d) Standard 3D meshing destroying all symmetries. (e) Criss-cross like meshing here keeps all symmetries (discrete rotation and reflection), also under (uniform) refinement. (f) ϕ_3 for SH on a "standard" mesh with $n_p = 6450$, compare to Fig. 10(a).

Similarly, the default meshing in 3D produces asymmetric meshes of type (d). Consequently, the continuation of highly symmetric branches such as the tubes and rhombs in Fig. 10 may be problematic: The solutions may jump (or, near bifurcation points, "slowly drift") to a less symmetric branch. For instance, over standard grids the tubes often jump to lamellas, and the rhombs to tubes (or at least strongly distort at larger amplitude). This can be alleviated by choosing meshes of type (e) in Fig. 13, which we also call pseudo criss–cross. Here we start with a regular cuboid grid, add all cuboid centers and face centers, and then do the Delauney meshing, and afterwards possibly some uniform ('red') mesh–refinement. We remark that starting with a coarse mesh and refinement vs starting with a fine grid produces similar but in general not equivalent results.

A lack of mesh symmetry is also often reflected in distorted eigenvectors at multiple BPs. Figure 13(f) shows one of the three distorted lamellas kernel vectors obtained for the same settings as in Fig. 10, but on a standard mesh of type (d) with $n_p = 6450$ points. This is not a problem for `cswibra`, which computes the same τ_1, τ_2, τ_3 as in Fig. 10(b), but branch switching more likely fails than on more symmetric meshes in the sense that the initial corrector jumps, e.g., from the rhombs predictor to the tubes branch.

To give the user some easy control over the meshing, the calls `pde=stanpdeo2D(lx,ly,nx,ny,sw)` and `pde=stanpdeo3D(lx,ly,lz,nx,ny,nz,sw)` have, besides the obvious arguments $l_x, n_x, \ldots$, the

struct `sw` as an auxiliary argument. Currently, this can have two fields, namely

- If `sw.sym=1`, then meshes of type (c) (2D) and (e) (3D) from Fig. 13 are generated.
- In 2D, if `sw.sym=2`, then we generate genuine type (b) criss cross meshes.
- If `sw.ref` > 0, then `sw.ref` refinement steps are executed after the initial meshing.

See `sh/shinit.m` and `cmds2dsq.m, cmds2dhex.m, cmds3dSC.m, cmds3dBCC`, and the `cmds*` scripts in our next example `schnakpat` for templates and details.

Remark 3.3. (a) `OOPDE`, like many other FEM packages offers additional elements, for instance bilinear rectangular elements (in 2D) and triangular prism elements (in 3D). For some applications, these show some advantages, but in this tutorial we restrict to the triangle and tetrahedra elements.

(b) An important strength of the FEM is the option of adaptive mesh refinement. For `pde2path`, this is discussed in some detail in [RU19] (1D and 2D, using error estimators) and [Uec19c] (2D and 3D, using the anisotropic mesh adaptation package `trullekrul`). However, for the (roughly harmonic) periodic patterns considered here, mesh adaptation is typically not very efficient, and, moreover, it may break symmetries and introduce anisotropies into the problem. Therefore, mesh adaptation does not play a big role in this tutorial. On the other hand, for fronts between patterns and constant solutions, mesh adaptation may be quite efficient as the constant part obviously only needs a coarse mesh. In Fig. 9 we gave one example for this in 2D, and in Fig. 18 we do the analog in 3D. ⌋

3.8 Problems with 'many' solutions, warnings, tips and tricks

3.8.1 General remarks

In §3.4 (2D) and §3.5 (3D) we considered small (almost minimal) domains. Over larger domains, the number of patterns resulting from the Turing instability and secondary bifurcations quickly becomes quite large, which can be a serious problem for the numerics. For illustration, and for comments on how to deal with these problems, here we double the domain from Fig. 7, see Fig. 14, and the script `cmds2dhexb.m`. We focus on the stripe branch `s` and its secondary bifurcations, which due to the many secondary bifurcation points (and in contrast to `cmds2dhex.m`) we now continue using `p.sw.bifcheck=2`. The `s` branch solutions again gain stability near $\lambda_1 \approx -0.01$, but now, with `dsmax=0.1`, the number of unstable eigenvalues jumps from 2 to 0 in the continuation across λ_1, leading to the localization of `BP16` in Fig. 14(a,b). The *three* smallest eigenvalues then are $\mu_{1,2,3} = -0.000024, -0.016013, 0.04903$, and the corresponding eigenvectors (1st component) are shown in (c). It turns out that:

- To each of these eigenvectors there is a branch bifurcating from `s`, although only approximately at `BP16`, and these eigenvectors are also (approximately) returned by `cswibra` at `BP16`.
- Thus we can simply call `seltau` after `cswibra`, and obtain the bifurcating branches `b1a, b1b`, and `b1c` (not shown).

In particular, the `b1b` branch are the beans (light brown (e)), from which a snaking branch (dark brown (f)) of fronts between hexagons and stripes bifurcates. Example plots from the (dark blue) branch `b1a` associated to `phi-3` are shown in (g). This reconnects to the stripes at low amplitude, and has (small) stable segments. Similarly, calling `cswibra` at many other BPs on the stripe branch yields bifurcations to various branches of patterns which have (small) stable segments. In fact, once the bifurcation points become 'sufficiently dense' on a given branch, we can more or less

- call `q(c)swibra` at *any* point, including regular points. Typically, the eigenvectors belonging to small eigenvalues are then sufficiently close to the kernel vectors at a nearby bifurcation point, and usually branch switching via `seltau` works.

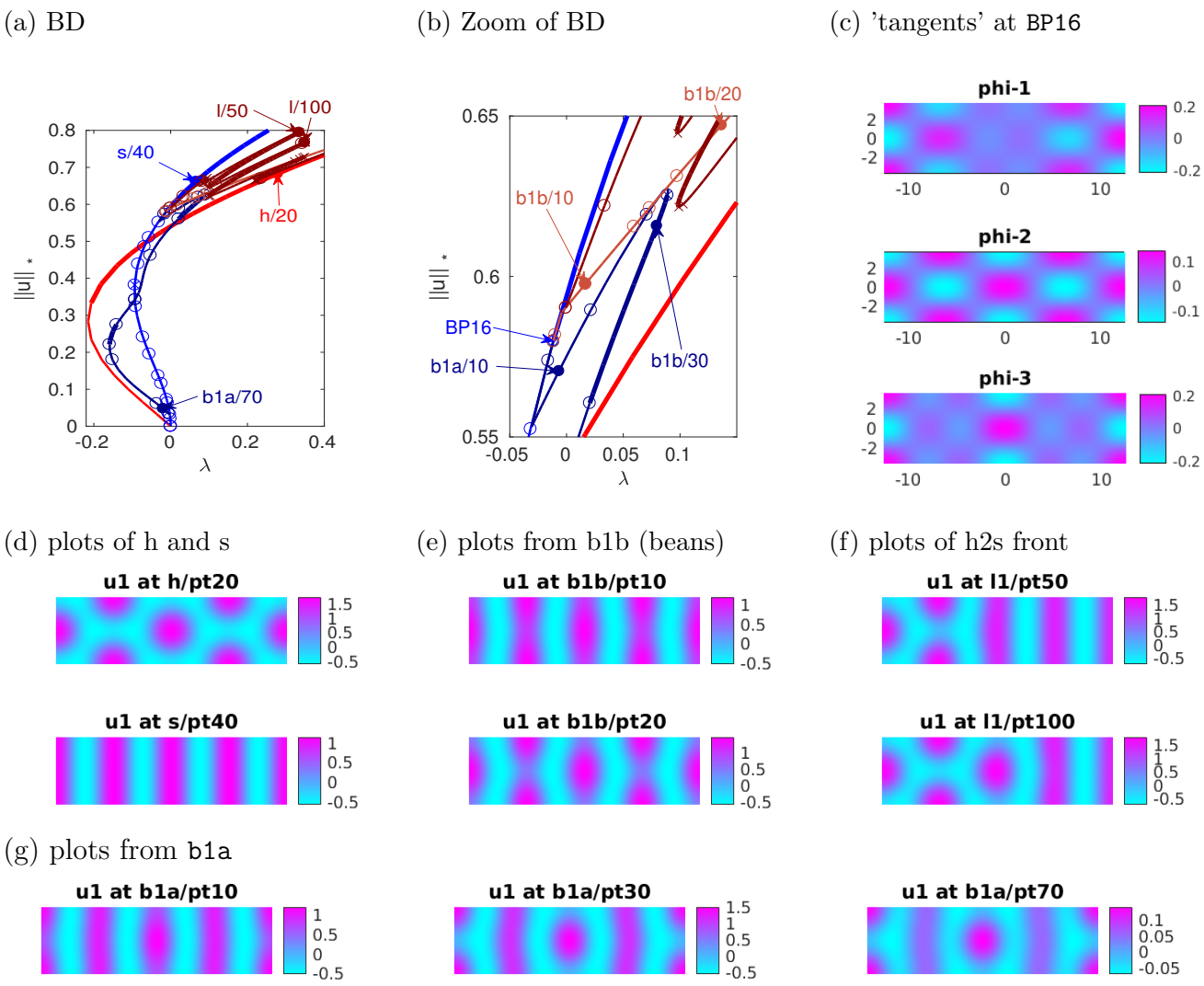


Figure 14: (3) on Ω=$(-2l_x, 2l_x)\times(-l_y, l_y)$, l_x=2π, l_y=$2\pi/\sqrt{3}$, $\nu = 1.3$, example results from `cmds2dhexb.m`. See text and `cmds2dhexb.m` for details.

In this sense, already on this still small domain

- it becomes essentially impossible to obtain a 'nearly complete' bifurcation diagram that contains at least the stable solutions at small to intermediate amplitude, $\lambda \in (-0.2, 0.4)$, say.

Moreover, we want to stress that in these circumstances, the use of `pmcont` instead of `cont` seems crucial to avoid (reduce) uncontrolled branch switching. Here we recall that:

- Branch jumping does not produce 'wrong solutions', but a wrong bifurcation picture.
- For `pmcont` it is important to choose `dsmin` 'sufficiently large', as a too small `ds` (obtained via many stepsize reductions possible for small `dsmin`) leads to essentially the same behavior of `pmcont` as `cont`, e.g., branch jumping.
- Rather use `p.pm.resfac` (residual decrease in each Newton step, default 10^{-3}) and `p.pm.mst` (number of different length predictors, default 4) to tune `pmcont`. Smaller `p.pm.resfac` means that more predictors are discarded (stricter behavior of `pmcont`). Often, it also helps to (maybe only on a 'difficult' segment of the branch) relax `p.nc.tol` (residual tolerance, default 10^{-8}) somewhat, e.g., set `p.nc.tol` = 10^{-6}.

Another option is adaptive mesh-refinement. This often helps if patterns start to 'drift' under continuation, as it introduces a (helpful) mesh-inhomogeneity, which may pin patterns at the 'right'

(desired) positions. It is not needed here, but at the end of `cmds2dhexb.m` we give an example of such an adaptive mesh–refinement on the `l1` branch, after which the branch continues qualitatively as before. Finally, consider the remarks on mesh–symmetry from §3.7.

In any case, if there are 'too many' solutions close to each other, the continuation is more complicated and may fail by, e.g.:

- Missing (important) bifurcation points due to a too large stepsize (which may be necessary to avoid undesired branch switching);
- Undesired branch switching even under strict settings for `pmcont`;
- Non-convergence of `pmcont` under too strict settings (too small `p.pm.resfac` and/or too large `p.nc.dsmin`).

Thus, to study pattern formation in 2D (or 3D, where the above problems usually become worse), we recommend to always start with a rather small domain. In particular on larger domain, a useful alternative to the 'continue and bifurcate' strategy used so far may be a direct search for patterns of interest, described next.

3.8.2 Deflation

So far we implicitly assumed that all 'relevant' solution branches can be found by starting from a known (trivial) branch, continuation and branch switching. However, there may be (relevant) branches not connected to the starting branch, e.g., bifurcations may be imperfect, or a trivial branch may not be available and it may be unclear how to find a first solution. In this case, we may simply generate, e.g., a number of initial guesses for u at fixed λ, run Newton loops and hope to converge to a solution.[2] However, if we aim to find to several solutions at fixed λ, then using several different initial guesses may not work, as we may always converge to the same solution, or not converge at all.

There are (at least) two methods aiming to improve this. The first are called minimax algorithms, which, essentially, exclude the span of previously found solutions from the search directions for new solutions. This has been implementated in `Matlab` for (scalar) PDEs in [LZ01, LZ02] and linked with `pde2path` in [Kue15].

A related method which is readily available in `pde2path` is deflation, which in the context of continuation has been (re)proposed in [FBF15], see also [FBB16], and [CKF18]. Assuming that we already have a number of solutions $(u^{(1)}, u^{(2)}, \ldots, u^{(l)})$, $l \geq 1$, of $G(u) = 0$ (at fixed λ), the idea is to modify the problem to

$$F(u; u^{(1)}, \ldots, u^{(l)}) := \mathcal{D}(u; u^{(1)}, \ldots, u^{(l)})G(u) = 0, \tag{38}$$

with a deflation operator $\mathcal{D}$ which, essentially, goes to ∞ fast enough as $u \to u^{(j)}$ for $j \in \{1, \ldots, l\}$ to exclude the roots $u^{(j)}$ of G from F, i.e.,

$$\liminf_{i\to\infty} \|F(u; u^{(1)}, \ldots, u^{(l)})\| > 0 \text{ for all sequences } (u_i) \text{ such that } u_i \to u^{(j)}, j = 1, \ldots, l. \tag{39}$$

At the same time we require $F(u; u^{(1)}, \ldots, u^{(l)}) = 0 \Leftrightarrow G(u) = 0$ for all further roots of G. A simple

[2] As Newton loops only converge locally, it is sometimes helpful to first run some time-integration to decrease the residual, i.e., to get closer to a solution, see §3.8.3 for some concrete examples.

choice is (the scalar function)

$$\mathcal{D}(u; u^{(1)}, \dots, u^{(l)}) = \left(\alpha_1 + \Pi_{j=1}^{l} \|u - u^{(j)}\|_q^q\right)^{-1}, \tag{40}$$

where $\|\cdot\|_q$ is the standard q–norm in $\mathbb{R}^n$, and q and α are suitable parameters, and naturally the roots $u^{(j)}$ must be considered with their multiplicity for (39) to hold. In [FBF15, FBB16] it is shown how this choice 'deflates' the domain of attraction of known roots of G for the Newton loops for F, and illustrated by various examples, including nonlinear PDE problems (after FEM discretization). In this context, $\partial_u G$ is sparse, but

$$\partial_u F = \mathcal{D}\partial_u G + G\partial_u \mathcal{D} \tag{41}$$

is dense. However, since $G \in \mathbb{R}^{n_u \times 1}$ and $\partial_u \mathcal{D} \in \mathbb{R}^{1 \times n_u}$, $\partial_u F$ is a rank-one correction of $\mathcal{D}\partial_u G$, which for instance can be handled with Sherman–Morrison–Woodbury (SMW) formulas [PTVF07, §2.7.3], see also §5.6.

For the (discretized) PDEs it may be useful to replace $\|u\|_q$ by $\|u\|_{L^q(\Omega)}$ (discretized), and we choose the default $q = 2$ and $\alpha_1 = 1$. Moreover, we found it more robust to modify $\mathcal{D}$ to

$$\mathcal{D}(u; u^{(1)}, \dots, u^{(l)}) = 1/\left(\alpha_1 + \Pi_{j=1}^{l} \min(\|u - u^{(j)}\|_q^q, \alpha_2)\right), \tag{42}$$

with default $\alpha_2 = 1$. The min is used because for large l the term $\Pi_{j=1}^{l}\|u - u^{(j)}\|_q^q$ may otherwise become large even if u is close to some $u^{(j)}$ (but far from others). This $\mathcal{D}$ is now longer differentiable on the null sets $\|u - u^{(j)}\|_q = \alpha_2$, but in practice this is not a problem, and an advantage is that this way $\mathcal{D}$ is flat away from the known roots, $\mathcal{D} = 1/\alpha_2$, and hence $G\partial_u\mathcal{D} = 0$ in (41). If $\partial_u\mathcal{D} \neq 0$, then often lumped approximate Jacobians also work as an alternative to SMW formulas in the Newton loops for (41).

Remark 3.4. a) The choices of $(q, \alpha_1, \alpha_2) = (2, 1, 1)$ and of the norm $\|u\|_2$ vs $\|u\|_{L^2(\Omega)}$ are essentially heuristic, and may have to be adapted for any given problem, or even depending on the initial guess for the deflated Newton loop and the solutions already computed. See [FBF15, §4.3,§4.4] for a detailed discussion of the hard–to–predict influence of q and α_1 on how many solutions will be found. On the other hand, [FBF15] reports that for most choices of (q, α) at least some solutions are found, and the same holds for the example below and further examples in `pde2path`.

b) In [CKF18] the values $(q, \alpha_2) = (2, 1)$, the $\|\cdot\|_{L^2(\Omega)}$ norm, and $\alpha_3 = \infty$, are used, but with a modified $\mathcal{D}$ that for a given solution deflates the full group orbit under rotation. ⌋

Deflation for the SH equation on $\Omega = (-2\pi, 2\pi)^2$ with pBC. In `shpbc/cmdssq_defl.m`, as an example we use deflation to compute solution branches not connected to the trivial branch $u \equiv 0$ for the SH equation on $\Omega = (-2\pi, 2\pi)^2$ with pBC, with sample results given in Fig. 15. Here we simply init with $u = 0$ at $\lambda = 1$, and subsequently run a deflation with standard parameters $q = 2$, $(\alpha_1, \alpha_2, \alpha_3) = (1, 1, 1)$ and initial guesses obtained from *random* Fourier coefficients of wave vectors near the unit circle such as $k = (1, 0.25), k = (0.25, 1)$ and so on, combined with random spatial shifts. If this yields convergence, then the found solution is added to the list `p.defl.u` of known solutions, and the next deflation is tried with the same initial guess. After no more solutions are found from this initial guess, the next initial guess is tried. For 10 initial guesses, this give about 4 solutions on average, which, due to the rather unstructured character of the initial guesses are typically of the shape in Fig. 15(a), rather than regular stripes or spots. After the deflation, we continue the solutions

2 and 3 in λ, yielding Fig. 15(b,c). Continuing in negative direction, the branches show folds, and altogether they are not (directly) connected to any of the primary bifurcating branches from Fig. 12. Incidentally, at the starting point $\lambda = 1$, both solutions are stable, but generally mostly unstable solutions were found in the deflation.

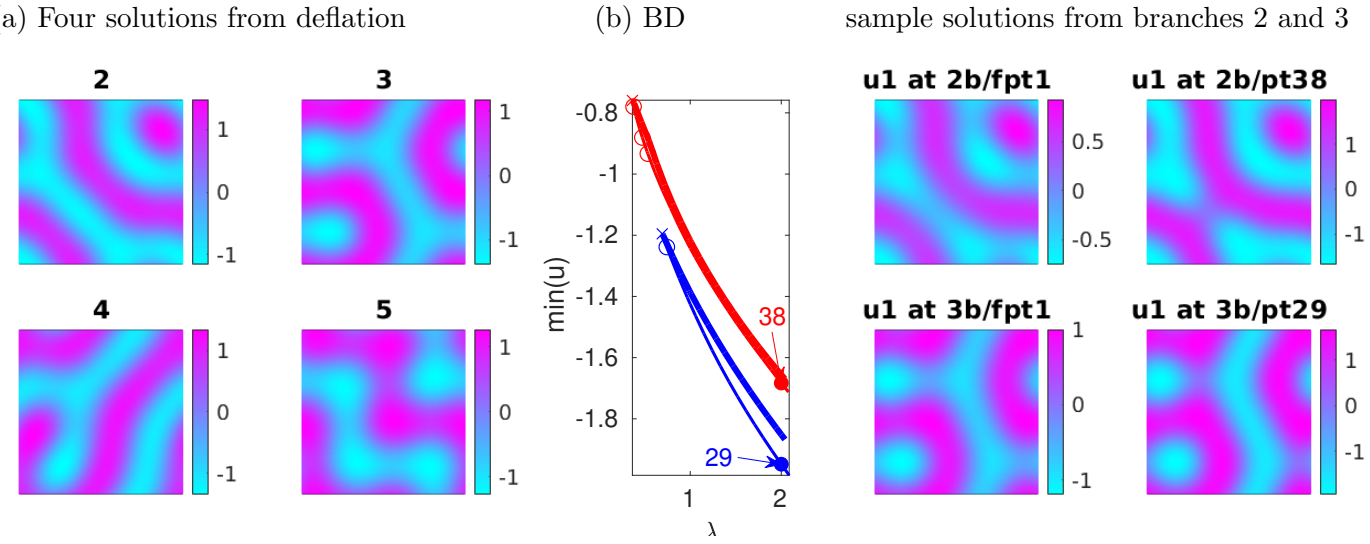


Figure 15: Deflation for the SH equation (3) at $\lambda = 1$ over $\Omega = (-2\pi, 2\pi)^2$ with pBCs in x and y, $\nu = 0$. Using $u \equiv 0$ as primary known solution, ten searches from initial guesses with random ($|k| \approx 1$) Fourier coefficients, yield the four solutions from (a). (b,c) Continuation of 2 and 3 and sample solutions.

3.8.3 Patterns from (educated) guesses and time–integration, isolas

In case one is primarily interested in a particular pattern u^*, which one knows to exist, there is the option to use a rough initial guess of $\tilde{u}$ for u^* and aim to converge to u^* by a Newton loop. If one additionally knows (or expects) the pattern to be stable, then it might be helpful or even necessary to first improve the initial guess by running some time integration, aka direct numerical simulation (DNS). After the system has then come sufficiently closed to a (the) desired stationary solution u^*, again a Newton loop can be tried to compute u^*.

2D. A simple example is given in `sh/cmds2dtint.m`, with some results plotted in Fig. 16. Here we use same domain as in Fig. 14 and aim to directly obtain a 'hex2stripes' front as in Fig. 14(f). For this we use initial guesses of the form

$$u_1(x,y) = \begin{cases} \cos(x) + \cos(x/2)\cos(\sqrt{3}y/2) & x \le 0 \\ \alpha\cos(x) & x \ge 0 \end{cases}, \quad u_2(x,y) \equiv 0, \tag{43}$$

α=2. Then, even though we only seed u_1, and only with a rough guess, a Newton loop on this $\tilde{u}$=(u_1, u_2) takes us directly to our 'desired pattern' `solution 1` in Fig. 14(a). On the other hand, if we take a guess $\tilde{u}$ too far off, then a direct Newton loop may not converge, or may converge to an 'undesired pattern'. For instance (43) with α=4 in Fig. 14(b), leads to the stripe pattern `solution 2`.

Often, it helps to use the guess $\tilde{u}$ as an initial condition and run some time–steppers. Time integration is not a core feature of `pde2path`, but we do provide a number of simple semi-implicit

time steppers, described in [?], which essentially only need the struct `p` as main input. The main time-steppers are `tint, tintx` (general purpose, where the `x` stands for more comprehensive output such as time-series of the residuals $\mathrm{res}(t) = \|G(u(t)\|_\infty)$, and `tints, tintxs` (for semilinear systems, with pre-factoring of the stiffness matrix). If we use this on `guess 2` until $t = 5$, then we obtain the residuals shown in Fig. 14(c), and a subsequent Newton loop takes us to the desired `solution 1`.

```
%% patterns from initial guesses; long 2D hex-domain; init and zero-branch
lx=4*pi;nx=round(3*lx);ly=2*pi/sqrt(3);lam=0.2;nu=1.3;par=[lam; nu]; sw.sym=2;
sw.ref=1; ndim=2; p=shinit(p,nx,lx,ly,ndim,par,sw); dir='tint'; p=setfn(p,dir);
po=getpte(p); x=po(1,:)'; y=po(2,:)'; % extract coord from p
%% a 'good' initial guess for hex2str front, and hex2str front from Newton loop
p.u(1:p.np)=(cos(x)+cos(x/2).*cos(sqrt(3)*y/2)).*(x<=0)+2*cos(x).*(x>0);
spl(p,''); title('initial guess 1'); r=norm(resi(p,p.u),'inf');
[u,res,iter]=nloop(p,p.u); p.u(1:p.nu)=u(1:p.nu);
fprintf('initial res=%g, res=%g after %i iteration\n',r,res,iter);
spl(p,''); title('solution 1'); pause, clf(2); p=pmcont(p,20); % plot, then cont
%% a 'bad' initial guess for hex2str front, Newton loop goes to stripes
u0=(cos(x)+cos(x/2).*cos(sqrt(3)*y/2)).*(x<=0)+4*cos(x).*(x>0); p.u(1:p.np)=u0;
spl(p,''); title('initial guess 2'); r=norm(resi(p,p.u),'inf');
[u,res,iter,Gu,Glam,p]=nloop(p,p.u); p.u(1:p.nu)=u(1:p.nu);
fprintf('initial res=%g, res=%g after %i iteration\n',r,res,iter);
spl(p,''); title('solution 2');
%% to obtain hex2str front, do a few steps with tintxs: preparations
p.u(1:p.np)=u0; t1=0;ts=[];nc=0;dt=0.01;nt=500;pmod=20;smod=100; p.mat.Kadv=0;
%% the tint loop, repeat this cell until residual is small (here just once)
[p,t1,ts,nc]=tintxs(p,t1,ts,dt,nt,nc,pmod,smod,@nodalf);
%% plot time series of res
tss=5; plot(ts(1,tss:end),ts(2,tss:end));axis tight;legend('res');xlabel('t');
%% Newton loop after tint, then cont
[u,res,iter]=nloop(p,p.u); p.u(1:p.nu)=u(1:p.nu);
fprintf('initial res=%g, res=%g after %i iteration\n',r,res,iter);
plotsol(p,1,1,2); pause; clf(2); p=pmcont(p,20);
```

Listing 6: `sh/cmds2dtint.m`. Obtaining solutions from initial guesses, possibly combined with some time-integration (lines 17-21).

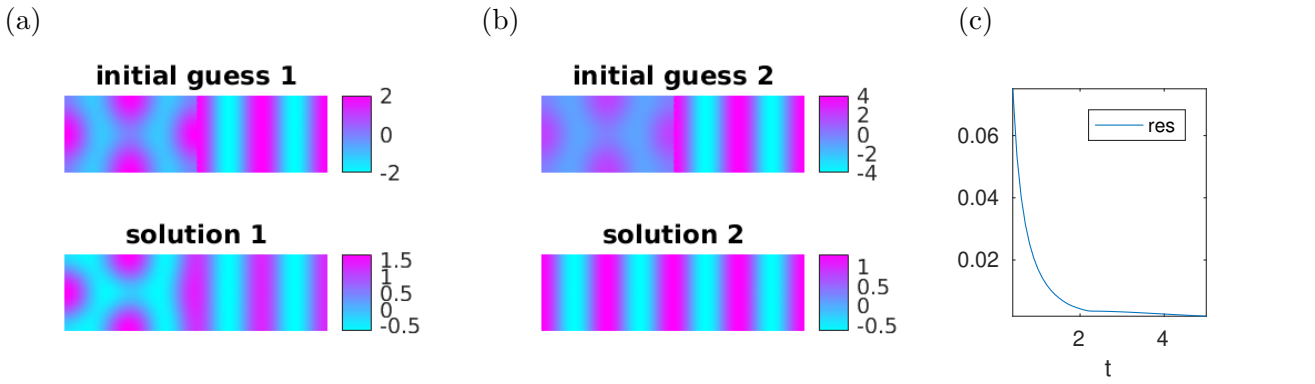


Figure 16: Obtaining solutions from guesses, if necessary including time-integration. (a) A 'reasonable' initial guess for a hex-to-stripes front, yielding the desired solution directly from a Newton loop, $\nu = 1.3$, $\lambda = 0.2$. (b) A 'bad' initial guess for a hex-to-stripes front; here the Newton loop gives the stripe solution. However, if we run `tintxs` on initial guess 2, then at $t = 5$ the solution is sufficiently close to the front for a Newton loop to converge to this desired solution.

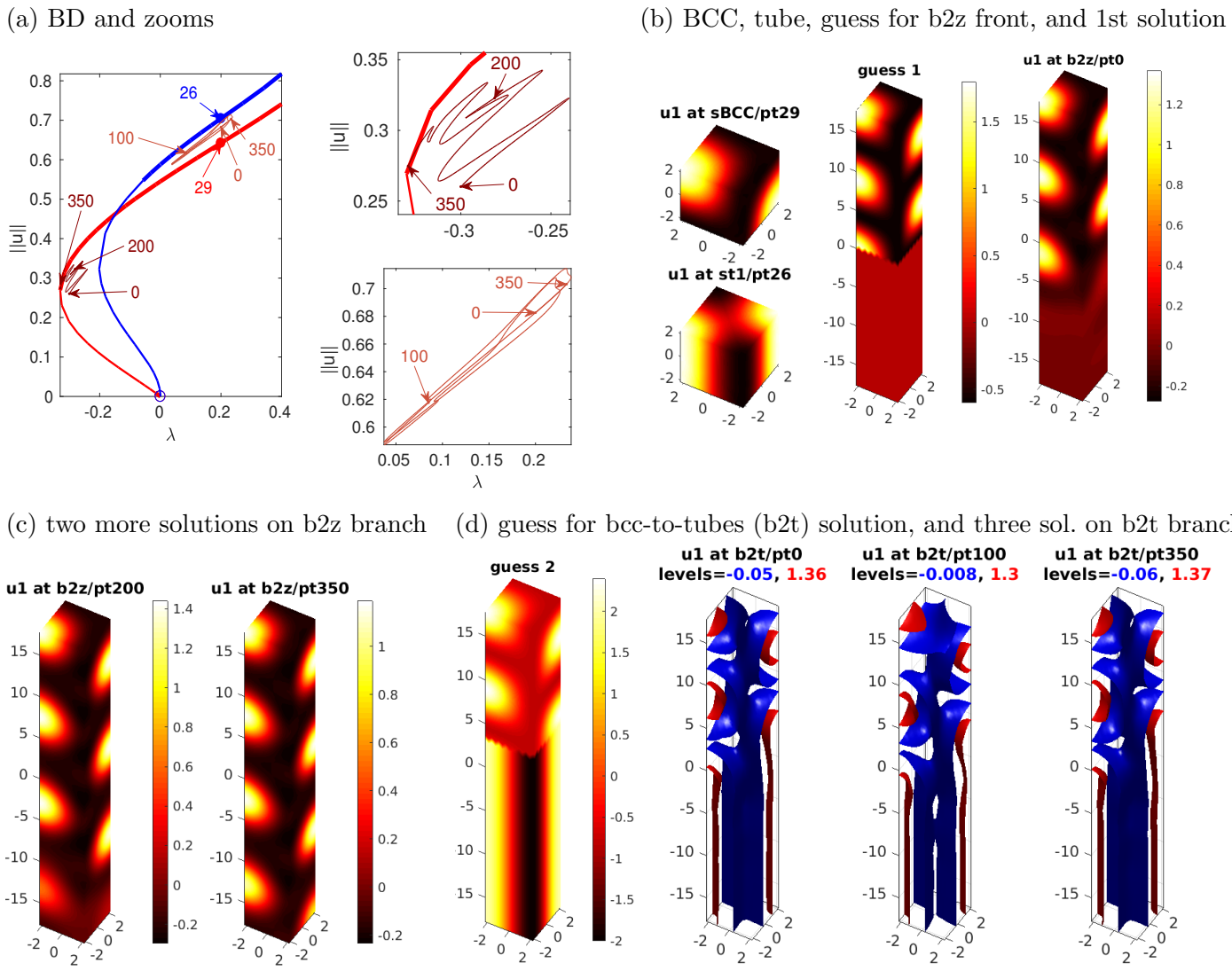


Figure 17: Results from `sh/cmdsBCClong.m` for (3) on $\Omega = (-l_x, l_x)^2 \times (-l_z, l_z)$, $l_x = \pi/\sqrt{2}$, $l_z = 8l_x$, $\nu = 1.5$. (a) BD of BCCs (red), tubes (blue), and `b2z` front branch (dark brown) and `b2t` isola (light brown). (b) BCC and tube plotted over small domain; guess for `b2z` front, and solution obtained from Newton loop. (c) two more solutions on the `b2z` branch. (d) `b2t` guess, and solution plots. $n_p = 17375$ grid points and $n_t = 101577$ tetrahedral elements. Computation of `b2z` and `b2t` branches takes about 20Min on an i7 laptop computer.

3D. In Fig. 17 we proceed similarly in 3D. Here, as in Fig. 5 (1D) and in Fig. 8 (2D), for 'sufficiently long' 3D cuboids we expect snaking branches of localized BCCs in the bistable range of BCCs and the trivial solution. In `cmdsBCClong.m` we first compute a BCC branch (red) and a tubes branch (blue) on the minimal domain $\Omega = (-l_x, l_x)^3$, $l_x = \pi/\sqrt{2}$, see the first two plots in Fig. 17(b).[3] Then we let $\Omega = (-l_x, l_x)^2 \times (-8l_x, 8l_x)$ and $\lambda = -0.3$, and try the guess

$$u_1(x,y,z) = \begin{cases} 0.4\mathrm{Re}\left[\sum_{j=1}^{6} \exp(\mathrm{i}k_j \cdot (x,y,z))\right], & z \geq 0 \\ 0 & z \geq 0 \end{cases}, \quad u_2(x,y,z) \equiv 0, \tag{44}$$

[3]For speed and convenience, the red and blue BCC and tubes branches in the left panel of (a) are from the minimal domain $\Omega = (-l_x, l_x)^3$, including the stability, but we can obtain the same branches on $\Omega = (-l_x, l_x)^2 \times (-8l_x, 8l_x)$, with the same stability for the BCCs and almost the same stability for the tubes.

with k_j from (31) to obtain a BCC-to-zero front `b2z`. A Newton loop takes us to `b2z/pt0`. Continuing this branch we find it connected to the BCC branch near zero (not shown) and near the BCC fold (zoom in (a), top right panel). For speed, in this continuation we switch off bifurcation detection and spectral computations, and instead here remark that this snaking branch consists of alternating stable and unstable segments, as expected.

As an additional example for the multitude of patterns on this long domain, similar to Fig. 16 we seek a BCC-to-tubes front branch `b2t` in the bistable range of BCCs and tubes. We let $\lambda = 0.2$, and as a guess use

$$u_1(x,y,z) = \begin{cases} 0.4\mathrm{Re}\left[\sum_{j=1}^{6} \exp(\mathrm{i}k_j \cdot (x,y,z))\right], & z \geq \pi \\ \cos((x+y)/\sqrt{2}) + \cos((x-y)/\sqrt{2}), & z \leq \pi \end{cases}, \quad u_2(x,y,z) \equiv 0, \tag{45}$$

see the first plot in Fig. 17(d). A Newton loop takes us to the solution `b2t/pt0`. Continuing this branch we find that it forms an isola (bottom right panel in (a)): After going back and forth twice, near the 340th continuation point it returns to `b2t/pt0`. For speed we again switch off the stability and bifurcation detection computations, and remark that by checking stability a posteriori we find that significant segments of this branch consist of stable solutions.

Remark 3.5. In [UW20], similar results can be found for the 3D Brusselator RD-system, i.e., snaking branches of fronts between BCCs and the trivial (spatially homogeneous) solution (`b2z`–branch), and between BCCs and tubes (`b2t`–branch). There, the `b2z`–branches were obtained via bifurcation from (subcritical) BCC–branches, and a `b2t`–branch via "educated" initial guesses. ⌋

In Fig. 18 we compare results from mesh adaptation by `trullekrul` (magenta branch) with the original `b2z` branch (brown) and the same branch on a finer uniform mesh (blue, on top of the brown branch). The result is that using `trullekrul` we can compute the branch about 50% faster than on the original mesh, while the blue branch takes about three times as long as the brown one. We refer to `cmdsBCClongada.m` for details and the `trullekrul` parameter choices [Uec19c]. See also Fig. 9 and Listing 4 for the analogs in 2D.

4 Demo `schnakpat`

In the demo `schnakpat` we consider the (modified) Schnakenberg reaction diffusion system

$$\partial_t U = D\Delta U + F(U), \quad U = \begin{pmatrix} u \\ v \end{pmatrix}, \quad F(U) = \begin{pmatrix} -u+u^2v \\ \lambda - u^2 v \end{pmatrix} + \sigma \left(u - \frac{1}{v}\right)^2 \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \tag{46}$$

with diffusion matrix $D = \left(\begin{smallmatrix} 1 & 0 \\ 0 & d \end{smallmatrix}\right)$ and parameters $\lambda > 0$ and $\sigma \in \mathbb{R}$. In suitable parameter regimes, (46) shows Turing bifurcations from the homogeneous branch $(u,v) = (\lambda, 1/\lambda)$, and, in a nutshell, we may more or less expect all phenomena explained for the SH equation in the previous sections also in (46). The term involving σ does not change the homogeneous branch or the linearization around it, but has been introduced to tune the primary bifurcation from super– to subcritical. The system (46) has already been considered as a `pde2path` and pattern formation model problem in [UWR14, §4.2] (with $\sigma = 0$), and in [UW14] and [dW17]. Here we want to give a concise and updated demo which besides illustrating the use of `qswibra` and `seltau` also recovers the main results from [UW14]. Table 5 gives an overview of the involved files. We fix $d = 60$ throughout, and focus on five tasks, namely:

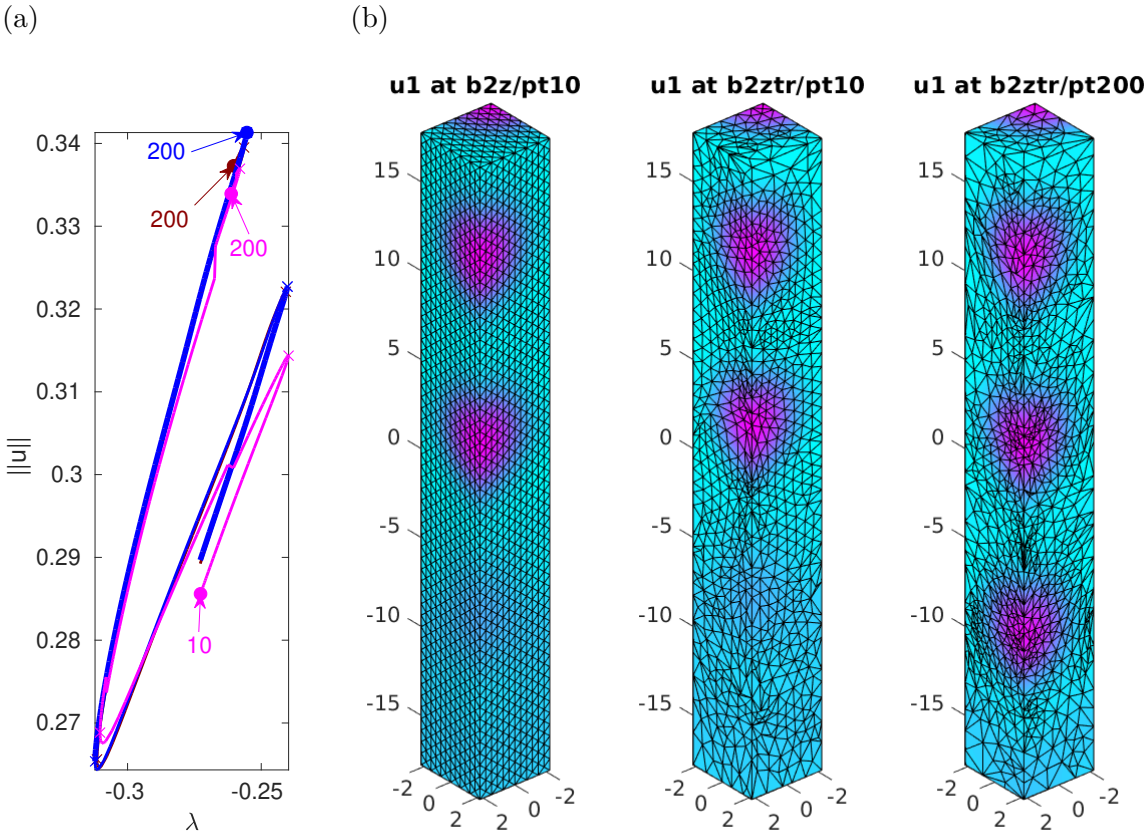


Figure 18: Results from `sh/cmdsBCClongref.m` for (3) on $\Omega = (-l_x, l_x)^2 \times (-l_z, l_z)$, $l_x = \pi/\sqrt{2}$, $l_z = 8l_x$, $\nu = 1.5$. (a) comparison of the same branch on different meshes. Brown (b2z): uniform mesh, $n_p = 11360$; blue: original uniform mesh, $n_p = 17375$, on top of brown branch; magenta (b2ztr): adaptation by `trullekrul`, first reducing `b2z/pt10` to $n_p = 5982$, then gradually increasing to $n_p = 8484$ (at pt200, due to the growth of the next spot). The initial adaptation reduces the L^2 norm form the brown branch, and each subsequent adaptation also yields a small 'jump' on the magenta branch. (b) sample plots.

- explain a trick to let `pde2path` display the dispersion relation for homogeneous states;
- compute a basic bifurcation diagram of 1D patterns, including snaking branches of localized patterns;
- generate a basic bifurcation diagram of 2D patterns over small 2D domains;
- compute snaking branches of localized 2D patterns over long 2D domains;
- compute the primary bifurcations in 3D for the SC and BCC lattices.

4.1 1D: computing the dispersion relation, basic branches, and snaking

For (46) (with $d = 60$ fixed) we know the parameter value for first Turing bifurcation from the homogeneous branch $(u, v) = (\lambda, 1/\lambda)$ and the critical wave number, namely $\lambda_c = \sqrt{60}\sqrt{3 - \sqrt{8}} \approx 3.21$ and $k_c = \sqrt{\sqrt{2} - 1}$. Nevertheless, in `cmds1d` we start the 1D computations on a small domain to illustrate the usage of `spufu` (see Listing 7) to plot the dispersion relation, see Fig. 19. `spufu.m` is a modification of the `pde2path` library function `stanufu`, and should be easily adaptable to any RD system. See also demo `hopfdemos/extbru` explained in [Uec20a] for a 3 component case.

```
% addition to STANUFU: plot the dispersion relation!
n=p.np;nu=p.nu;par=p.u(p.nu+1:end);u=[p.u(1);p.u(n+1)]; % hom-state 2-vector
```

Table 5: Scripts and functions in `demos/schnakpat`.

script/function	purpose, remarks
cmds1d	display the dispersion relation, compute a basic 1D bifurcation diagram, and fold continuation, yielding Fig. 19–21.
cmds2dsq	compute a basic bifurcation over a square, primary bifurcations are pitchforks.
cmds2da	compute a basic bifurcation over a small rectangle, Fig. 22.
cmds2db	compute a branch of spots embedded in stripes over a long 2D domain, Fig. 23.
cmds3dSC/BCC	compute basic bifurcation diagrams over SC and BCC cubes.
schnakinit	initialization, 1D and 2D
sG, nodalf, sGjac	as usual
spjac	Jacobian for fold continuation
spufu	auxiliary function for plotting the dispersion relation
schnakbra	modification of `stanbra` (to include the L^8 norm on the branch)

```
u=[u;par];p.np=1;p.nu=2;[f1u,f1v,f2u,f2v]=njac(p,u); J=[[f1u f1v];[f2u f2v]];
kv=0:0.01:1.5; kl=length(kv); muv=zeros(2,kl); % provide wave-nrs and mem
d=par(3); % diffusion param.; this and k-range usually only problem dep. things
for i=1:kl %  now loop over wave-nr and compute Evals
    k=kv(i); K=[[k^2 0];[0 d*k^2]];  A=J-K; % Jac in Fourier space
    mu=eig(A); [mus, ix]=sort(real(mu),'descend'); % sorted eigenvalues
    for j=1:2; muv(j,i)=mu(ix(j)); end
end
figure(10); clf; plot(kv,real(muv(1,:)),kv,imag(muv(1,:))); % plot leading Eval
```

Listing 7: (Selection from) `schnakpat/spufu.m`. Modification of (addition to) `stanufu` to plot the dispersion relation. In line 22 we extract (u, v) at just one point, shorten the vector of unknowns accordingly, and compute the local Jacobian $\partial_{\vec{u}} f$ of the 'nonlinearity' f given in `nodalf`, where we use that this is already encoded in `njac` (and called accordingly in sGjac). To compute $\mu(k)$ we then loop over k and numerically solve the pertinent 2×2 eigenvalue problem. This can be modified to other two–component or general N–component systems in a straightforward way, where essentially N, the pertinent wave-number range `kv`, and the diffusion constant(s) are the problem dependent points in `spufu`.

```
%% C1: init on small (arbitrary) 1D domain, and use spufu to plot disp rel.
p=[];lx=1;nx=20; par=[3.5,-0.6,60]; p=schnakinit(p,lx,nx,par); p.nc.dsmax=0.5;
p.fuha.ufu=@spufu; % set user function to "spectral plot ufu"
p.sol.ds=-0.1; p=setfn(p,'dummy'); p=cont(p,20); % continue for plotting disp
%% C2: init on larger domain, with rather large sigma to have subcrit. stripes
p=[]; kc=sqrt(sqrt(2)-1); lx=5*2*pi/kc; nx=500;
p=schnakinit(p,lx,nx,par); p=setfn(p,'h1D');
p=findbif(p,6); % many bif-points close to each other, use findbif
p=cont(p,20);  % a few more steps (for later plotting)
%% C3: stripes 1,2,3,6, and 1st snake on stripes 1
p=swibra('h1D','bpt1','1D1',0.1); p=cont(p,5);
```

```
p=swibra('1D1','bpt1','sn1D',0.1); p=cont(p,110);
```

Listing 8: (Selection from) `schnakpat/cmds1d.m`. In C1 we use `spufu` (on a small domain) to display the dispersion relation. In C2 we then start the computations on a large domain, $l_x = 5\pi/k_c$, which means that the primary Turing branch T1 ($k = k_1 := k_c \approx 0.64$) has 10 periods in Ω. Then we follow the Turing branches T2 ($k = k_2 \approx 0.61$), T3 ($k = k_3 \approx 0.7$) and T6 ($k = k_6 \approx 0.58$) since in particular the branch T6 with only 7.5 periods in Ω moves furthest to the right. Additionally, we follow a front bifurcating on T1. The remainder of the script deals with plotting, and with fold continuation.

Listing 8 shows the start of `schnakpat/cmds1d.m`. After finding k_c in C1, in C2 we restart the computations on a domain tuned to the critical mode $\cos(k_c x)$, i.e, of length $10\pi/k_c$, see Fig. 20. We

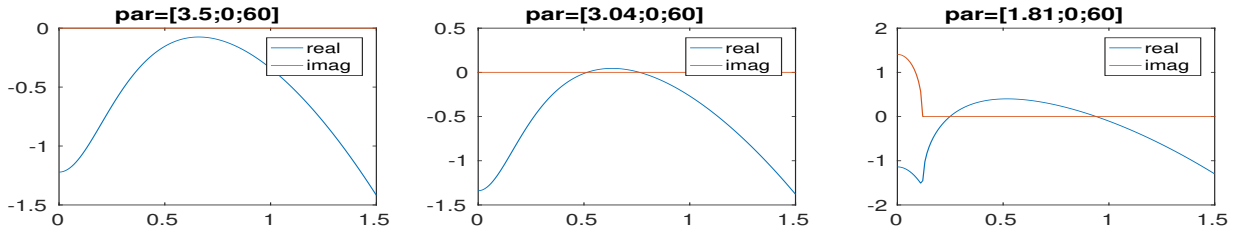


Figure 19: Preparatory step (not strictly necessary) for (46): plotting the dispersion for different λ.

follow the Turing branches T1, T2, T3 and T6, associated to the first three and the sixths branch point (counting from the right), and a snaking branch S1 bifurcating from T1. The rather large value of σ has the disadvantage that the periodic patterns are somewhat nonphysical because u does not stay positive. However, an interesting feature of $\sigma = -0.6$ is that the 'most subcritical' branch is not the primary Turing branch T1, but (here) T6 with $k = k_6 \approx 0.58$. Moreover, on S1 the periodic patterns have wave-number k near k_6, and in particular the snake reconnects not to T1 but to T6.

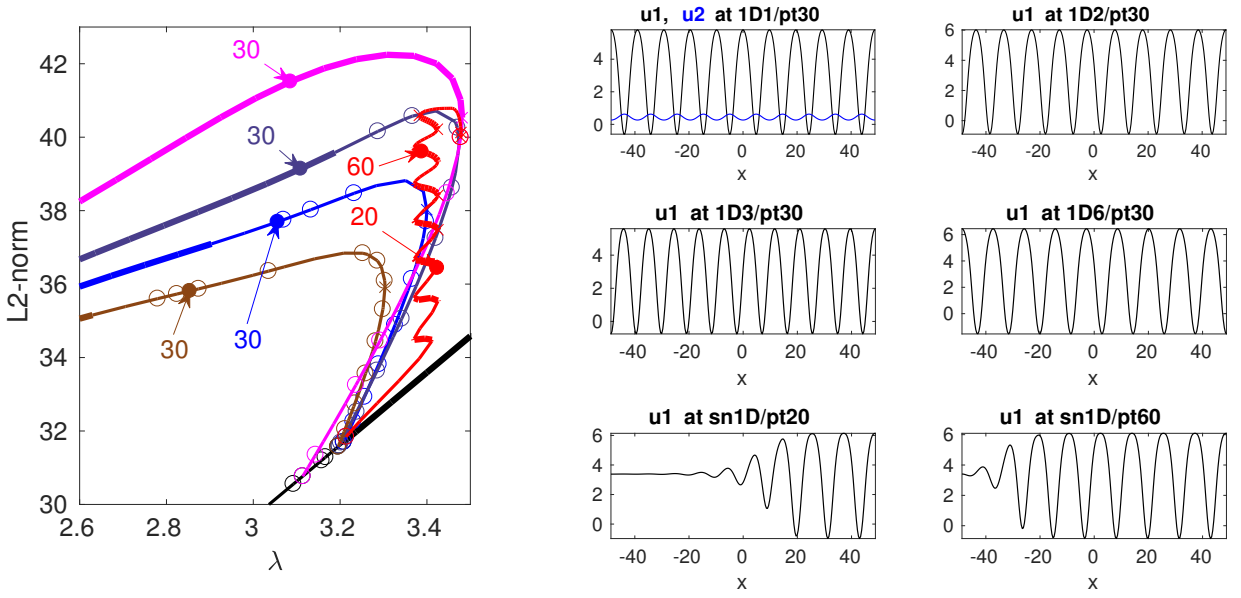


Figure 20: (46), $(\sigma, d) = (-0.6, 60)$, $l_x = 10\pi/kc \approx 48.8132$. Turing branches T1 (blue), T2 (dark blue), T3 (brown) and T6 (magenta), and a snaking branch of a front bifurcating from T1 but reconnecting to T6. In the solution plot of 1D1/pt30 we use the setting `plotsol('1D1','pt30',1,[1 2],'cl','k','b');` to plot both components.

In the remainder of `cmds1d` we follow the folds on T1, T3 and T6 as functions of σ, see Fig. 21. This illustrates the role played by σ for the structure of the bifurcation diagram: The primary branch T1 bifurcates subcritically only for $\sigma < \sigma_0 \approx -0.3$. Moreover T1 extends furthest to the right for all $\sigma > \sigma_1 \approx -0.5$ and becomes stable at its fold, respectively is stable directly after bifurcation for $\sigma > \sigma_0$.

4.2 2D: basic bifurcation diagram, and branches of localized patterns

To compute the 'standard' bifurcation diagram of stripes and hexagons in Fig. 22 we first let $\sigma = 0$ and proceed similarly as for the SH stripes and hexagons in Fig. 7: Following the homogeneous branch over a domain $\Omega = (-l_x, l_x) \times (-l_y, l_y)$ with $l_x = \pi/k_c$ and $l_y = l_x/\sqrt{3}$, we find a double branch point at $\lambda = \lambda_c = \sqrt{60}\sqrt{3 - \sqrt{8}} \approx 3.21$. We then use `qswibra` to switch to the hexagon branch, which we

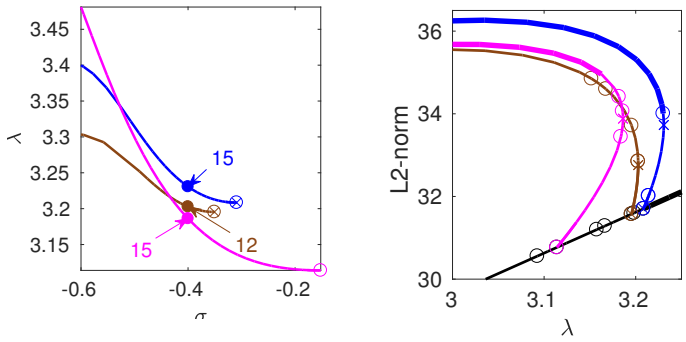


Figure 21: Continuation of folds on T1,T3 and T6 in σ, and behavior of branches at $\sigma = -0.4$.

follow in "both directions" (positive and negative `ds`) to subsequently discuss secondary connections between the spots and "+" stripes, and the gaps and "-" stripes. Since here the kernel vectors $\phi_{1,2}$ are clean stripes, we skip a call to `cswibra` and use `gentau` to follow the stripe branches. The mixed mode connections between stripes and spots are obtained from branch–switching where the stripes lose/gain stability. See `cmds2da.m` and Fig. 22. Additionally we remark that on the gap branch there is a Hopf bifurcation point near $\lambda = 2.75$. We do not discuss this here, but the bifurcating branch of oscillating gaps can be obtained from the commands at the end of `schnakpat/cmds2da.m`

The mixed mode branches and the associated bistability ranges, for instance between "+" stripes and spots, by analogy with the SH equation suggest the existence of localized patterns over patterns, e.g., of spots embedded in stripes. A multitude of such solutions has been discussed in [UW14], and in Fig. 23 we only illustrate one example, computed in `cmds2db.m`. Here we essentially increase the domain length in x, and then find bifurcation points on the mixed mode branches where branches of localized patterns bifurcate, which return to the mixed mode branch at the other end. The only non-standard setup in the software is that we modify `stanbra` to `schnakbra` and set `p.fuha.outfu=@schnakbra`. Here we append the (normalized) L^8 norm $\|u\|_8 = \left(\frac{1}{|\Omega|}\int_\omega u^8\,\mathrm{d}x\right)^{1/8}$ to the branch output, because this yields a bigger difference between spots and stripes than the L^2 norm, and is therefore more suitable for plotting.

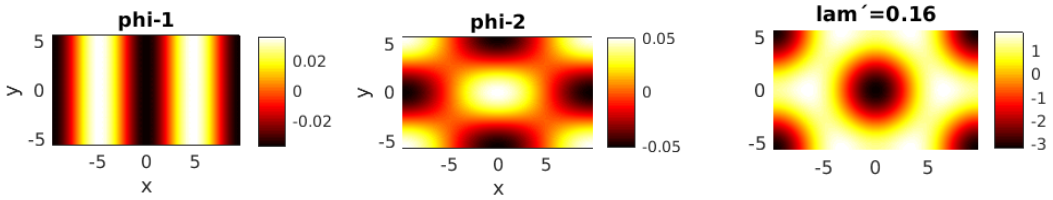
```
%% commands for Schnakenberg on a small 2D domain with hex lattice
p=[]; kc=sqrt(sqrt(2)-1); lx=2*pi/kc; ly=lx/sqrt(3); par=[3.3, 0, 60];
nx=35;sw.sym=2;p=schnakinit(p,[lx,ly],nx,par,sw); % init with criss-cross mesh
p.pm.resfac=1e-4; p.sol.ds=-0.1; p=setfn(p,'hom');pause; p=cont(p,30);
%% hex via qswibra, continue in both directions
p0=qswibra('hom','bpt1'); p0.nc.dsmin=0.1; p0.sw.bifcheck=1; pause
p=seltau(p0,3,'h-',2); p.sol.ds=0.1; p=pmcont(p,30);
p=seltau(p0,3,'h+',2); p.sol.ds=-0.1; p=pmcont(p,30);
%% s+ and s- via gentau
p=gentau(p0,1,'s+'); p.sol.ds=-0.05; p=pmcont(p,20);
```

Listing 9: (Selection from) `schnakpat/cmds2da.m`. Here the kernel vectors are ϕ_1=stripes and ϕ_2=patchwork quilt, and `qswibra` computes the pertinent linear combination and $\lambda'(0)$ for the hex branch. From the inspection of ϕ_1, ϕ_2, for the stripe branch we then directly use `gentau`.

Remark 4.1. The analogs of Fig. 10 and Fig. 11 for the Schnakenberg model in 3D are computed in `cmds3DSC.m` and `cmds3DBCC.m`, and we only remark that:

- For (46) in 3D, the choice of a pseudo criss-cross meshes seems even more vital than for (3); over standard meshes, solutions quickly loose symmetry.
- The SC lamellas and rhombs can be continued very robustly via `cont`, while the tubes need

(a) Two numerical kernel vectors at the first BP, and a tangent τ from `qswibra`



(b) Basic bifurcation diagram, including 'bean' (mixed mode) branches

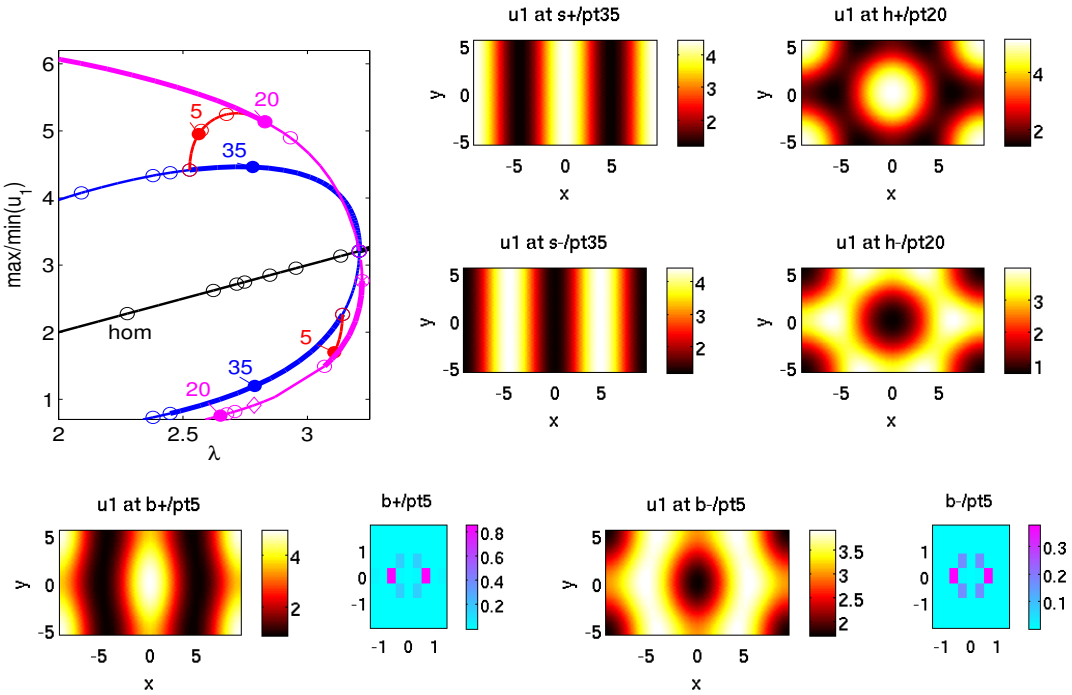


Figure 22: Results from `schnakpat/cmds2da.m` for (46) on a small rectangular domain $\Omega = (-l_x, l_x) \times (-l_y, l_y)$, $l_x = \pi/k_c$, $l_y = l_x/\sqrt{3}$, corresponding to a hexagonal dual lattice. (a) kernel at the first BP, and 'hex' bifurcation direction obtained from qswibra. (b) Bifurcation diagram and example solution plots; stripes (blue), hexagons (magenta), and mixed modes or beans (red), including Fourier plots.

`pmcont`, and at larger amplitude still tend to drift to lamellas.

- On the BCC domain, the tubes continue very robustly and become stable at large amplitude, while the continuation of the BCCs becomes more difficult because they tend to loose symmetry also over pseudo criss-cross meshes.

5 Five intermezzi

We give five short intermezzi dealing with more or less classical problems in pattern formation. The main purpose is to explain in a concise way how additional features of `pde2path` can be exploited, for instance branch-point continuation in the demo `shEck` (§5.1) to approximate the Eckhaus instability of rolls in the 1D SH equation, and the coupling with additional equations in the Cahn–Hilliard demo `CH` (§5.2). The demo `hexex` (§5.4) illustrates the use use of `gentau` to deal with multiple BPs of higher indeterminacy, while `chemtax` (§5.5) revisits a chemotaxis system from [UWR14, §4.1] to explain how to deal with quasilinear terms in the `OOPDE` setting. Finally, in the demo `shgc` (§5.6) we consider a SH equation with a global coupling, which thus requires customized linear system solvers and eigenvalue solvers.

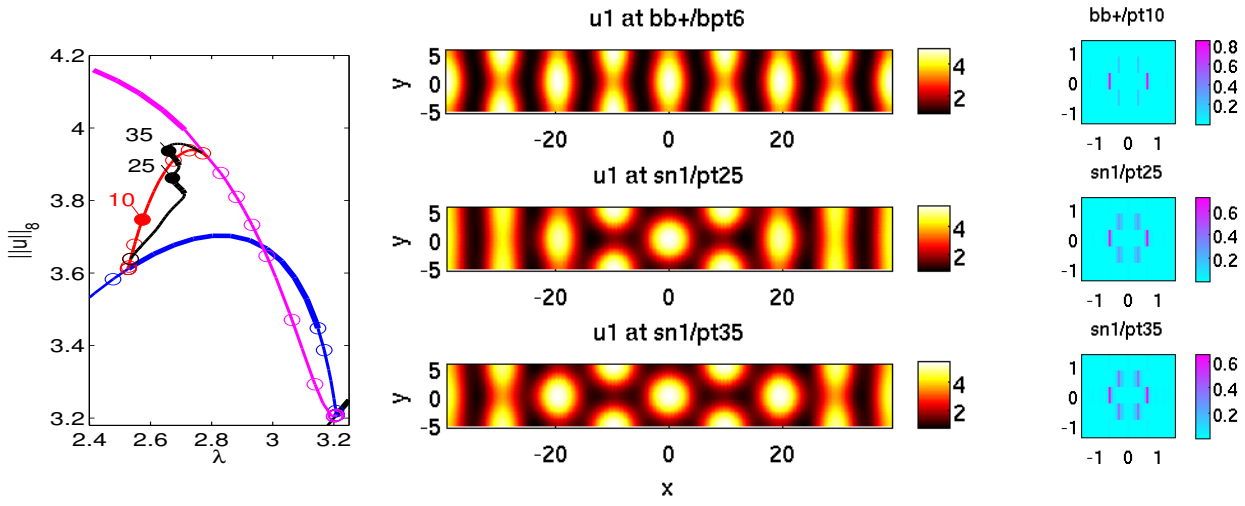


Figure 23: (46) on a long rectangular domain $\Omega=(-l_x,l_x)\times(-l_y,l_y)$, $l_x = 8\pi/k_c$, $l_y = \pi/(\sqrt{3}k_c)$. On the bean branch (red) between stripes (blue) and spots (magenta), there are bifurcation points leading to snaking branches of localized patterns. Right column: Fourier spectrum of $u_1 - \langle u_1\rangle$.

5.1 Approximation of the Eckhaus curve by BP continuation: `shEck`

An important result for the classical 1D SH equation (with a scaling parameter ℓ, which we first fix to $\ell = 1$)

$$\partial_t u = -(1+\ell^2\partial_x^2)^2 u + \lambda u - u^3, \tag{47}$$

is the so–called Eckhaus instability of stripes [Eck65]. In detail, for $x \in \mathbb{R}$ such that we have the continuum $k \in \mathbb{R}$ of admissible wave numbers, we have the supercritical bifurcation of stripes with wave number k at $\lambda = (1-k^2)^2$. However, except for $k = 1$ the bifurcating stripes are not stable directly at bifurcation, but only for

$$\lambda > \lambda_{\rm Eck}(k) := 3\kappa^2 - \kappa^3 + \mathcal{O}(\kappa^4), \quad \kappa = k^2 - 1. \tag{48}$$

Here we illustrate how this Eckhaus curve can be approximated on a finite domain via BP continuation. In `pde2path`, this is done similar to fold continuation, i.e., via the extended system [Mei00, §3.3.2]

$$H(U) = \begin{pmatrix} G(u,\lambda) + \mu M\psi \\ G_u^T(u,w)\psi \\ \|\psi\|_2^2 - 1 \\ \langle \psi, G_\lambda(u,w)\rangle \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad U = (u,\psi,w), \tag{49}$$

where (u,λ) is a (simple) BP (for the continuation in λ), ψ is an adjoint kernel vector, $w = (\lambda,\mu)$ with $w_1 = \lambda$ the primary active parameter and $w_2 = \mu$ as additional active parameter.

Remark 5.1. (a) The extended system (49) is regular at simple BPs, see [Moo80, Mei89, Mei00], and thus can be used for localization of (simple) BPs if a sufficiently good initial guess $(u,\psi,\lambda,0)$ is available. However, we (currently) hardly use this option, mainly because the implementation of $\partial_u(G_u^T\psi)$ (see below) requires some additional effort, and the localization by bisection is usually fast and accurate enough. See [UW17] for an example for BP localization via extended systems.

(b) Freeing a second parameter, i.e., setting $\tilde{w} = (\tilde{\lambda}, w)$ with *a new* $\tilde{\lambda}$, dropping the $\tilde{}$ and augmenting (49) with the usual arclength condition $p(U, \mathrm{d}s) = 0$, $U = (u, \psi, \lambda, w) \in \mathbb{R}^{2n_u+3}$, (49) can also be used for BP continuation, in which we are interested here. ⌋

To prepare the use of (49) for BP continuation, `pde2path` provides the call `p=bpcontini(p,newpar,...)`, which (internally) doubles the number n_u, of unknowns, stores ψ at $u_{n_u+1,\ldots,2n_u}$, and shifts the parameters to the pertinent new positions $2n_u + 1$:end (with n_u =old n_u). The main task then is to set up $\partial_u(G_u^T\psi)$ for the Jacobian

$$J_H = \begin{pmatrix} G_u & \mu M & G_\lambda & M\psi \\ \partial_u(G_u^T\psi) & G_u^T & \partial_\lambda(G_u^T\psi) & 0 \\ 0 & 2\psi^T & 0 & 0 \\ \psi^T\partial_\lambda G_u^T & G_\lambda^T & \psi^T\partial_\lambda G_\lambda & 0 \end{pmatrix}, \tag{50}$$

while G_u is already available, and all other derivatives can efficiently be done numerically and hence automatically. However, for semilinear problems $G(u, \lambda) = K(\lambda)u - Mf(u, \lambda)$, where the stiffness matrix $K(\lambda)$ does not depend on u, $\partial_u(G_u^T\psi)$ has a simple form. For instance, for a 2-component system with $f = \begin{pmatrix} f(u_1, u_2) \\ f_2(u_1, u_2) \end{pmatrix}$ we have

$$\partial_u(G_u^T\psi) = \left(\partial_u \begin{pmatrix} f_{1,u_1}\psi_1 + f_{2,u_1}\psi_2 \\ f_{2,u_1}\psi_1 + f_{2,u_2}\psi_2 \end{pmatrix}\right) M^T = \begin{pmatrix} f_{1,u_1u_1}\psi_1 + f_{2,u_1u_1}\psi_2 & f_{1,u_1u_2}\psi_1 + f_{2,u_1u_2}\psi_2 \\ f_{1,u_1u_2}\psi_1 + f_{2,u_2u_1}\psi_2 & f_{1,u_2u_2}\psi_1 + f_{2,u_2u_2}\psi_2 \end{pmatrix} M^T, \tag{51}$$

where expressions such as $f_{i,u_j,u_k}\psi_m$ are to be understood as pointwise multiplication. Thus, `bpjac` from Listing 10 yields the desired $\partial_u(G_u^T\psi)$, while Listing 11 shows pertinent cells from the script `cmdsEck.m`, and Listing 12 shows how to put the effective wave number k onto the branch for plotting.

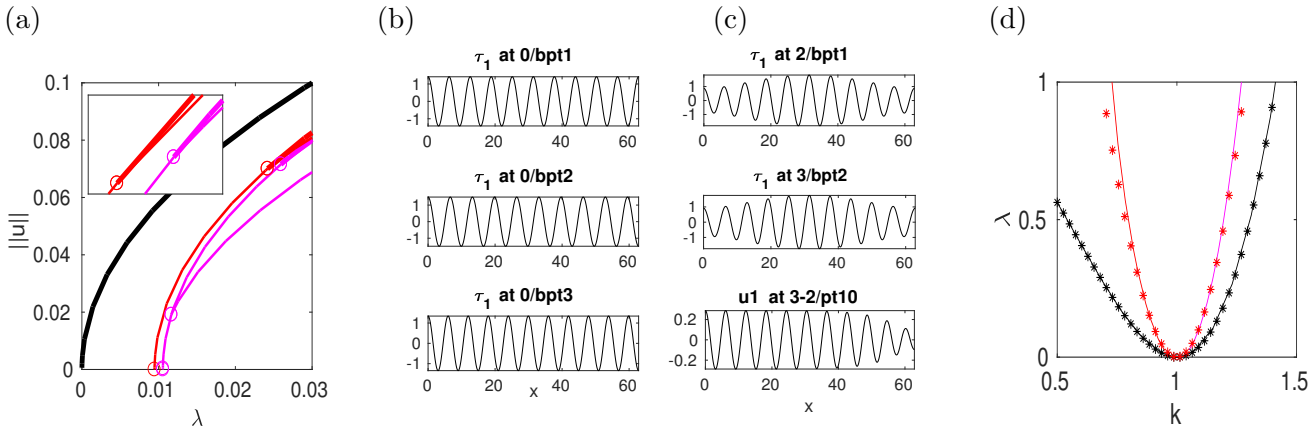


Figure 24: (a) First 3 bifurcating branches on $\Omega = (0, 20\pi)$, with wave numbers $k = 1, 19/20$ and $21/20$, and secondary bifurcations. Inset zoom near $\lambda = 0.025$. (b) tangent plots at the BPs on $u \equiv 0$. (c) tangent plots at the 'Eckhaus points' `b2/bpt1` and `b3/bpt2` (first two panels), and continuation of the branch bifurcating at `b3/bpt2`. (d) BP continuation, yielding the black existence curve, and two approximations of $\lambda_{\mathrm{Eck}}(k)$. These three curves are also compared with the formulas $\lambda = (1 - k^2)^2$ and $\lambda = \lambda_{\mathrm{Eck}}(k)$ from (48) (dots).

To compute (approximate) λ_{Eck} we consider (47) on $\Omega = (0, l_x)$ with $l_x = 20\pi$ and homogeneous Neumann BCs. The first three BPs from $u \equiv 0$ then have $k = 1$, $k = 19/20$ (dilated pattern) and $k = 21/20$ (compressed pattern), respectively, see Fig. 24(a,b). The 2nd and 3rd branches `b2` and

b3 start with 1 and 2 unstable eigenvalues, respectively, and they both gain stability at $\lambda \approx 0.025$. The bifurcating branches at these secondary bifurcation correspond to long wave modulations of the patterns, see Fig. 24(c). In (d) we then continue the primary bifurcation point `0/bpt1` ($(u,\lambda)=(0,0)$) and the secondary BPs `b2/bpt1` and `b3/bpt3` in the scaling parameter ℓ, i.e., set $w=(\ell,\lambda)$ in (49). For `0/bpt1` we naturally obtain the (continuous) existence-of-patterns curve $\lambda_{\rm ex}=(1-k^2)^2$ (black), where $k=\ell k_0$ with $k_0=1$, i.e., periodic solutions with minimal period k exists for $k>\lambda_{\rm ex}(k)$. Similarly, for `b2/bpt1` (with $k_0=19/20$) and `b3/bpt2` (with $k_0=21/20$) we obtain the red/magenta approximations of $\lambda_{\rm Eck}$.

```
function duGuph=bpjac(p,u) % for SH, gen.form, but only top left block nonzero
n=p.np; u1=u(1:p.np); par=u(2*p.nu+1:end); nup=par(2); ov=ones(n,1);
f1uu=2*nup*ov-6*u1; % only nonzero entry
f1uv=0*ov; f1vv=f1uv; f2uu=f1uv; f2uv=f1uv; f2vv=f1uv;
ph1=u(p.nu+1:p.nu+p.np); ph2=u(p.nu+p.np+1:2*p.nu);
M1=spdiags(f1uu.*ph1+f2uu.*ph2,0,n,n); M2=spdiags(f1uv.*ph1+f2uv.*ph2,0,n,n);
M3=spdiags(f1uv.*ph1+f2uv.*ph2,0,n,n); M4=spdiags(f1vv.*ph1+f2vv.*ph2,0,n,n);
duGuph=-[[M1 M2]; [M3 M4]]*p.mat.M;
```

Listing 10: `shEck/bpjac.m`. ψ is in `u(p.nu+1:2*p.nu)`, and the parameters are at `u(2*p.nu+1:end)`.

```
figure(2); clf; p=bpcontini('b2','bpt1',3,'bpc2a'); p.k=19/20; p.sol.ds=-0.01;
p.plot.bpcmp=1; p.sw.spcalc=0; p.sw.bifcheck=0; p.nc.dsmax=0.05;
p.sw.spjac=1; p.fuha.spjac=@bpjac; huclean(p); p=cont(p,20);
```

```
% k=l*k_branch at pos.5 (due to param-doubling during BP cont, lam at pos 1)
figure(3); clf; hold on; p=loadp('bpc0a','pt10'); plotbradat(p,3,5,1);
p=loadp('bpc0b','pt10'); plotbradat(p,3,5,1);
p=loadp('bpc2a','pt20'); aux.ps='-r'; plotbradat(p,3,5,1,aux);
p=loadp('bpc3a','pt20'); aux.ps='-m'; plotbradat(p,3,5,1,aux);
k=linspace(0.5,1.5,40); kap=k.^2-1;
figure(3); hold on; plot(k,kap.^2,'*k'); % analytical existence line
plot(k,3*kap.^2-kap.^3,'*r'); % analytical stability line
axis([0.5 1.5 0 1]); xlabel('k'); ylabel('\lambda'); set(gca,'fontsize',16);
```

Listing 11: Two cells from `shEck/cmdsEck.m`. In line 1 of the top cell we initialize BP continuation in the scaling parameter ℓ by calling `bpcontini`, and save the original wave number $k_0=19/20$ of `b2` in p.k. This is used to put the correct $k=k_0\ell$ onto the branch. The bottom cell contains the plot commands to obtain Fig. 24(d).

```
function out=shbra1d(p,u)
M=getM(p); n=p.np; try k=p.k;catch;k=1;end; keff=k*u(p.nu+3); % eff. wave-nr
out=[u(p.nu+1:end); keff; sqrt(u(1:n)'*(M(1:n,1:n)*u(1:n)))/sqrt(p.0m);
     max(u(1:n)); min(u(1:n))];
```

Listing 12: `shEck/shbra1d.m`, to put $k=k_0\ell$ onto the output branch.

Remark 5.2. In 1D, the Eckhaus curve $\lambda{=}\lambda_{\rm Eck}(k)$ is the (lower) boundary of the Busse balloon $\lambda{>}\lambda_{\rm Eck}(k)$, which is the parameter set of stable (1D) patterns. In 2D, the Busse balloon for the SH equation is further delimited by the so called zig-zag instability curve $k>\sqrt{1-\lambda^2/512}+\mathcal{O}(\lambda^4)$, for small λ. In general, the boundary of the Busse balloon for patterns consists of various instability curves, such as Eckhaus–, zig-zag–, cross-roll– and other instabilities, and the Busse balloon is typically a bounded set, even if, as in the SH equation, the existence region of the pattern is unbounded. The (asymptotic) computation of Busse balloons is an important but complicated problem, see, e.g., [Hoy06, §8]. The above example and further tests suggest that at least in simple cases the continuation of BPs (and of Hopf BPs) may be a simple but efficient method to approximate Busse balloons. However, further details will appear elsewhere. ⌋

5.2 Cahn-Hilliard: Demo `ch`

The Cahn–Hilliard problem models 'spinodal decomposition' of an alloy, and consists in finding stationary points (in particular minimizers) of the energy

$$E_\varepsilon(u) = \int_\Omega \frac{1}{2}\varepsilon^2\|\nabla u\|^2 + W(u)\,\mathrm{d}x, \text{ under the mass constraint } \frac{1}{|\Omega|}\int_\Omega u\,\mathrm{d}x = m \tag{52}$$

and zero flux–BCs. Here $\Omega \subset \mathbb{R}^d$ is a bounded domain, $\varepsilon > 0$ is a parameter for the so–called interface energy, and W is a double well potential, e.g., $W(u) = -\frac{1}{2}u^2 + \frac{1}{4}u^4$. A detailed bifurcation *analysis* of the problem on the unit square can for instance be found in [Kie12, §III.2.6]. Here we are mainly interested in the implementation of the constraint $\frac{1}{|\Omega|}\int_\Omega u\,\mathrm{d}x = m$, and give a few numerical results for illustration.

We let $\Omega = (-1/2, 1/2)^d$ such that $|\Omega| = 1$. The first variations wrt u and λ of the Lagrangian $L(u,\lambda) = E_\varepsilon(u) + \lambda(\int_\Omega u\,\mathrm{d}x - m)$ then yield the Euler–Lagrange equations

$$\text{(a)}\quad G(u) := -\varepsilon^2\Delta u + W'(u) - \lambda \stackrel{!}{=} 0, \qquad \text{(b)}\quad q(u) := \int_\Omega u\,\mathrm{d}x - m \stackrel{!}{=} 0. \tag{53}$$

Remark 5.3. a) See, e.g., [GM88] for the connection of (52) to so-called phase transitions. In [Mod87] the following result has been proved, for $\Omega \subset \mathbb{R}^d$ a bounded domain with Lipschitz boundary:

> For $\varepsilon > 0$, let u_ε be a minimizer of E subject to $\int_\Omega u\,\mathrm{d}x = 0$. For $\varepsilon \to 0$, the sequence (u_ε) converges (in $L^1(\Omega)$) to a function u_0 which only takes values in ± 1, and with the interface $I{=}\partial\{u{=}1\} \cap \partial\{u{=}-1\}$ having minimal length (2D) resp. area (3D). (54)

Results of this type have since been extended and refined, and have been transfered to Allen–Cahn and Cahn–Hilliard problems on Riemannian manifolds (see also §6, including min–max type results for stationary points of E (saddle–points), and showing important connections between the nonlinear elliptic PDE (53) and the theory of embedded minimal hypersurfaces. See, e.g., [GHP03, dRGGHP03, Ton05, Pac12, Gua18].

b) The double well $W(u) = -\frac{1}{2}u^2 + \frac{1}{4}u^4$ is often modified to $\tilde{W}(u) = \frac{1}{4}(u^2-1)^2$. Naturally, the absolute values at the minima $u = \alpha$ and $u = \beta$ (here at $\alpha = -1, \beta = 1$) play no role, as only W' appears in Euler–Lagrange equations, $\alpha \neq \beta$ may be located anywhere in $\mathbb{R}$, and W at the minima need not have equal values. However, a useful canonical form of the energy (without loss of generality returning to $\tilde{W}$) is

$$E_\varepsilon(u) = \int_\Omega \frac{\varepsilon}{2}|\nabla u|2 + \frac{1}{\varepsilon}\tilde{W}(u)\,\mathrm{d}x, \tag{55}$$

because in this form we have

$$E_\varepsilon(u_\varepsilon) \to 2\sigma|I| \text{ as } \varepsilon \to 0 \tag{56}$$

for local minimizers and saddle points u_ε, where $\sigma = \int_{-1}^{1}\sqrt{\frac{1}{2}\tilde{W}(u)}\,\mathrm{d}u = \sqrt{2}/3$ is the so–called energy constant, cf., e.g., [Gua18, Theorem B].

c) The dynamic Cahn–Hilliard equation is obtained from taking the conserved flux

$$\partial_t u = -G(u) := \nabla \cdot [\nabla \delta_u E_\varepsilon(u)] = -\Delta[\varepsilon^2 \Delta u - W'(u)], \tag{57}$$

also called H^{-1} gradient, where δ_u denotes the variational derivative. For zero-flux BCs, i.e., $\partial_n u = \partial_n \Delta u = 0$ on $\partial\Omega$, this conserves the mass $\int_\Omega u \, dx$, but for the continuation of the steady problem $0 = G(u)$ we again need to explicitly enforce the mass conservation, see, e.g., [EGU+19]. Further, see, e.g., [Ell89] for basic results on the existence of solutions of (57), their numerical approximation, and their basic dynamical behaviour, which can roughly be characterized as follows: Starting from essentially random initial data (with mass 0), the solution rapidly evolves to a 'fine grained' structure with complex interfaces between the phases $u = \pm 1$, also called 'fat spaghettis'. After this initial phase, a slow coarsening process sets in, during which interfaces move and disappear (regions of pure phases $u \approx 1$ or $u \approx -1$ coming together), on longer and longer time scales. See also [Mir19, DF20] for comprehensive reviews of other Cahn–Hilliard type equations obtained to describe diffusive interfaces in a variety of settings and applications, and of their analytical and numerical treatment. ⌋

As parameter vector in (53) we have $(m, \varepsilon, \lambda)$. We first continue solutions in m with ε fixed and the Lagrange parameter λ as free parameter for the constraint q=0 from (53). On the FEM level, $\int_\Omega u(x) \, dx$ is conveniently approximated as `vM*u`, where `vM=sum(M,1)` and M is the FEM mass matrix. Thus, the constraint q and its derivative are easily implemented as in Listing 13, and at startup we let `p.fuha.qf=@qf; p.fuha.qfder=@qfder; p.sw.qjac=1; p.nc.nq=1; p.nc.ilam =[1,3]`. For plotting we also put E_ε from (55) on the branch, see Listing 14.

```
function q=qf(p,u) % mass constraint
m=u(p.nu+1); u=u(1:p.nu); q=p.mat.vM*u/p.Om-m;
```

```
function qu=qfder(p,u); qu=(1/p.Om)*p.mat.vM;  % pa_u q of mass constraint
```

Listing 13: `qf.m` and `qfder.m` from `ch`, computing the constraint (53b), and its derivative $\partial_u q$.

```
function [E1, E2]=chE(p,u)  % energy for CH
par=u(p.nu+1:end); eps=par(2); u=u(1:p.np);
ux=p.mat.Dx*u; uy=p.mat.Dy*u; uz=p.mat.Dz*u; % for |grad u|^2
W=0.25*(u.^2-1).^2; ux2=ux.^2+uy.^2+uz.^2; sig=sqrt(2)/3;
dens1=0.5*eps^2*ux2+W-0.25; E1=sum(p.mat.M*dens1); % physical E
dens2=0.5*eps*ux2+W/eps; E2=sum(p.mat.M*dens2)/(2*sig); % interface-length E
```

Listing 14: The energy E from (52). The 1st order differentiation matrices `Dx, Dy`, and `Dz`, and the 'integration vector' `p.mat.vM=sum(p.mat.M,1)`, are generated in `oosetfemops`.

1D. In Fig. 25(a) we start with a basic BD of E_ε over m for $\Omega = (-0.5, 0.5)$ and $\varepsilon = 0.1$. Due to this rather large ε, or, equivalently, the rather small domain, here we only have three BPs up to $m = 0$ on the black spatially homogeneous branch, with bifurcating branches with 1,2 and 3 interfaces, see the sample plots in (b). The bifurcation of the first branch is subcritical and to lower energy, and the 1–interface solutions become stable in a fold at $m = m_0 \approx -0.66$. In the last panel of (b) we plot λ over E, which illustrates that λ is small on `a1` in the "spinodal region" extending from $m = -m_0$ to $m = m_0$. Also, on `a1` in the spinodal region $E_\varepsilon(u_\varepsilon)$ approximates well the number of interfaces (namely 1); the approximation is worse for `a2` and `a3`, as in fact ε is "too large" (or the domain too small) for these solutions to approach the two minima $u = \pm 1$ of the double well potential between the interfaces.

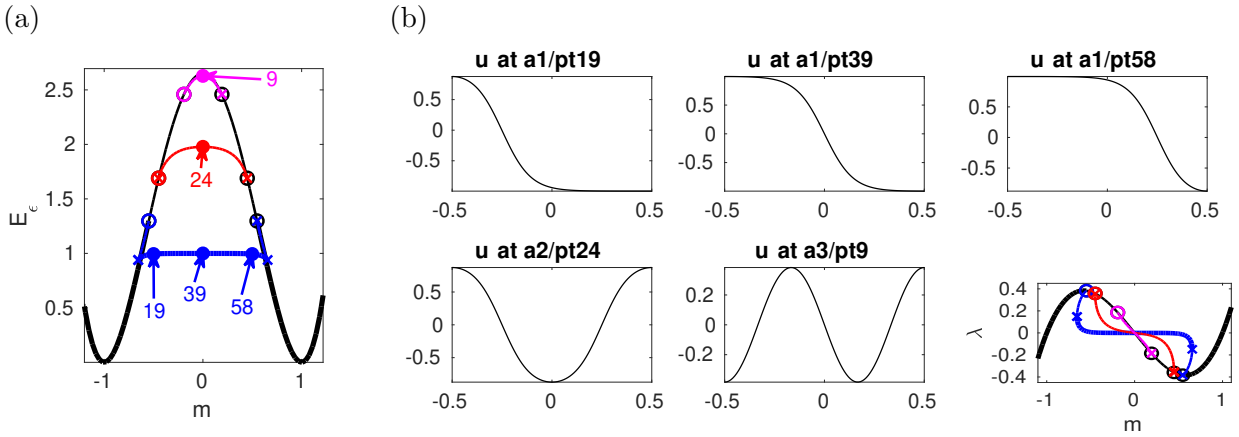


Figure 25: Resuts from `cmds1D.m` for (52) on $\Omega = (-0.5, 0.5)$, $\varepsilon = 1/10$. (a) BD of E_ε over m, homogeneous branch (black), and three bifurcating patterns with 1 (blue), 2 (red) and 3 (magenta) interfaces. (b) Sample plots; the upper row shows how the front moves right as the mass increases, and the last panel shows the Lagrange multiplier λ over m.

Figure 26 shows some results for FP continuation for the first FP on the 1–interface branch `a1` from Fig. 25. This is switched on via `p=spcontini('a1','fpt1',2,'fpc')`, which automatically uses the old primary parameter m as the new secondary parameter, such that now altogether `p.nc.ilam=[2 1 3 4]`, where the new fourth parameter is the q–component of the (extended) eigenvector (ϕ, ϕ_q). The FP position m in (a) decreases with ε, as expected. Using, e.g., `p=spcontexit('fpc','pt10','b1-a')` we return to continuation in m from the obtained FPs, with fixed $\varepsilon \approx 0.045$. Going in both directions from the new FP yields the darker blue branch in (b) (with label 20), and similarly exiting FP continuation at $\varepsilon \approx 0.02$ yields the green branch. The solution plots in (c) show how the interfaces steepen up as ε decreases. The energy E_ε in (b) does *not* converge to 1, i.e., to the number of interfaces; however, this is a numerical effect as here we compute with a fixed discretization (with `np=100` points), which fails to resolve the steep fronts at small ε. This naturally calls for mesh adaptation at the interfaces, see the end of `cmds1D`, but we discuss this in more detail in 2D and 3D in the next two subsections.

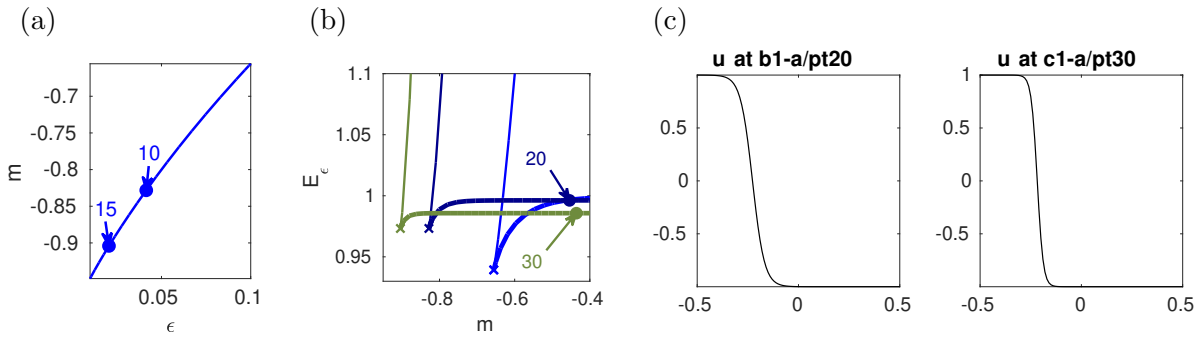


Figure 26: FP continuation, then return to normal continuation, from `ch/cmds1D.m`. (a) The primary nontrivial branch(es) for $\varepsilon = 0.1, \varepsilon \approx 0.045$ and $\varepsilon \approx 0.02$. (c) sample plots from (b).

2D. In Fig. 27, obtained in `cmds2D.m`, we return to $\varepsilon = 0.1$ and the more interesting 2D case $\Omega = (-0.5, 0.5)^2$, starting with a discretization of `np=1741` points (see Fig. 27(f) for a mesh–plot on this basic mesh). The homogeneous branch is as before, but now naturally contains more BPs. By D_4 symmetry, the first is double, with horizontal and vertical stripes in the kernel, which extend the

1D solutions in the second direction, and using `cswibra` we bifurcate to a stripe and a spot branch. Both bifurcate in subcritical pitchforks, but the spots extend to lower m and then become stable in a fold, while the primary stripes only becomes stable shortly after their fold. For both, E_ε yields a reasonable approximation of the length $|I|$ of the interface $I = \{x : u(x) = 0\}$. Up to $m \approx -0.35$, the quarter circle spot (with radius r and hence $|I| = r\pi/2$) has lower E_ε than the stripe (with $|I| = 1$). See (a), and the sample plots in the first row of (b). The remaining solutions (with sample plots in the 2nd row in (b)) are all unstable.

In (c) we zoom onto the blue spot branch near $m = 0$, where it shows two BPs to unstable "oval" interfaces, and a loop, where the solution changes from a "+"–spot in the upper left corner to a "-"–spot in the lower right, with a diagonal interface on the top point `pt84`. In (d) we do a FP continuation of the right FP from (c) in (ε, m) to decrasing ε, leading to steeper interfaces *and* the motion of the FP to larger m. In (e) we return to continuation in m from the FP `fp2c/pt20` ($\varepsilon \approx 0.0423$). The BD illustrates that also the BPs have moved to the other side of $m = 0$ (compared to Fig. 27(c)), and quite close to the FPs, such that the stability region of the spots is now larger. In (f) we show the solution `pt26` at $m = 0$ with a diagonal interface on the original mesh, and after mesh adaptation, and similar for solution `pt68` in (g). For the original mesh we have $E_\varepsilon \approx 1.32$ and after adaptation $E_\varepsilon \approx 1.25$. If we estimate the radius of the quarter circle interface $I = \{u = 0\}$ at `pt68` by 0.77, then $|I| \approx 1.21$.

3D. In Fig. 28, obtained from `cmds3D`, we essentially repeat for 3D the same steps as in 2D. Here we start with a rather coarse mesh of n_p=25^3=15625 points, and with a rather large $\varepsilon = \sqrt{1/20}$. This is partly to work with the coarse mesh, but also to keep things simple (initially) by avoiding too many BPs on the homogeneous branch. In (a) we show the homogeneous branch, and three branches from the 'lamella', 'tube' and 'spot' classes bifurcating at the first BP at $m \approx -0.445$. (b) shows three numerical kernel vectors, here obtained as two tubes (ϕ_1 and ϕ_2) and one lamella (ϕ_3). However, this depends on the discretization (and the starting vector for the inverse vector interation in `eigs`), and we might as well obtain three tubes, or three lamellas, or two lamellas and one tube as a basis for the (by cubic symmetry) three dimensional kernel. Confirm the discussion in §3.3. In particular, from the kernel vectors in (b) it is not immediately clear how to compose a spot branch.

Thus we again use `cswibra`, which finds 13 isolated solutions of the CBE, as it should, namely 3 (orientations of) lamellas, 3*2 tubes, and 4 spots. We select one branch from each isotropy class, and the first row in (c) shows samples of lamellas and tubes, while the second shows two solutions from the spot branch, all as indicated in (a). In particular, at low m (zoom in the second panel of (a)), the spots have the lowest E_ε and are stable after their fold, then lose stability to the tubes, and for $m > -0.334$ the lamellas are the only stable solutions. Given the relatively large ε and the coarse mesh, E_ε yields reasonable approximations of the interface lengths for all three solution classes, in particular $|I| = 1$ for the lamellas. In (d,e) we switch to continuation in ε and hence steeper interfaces. For the red (lamella) branch in (e) we switch on mesh adaptation each 5^{th} step during the continuation in ε. Without this, E_ε, which starts at $E_\varepsilon \approx 0.975 < 1$ at $\varepsilon = \sqrt{1/20}$, decreases for $\varepsilon < 0.12$, while it should converge to 1. This mesh adaptation produces the jumps on the red branch in (e). (f) illustrates the adapted mesh at the final lamella `pt15`, with `np=17700`.

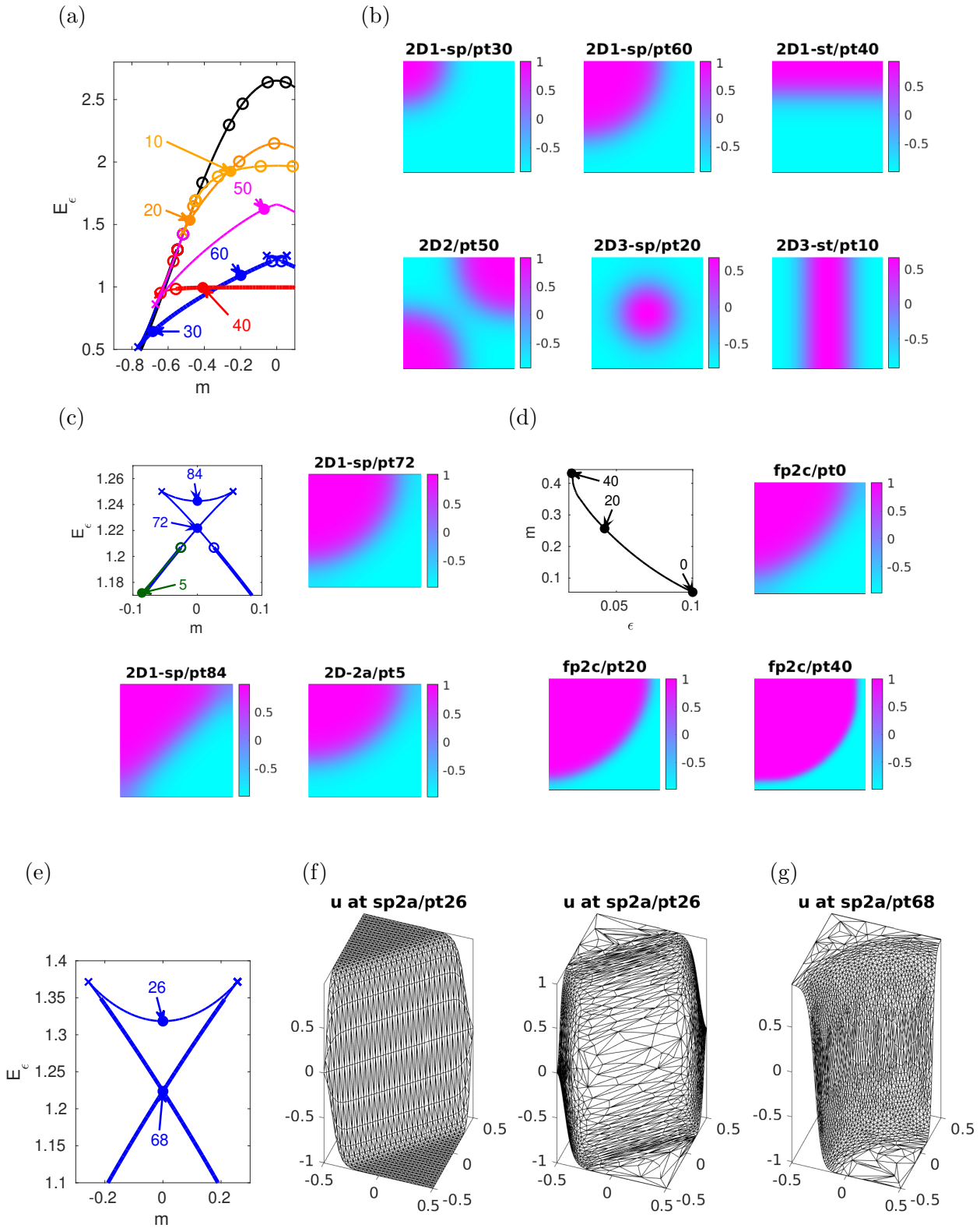


Figure 27: Results from `cmds2D.m` for (53) on the unit square, $\varepsilon = 1/10$. (a) Bifurcation diagram of E_ε over m. The first BP is double, with spots (blue) and stripes (red) bifurcating subcritically, with the spots becoming stable in their first fold, the stripes shortly after. The further BPs may be simple (BP2, magenta) or double (BP3, again with spots and stripes, orange branches). Symmetric behavior for $m > 0$. (b) Sample plots from (a). (c) Zoom into primary spot branch near $m = 0$, with secondary bifurcation 2D-2a (green). (d) Continuation of the right FP from (c). (e)–(f) return to continuation in m from the FP in (d) at $\varepsilon = 0.0423$, with mesh adaptation. (f) `pt26` on the original mesh (1741 points), and after mesh adaptation (to 1150 points). (g) The solution after the loop and mesh adaptation (to 1538 points).

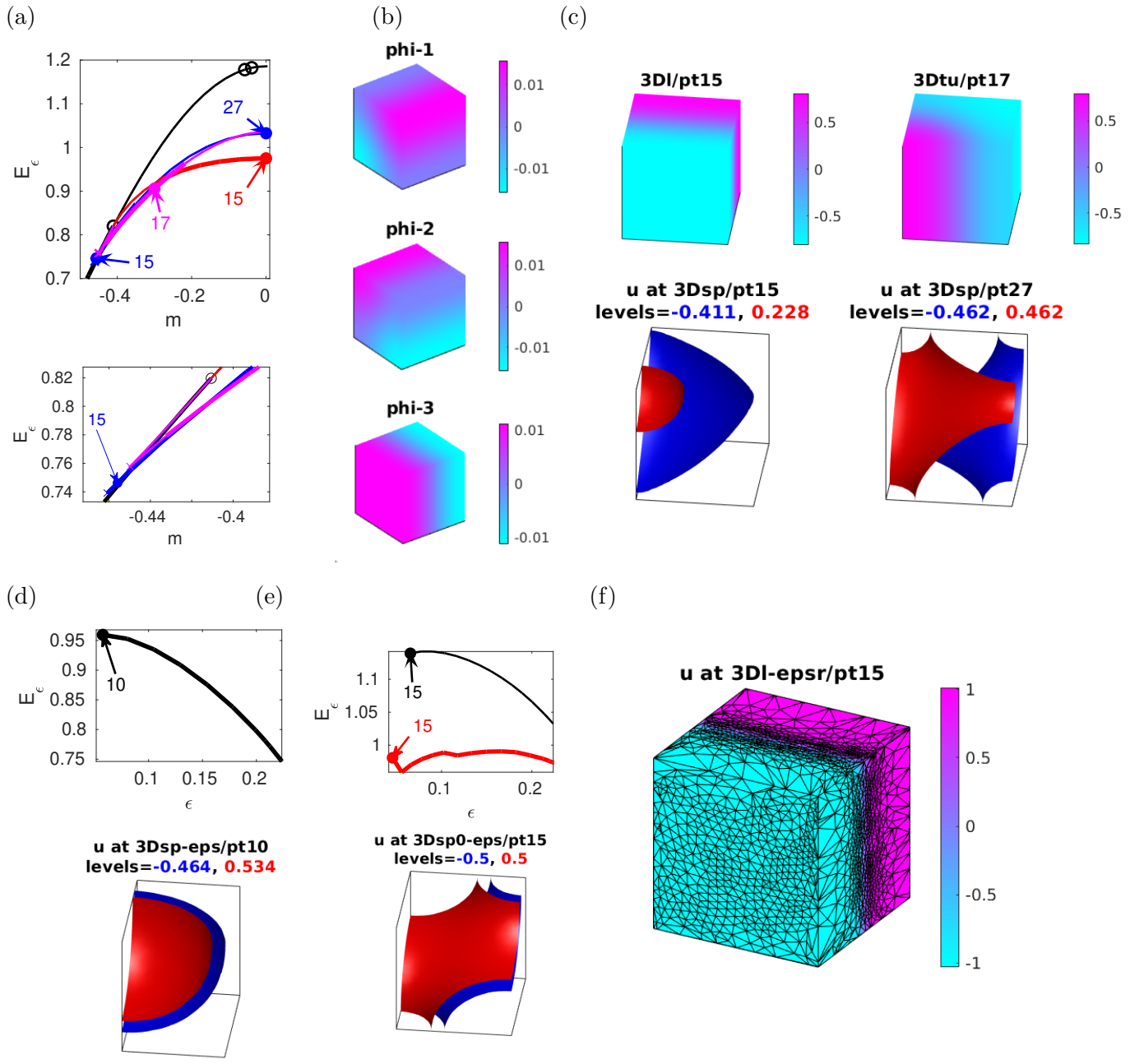


Figure 28: Results from `cmds3D` for (53) on the unit cube, n_p=25^3=15625 initially. (a) ε=$\sqrt{1/20}$, homogeneous branch (black), and first three bifurcating branches: spots (blue), lamellas (red), and tubes (red). (b) the three numerical kernel vectors at the first BP from the black branch: 2 tubes and 1 lamella. (c) sample plots from (a). (d,e) continuation of `3Dsp/pt15` (d), and `3Dsp/pt27` and `3Dl/pt15` (e), to decreasing ε, yielding steeper interfaces. (f) Adaptated mesh at `pt15` on the red lamella branch from (e).

5.3 Pearling in the functionalized Cahn–Hilliard equation: Demo `fCH`

A binary mixture of a solvent (e.g. water) and (hydrophobic) molecules (polymers or lipids) can be 'functionalized' by adding hydrophilic side chains to the molecules. Intuitely, this may favor the formation of bilayer interfaces ("channels", see Fig. 29(a) for examples in the model given below), where the hydrophilic (hydrophobic) ends of the molecules point to the solvent (to the inside of the channel). In Fig. 29, black corresponds to the low and yellow to a high concentration of molecules. If the mixture is dilute, the channels show a pearling instability, and if there is 'excess mass', i.e., a relatively high volume fraction of molecules, then channels tend to bend, which is also called meandering.

The model we consider is from [GHY11, DHPW14], to which we also refer for further background. Letting $u : \Omega \to \mathbb{R}$ be the volume fraction of solvent and molecules, with $u = -1$ corresponding to pure solvent and some $u_+ > 0$ corresponding to saturation, the free energy of the mixture reads

$$\mathcal{F}(u) = \int_\Omega \frac{1}{2}(\varepsilon^2\Delta u - W'(u))^2 - \varepsilon^\beta \left(\frac{\eta_1}{2}\varepsilon^2|\nabla u|^2 + \eta_2 W(u)\right)^2 \mathrm{d}x, \tag{58}$$

where $0 < \varepsilon \ll 1$ is related to interface thickness, and $W : \mathbb{R} \to \mathbb{R}$ is a double–well potential with typically unequal strict minima at -1 and $u_+ > 0$, and a strict maximum at $u = 0$. The first summand is the squared variational derivative of the Cahn–Hilliard energy $E_\varepsilon(u)$ (with ε^2 instead of ε), cf. (52), and can thus be seen as a bending energy, also called Willmore functional. The second summand represents the functionalization, where $\eta_1 > 0$ and $\eta_2 \in \mathbb{R}$ model the strenght of the hydrophilicity in terms of interfaces and volumes, respectively. Thus, $\eta_1 > 0$ means that interfaces lower the free energy.

For $\beta = 1$ ($\beta = 2$) we have so called strong (weak) functionalization, and as in [DHPW14] we focus on the strong case. The evolution of the system is assumed to be given by a gradient flow $\partial_t u = -\mathcal{G}\delta_u\mathcal{F}(u)$, where the choice of the gradient operator $\mathcal{G}$ together with the BCs must ensure mass conservation, i.e., $\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega u(t,x)\,\mathrm{d}x = 0$. The simplest choice is the projection

$$\mathcal{G}f = \Pi f := f - \frac{1}{|\Omega|}\int_\Omega f(x)\,\mathrm{d}x, \tag{59}$$

leading to the evolution equation

$$\partial_t u = -\mathcal{G}[(\varepsilon^2\Delta - W''(u) + \varepsilon\eta_1)(\varepsilon^2\Delta u - W'(u)) + \varepsilon\eta_d W'(u)], \tag{60}$$

where $\eta_d = \eta_2 - \eta_1$. In suitable parameter regimes, (60) is very rich in pattern formation. The basic building blocks (in 2D) are straight and curved bilayer interfaces between $u \equiv -1$ and u near u_+, which show "pearling" and "meander" instabilities as illustrated in Fig. 29(a).

Table 6: Selected scripts and functions in `pftut/fCH`. Other files, fchinits, oosetfemops, sG (with nodalf), and sGjac, more or less as usual.

script/function	purpose,remarks
cmds1	script to continue a channel, and meandering and pearling bifurcations from it, Fig. 29
cmds2	script for continuation in η_1 and DNS, Fig. 30
e2rs_ad_hoc	trivial elements-to-refine-selector based on $\|u+2\|$
qf2, qf2jac	the phase conditions (61c) and (62), and the derivatives
nodalft	modification of `nodalf` (the 'nonlinear terms' in (61)), setting $\gamma = 0$ and including (59) for mass conservation.

A pde2path–setup for (60) and some results were already presented in [DRUW14], but without stability considerations. Moreover, here we present a simpler setup, and want to explain further tricks to compute patterns in (60) more robustly. See Table 6 for an overview of the used files. Setting $v = \varepsilon^2\Delta u - W'(u)$, the problem can be written as the two component system

$$\partial_t u = -\varepsilon^2\Delta v + W''(u)v - \varepsilon\eta_1 v - \varepsilon\eta_d W'(u) + \varepsilon\gamma, \tag{61a}$$

$$0 = -\varepsilon^2\Delta u + W'(u) + v, \tag{61b}$$

where γ is the (scaled) Lagrange multiplier for mass conservation in (60), over domains Ω with Neumann BCs for u and v. We take γ as an additional unknown, and add the equation

$$q(u) := \frac{1}{|\Omega|}\int_\Omega u\,\mathrm{d}x - m = 0, \tag{61c}$$

where m is a reference mass, also taken as a parameter, which here we shall use as the main continuation parameter as in the CH equation in §5.2. Moreover, as we shall be interested in rather dilute mixtures m near -1, it turns out that often phase conditions are needed or at least very useful for robust continuation of patterns. For instance, for the channels in Fig. 29 we use the phase condition

$$\langle \sin(x), u\rangle = \int_\Omega \sin(x)u(x,t)\,\mathrm{d}x = 0, \tag{62}$$

fixing the channel in the middle of the domain, and add $s\partial_x u$ to the rhs of (61a). Thus, we now have four parameters $(\eta_1, \eta_2, \varepsilon, m)$, two additional unknowns (γ, s), and two additional equations (61c) and (62), i.e., $n_q = 2$.

For W we follow [DHPW14, §5] and let

$$W(u) = W_p(u+1) + 20(u - m_p + 1)^{p+1}H(u - m_p + 1), \text{ where } W_p(u) = \frac{1}{p-2}(pu^2 - 2u^p)$$

with $p = 3$, $m_p = (p/2)^{1/(p-2)} = 3/2$, and H being the Heaviside function. In [DHPW14, §5], additional to a strong analysis which establishes the existence of straight and curved channels and gives certain instability criteria for them, numerical time integrations are presented with $\varepsilon = 0.1$, $\eta_2 = 2$, and η_1 between 1 and 2. We use similar parameter regimes, but a somewhat larger $\varepsilon = 0.25$ to avoid very steep interfaces.

The starting straight channel in Fig. 29 is obtained from the guess

$$u_{ig}(x) = -1 + a/\cosh(bx), \quad v_{ig} = -W'(u_{ig}) \tag{63}$$

with $a = 1.25$, $b = 20/3$, which yields the initial mass $m = m_0 \approx -0.853$, followed by a Newton loop for (61)–(62). It turns out that due to the large variety of patterns possible for (61), a careful choice and refinement of meshes is important, in particular mesh symmetry. Thus, here we use a *very* ad–hoc refinement strategy, i.e., we replace the error estimator `e2rs` by `e2rs_ad_hoc`, which simply selects the triangles according $|u + 2|$. For the initial guess (63), a 1–step mesh refinement from an initial criss–cross mesh of about 3500 points then yields the first solution u_0 in Fig. 29(a), on a mesh of about 9000 points, refined around the $x = 0$ line, and we keep this mesh for all the subsequent

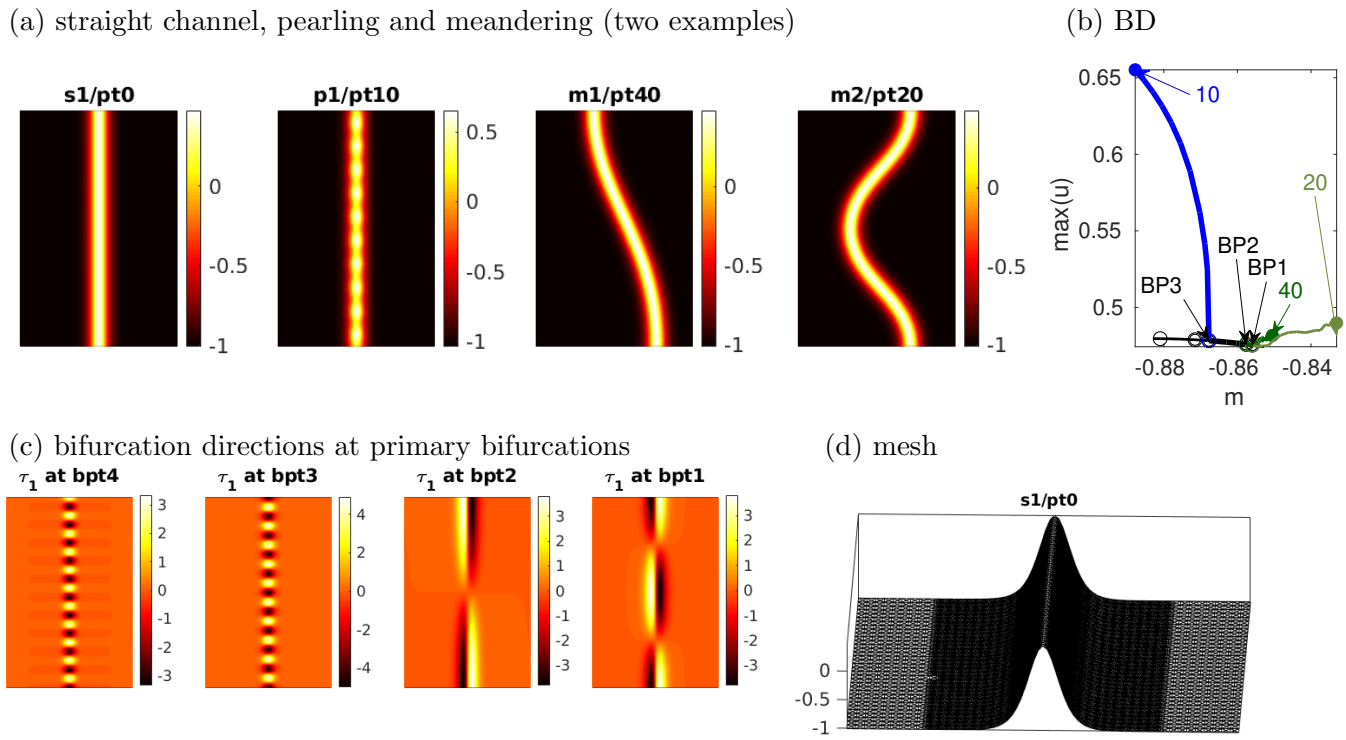


Figure 29: Example results from `cmds1`, $\Omega = (-2,2)\times(-3,3)$, $\varepsilon = 0.25, \eta_1 = 1, \eta_2 = 2$, continuation in m starting with initial in the form of straight chanel with $\approx= -0.853$, which is unstable. Continuation to smaller m first yields meandering BPs (green branches), then a stable segment for the straight channel, then pearling (blue branch `p1`), as also illustrated in the bifurcation directions in (c). (d) is meant to illustrate the straight channel shape and the used mesh.

computations.[4]

The straight channel solution u_0 at $m = m_0$ is unstable with 2 unstable eigenvalues of $\partial_u G$ for (61) (which do not take into account the constraints (61c) and (62)), and for DNS of (60). These DNSs are again based on the formulation as the DAE system (61), with both (61c) and the phase condition (62) switched off, and with $\gamma = 0$. We implement this by a small modification of the library function `tintxs` and of the nonlinearity function `nodalf` used for (61) to account for Π from (59). Continuing u_0 to smaller m, we first find two BPs BP1 and BP2 at $m \approx -0.856$ and $m = -0.858$, respectively, after which the straight channel is stable until BP3 at $m \approx -0.868$, followed by BP4 at $m \approx -0.872$ and further BPs at lower m. BP1 and BP2 are associated to meandering, while BP3 and BP4 yield pearling, see (c) for the bifurcation directions, which illustrate the dichotomy between pearling at low m (dilute mixture) and bending at 'high' m (excess mass of the molecules). The primary bifurcating meandering dark green branch `m1` is stable after a fold. The primary blue pearling branch `p1` bifurcates supercritically and is stable until $m \approx -0.9$ (secondary bifurcation not shown). Similar pearling branches bifurcate from the further BPs on the straight channel branch at lower m. Note that all these (pearling and meandering) bifurcations respect the (physical) constraint (61c), and the numerical constraint (62), which is needed for the continuation of the steady branches: Omitting it, we get an eigenvalue very close to 0 from shifting a solution in x, and relatedly we observe

[4] At the end of `cmds2.m` we use `trullekrul` to generate meshes which are clearly much better adapted to specific chosen solutions. However, using these meshes for continuation, with or without further adaptation during continuation, makes the continuation less robust: Often, genuine bifurcations on the symmetric mesh (which are also expected from theory) become imperfect bifurcations on the (anisotropic) `trullekrul` meshes.

that some solutions drift during the continuation, even on the rather narrow domain. The pearling instabilities at low m only depend weakly on the transverse length l_y, as they are of relatively high wave number in y, but of course the bending may set in earlier on longer domains.

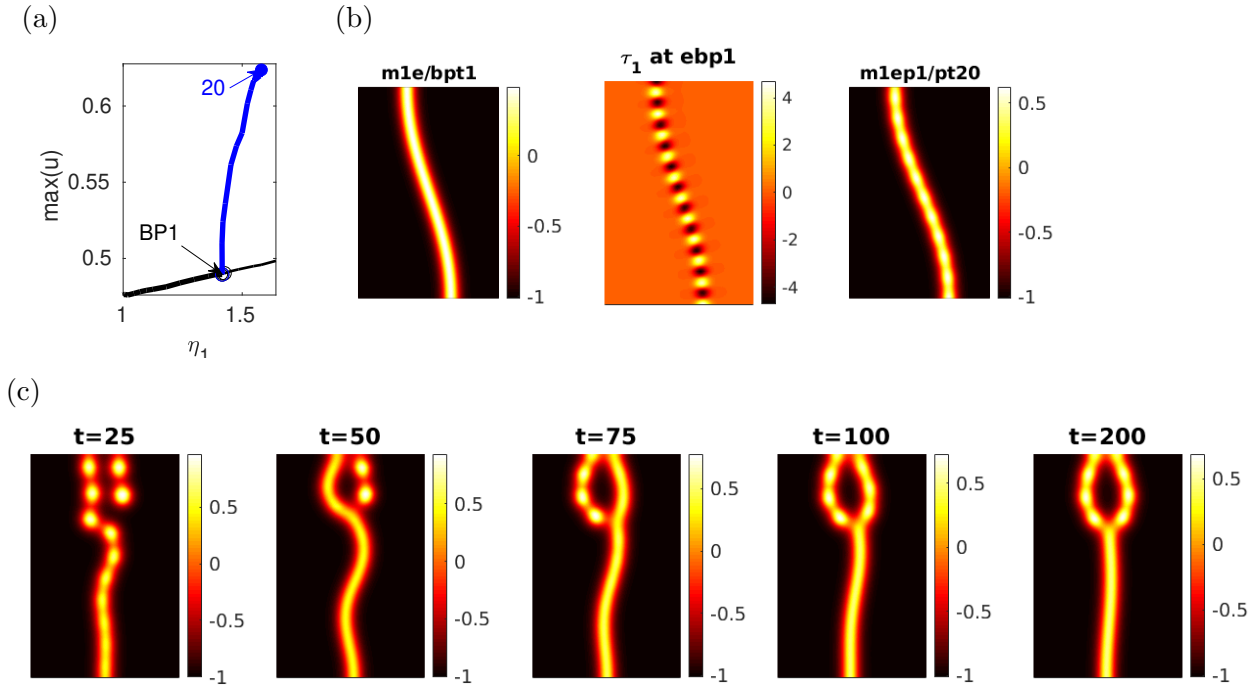


Figure 30: (a) Continuation of `m1/pt10` in η_1, giving pearling, with sample solutions and tangent in (b). Snapshots from the evolution starting from near `s1` at $m = -0.8$. At $t = 200$ the solution is quasi steady, and a Newton loop for the steady problem yields a (stable) steady solution with no visible change. The pearled loop at the top is characteristic for such DNS, i.e., occurs (located somewhere along a channel) in the vast majority of simulations.

In [DHPW14, Fig.5.1, Fig.5.2], an example is given where increasing η_1 yields pearling of a curved (circular) channel. To see this, in `cmds2` and Fig. 30(a,b) we continue the meandering solution `m1/pt10` in η_1. In Fig. 30(c) we show snapshots from a DNS starting near the unstable straight channel at $m = -0.8$ (with zero mass perturbation) which illustrates the typical behaviour at 'high' m ($m > -0.82$, say; of course, the m here also depends on the domain size in x, i.e., the same channels on domains twice as large in x would yield m values much closer to -1). Initially, some (non–small wave number) pearling occurs, but most of the spots recombine to a curved channel, with a pearled loop at the top. At $t = 200$ the solution is already quite close to a steady state, i.e., the residual is below 10^{-8}, and further DNS does not yield visible changes. Thus, we use this solution as initial guess for a Newton loop and obtain convergence to a steady state without visible change. Conversely, similar experiments at low m (close to -1) typically yield convergence to straight pearled channels, i.e., no meandering at all.

Remark 5.4. The results in Fig. 29 and Fig. 30 only scratch the surface of a numerical exploration of the fCH equation (60), for which continuation is difficult due to the different length scales involved, and the many solutions supported. We further refer to [CKP19] for remarkable DNS in 2D and 3D, which combined with analysis yield "qualitative bifurcation diagrams", which partition the parameter space into regions where pearling or meandering dominates. See also [PW17] for modeling of biological systems (consisting of different lipids) by multi–component fCH systems. Nevertheless, we believe that

numerical continuation and bifurcation as presented here may help to sort out the different parameter regimes more quantitatively. ⌋

5.4 Higher indeterminacy: Demo `hexex`

As already said, while `qswibra` and `cswibra` work robustly for most of the example problems we considered, they are not fail safe: the underlying QBE (25) and CBE (27) are only solved numerically via `fsolve`, and whether a solution is found may depend on the initial guesses for the Newton loops, and thus may require some trial and error. Similarly, whether a solution α is correctly or incorrectly identified as isolated or non-isolated may depend on the tolerance for the Jacobian determinant. Therefore we provide the auxiliary arguments in `aux` from Table 3, and the fallback routine `gentau` from Algorithm 3.1.

As an example, in the demo `hexex` we consider a problem from [Mei00, §6.8.2], namely

$$G(u,\lambda) := \Delta u + \lambda(u + u^3) = 0 \tag{64}$$

on a hexagon with unit side-lengths and Dirichlet BCs. On the trivial branch $u \equiv 0$, there is a simple bifurcation point at $\lambda = \lambda_1 \approx 7.14$, a double bifurcation point at $\lambda = \lambda_2 \approx 18$, and further bifurcations at $\lambda = \lambda_3 \approx 32.5$ (double), $\lambda = \lambda_4 \approx 37.6$ (simple), ... See Fig. 31(a) for the kernel vectors at λ_2. At the simple BPs we can use `swibra`. However, the problem is $D_6 \times Z_2$ equivariant, and thus we expect pitchfork bifurcations at the multiple bifurcation points which are at best 5-determined, cf. [Uec19b, Remark 3.1], and thus the bifurcation directions cannot be computed with `cswibra`, which correctly reports that only non–isolated solutions α are found (with the default setting of `isotol` $= 10^{-10}$).

Therefore we try `gentau`, for instance with the natural choice $\gamma = (1,0)$ and $\gamma = (0,1)$. This turns out to immediately yield two bifurcating branches, i.e., the tangents to these branches coincide with the numerical kernel vectors. Moreover, for mixed choices of γ, i.e., $\gamma = (\gamma_1, \gamma_2)$ with $\gamma_1\gamma_2 \neq 0$, if the first Newton loop converges, then the convergence is to one (isotropy class) of these two branches. In fact, this convergence occurs for a large majority of γ values, and only selected large vectors γ give non-convergence. In summary we conclude that exactly the two (classes) of distinct branches bifurcate, which fully agrees with the high-order determinacy analysis in [Mei00, §6.8.2]. Thus, `gentau`, possibly with some trial and error, can be an efficient method to find all pertinent bifurcating branches of determinacy $k \geq 4$.

The implementation of `pftut/hexex` is fairly standard, and thus we refrain from detailed comments. The only non-obvious issue is how to generate the hexagonal domain in the OOPDE setting. For this we use the class definition `hexpdeo`, for which we modify `stanpdeo2D` and use the method `grid.freeGeometry([x;y])`, see Listing 15. Alternatively we could just pass `x,y` from line 5 to `freegeompdeo.m`.

```
classdef hexpdeo< pde  % hexpdeo (classdef, modification of stanpdeo2D)
methods(Access = public)
  function o=hexpdeo(hmax) % constructor
     o.grid=grid2D; s3=sqrt(3);
     x=[1 0.5 -0.5 -1 -0.5 0.5]; y=[0 s3/2 s3/2 0 -s3/2 -s3/2];
     o.grid.freeGeometry([x;y])
     h=1; while h > hmax; o.grid.refineMesh; h = h/2; end
     o.fem=lagrange12D;
  end
```

```
10 end
```

Listing 15: `hexex/hexpdeo.m`. Using `grid.freeGeometry` to generate a hexagonal domain and coarse mesh, then do some uniform refinement.

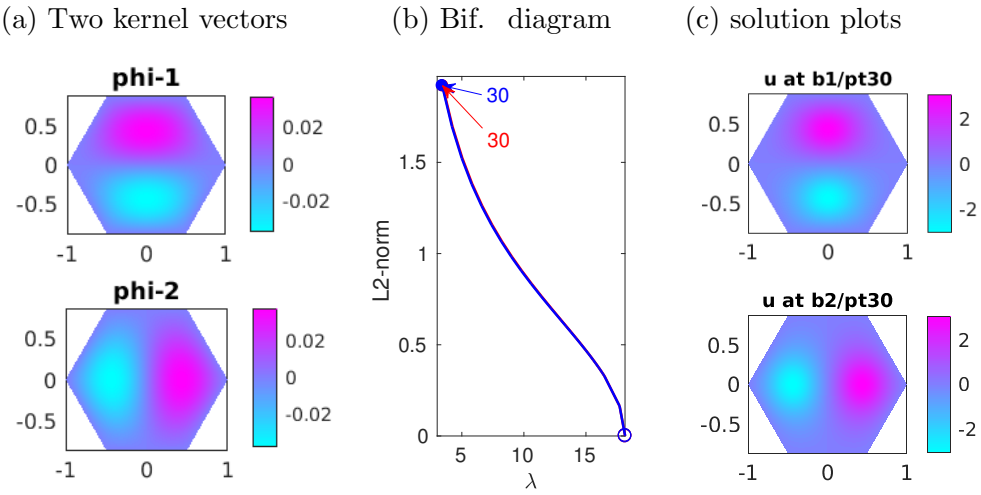


Figure 31: Results from the demo `hexex` for bifurcations at the second BP for (64).

5.5 A quasilinear system: Demo `chemtax`

In [UWR14, §4.1] we considered a reaction diffusion system with cross–diffusion as a quasilinear model problem, with the implementation based on the `Matlab pdetoolbox`. The (stationary) problem reads

$$0 = G(u) := -\begin{pmatrix} d\Delta u_1 - \lambda\nabla\cdot(u_1\nabla u_2) \\ \Delta u_2 \end{pmatrix} - \begin{pmatrix} ru_1(1-u_1) \\ \frac{u_1}{1+u_1} - u_2 \end{pmatrix}, \tag{65}$$

with homogeneous Neumann BCs for u_1, u_2. The trivial branch for (65) is $(u_1, u_2) = (1, 1/2)$. We take the chemotaxis coefficient $\lambda \in \mathbb{R}$ as bifurcation parameter, fix $d = 1/4$ and $r = 1.52$, and now implement (65) in the `OOPDE` setting, and, moreover, explain a general method for the efficient setup of Jacobians of nonlinear diffusion via `numjac`.

We split (65) as

$$G(u) = -D\Delta u - f(u) + \begin{pmatrix} \lambda C(u) \\ 0 \end{pmatrix}, \tag{66}$$

where $D = \mathrm{diag}(d, 1)$, and hence the first two terms are standard semilinear, as in, e.g., the Schnakenberg model (46). However,

$$C(u) = \nabla\cdot(c(u_1)\nabla u_2), \quad c(u_1) = u_1, \tag{67}$$

is a quasilinear cross–diffusion term, where we introduced $c(u)$ for generality. On the FEM level, (66) becomes

$$G(u) = \begin{pmatrix} dKu_1 - \lambda K_{12}(u_1)u_2 \\ Ku_2 \end{pmatrix} - Mf(u), \tag{68}$$

where K is the standard 1-component Neumann-Laplacian, M is mass matrix, and $K_{12}(u_1)u_2$ implements $\nabla\cdot(c(u_1)\nabla u_2)$. Thus, K_{12} has to be assembled in each call to `sG`, see Listing 16 below.

The Jacobian reads

$$G_u(u)\begin{pmatrix}v_1\\v_2\end{pmatrix}=\left[-\begin{pmatrix}d\Delta & -\lambda\nabla\cdot(u_1\nabla\cdot)\\ 0 & \Delta\end{pmatrix}+\begin{pmatrix}r(2u_1-1) & 0\\ -(1+u_1)^{-2} & 1\end{pmatrix}\right]\begin{pmatrix}v_1\\v_2\end{pmatrix}+\lambda\begin{pmatrix}\partial_{u_1}\nabla\cdot(v_1\nabla u_2)\\ 0\end{pmatrix},\quad(69)$$

and the last term is problematic from the FEM point of view as it contains Δu_1 and the FEM is based on the weak formulation. In [RU19, §5] (demo `acsuite/acql`), we treat the related problem of a quasilinear Allen–Cahn equation, where we approximate a term

$$\nabla\cdot((c_u(u)\nabla u)v)=\partial_x((c_u(u)u_x)v)+\partial_y((c_u(u)u_y)v)$$

as $K_x(c_u(u)u_xv)+K_y(c_u(u)u_yv)$. Thus, in `acql` we use the first order differentiation FEM matrices K_x, K_y, generated via `p.mat.Kx=fem.convection(grid,[1;0])` and `p.mat.Ky=fem.convection(grid,[0;1])`, respectively, and approximations of $\partial_x u$ and $\partial_y u$ via differentiation matrices `Dx` and `Dy`. This gives a rather large relative error between the Jacobians thus obtained and the numerical Jacobians, but the approximation is fast, and the continuation (the Newton loops) with these rough approximations of Jacobians works.

On the other hand, this trick (cf. Listing 16) does not work in general for (65), i.e., the continuation fails for some of the branches. Thus, we use `numjac` to obtain the problematic term $\partial_{u_1}\nabla\cdot(c(u_1)\nabla u_2)$ in an extra function `getKuvd`. For this we put the (here very simple) function $c(u_1)=u_1$ into the function handle `p.fuha.cfu=@cfu`. The resulting "hybrid" `sGjac` gives a relative error of order 10^{-8} to the full `numjac` (called for `p.sw.jac=0`), but is about one magnitude faster, and continuation (and BP detection/localization and subsequent branch switching) work without problems. Clearly, this setup can be generalized to other quasilinear problems, and in this sense the library functions `Kuv.m` and `getKuvd.m` (which also work for the quasilinear Allen-Cahn model, see `acsuite/acql`) should be seen as templates for adaption to a given problem.

```
function r=sG(p,u) % pde for chemotaxis model
% u_t=0.25*Lap u-lam*div(u*grad v)+r*u(1-u),   v_t=Lap v+(u/(1+u)-v)
lam=u(p.nu+1:end);d=0.25;r=1.52;u=u(1:p.nu);n=p.np;u1=u(1:n); u2=u(n+1:2*n);
f1=r*u1.*(1-u1); f2=u1./(1+u1)-u2; f=[f1;f2]; % semilin.nonlinearity
ut=p.mat.p2c*u1; gr=p.pdeo.grid; fem=p.pdeo.fem;
cc=p.fuha.cfu(ut,lam); % quasilin. coefficient, here simply c(u)=u
[K12,~,~]=fem.assema(gr,cc,0,0); % assemble matrix for cross-diff
K=p.mat.K; r=[d*K -lam*K12; 0*K K]*u-p.mat.M*f; % putting rhs together
```

```
function Gu=sGjac(p,u) % chemotaxis model
% u_t=0.25*Lap u-lam*div(u*grad v)+r*u(1-u), v_t=Lap v+(u/(1+u)-v)
d=0.25; r=1.52; lam=p.u(p.nu+1); u=u(1:p.nu); n=p.np; u1=u(1:n); u2=u(n+1:2*n);
f1u=r*(1-2*u1); f1v=0*f1u; f2u=1./((1+u1).^2); f2v=-ones(p.np,1);
Fu=[[spdiags(f1u,0,n,n),spdiags(f1v,0,n,n)]; % Jac of semilin. nonlin.
    [spdiags(f2u,0,n,n),spdiags(f2v,0,n,n)]];
ut=p.mat.p2c*u1; gr=p.pdeo.grid; fem=p.pdeo.fem; cc=p.fuha.cfu(ut,lam);
K=p.mat.K; [K12,~,~]=fem.assema(gr,cc,0,0);  % cross-diffusion
if 0 % approximate way, does not work for all branches here
  vx=p.mat.Dx*u2; vy=p.mat.Dy*u2; % 1st derivatives as coefficients
  KK=p.mat.Kx*spdiags(vx,0,n,n)+p.mat.Ky*spdiags(vy,0,n,n);
  Gu=[d*K+lam*KK -lam*K12;  0*K K]-p.mat.M*Fu;
else % using getKuvd to obtain \pa_u \div(c(u)\nabla v) via numjac
  Kuvd=getKuvd(p,lam,u1,u2);
```

```
  Gu=[d*K-lam*Kuvd -lam*K12;  0*K K]-p.mat.M*Fu;
end
```

Listing 16: `chemtax/sG.m` and `sGjac`, using `getKuvd` to numerically get $\partial_u \nabla \cdot (c(u)\nabla v)$.

Finally, in the script file `cmds2D.m` we also use a dirty little trick wrt to the localization of BPs and branch-switching: for BP detection and localization we use the `p.sw.bifcheck=2` setting with rather large `ds`. However, some of the BPs on the trivial branch are rather close together, and thus in some steps multiple (distinct) eigenvalues cross the imaginary axis. In this case, after localization we use `cswibra` with `aux.besw=0` to only compute (approximate) kernel vector. Subsequently calling `gentau` with the different respective kernel vectors succeeds, even though the branches in general do not "start" at the correct λ values. See Fig. 32 for the basic BD.

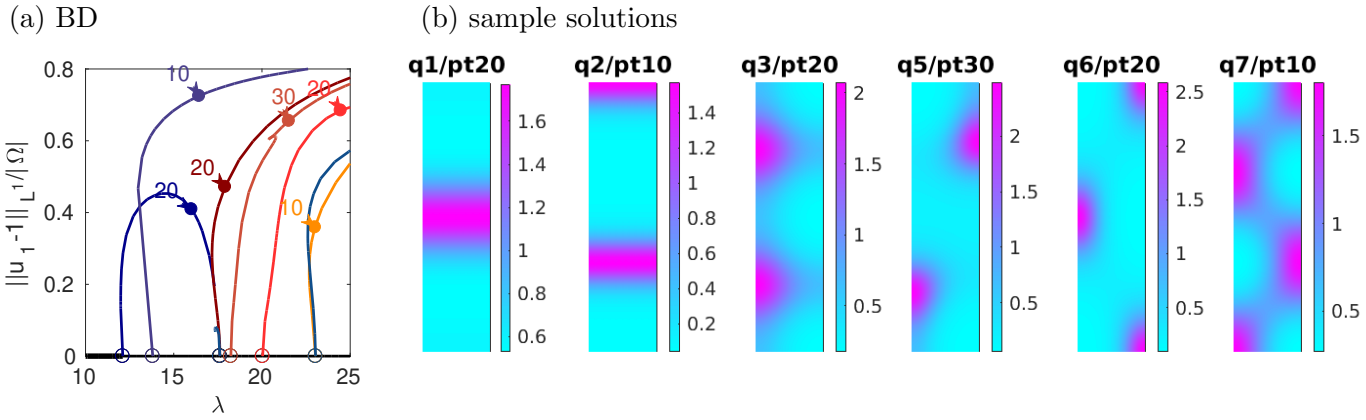


Figure 32: Basic BD and example solutions for the chemotaxis model (65) over $\Omega = (-0.5, 0.5) \times (-2, 2)$. Stripe branches `q*` in shades of blue, and spots `q*` in shades of red, with * increasing left to right.

5.6 Global coupling, and customized linear system solvers: Demo `shgc`

Interesting phenomena in pattern formation for RD systems (or SH type of equations) can occur under additional nonlocal or global coupling [FCS07, MD14, KT18, Sie18]. Here we explain a setup such that equations of type $M\partial_t u = -G(u)$, respectively the steady version $G(u, \lambda) = 0$, $u : \Omega \to \mathbb{R}^N$, can augmented by *global* coupling in the fairly general form

$$0 = G(u) + f_{\rm nl}(u, a), \quad a = \langle h(u) \rangle \,, \tag{70}$$

where $f_{\rm nl} : \mathbb{R}^{N+1} \to \mathbb{R}^N$ and $h : \mathbb{R}^N \to \mathbb{R}$ are general functions, and $\langle v \rangle = \frac{1}{|\Omega|} \int v(x)\,\mathrm{d}x$ denotes a global average. See Remark 5.5 for comments on *non–local* (but also non–global) coupling. Naturally, $f_{\rm nl}$ and h may also depend on parameters, and on x, such the averaging in $\langle h \rangle$ can be weighted.

A naive implementation of (70) yields full Jacobians, i.e.,

$$\frac{\mathrm{d}}{\mathrm{d}u} f_{\rm nl}(u, a)v = \partial_u f_{\rm nl}(u, a)v + \partial_a f_{\rm nl}(u, a) \, \langle h_u(u)v \rangle \,. \tag{71}$$

In the FEM discretization, the first term is sparse, and the second is a full matrix, but of rank 1. Thus, the purpose of this section is to explain a setup where this rank-1-correction can be treated efficiently by using Sherman–Morrison–Woodbury (SMW) formulas [PTVF07, §2.7.3]. This extends [UWR14, §4.3], where the idea was already used for a simple scalar problem with a simple linear

global coupling[5]. Moreover, we also provide customized interfaces to the `Matlab` function `eigs` for computing eigenvalues, such that also spectral computations and hence bifurcations can be treated without ever forming the full Jacobian.

A prototype problem is the globally coupled (quadratic–cubic) SH equation

$$\partial_t u = -(1+\Delta)^2 u + \lambda u + \nu u^2 - u^3 - \gamma \|u\|^2 u, \tag{72}$$

with parameters $\nu, \gamma \in \mathbb{R}$, $\|u\|^2 := \frac{1}{|\Omega|}\int u^2(x)\,\mathrm{d}x$, and (again) Neumann BCs $\partial_n u = \partial_n \Delta u = 0$, extending (3), and for instance considered in [FCS07]. For (72) with $\gamma > 0$, the nonlocal term simply acts as a reduction of the instability parameter λ: steady solutions $u(x;\lambda,\gamma)$ of (72) correspond to steady solutions $u(x;\lambda - \gamma\|u\|^2, 0)$ of (72) with $\gamma = 0$. In particular, for instance the branches of periodic and localized solutions from Figs. 5 and 8 get slanted to the right for $\gamma > 0$. Thus we obtain a slanted snaking, and in particular the snakes can move out of the bistable range of 0 and the periodic patterns, which in [FCS07] is proposed as a mechanism for the prevalence of localized states in certain systems. See Fig. 33 for some exemplary results obtained from numerical continuation of (72).

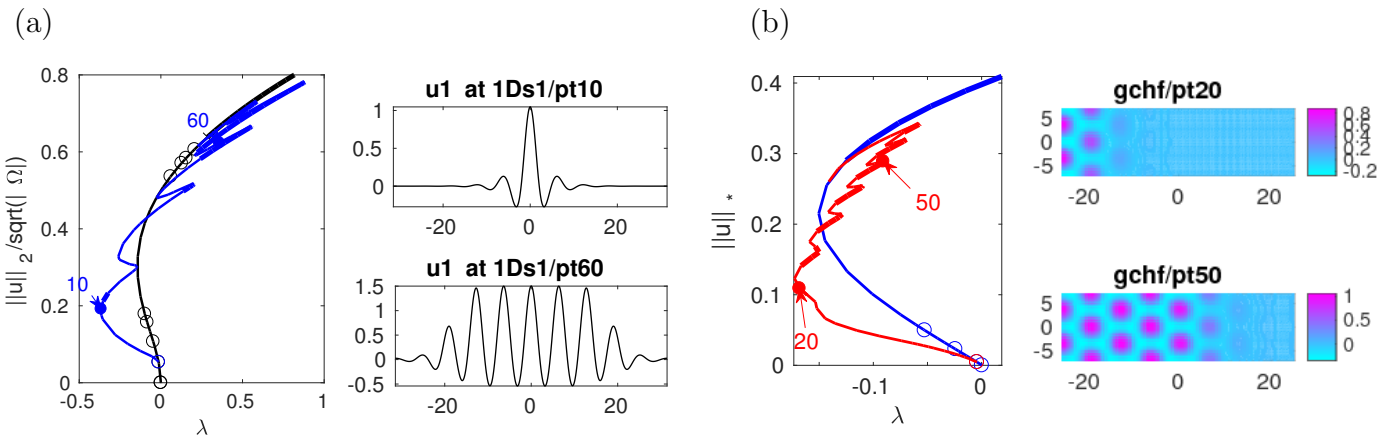


Figure 33: Slanted snaking for the globally coupled 2-3 SH equation (72). (a) $\Omega = (-10\pi, 10\pi)$, $\nu = 2, \gamma = 2$, compare to Fig. 5. (b) $\Omega = (-l_x, l_x) \times (-l_y, l_y)$, $l_x = 8\pi, l_y = 4\pi/\sqrt{3}$, $\nu = 1.3$, $\gamma = 1$, compare to Fig. 8.

Here we are mainly interested in the efficient implementation of (70) in `pde2path`, and use (72) only as an example, essentially ignoring the scaling relation to (3). Again setting $(u_1, u_2) = (u, \Delta u)$, (72) becomes,

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \partial_t \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} -\Delta u_2 - 2u_2 - (1-\lambda)u_1 + f(u_1) - \gamma\|u_1\|^2 u_1 \\ -\Delta u_1 + u_2 \end{pmatrix}, \tag{73}$$

$f(u) = \nu u^2 - u^3$, which is of the form (70) with $f_{\rm nl}(u,a) = \begin{pmatrix} -\gamma a u_1 \\ 0 \end{pmatrix}$ and $h(u) = u_1^2$. On the FEM level we obtain

$$\mathcal{M}\dot{u} = -(\mathcal{K}u - F(u) - \mathcal{M} f_{\rm nl}(u,a)), \tag{74}$$

$\mathcal{M}, \mathcal{K}, F$ as in (21), where $f_{\rm nl}(u,a)$ now means the vector $f_{\rm nl}(u,a) = -\gamma a(u_{1,1}, \ldots, u_{1,n_p}, 0, \ldots, 0)^T$,

[5]this has also been generalized in the demo `acsuite/acgc`

and where $a = \langle h(u)\rangle$ is evaluated as

$$\langle h(u)\rangle = a_{\rm av} h(u), \quad a_{\rm av} = \frac{1}{|\Omega|}\mathtt{sum}(\mathcal{M}), \quad h(u) = (h(u_{1,1}), \ldots, h(u_{1,n_p}), 0, \ldots, 0). \tag{75}$$

Since here $\mathcal{M} = \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix}$, the last n_p entries of $a_{\rm av}$ are zero anyway, but in a general setup we would rather recommend $a_{\rm av} = \frac{1}{|\Omega|}(\mathtt{sum}(M), \ldots, \mathtt{sum}(M))$ (N copies of the column sums of the 1-component mass matrix M), which of course could be used here as well. Similarly, the nonlocal part $A_{\rm nl} = \partial_a f_{\rm nl}(u,a)\,\langle h_u(u)\cdot\rangle$ of the Jacobian from (71) becomes

$$A_{\rm nl} = (\mathcal{M}\partial_a f_{\rm nl}(u,a))a_{\rm jac} \in \mathbb{R}^{n_u\times n_u}, \text{ where } \partial_a f_{\rm nl} \in \mathbb{R}^{n_u\times 1}, \quad a_{\rm jac} = \mathtt{sum}(\mathcal{M}\mathrm{diag}(h_u)) \in \mathbb{R}^{1\times n_u},$$

which again illustrates that $A_{\rm nl}$ is a full matrix but has but rank 1. The $\mathcal{M}$ in $(\mathcal{M}\partial_a f_{\rm nl})$ comes from the $\mathcal{M}$ multiplying $f_{\rm nl}$ in (74), while the $\mathcal{M}$ in $a_{\rm jac}$ is as in (75).

The SMW formula for solving linear systems $(K - ca^T)z = b$ reads

$$z = K^{-1}b + \alpha(K^{-1}c)(a^T K^{-1})b, \quad \alpha = \frac{1}{1 - a^T K^{-1}c}. \tag{76}$$

Thus, the idea is as follows: in `sGjac` we separately assemble the sparse part $G_u(u) + \partial_u f_{\rm nl}$ of G_u corresponding to K in (76), and the vectors $c = \partial_a f_{\rm nl}$ and $a_{\rm jac}$. We then pass these to SMW linear system solvers, also in the inverse vector iteration underlying `eigs` for spectral computations. Since the default interfaces for the linear system solvers do not account for the vectors c and $a = a_{\rm jac}$, these are passed via the global `pde2path` struct `p2pglob`. Table 7 lists the pertinent files from demo `gcsh`, and some functions from `libs` which have not yet been documented otherwise, and Listings 17–19 show the essential modifications compared to the demo `sh`.

Table 7: Scripts and functions in `/demos/gcsh`, with comments on the changes compared to `/demos/sh`, and functions from `/libs/linalg` and `libs/p2p` pertinent to global coupling problems.

script/function	purpose, remarks
cmds1d, cmds2dhexfro	scripts for 1D and 2D examples, see Fig. 10
shinit	init, sets `p.fuha.lss=@gclss;p.fuha.blss=@gcblss` as linear system solvers
oosetfemops	set FEM matrices, also stores the 'averaging vector' `p.avvec`
sG,nodalf,sGjac	encodes G with 'nonlinearity' in nodalf, and Jacobian; sG also sets the `global cvec`, and `sGjac` the `global avjvec` needed by the linear system solvers.
shbra1d	modification of `stanbra` for putting the normalized L^2 on the branch
fnl, hfu	functions $f_{\rm nl}$ and h for global coupling from (72)
fnljac, hjac, fnl_a	Jacobians of $f_{\rm nl}$ and h, and $\partial_a f_{\rm nl}$.
gclss	implements (76), with vectors $c =$ `cvec` and $a =$ `avec` passed in `p2pglob`
gcblss	implements the version of (76) for the bordered systems of arclength continuation; here a and c are augmented by a single 0
gclsseigs	version of (76) for `eigs` (inverse vector iteration); uses `lsslueigs` to solve Az=b, where A is LU-prefactored due for repeated solves with the same A.
gcafun	interface routine for `eigs` which contains the actual call to `gclsseigs`
lsslueigs	version of `lsslu` which stores LU factorizations as globals.

```
function p=shinit(p,nx,lx,ly,ndim,par,varargin)  % GCSH as 2 component system
```

```
p=stanparam(p); p.nc.neq=2; p.ndim=ndim; p.fuha.sG=@sG; p.fuha.sGjac=@sGjac;
p.fuha.lss=@gclss; p.fuha.blss=@gcblss; % Sherman-Morrison versions
p.sw.runpar=0;  % switch off parfor in pmnewtonloop (clashes with global vars)
p.sw.eigssol=1; % use Sherman-Morrison in eigs (gclsseigs)
```

Listing 17: `shgc/shinit.m` (lines 1-6). The diffferences to `sh/shinit.m` are in lines 3-5.

```
function r=sG(p,u) % rhs for SH with global coupling in fnl
f=nodalf(p,u)+fnl(p,u); r=p.mat.K*u(1:p.nu)-p.mat.M*f;

function f=fnl(p,u)  % f(u,<h(u)>) for global coupling 0=G(u)+fnl(u,<h(u)>)
h=hfu(p,u); a=p.avvec*h; ga=u(p.nu+3); u1=u(1:p.np); f=[-ga*a*u1; 0*u1];

function h=hfu(p,u) % h for GC
u=u(1:p.np); h=[u.^2; zeros(p.np,1)];
```

Listing 18: `shgc/sG.m`, `shgc/fnl.m` and `shgc/hfu.m`, which are straightforward, as are the Jacobians and fnl_a. `p.avvec` is precomputed in `oosetfemops`.

```
function Gu=sGjac(p,u) % jac for SH with GC
global p2pglob; % p2pglob.avec, cvec computed here and used in, e.g., gclss
hj=hjac(p,u); p2pglob.avec=sum(p.mat.M*spdiags(hj,0,p.nu,p.nu))./p.Om;
p2pglob.cvec=p.mat.M*fnl_a(p,u);
[f1u,f1v,f2u,f2v]=njac(p,u); n=p.nu/2;
Fu=[[spdiags(f1u,0,n,n),spdiags(f1v,0,n,n)];
    [spdiags(f2u,0,n,n),spdiags(f2v,0,n,n)]];
Gu=p.mat.K-p.mat.M*(Fu+fnljac(p,u));
```

Listing 19: `shgc/sGjac.m`. In `sG`, the term `fnl` is first added to the 'nonlinearity' f, and then multiplied by $\mathcal{M}$=`p.mat.M`. In other words, `fnl` and hence also `fnljac` and `fnl_a` contain no $\mathcal{M}$, and for the derivatives $\mathcal{M}$ is taken into account here.

With the modification of the lss setup, the script files for (72) are as for (3), see `gcsh/cmds1d.m` and `gcsh/cmds2dhexfro.m`, and the computations for (72) run almost as fast as for (3). In 3D (or, more generally, for large n_u) it turns out that combining `gclsseigs` with iterative linear system solvers yields further speed advantages, but this will be described elsewhere.

Remark 5.5. More general nonlocal couplings are often given as $f_{\rm nl}(u)(x) = \int_\Omega \kappa(x,\xi)h(u(\xi))\,{\rm d}\xi$, with a kernel $\kappa : \Omega\times\Omega \to \mathbb{R}$, often of the form $\kappa(x,\xi) = k(x-\xi)$, specifically with Gaussians $k_{\rm Gauss}(y) = \alpha_1 {\rm e}^{-\|y\|^2/\alpha_2}$. Our global coupling corresponds to $k\equiv 1$. For general nonlocal couplings, Jacobians naturally are again nonlocal, and importantly no longer of the form "local + rank–1–correction". However, preliminary results indicate that at least for fast-decaying kernels, e.g., small α_2 in $k_{\rm Gauss}$, the full Jacobians can be well approximated by reasonably sparse Jacobians by dropping entries below a certain threshold. This way, nonlocal nonlinearities can still be treated efficiently in `pde2path`, including a bifurcation analysis. Details will be given elsewhere. ⌋

6 Pattern formation on curved surfaces

In [Mur89, Chapter 15], see also, e.g., [VAB99, KEVG18, KBFVG18] and the references therein, and [NLCAS17] for a review, the influence of the geometry and specifically curvature on biological pattern formation on surfaces is discussed. Similarly, in [SLT+15], a Swift-Hohenberg equation on spheres and tori is used as a model for elastic surface patterns, and one interesting result, obtained via time–integration, is that a higher curvature favors hexagons, and lower curvature favors stripes, possibly

with defects, or, more generally, labyrinths. Additionally, there are a number of results on pattern formation on curved surfaces in certain singular limits where spots become strongly localized (i.e., far from onset), and where the interaction of spots is described by DAEs for the spot locations. See, e.g., [JLTW16, TT19] and the references therein. For this, one major step in the analysis are accurate approximations of certain Green's functions on the curved surface, and an interesting (semi-analytical) result is that on tori spots slowly drift towards the inner radius.

Already in 1953 [Tur92], see also [Tur54], Turing aimed at explaining the patterns formed by the silica skeletons of radiolaria (see Fig. 38(a) on page 71) via two component RD models, and computed the diffusion driven instability of RD systems based on the spherical harmonics as eigenfunctions of the spherical Laplace–Beltrami operator (LBO) $\Delta_{\mathcal{S}_R}$. Since then, a lot of 'model-independent' theory on bifurcations with $O(3)$ symmetry has been developed, see, e.g., [IG84, CLM90, Mat03, Mat04, Cal04], and one of our aims here is to use `pde2path` to recover some of these theoretical results, and to follow some pertinent branches, focusing on the Schnakenberg model on spheres, and on tori, which is simpler in the symmetry sense.

Additionally, in §6.6 we patch together two surfaces via a common boundary, and in §6.7 we study a simple model for cell polarization from [CEKM$^+$19], consisting of linear diffusion in the 3D bulk Ω of the cell (the cytosol), coupled a reaction–diffusion equation on the surface $\Gamma = \partial\Omega$ (the cell membrane).

However, we start with (quadratic–cubic) Allen–Cahn (AC) equations on tori and spheres as simpler toy problems, namely

$$\partial_t u = -G(u) \stackrel{!}{=} 0, \quad G(u) = -\Delta_{\rm LB} u - \lambda u - u^2 + \gamma u^3, \quad \lambda \in \mathbb{R}, \gamma > 0, \tag{77}$$

where $\Delta_{\rm LB}$ denotes the LBO over the given surface. Equations similar to (77) have also been used as model problems (over 1D, 2D and 3D 'flat' domains, i.e., with the standard Laplacian and various BCs) in [RU19], and of course (53) and (64) are also quite similar.

We use a standard parametrization

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = \begin{pmatrix} (R + \rho\cos y)\cos x \\ (R + \rho\cos y)\sin x \\ \rho \sin y \end{pmatrix} \in \mathbb{R}^3, \quad (x, y) \in \Omega = [-\pi, \pi)^2, \tag{78}$$

of the (surface of the) torus $\mathcal{T}_{R,\rho}$ with major radius $R > 0$ and minor radius $0 < \rho < R$. The associated LBO is denoted by $\Delta_{\mathcal{T}_{R,\rho}}$ and given by

$$\Delta_{\mathcal{T}_{R,\rho}} u(x, y) = \frac{1}{\rho^2(R + \rho\cos y)} \partial_y((R - \rho\cos y)\partial_y u) + \frac{1}{(R + \rho\cos y)^2} \partial_x^2 u, \tag{79}$$

with periodic BCs in x (think azimuth ϕ) and y (think elevation θ). Thus, $\Delta_{\mathcal{T}_{R,\rho}}$ is translationally invariant in x, i.e.,

$$\Delta_{\mathcal{T}_{R,\rho}} S_\xi u(x, y) = S_\xi \Delta_{\mathcal{T}_{R,\rho}} u(x, y), \text{ where } S_\xi u(x, y) = u(x - \xi, y), \tag{80}$$

and invariant under the two discrete mirror symmetries

$$m_x u(x, y) = u(-x, y) \quad \text{and} \quad m_y u(x, y) = u(x, -y), \tag{81}$$

which will be important to determine whether bifurcations are pitchforks or transcritical.

We parameterize spheres $S_R = \{\tilde{x} \in \mathbb{R}^3 : \|\tilde{x}\|_2 = R\}$ in a slightly non–standard way by

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = \phi(x,y) := R \begin{pmatrix} \cos y \cos x \\ \cos y \sin x \\ \sin y \end{pmatrix} \in \mathbb{R}^3, \quad (x,y) \in \Omega = [-\pi, \pi) \times (-\pi/2, \pi/2), \tag{82}$$

such that $S_R = \phi(\Omega) \cup \{N, S\}$, where $N, S = (0, 0, \pm R)$, and the equator is at $y = 0$. Here the associated LBO is

$$\Delta_{S_R} u(x,y) = \frac{1}{R^2} \left(\frac{1}{\cos^2 y} \partial_x^2 u + \frac{1}{\cos y} \partial_y (\cos y \partial_y u) \right), \tag{83}$$

with periodic BCs in x, and one reason why we first consider the torus is that the coordinate singularity of Δ_S at $y = \pm\pi/2$ warrants some tricks. Moreover, the symmetry group of PDEs of type (77) (and of system of such type) over S_R is $O(3)$, which makes the bifurcation behavior richer than over the torus.

Remark 6.1. a) The Laplace–Beltrami operator on an m–dimensional C^2 submanifold $S \subset \mathbb{R}^d$ is defined as follows. Given a point $\xi \in S$ and a local chart $\psi \in C^2(\Omega, S)$ with $\Omega \subset \mathbb{R}^m$ open and $\psi(x) = \xi$, let $g_{ik} = \langle \partial_{x_i}\psi, \partial_{x_k}\psi \rangle$ be the coefficients of the metric $(g_{ik})_{i,k=1,\dots,m}$, let (g^{ik}) be the inverse of (g_{ik}), and let $g = \det(g_{ik})$ be the determinant.[6] Then,

$$\Delta_S U(\xi) = \frac{1}{\sqrt{g}} \sum_{i,k=1}^{m} \partial_{x_i} (\sqrt{g} g^{ik} \partial_{\xi_k} u)(x), \tag{84}$$

where $U(\xi)$=$u(\psi(x))$, and $\Delta_S u$ is coordinate invariant, i.e., independent of the chart ψ.

b) Given the strong form $\partial_t u - \Delta_S u - f(u) = 0$ of a PDE on a manifold S, there are different options for the weak form and the associated FEM discretization. After multiplication by a test function $v : S \to \mathbb{R}$, we can either require

$$\begin{aligned} &\text{(a)} \int_S (\partial_t u - \Delta_S u - f(u)) v \, \mathrm{d}S = 0, \quad \text{or} \\ &\text{(b)} \int_\Omega (\partial_t u - \Delta_S u - f(u)) v \, \mathrm{d}(x_1, \dots, x_d) = 0, \end{aligned} \tag{85}$$

where in case (b) v is interpreted directly as a function on Ω, and in case (a) for computation we also need to return to coordinates and hence $\mathrm{d}S = \sqrt{g}\, \mathrm{d}(x,y)$. This yields 2 sets of FEM matrices, namely

$$\text{(a) } M_{ij} = \int_\Omega \phi_i \phi_j \sqrt{g} \, \mathrm{d}(x,y), \quad K_{ij} = \int_\Omega \langle c \nabla \phi_i, \phi_j \rangle \sqrt{g} \, \mathrm{d}(x,y), \quad \text{or,} \tag{86}$$

$$\text{(b) } M_{ij} = \int_\Omega \phi_i \phi_j \, \mathrm{d}(x,y), \quad K_{ij} = \int_\Omega \langle c \nabla \phi_i, \phi_j \rangle \, \mathrm{d}(x,y), \tag{87}$$

where in both cases c is used to express the coefficients in (84), and similar expressions hold for F (the discretization of f). For $g \neq 0$, both options are equivalent as the factor $\sqrt{g}$ in (a) cancels, and it then depends on the problem, which option is more useful and/or robust.[7] One advantage of (86) is

[6] For $m = 2$ and $d = 3$, $\sqrt{g}$ yields familiar surface element, i.e., $\mathrm{d}S = \|\partial_x \psi \times \partial_y \psi\| \, \mathrm{d}(x,y) = \sqrt{g} d(x,y)$.

[7] In principle, the different weights in the weak forms (85) weight the different components of the algebraic equations

that for instance integral expressions such as $\int_M u\,\mathrm{d}S$ can directly be evaluated as `sum(M*u,1)`, where `M` is the mass matrix in (86); for (87), we would typically use $\int_M u\,\mathrm{d}S = \texttt{sum}(\texttt{M} * (\texttt{u}. * \texttt{dS}), 1)$, where `dS` is a column vector encoding the surface element.

However, in most of the examples (exception `chtor`) below we do not use integral expressions and thus opt for (87), as it is slightly simpler to code. ⌋

6.1 An Allen–Cahn equation on tori: Demo `actor`

The (quadratic–cubic) Allen–Cahn equation (77) over $\mathcal{T}_{R,\rho}$ reads

$$G(u) = -\Delta_{\mathcal{T}_{R,\rho}} u - \lambda u - u^2 + \gamma u^3 \stackrel{!}{=} 0, \quad \lambda \in \mathbb{R}, \gamma > 0, \tag{88}$$

with trivial solution $u \equiv 0$. We choose $(R, \rho, \gamma) = (2, 1, 1)$ as base parameters, and first continue in λ, but subsequently also continue in R. The eigenvalues λ of $-\Delta_{\mathcal{T}_{R,\rho}}$ have for instance been computed (and continued in the ratio R/ρ) numerically in [GS08], and one crucial observation is that they are generically of multiplicity 1 (with eigenfunctions independent of 'angle' x) or 2 (with eigenfunctions dependent on x, which thus gives a two dimension kernel via shift by half a period). At certain ratios R/ρ there are crossings of eigenvalues and hence eigenvalues of higher multiplicity, but we do not discuss these here.

Figure 34(a) shows kernel vectors at the branch points 2 to 7, from the $u \equiv 0$ branch[8]. The bifurcation points have multiplicities $1, 2, 2, 1, 1, 2, 2, \ldots$, with λ–values agreeing with those in [GS08]. We label the associated branches `b1,...,b7`, and (b) shows a basic bifurcation diagram of (88). The symmetries (80) and (81) explain why only the first and fifths bifurcations are transcritical, and the others are pitchforks, even though we have quadratic terms in (88) and thus no up-down symmetry of the equation. For instance, $S_\pi \phi_2 = -\phi_2$, and hence the two direction $\pm\phi_2$ for `b2` at bifurcation are related by symmetry, and this is inherited by `b2`. Similarly, $S_{\pi/2}\phi_3 = -\phi_3$, $m_y\phi_4 = -\phi_4$, $m_y\phi_6 = -\phi_6$ (or again $S_\pi\phi_6 = -\phi_6$), and $S_{\pi/4}\phi_7 = -\phi_7$, with m_y from (81). However, no such hidden up-down symmetry holds for ϕ_1 or ϕ_5, and hence the bifurcations with these tangents are transcritical. In the (spatially homogeneous) bifurcation at $\lambda = 0$, the subcritical leg corresponds to $u > 0$ and becomes stable after the fold, and the leg with $u < 0$ is stable throughout. In (c,d) we show a selection of solutions at $\lambda = 2$, and in (e,f) we continue two of these solutions in R, just for illustration.

Regarding the implementation, we comment that:

1. To generate the LBO (79) we use the function `K=LBtor(p,R,rho)`; see its source for the straightforward implementation. As long R and ρ are kept fixed, `K=LBtor(p,R,rho)` can be preassembled as usual, but for continuation in R or ρ we naturally need to call `LBtor` in each step; see `sG.m` and `sGjac.m`, which otherwise are completely standard.
2. The implementation of periodic BCs is as in §3.6. Again we call `p=box2per(p,[1 2])` during initialization (see `acinit.m`), and `filltrafo` after each assembly of a system matrix such as `K` (associated to (79)) or `M`.
3. The translational invariance of (88) in x requires a phase condition (PC) for the continuation of x–dependent solutions as in (32), see also Remark 3.2. However, in contrast to §3.6 we do not

obtained, and hence should be chosen with some care. Note that we can always introduce additional weights in the integrals, as long as we also take them into account in the integration by parts and the BCs.

[8] all named τ_1 because it's the first (and only) component of the function u; however in the following we call them $\phi_2, \ldots \phi_7$ for convenience. The eigenfunction at the first branch point $\lambda = 0$ is $\tau_1 = \phi_1 \equiv 1$.

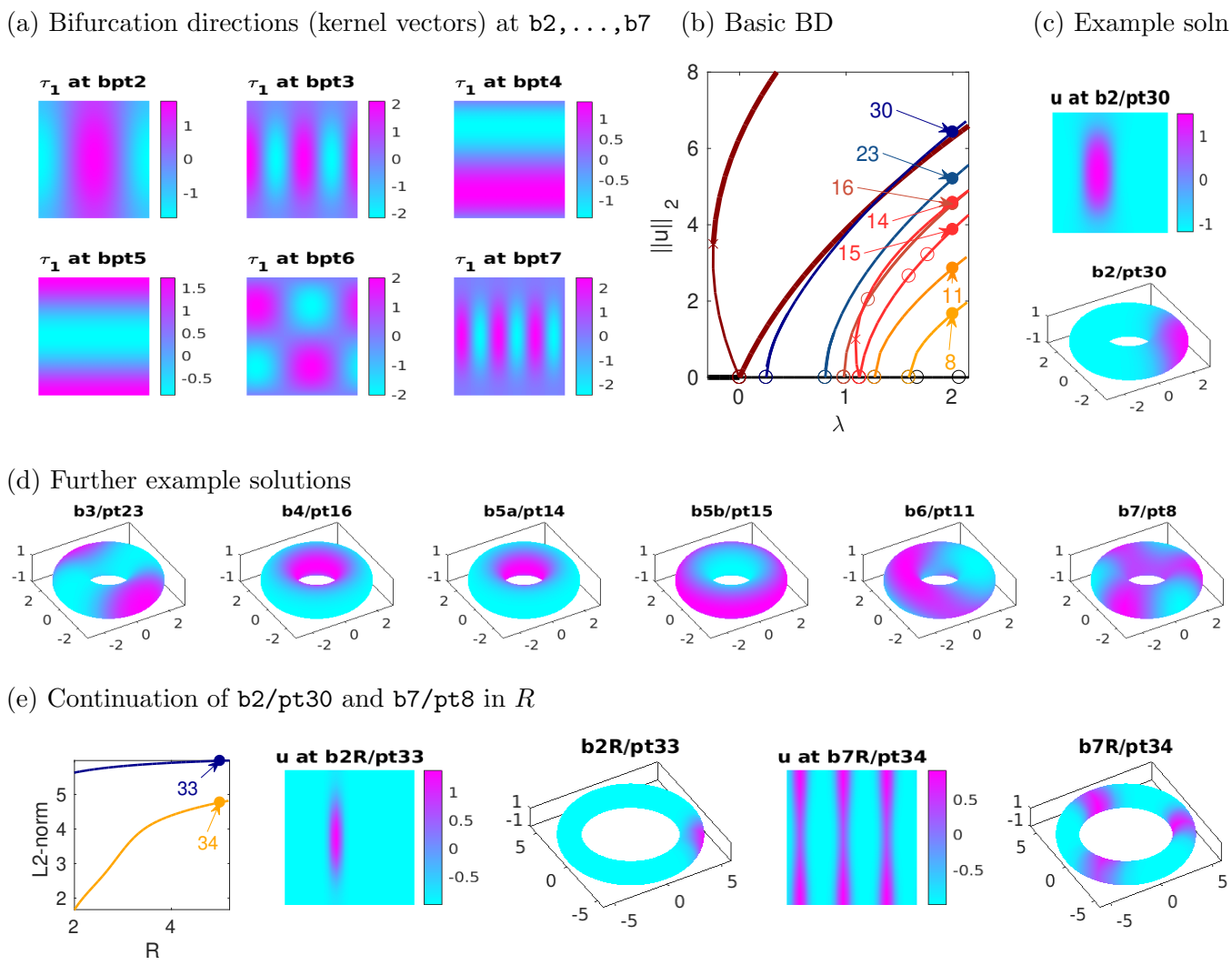


Figure 34: The Allen–Cahn equation (88) on tori, `nt=3600` triangular elements. (a) Kernel vectors at BPs 2 to 7 (only one shown at the double BPs 2,3,6 and 7). (b-d) Continuation in λ and example solutions. (e) Continuation in R of two solutions from (b).

need to use `q(c)swibra` for these branches, because modulo translations the multiplicity is 1.

4. To produce the surface plots in Fig. 34 we use the convenience function `torplot`.

6.2 Spheres: Demo `acS`

The symmetry group Γ of a PDE with space ($\tilde{x}$) independent coefficients on a sphere S_R of radius R is $O(3)$, generated by two rotations (around the $\tilde{x}_1$ axis and the $\tilde{x}_3$ axis, say) and one reflection ($\tilde{x}_1 \mapsto -\tilde{x}_1$, say). The consequences of these symmetries for bifurcations from a branch of spatially homogeneous solutions have been studied in detail in, e.g., [IG84, CLM90, Mat03, Mat04, Cal04]. Here we want to recover and illustrate some of these results numerically, again first using the AC equation as a toy model, i.e.,

$$0 = G(u) = -\Delta_{S_R} u - \lambda u - u^2 + \gamma u^3, \quad \lambda \in \mathbb{R}, \gamma > 0. \tag{89}$$

We parameterize S_R by (82) with $\Delta_{\mathcal{S}_R}$ given in (83), i.e., $\Delta_{S_R}u(x,y) = \frac{1}{R^2}\left(\frac{1}{\cos^2 y}\partial_x^2 u + \frac{1}{\cos y}\partial_y(\cos y \partial_y u)\right)$. The eigenfunctions of $\Delta_{\mathcal{S}_R}$ are the spherical harmonics

$$Y_\ell^m(x,y) = \mathrm{e}^{\mathrm{i}m\phi}P_\ell^m(\sin y), \quad m \in \mathbb{N},\ \ell = m, m+1, \ldots, \tag{90}$$

where $P_\ell^m(s)$ are the associated Legendre functions, and the associated eigenvalues are

$$\mu_\ell = \frac{1}{R^2}\ell(\ell+1), \text{ of multiplicity } 2\ell+1. \tag{91}$$

For given $\ell \in \mathbb{N}$ we have one eigenfunction with $m = 0$ (no x–dependence) and two eigenfunctions $\cos(mx)P_\ell^m(\sin y)$ and $\sin(mx)P_\ell^m(\sin y)$ for each $1 \le m \le \ell$. Thus, for (89) we have bifurcation points at

ℓ	0	1	2	3	...
$R^2\lambda$	0	2	6	12	...
multiplicity	1	3	5	7	...

(92)

Moreover, from symmetry and properties of the Y_ℓ^m it is known that bifurcations at even ℓ are generically transcritical while those at odd ℓ are all pitchforks.

To study these bifurcations numerically, using (82) and the ideas from §3.6 for branch switching at BPs of higher (discrete) multiplicity and with additional continuous symmetries, we need to deal with the coordinate singularity of Δ_S at $y = \pm\pi/2$, and we should also choose meshes that are coarser for y near $\pm\pi/2$ than near $y = 0$. For the first problem we use a standard trick and slightly cut off the sphere near the poles, and instead of $\Omega = (-\pi,\pi)\times(-\pi/2,\pi/2)$ choose

$$\Omega = (-\pi,\pi)\times(-\pi/2+\delta,\pi/2-\delta) \tag{93}$$

with a small $\delta > 0$ (typically $\delta = 10^{-3}$), and pBCs in x and homogeneous Neumann BCs in y. The pertinent branches for (89) are still like on the full sphere, and the choice of δ (in the range 10^{-4} to 10^{-2}) plays no visible role. Concerning suitable meshes, we use a modification `sppdeo` of `stanpdeo2D`, where we provide a set of points in Ω which becomes coarser as $|y| \to \pi/2-\delta$, and then use `Matlab`'s `delaunayTriangulation` method to set up the mesh.

Figure 35 shows results for the continuation of steady solution branches of the AC equation on a sphere of radius 3. In (a) we just sketch the idea of the meshes, (b) shows a basic bifurcation diagram, and (c,d) show example solutions. The results completely agree with theoretical predictions. In the script `cmds.m` (see Listing 21) we use `qswibra(...,aux)` (for $l = 2$) and `cswibra(...,aux)` (for $l = 1, 3$) to remove the spurious multiplicity from the translational invariance in x by choosing appropriate active kernel lists `aux.ali`, cf. §3.6. We then obtain 1,2, and 3 distinct branches, at the 2nd, 3rd and 4th BP, respectively. To follow the branches with x–dependence we use the convenience function `conpc` which switches on the x phase-condition after a few initial steps. In `cmds.m` we also compute a number of branches at further BPs, yielding interesting patterns as predicted from the Y_ℓ^m, but naturally these are all unstable in the AC equation.

```
function p=conpc(p,ds,n1,n2) % convenience function to continue with PC
% first n1 initial steps without PC, then n2 steps with PC
p.sol.ds=ds; p=cont(p,n1); spplot(p); pause;
```

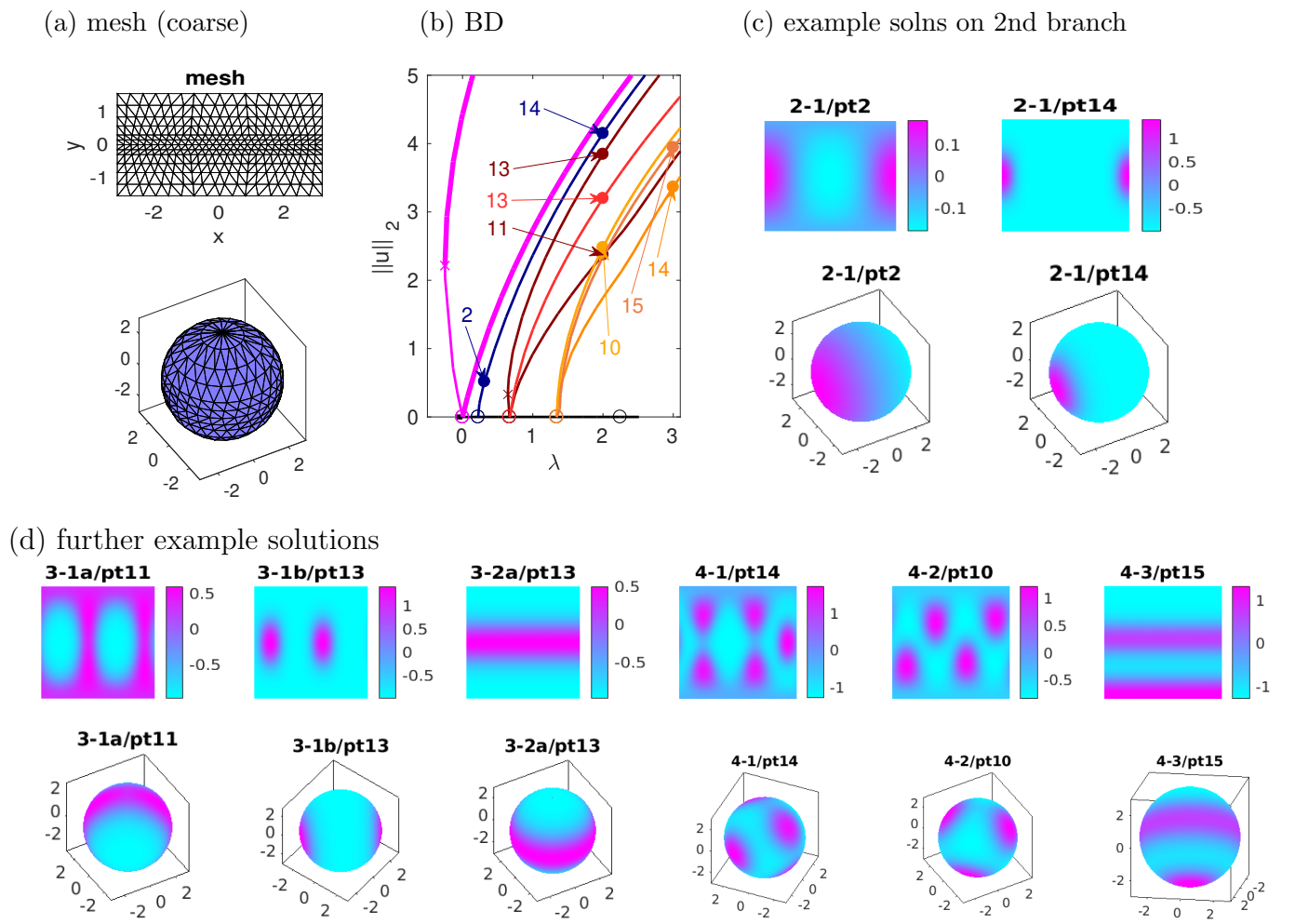


Figure 35: The Allen–Cahn equation (88) on a sphere of radius 3 (with cutoff $\delta = 10^{-3}$). (a) Idea of the meshing, with coarser meshes for $|y| \to \pi/2 - \delta$. (b) Basic bifurcation diagram of branches from the first 4 BPs. (c,d) example solutions, where `2-1, 3-1a, 3-1b, 4-1, 4-2, 4-3` correspond to the blue (twice), brown (twice), red, and orange (triple) branches, respectively. `nt=4470` triangular elements used for the computations.

```
p.nc.nq=1; p.nc.ilam=[2 4]; p=cont(p,n2); spplot(p);
```

Listing 20: `acS/conpc.m`; convenience function to continue with phase condition in x

```
%% init and trivial branch
p=[]; par=[3 -0.1 1 0]; % parameters [R lambda gamma s]  (s=speed for x-PC)
lx=pi; del=1e-2; ly=pi/2-del; nx=31; ny=15; ref=1;
p=acinit(p,lx,ly,nx,ny,par,ref); p=setfn(p,'tr3'); p=cont(p,20);
%% BP1, simple, spat.homogen.branch(es)
p=swibra('tr3','bpt1','1a',0.1); p=cont(p,20);
p=swibra('tr3','bpt1','1b',-0.1); p=cont(p,20);
%% BP2, l=1, pitch, dim(ker)=2l+1=3, use aux.ali to select kernel vectors
aux=[]; aux.besw=0; aux.m=3;  aux.isotol=1e-3;
aux.ali=[1 3]; aux.besw=1; % in first run, comment out this line to see EFus
p0=cswibra('tr3','bpt2',aux); p0.sw.bifcheck=0; p0.nc.tol=1e-6;
%% one spot
p=seltau(p0,1,'2-1',3); p.file.smod=5; p=conpc(p,0.05,2,20);
%% BP3, l=2, dim(ker)=2l+1=5, trans, 2 branches (modulo symmetry)
aux=[]; aux.isotol=1e-2; aux.besw=0; aux.m=5;
aux.ali=[1 3 5]; aux.besw=1; % selecting kernel vectors
```

```
p0=qswibra('tr3','bpt3',aux);
```

Listing 21: `acS/cmds.m`, using `aux.ali=[1 3]` in `qswibra(...,aux)` for $\ell = 1$ to select the 'active' kernel vectors. The function `spplot` is a convenience function for the surface plot.

Intermezzo: The Cahn–Hilliard problem on a torus. In §5.2 we discussed the Cahn–Hilliard problem of spinodal decomposition on boxes (1D, 2D and 3D), including the relation of the interfaces between $u = -1$ and $u = 1$ to minimal embedded surfaces. Here we briefly transfer this to curved surfaces, focussing on the torus. In the demo `chtor` we consider

$$G(u) := -\varepsilon^2 \Delta_{\mathcal{T}_{R,\rho}} + W'(u) - \lambda \stackrel{!}{=} 0 \quad q(u) := \frac{1}{|\Omega|} \int_\Omega u \, \mathrm{d}S - m \stackrel{!}{=} 0, \tag{94}$$

on $\mathcal{T}_{R,\rho}$, (i.e., $\Omega = [-\pi, \pi)^2$ with (78)), with $W(u) = \frac{1}{4}(u^2 - 1)^2$, and the associated energy

$$E_\varepsilon = \frac{1}{2\sigma} \int_{\mathcal{T}_{R,\rho}} \frac{\varepsilon}{2} |\nabla_{\mathcal{T}_{R,\rho}} u|^2 + \frac{1}{\varepsilon} W(u) \, \mathrm{d}S, \tag{95}$$

where $\sigma = \int_{-1}^{1} \sqrt{\frac{1}{2}\tilde{W}(u)} \, \mathrm{d}u = \frac{\sqrt{2}}{3}$. We evaluate $\int_{\mathcal{T}_{R,\rho}} W(u(x)) \, \mathrm{d}S$ as `sum(M.*(W(u).*dS),1)`, where `dS` $= (R + \rho\cos(y))\rho$ is computed in the convenience function `surfelem` cf. Remark 6.1(b), and `chE` for the implementation. Similarly,

$$\int_{\mathcal{T}_{R,\rho}} |\nabla_{\mathcal{T}_{R,\rho}} u|^2 \, \mathrm{d}S = \int_{\mathcal{T}_{R,\rho}} (-\Delta_{\mathcal{T}_{R,\rho}} u) u \, \mathrm{d}S = \texttt{sum((Ku).*u.*dS,1)}.$$

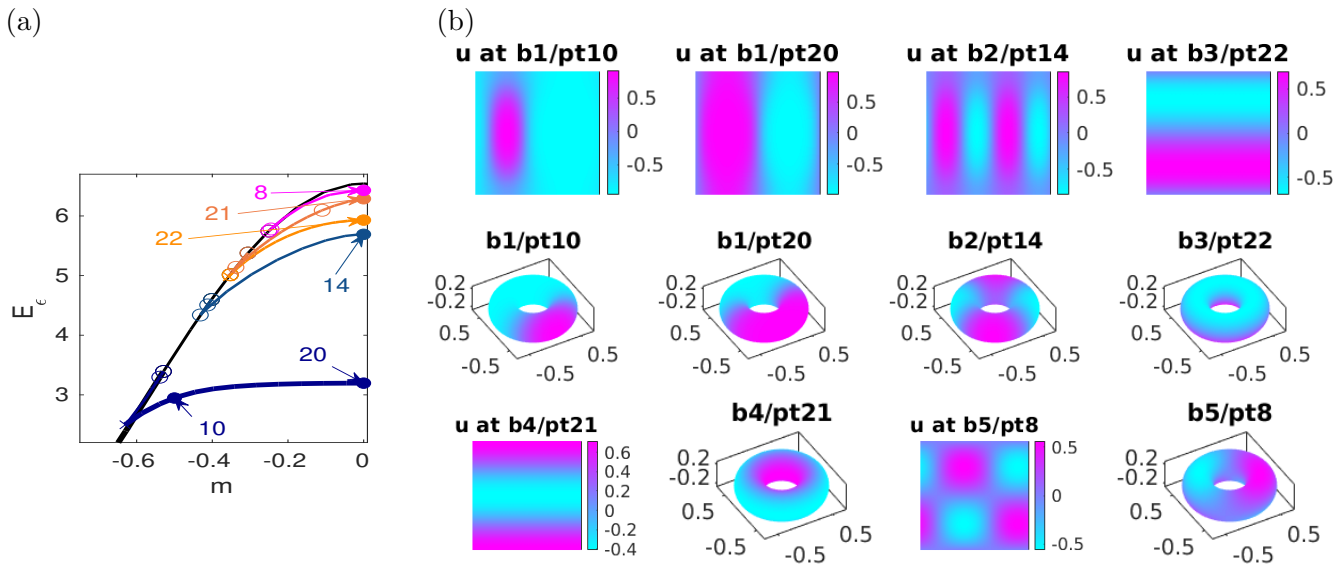


Figure 36: Results from `chtor/cmds1.m`. Bifurcation diagram (a) and sample solutions (b) (top views over the preimage $\Omega = [-\pi, \pi)^2$, and torus plots, $R = 0.5, \rho = 0.25$) for the Cahn–Hilliard problem (94) with $\varepsilon = 0.2$. The primary instability of the homogeneous branch $u \equiv m/|\mathcal{T}_{R,\rho}|$ (black) is to a branch (`b1`, dark blue) with two "vertical rings" as interface, followed by a branch with four vertical rings (`b2`, lighter blue), then horizontal rings of type "inside–outside" (`b3`, yellow) and "up–down" (`b4`, orange), and finally spots (`b5`, magenta).

Figure 36 shows results for (94) with $R = 0.5, \rho = 0.25$, and rather large $\varepsilon = 0.2$, starting with $n_p \approx 1750$ mesh–points. For $m = 0$ and $\varepsilon \to 0$ we expect the interface lengths

$$|I_{b1}| = \pi \text{ (two circles of radius } \rho = 1/4), \quad |I_{b2}| = 2\pi \text{ (twice } |I_{b1}|),$$
$$|I_{b3}| = 2\pi((R+\rho)+(R-\rho)) = 2\pi, \quad \text{and } |I_{b5}| = |I_{b1}| + |I_{b3}| = 3\pi.$$

On `b4` we expect the limit interfaces at heights $\pm\rho \sin y_0$, where y_0 solves

$$2\rho \sin(y_0) = (\pi - 2y_0)R, \tag{96}$$

which for $\rho = 1/4$ and $R = 1/2$ yields $y_0 \approx 1.12$, and hence interface length $|I_{b4}| = 2 * 2\pi * (R + \rho\cos y_0) \approx 7.65$. These predictions are checked in Fig. 37, where at the start of the continuation in ε we do some mesh refinement. However, for instance (a) shows that further mesh refinement is needed as $\varepsilon \to 0$, as in, e.g., Fig. 28(e).

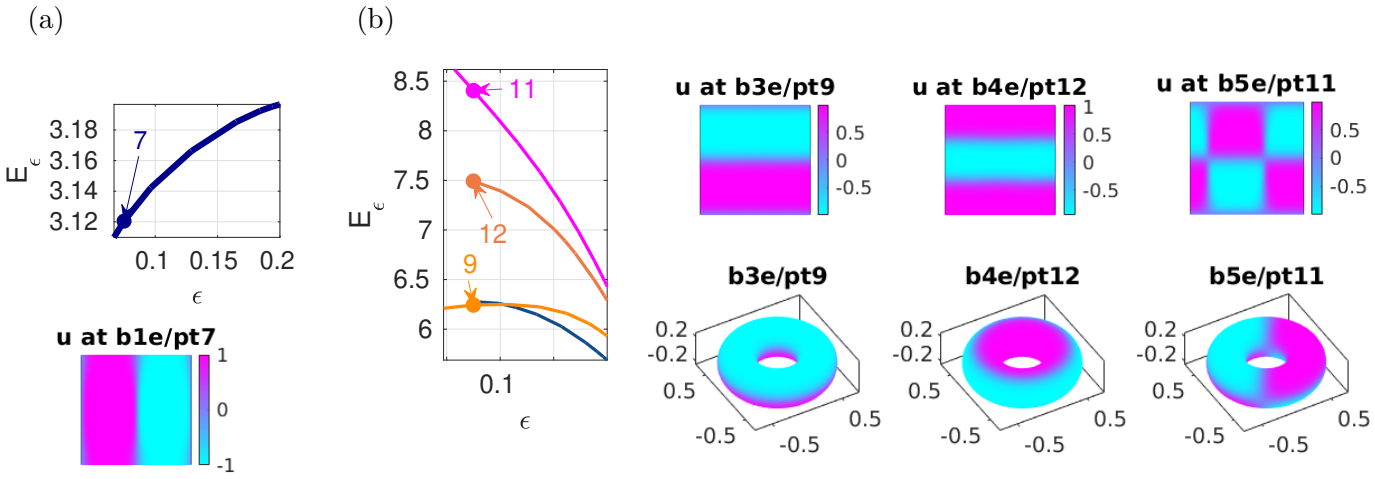


Figure 37: Continuation to $\varepsilon = 0.075$ of solutions from Fig. 36 at $m = 0$. Primary spinodal branch (a), and further branches (b).

6.3 Schnakenberg on spheres: Demo `schnakS`

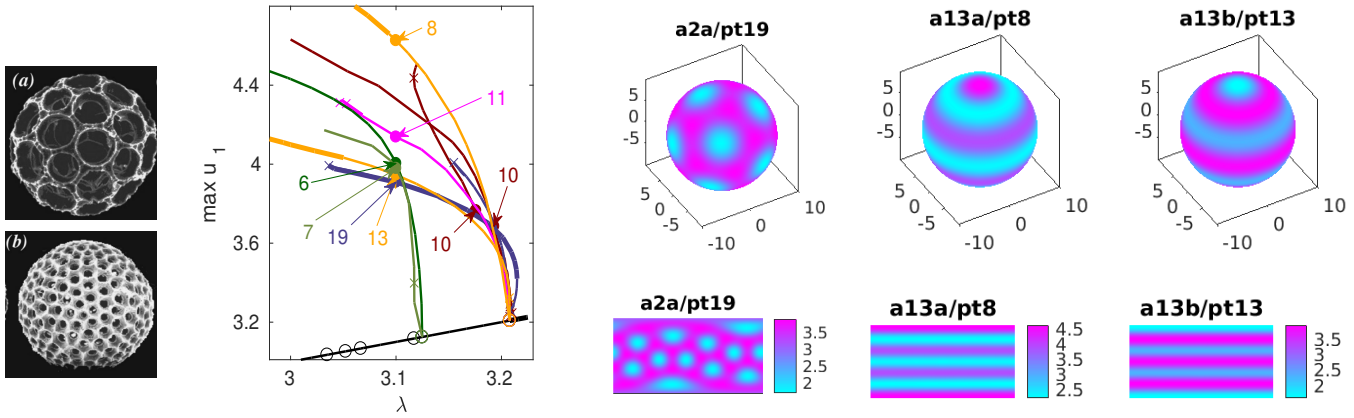
The Schnakenberg model on spheres $\mathcal{S}_R$ is obtained by replacing Δ in (46) by $\Delta_{\mathcal{S}_R}$ from (83). Thus we consider

$$\partial_t U = D\Delta_{\mathcal{S}_R} U + F(U),\ U = \begin{pmatrix} u \\ v \end{pmatrix},\ D = \begin{pmatrix} 1 & 0 \\ 0 & d \end{pmatrix},\ F(U) = \begin{pmatrix} -u+u^2v \\ \lambda - u^2 v \end{pmatrix} + \sigma \left(u - \frac{1}{v}\right)^2 \begin{pmatrix} 1 \\ -1 \end{pmatrix}. \tag{97}$$

In Fig. 38(b)-(d) we fix $(\sigma, d) = (0, 60)$, and use λ as our primary bifurcation parameter, on a sphere of radius $R = 10$. At the first BP1 `bpt1` from the homogeneous branch $U = (\lambda, 1/\lambda)$ at $\lambda_c \approx 3.208$ we then have $\ell = 6$. Hence $m = \dim N(G_u) = 13$ and the bifurcations are generically transcritical. At the second BP `bpt2` we have $\ell = 5$, and hence $m = 11$ and all bifurcations are pitchforks. To find (a selection of) the bifurcating branches we use similar ideas as in §6.2, i.e., we first call `qswibra` (at `bpt1`) with `aux.besw=0` and inspect the kernel vectors. Then selecting appropriate `aux.ali` we call `qswibra` with `aux.besw=1` to find bifurcation directions. At `bpt2` we proceed analogously with

`cswibra`. However, both algorithms do run into problems for the still high–dimensional kernels (of dimension $\ell+1$), and to find solutions of the branching equation we need to increase `aux.soltol`, and to identify solutions as isolated we need to decrease `aux.isotol`. Still, depending on the choices of these parameters, typically only a subset of the expected branches is found, and thus we also use `gentau` to, e.g., start the horizontal stripes branches `a13*` (with *=a,b for the two 'legs') and `b1`.

(a) Real life (b) Some primary bifurcations (c) Three example solutions, the first one stable



(d) Further example solutions

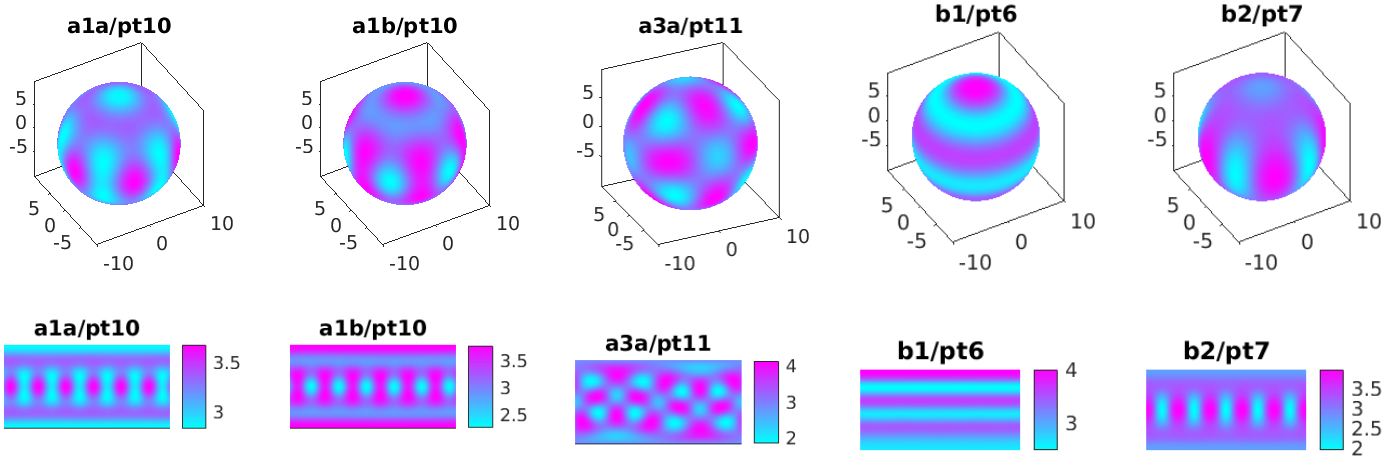


Figure 38: Pattern formation on spheres. (a) Silica skeletons of two radiolaria, scale about 1mm, from [VAB99]. (b)-(d) (97) on a sphere of radius $R=10$, `nt=10000` triangular elements. (b) *Some* branches bifurcating at the first BP ($l=6$, hence transcritical) and second BP ($l=5$, hence pitchfork)), colors: `a1` red, `a2` blue, `a13` orange, `a3` magenta. `b1,b2` green. (c,d) example solutions.

Remark 6.2. a) For $R=10$, we have $|S_R|=4\pi R^2\approx 1257$, which roughly corresponds to a 'not–too small' flat square of side-lengths $7\pi/k_c$, where $k_c{=}\sqrt{\sqrt{2}{-}1}$ is the critical wave number from the flat case, cf. §4. Moreover, $\lambda_c(R)|_{R=10}{\approx}3.208$ is already quite close to the (flat) infinite domain value $\lambda_c{=}\sqrt{60}\sqrt{3{-}\sqrt{8}}\approx 3.21$, and λ only changes slightly for larger R. The wave number ℓ changes in the size parameter R, but the wavelength (on S_R) essentially stays the same. Thus, we conclude that for $R=10$ (and larger) we are in the regime where the patterns from the first (and second) BP are intrinsic, and not an artifact of the domain size. See also the discussion of intrinsic versus extrinsic patterns in [LBFS17, §6].

b) Besides the difficulties due to larger ℓ, the main consequence of a larger R is that the BPs come

closer together, and that the spectral gaps between the $2\ell+1$ (approximate) zero eigenvalues and the rest of the spectrum decrease, exactly as in the flat case. For $R = 10$, the gap size at `bp1` is about 0.025 (the value of the 14th eigenvalue), and at `bpt2` the gap size is about 0.013.

c) The case $\ell = 6$ has for instance been discussed in a detailed model–independent way in [Mat03, §5.1]. The 1(real)+6(complex)–dimensional amplitude equations of quadratic order yield four branches (modulo isotropy) of patterned solutions, namely:

Isotropy	#neg. Eigenval. close to bif.	Example in Fig. 38
$O(2)$ (e.g., rotation around $\tilde{x}_3$ axis)	3	`a13`
I (icosaeder group)	1	`a2`
O (octaeder group)	5	`a3`
D_6 (symmetry group of reg. hexagon)	4	`a1`

In particular, while all bifurcating branches are unstable at bifurcation, the I branches are preferred in the sense of a certain variational principle [Bus75], and in the sense that they have only one unstable direction and hence the potential to stabilize in a fold. This is what happens to the `a2a` branch in Fig. 38. We also recover the other numbers of unstable eigenvalues near bifurcations, and note that the stripes `a13a` and `a13b` both become stable at rather large amplitude.

d) Additionally, and in contrast to the Schnakenberg model over flat domains in §4.2, we note that patterned branches typically do not extend to small λ, but show a number of folds for λ still 'not far' from bifurcation, i.e., near $\lambda > 3$, say. See, e.g., branches `a1b` and `a2a/b`. This may partly be due to 'branch jumping' in the continuation, which as usual we try to mitigate by using `pmcont`, cf. §3.8.3, but it seems that the spherical geometry gives more restrictions on the λ range of solutions than 'flat boxes' with Neumann BCs, cf., e.g., Fig. 22. A similar effect also occurs if we try to continue the branches from $R = 10$ to larger R and smaller R, see Fig. 39.

(e) In any case, if one wants to study patterns and bifurcations on (significantly) larger spheres ($R > 20$, say), then time integration (cf. §3.8.3) to identify stable branches should be used first. ⌋

6.4 Schnakenberg on tori: Demo `schnaktor`

Following §6.1, in Figures 40 and 41 we consider (46) on tori, now replacing Δ by $\Delta_{\mathcal{T}_{R,\rho}}$, i.e.,

$$\partial_t U = D\Delta_{\mathcal{T}_{R,\rho}} U + F(U), \tag{98}$$

with D and $F(U)$ as in (97), and $U = U(x, y)$, $(x, y) \in [-\pi, \pi)^2$ with pBCs in x and y. We use the base parameters $(\sigma, d) = (-0.1, 60)$, and again λ as the primary bifurcation parameter. The torus is simpler than the sphere in the sense that the bifurcations from the trivial branch are generically simple (modulo the double multiplicity due to translational invariance in azimuth), and naturally (97) inherits the symmetries (80), (81) from $\Delta_{\mathcal{T}_{R,\rho}}$, and these can be used to decide which bifurcations are transcritical and which are pitchforks. However, even over the relatively small torus in Fig. 40 with $(R, \rho) = (12, 4)$ and hence $|\mathcal{T}_{R,\rho}| = 4\pi\rho R \approx 600$, already more than 25 eigenvalues cross 0 between $\lambda = 3.21$ and $\lambda = 3.15$, and thus in the implementation (see `cmds1.m`) we now use a trick as for (65) to obtain a first bifurcation diagram close to the first bifurcation from $U \equiv (\lambda, 1/\lambda)$ in a simple way:

- We only localize "the first" (where we don't care if it is really the first) BP using the `bifcheck=2` setting. We then call `cswibra` but only to compute (here `m=8`) 'nearby' eigenvectors, i.e., with the argument `aux.besw=0`. Then we use `gentau` to select different approximate numerical kernel vectors as predictors for the branch switching.

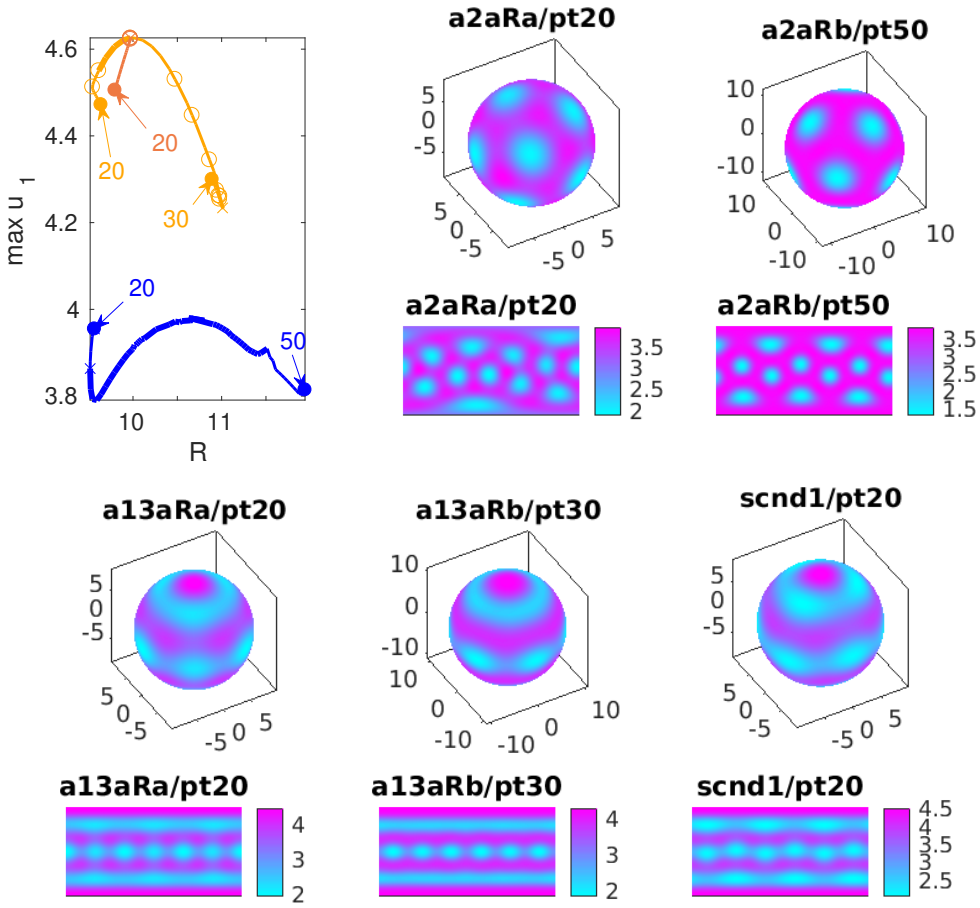


Figure 39: Continuation in R of a pentagonal (blue branch) and a stripe (orange branch) solution from Fig. 38, and one secondary branch (brown). Both types of primary branches solutions seem to exist only in a rather small range in R, i.e., both show folds at R not much smaller/larger than the starting value $R = 10$, and there is a strong tendency to 'branch jumping' during the continuation. See text for further comments. .

Figure 40(a,b) shows results for the first four branches thus obtained. The bifurcations are pitchforks for symmetry reasons as explained for (88). In `cswibra` we use $m = 8$ because the first four BPs are all x–dependent and hence double, such that, e.g., to obtain `c3` we use `p=gentau(p0,[0 0 0 0 1],'c3')`. Due to the x–dependence we also again switch on a PC similar to (35) after one step. The 4 branches are all unstable throughout, except for `c3` which becomes stable after the fold, before again loosing stability in a bifurcation to a branch of solutions with patterns wrapping around the torus, see Fig. 40(c). In summary:

- Already on $\mathcal{T}_{12,4}$, (98) is very rich in pattern formation, but the only stable pattern we found consist of a rather regular arrangement of hexagons, see `c3/pt13` in Fig. 40(b). There are also patterns which consist of stripes, and patterns which combine spots and stripes, but these are illustrated for a larger torus below.
- The implementation re–uses `LBtor` and other ideas from §6.1 in a straightforward way, and additionally we use `cswibra` with `aux.besw=0` as explained above.

In Fig. 41 we consider (98) on a larger half torus with $(R, \rho) = (25, 10)$, and hence $|\mathcal{T}_{R,\rho}| = 1000\pi$. Here 'half' means that $(x, y) \in (-\pi/2, \pi/2) \times [-\pi, \pi)$, and we use pBCs only in y but homogeneous Neumann BCs in x, which is switched on via `p=box2per(p,2)`. The idea is to reduce computational costs using that solutions that are even in x can be mirrored at $x = \pm\pi/2$ to obtain solutions on the

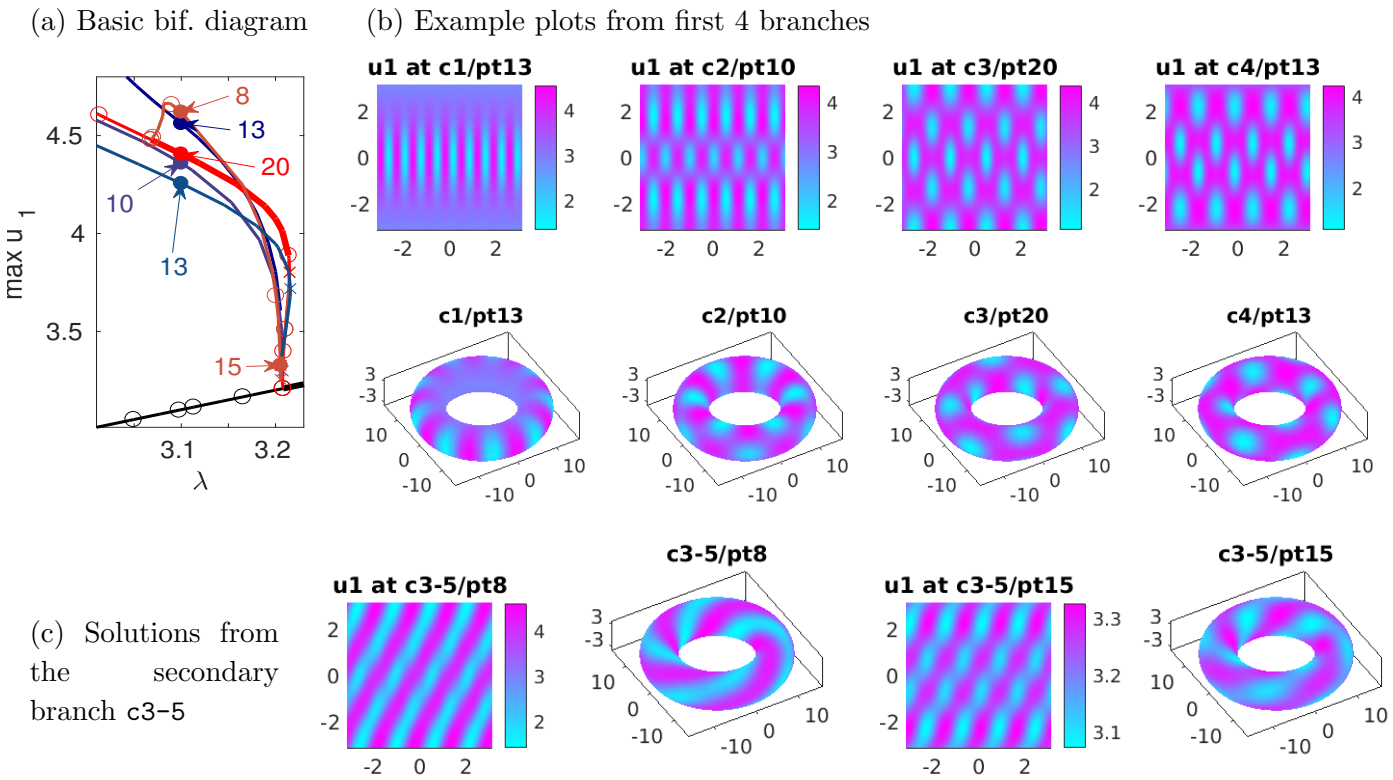


Figure 40: (98) on a rather small torus, where however already many primary bifurcation occur close together near $\lambda \approx 3.2$. Branches `b1` (dark blue), `b2` (light blue), `b3` (red), `b4` (blue again), and 2 secondary branches `c3-4` (orange) and `c3-5` (brown). This BD is only a small selection, of which only `c3` is stable in a certain λ range, see (b) for example plots at $\lambda = 3.1$. (c) shows solutions from secondary branches bifurcating from `c3`, where `c3-5` reconnects to `c3` at small amplitude. `nt=10000` triangular elements.

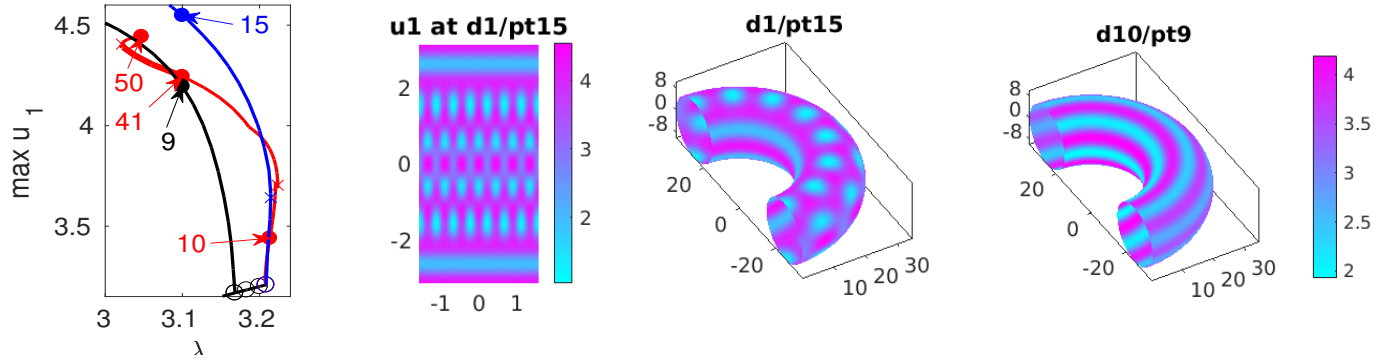
full torus. Otherwise the implementation with script `cmds2.m` uses the same ideas as `cmds1.m`. In (b) we show two exemplary solutions, where the branch `d10` consists of pure stripes, obtained from a primary bifurcation at the 4th BP from the homogeneous branch (in the sense of the trick explained after (97)). In (c) we show solutions from the red branch in (a), which becomes stable roughly at $\lambda = 3.1$. At the outer radius there are somewhat regular hexagons, which get closer to stripes for $|\vartheta| \to \pi$, i.e., near the inner radius. For smaller λ the branch shows a fold where the 'inner spots' merge to stripes and the branch looses stability.

This last result at least phenomenologically agrees with the results from [SLT+15], where, in the bistable regime between spots and stripes for a SH equation, the spots tend to sit on the 'outside' ($-\pi/2 < \vartheta < \pi/2$), and the stripes on the 'inside' ($|\vartheta| > \pi/2$). The mean curvature $H = -\frac{1}{2}\left(\frac{\cos\vartheta}{R+\rho\cos\vartheta} + \frac{1}{\rho}\right)$ is smaller on the inside, and thus these results also agree with similar results on spheres, for which high curvature favors hexagons, and low curvature favors (labyrinthine) stripes.

6.5 Schnakenberg on cones

In, e.g., [PSGnP+04], pattern formation on cones (coninal surfaces) is studied numerically by DNS, with a focus on growing (in time) cones, and roughly the results are as follows: for pointed cones, stable patterns take the form of rings near the tip, which (for suffiently large cones) transform into spots near the base. For blunter cones, rings near the tip become less dominant. In `schnakcone` we

(a) Partial BD (b) Example solutions on `d1` and `d10`, all unstable



(c) Example solutions on `d7`, where, e.g., `pt41` corresponds to a stable solution

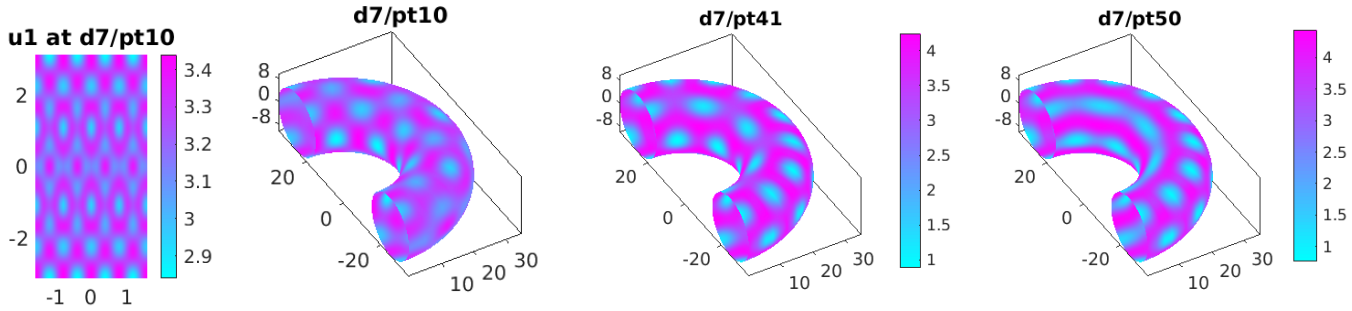


Figure 41: (98) on a 'half torus', with Neumann BCs in x direction. (a) BD of three primary bifurcations from the trivial branch, with two example solutions from the unstable branches `d1` (blue) and `d10` (black) in (b). (c) shows example solutions from the red branch `d7` with for instance `pt41` stable. `nt=12800` triangular elements.

consider the Schnakenberg problem on cones C_a, which we parameterize as

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = \phi(x,y) := \begin{pmatrix} ax \\ ay \\ -(x^2+y^2)^{1/2} \end{pmatrix} \in \mathbb{R}^3, \quad (x,y) \in \Omega, \tag{99}$$

where $\Omega \subset \mathbb{R}^2$. For Ω a disk (of radius 1, say), we have a classical circular cone, with "base" at $\tilde{z} = -1$ of radius a, and tip at $(\tilde{x}, \tilde{y}, \tilde{z}) = (0,0,0)$. This is essentially what we have in mind, but for convenience we (slightly) break the rotational invariance of C_a by considering elliptical domains Ω_e instead of disks, namely

$$\Omega_e = \{x^2/e^2 + y^2 \leq 1\} \tag{100}$$

with major axis $e \geq 1$. The metric tensor is $g_{ij} = \begin{pmatrix} a^2+x^2/r^2 & xy/r^2 \\ xy/r^2 & a^2+y^2/r^2 \end{pmatrix}$, $r = (x^2+y^2)^{1/2}$, the surface element is $a\sqrt{1+a^2}$, and the LBO reads

$$\begin{aligned} \Delta_{C_a} u(x,y) = \frac{1}{a^2(1+a^2)} \Big(& \partial_x((a^2 + y^2 r^{-2})\partial_x u) - \partial_y(xyr^{-2}\partial_x u) \\ & -\partial_x(xyr^{-2}\partial_y u) + \partial_y((a^2 + x^2 r^{-2})\partial_y u) \Big). \end{aligned} \tag{101}$$

Because in (100) we fix the (preimage) domain size (near the unit disk), we introduce a scaling parameter $\varepsilon > 0$ to vary the cone size. Our Schnakenberg problem on cones thus reads

$$\partial_t U = \begin{pmatrix} \varepsilon^2 \Delta_{C_a} & 0 \\ 0 & d\varepsilon^2 \Delta_{C_a} \end{pmatrix} U + F(U), \quad F(U) = \begin{pmatrix} -u + u^2 v \\ \lambda - u^2 v \end{pmatrix}, \quad (x, y) \in \Omega_e, \tag{102}$$

as usual with Neumann BCs on $\partial\Omega_e$. [9] Thus, all lengths are scaled with $1/\varepsilon$, and for instance the effective surface of the (circular) cone associated to Ω_1 is $a\sqrt{1+a^2}\pi/\varepsilon^2$. The implementation of (102) uses `diskpdeo2b.m` to generate Ω_e, `LBcone.m` to generate the LBO Δ_{C_a} from (101), a preparatory mesh–refinement near $r = 0$ to suitably resolve the tip, a custom function `coneplot.m` to plot solutions, but otherwise proceeds as usual.

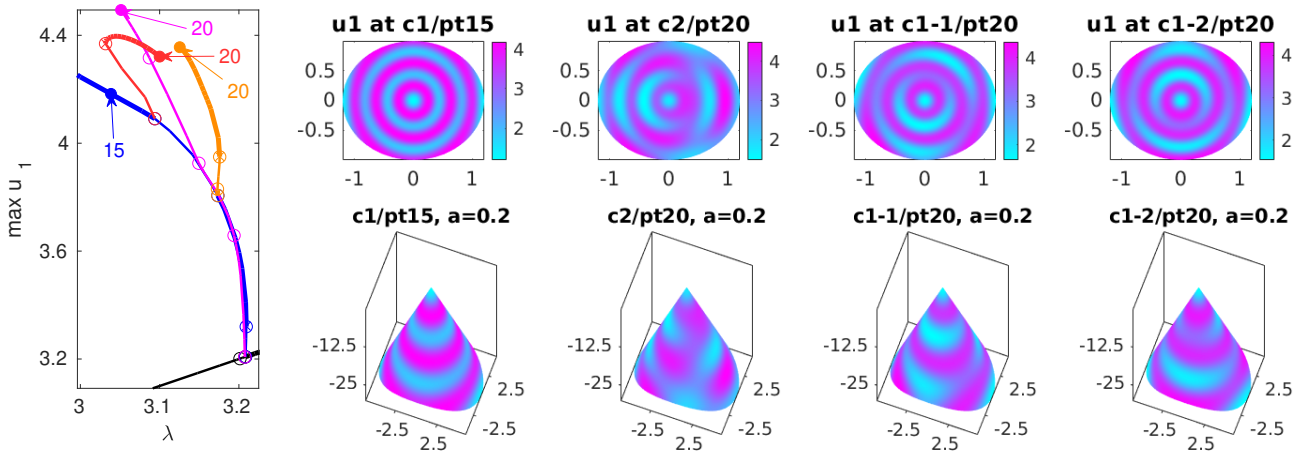


Figure 42: (98) on the (acute) cone associated to $\Omega_{1.2}$, $\varepsilon = 0.04$, base radius $a = 0.2$. Basic BD, obtained from `schnakcone/cmds1.m`, and sample plots over the associated cones, including the scaling by $1/\varepsilon$. Two primary bifurcating branch c1 (blue) and c2 (magenta), and two secondary branches (c1-1, orange) and (c2-2) bifurcating from c1.

In Fig. 42 we start with a rather pointed cone, $a = 0.2$, and follow the first two branches bifurcating from the usual homogeneous branch $(u, v) = (\lambda, 1/\lambda)$. The primary bifurcation (`c1`, blue) is to rings (near the tip), as expected, but the branch becomes unstable[10] to distorted rings and later gains stability again. The second branch (`c2`, magenta) is more spot like. It is unstable throughout, but there are secondary bifurcations to branches with stable solutions showing rings at the tip, which once more shows a richness of pattern formation on a rather small domain.

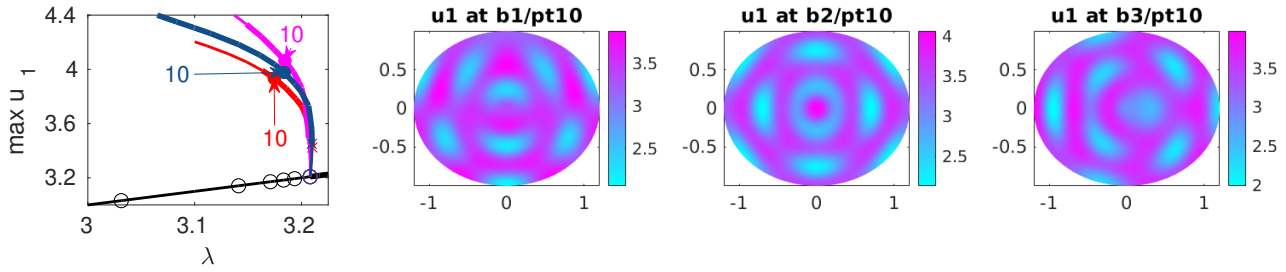
In Figure 43 we set $a = 0.5$ and $\varepsilon = 0.05$. The first three BPs from the homogeneous branch $(u, v) = (\lambda, 1/\lambda)$ are then very close together, and we use the same trick as in §6.4 and simply compute a number of eigenvectors to small eigenvalues at the first BP, and then use `gentau` for predictors for bifurcating branches. Figure 43(a) shows three branches thus obtained, and three stable sample solutions (on $\Omega_{1.2}$). In (b) we continue the solution `b1/pt10` in the 'base radius' a, yielding the branch `b1a`.[11] Larger a also corresponds to a larger surface area of the cone, and associated with this we find

[9] For $e = 1$, to fix the rotational invariance we can include phase conditions in a straightforward way as in §6.1–§6.3. However, as we shall compute on rather large domains (small ε, see (102)), where again we already have a large number of patterns, here we simplify the numerics slightly by breaking the rotational invariance.

[10] This is due to the non–planar base of the cone; for a genuine circular cone with $a = 0.2$ the rings are stable for all $\lambda > 2.8$, see `cmds1b.m`.

[11] Typically, such continuation in a yields folds and the branches do not extend to significantly smaller or larger a,

(a) Primary bifurcations and sample solutions



(b) continuation in a and sample solutions

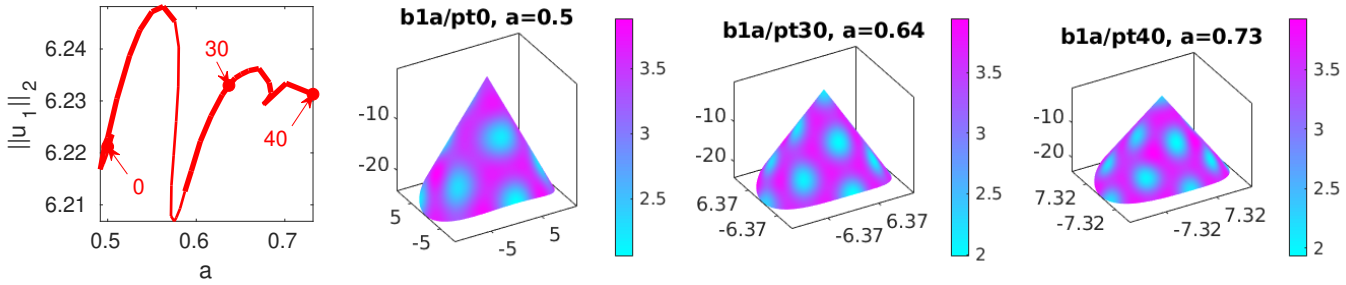


Figure 43: (98) on the cone associated to $\Omega_{1.2}$, $\varepsilon = 0.05$, base radius $a = 0.5$ in (a). (a) Three branches, b1 (red), b2 (magenta), and b3 (blue), bifurcating near the primary instabiliy of the homogeneous branch, and stable sample solutions. (b) continuation of the sample solution `b1/pt10` in a, with sample plots over the associated cone(s).

the growth of additional spots on `b1a`.

6.6 A coupled problem on a cylinder with a lid: Demo `accyl`

As an example how to patch together two surfaces, or, more generally, two problems defined on two domains, we consider the coupled problem

$$0 = G_1(u_1) := -c\Delta_{C_R} u_1 - \lambda u_1 - u_1^3 + \gamma u_1^5 \quad \text{in} \quad \Omega_1 = \{\tilde{x} \in \mathbb{R}^3 : \tilde{x}_1^2 + \tilde{x}_2^2 = R^2, |\tilde{x}_3| \le l_y\}, \tag{103a}$$

$$0 = G_2(u_2) := -c\Delta_R u_2 \quad \text{in} \quad \Omega_2 = \{\tilde{x} \in \mathbb{R}^3 : \tilde{x}_1^2 + \tilde{x}_2^2 \le R^2, \tilde{x}_3 = l_y\}, \tag{103b}$$

with BCs given by

$$u_2 = u_1 \text{ on } \Gamma_1 = \partial\Omega_2 \cap \partial\Omega_1, \quad \partial_n u_1 = 0 \text{ on } \partial\Omega_1, \tag{103c}$$

where n is the outer normal to the surface, i.e., here $n = (0, 0, 1)$. As Ω_1 is a cylinder with radius R and height $2l_y$, we use the parametrization

$$\Omega_1 = \phi(Q_1),\ Q_1 = (-\pi, \pi) \times (-l_y, l_y), \quad \phi(x, y) = (R\cos x, R\sin x, y).$$

but rather yield closed loops. In particular, this happens if we start from solutions at $a = 0.2$ from Fig. 42, but for instance `b1a` extends to larger a. See also the a–continuation of the ring branch in `cmds1b.m`, which extends to large a, with interesting changes of stability as further rings are added as a increases.

Then Δ_{C_R} in (103a) is given by $\Delta_{C_R}u_1(x,y) = \frac{1}{R^2}\partial_x^2 u(x,y) + \partial_y^2 u(x,y)$, and $\partial_n u_1 = \partial_y u_1$. For the disk Ω_2 as the top lid of the cylinder we choose

$$\Omega_2 = \psi(D),\ D = \{x^2+y^2<1\},\ \psi(x,y) = (Rx, Ry, l_y),\ \text{such that } \Delta_R u_2 = \frac{1}{R^2}(\partial_x^2 + \partial_y^2)u_2. \quad (104)$$

The BCs (103c) on Γ_1 can be classified as 'Kirchhoff BCs', i.e., of type

$$A\begin{pmatrix} u_1 \\ \partial_n u_1 \end{pmatrix} = B\begin{pmatrix} u_2 \\ \partial_n u_2 \end{pmatrix} \text{ with } A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } B = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}. \quad (105)$$

Essentially, $u_1|_{\Gamma_1}$ gives the boundary values for the Poisson problem (103b) for u_2. However, we do not claim any physical significance for (103), but rather use it as an example to illustrate how to set up a coupling via BCs of two otherwise unrelated PDEs, which live over different domains. This is different from the systems considered so far, where u_1, u_2 were components of a vector valued function $u = (u_1, u_2)$ over a common domain. One motivation is that problems of type (103) may occur when one wants to patch together charts for a surface. Also note that in (103a) we now have a cubic-quintic AC equation instead of the quadratic-cubic case in §6.1.

Figure 44 shows some results for (103), and Table 8 gives an overview of the files used, including some local modifications of standard `pde2path` library functions. The main ideas of the implementation are:

1. We first define the cylinder (mantle) as a standard PDE object `p.pdeo=stanpdeo2D(lx,ly,nx,ny)`, $l_x = l_y = \pi$, `nx,ny`=numbers of discretization points in x and y, and switch on the periodic BCs in x via `p=box2per(p,1)`; see `acinit.m`.
2. We then identify the top boundary Γ_1 (its (x,y) values `x1` and its point indices `i1` in the grid for Ω_1) via the method `[x1,i1]=pde.grid.bdseg(3)` from `grid2D`, and use `t=x1(1,1:end)` and `geo=polygong(cos(t),sin(t))` to generate (the geometry for) a discretization for the top disk, i.e., a *second* PDE object `p.p2`.
3. To conveniently encode the PDEs (103) and Jacobians (see `sG.m` and `sGjac`), and specifically the coupling conditions $u_1|_{\Gamma_1} = u_2|_{\Gamma_2}$, we store the indices `p.i1` and `p.i2` of the points on Γ_1 for both meshes, such that the associated solution components are `u1(i1(:))` and `u2(i2(:))`, respectively. More specifically, let n_i, n_p and n_{p2} be the number of points on the interface Γ_1, and in the meshes for Ω_1 and Ω_2, respectively. We then generate matrices `p.Q2` $\in \{0,1\}^{n_{p2}\times n_i}$, `p.S1` $\in \{0,1\}^{n_i\times n_p}$, and `p.S2` $\in \{0,1\}^{n_i\times n_{p2}}$ such that $\tilde{u}_j = S_j u_j = u_j|_{\Gamma_1}$ and such that $Q_2\tilde{u}_2(x) = \tilde{u}_2(x)$ if $x \in \Gamma_1$, 0 else.
4. In `acinit`, it remains to initialize `u=[u1;u2;pars]` in a standard way (with $u_1 \equiv 0$ and $u_2 \equiv 0$). Additionally, since Ω_1 is periodic (in x) but Ω_2 is not, it is useful to remember `p.nu1` (compared to `p.np`) as the number of actual unknowns in Ω_1.
5. With these preparations we can set up `oosetfemops`, see Listing 22. Essentially, here we first set up the matrices `Kphi` and `Kz` corresponding to $-\partial_\phi^2$ and $-\partial_z^2$ in Ω_1 to later build the LBO $-\Delta_{C_R}$ = `Kphi/R^2 + Kz^2`, and `Dphi`= ∂_ϕ to implement a PC similar to (35) on Ω_1. Afterwards, we assemble the stiffness and mass matrices `p.mat.K2` and `p.mat.M2` on Ω_2, and finally put together the total mass matrix `p.mat.M`.
6. The main issue in `sG` is the coupling $u_1 - u_2 = 0$ on Γ_1, for which we use a stiff spring approximation [RU19]. We identify the boundary nodes of u_1 and u_2 via `p.S1` and `p.S2` and add `p.sf*p.Q2*(p.S1*p.mat.fill*u1-p.S2*u2)` to the rhs of (103b). This can easily be differenti-

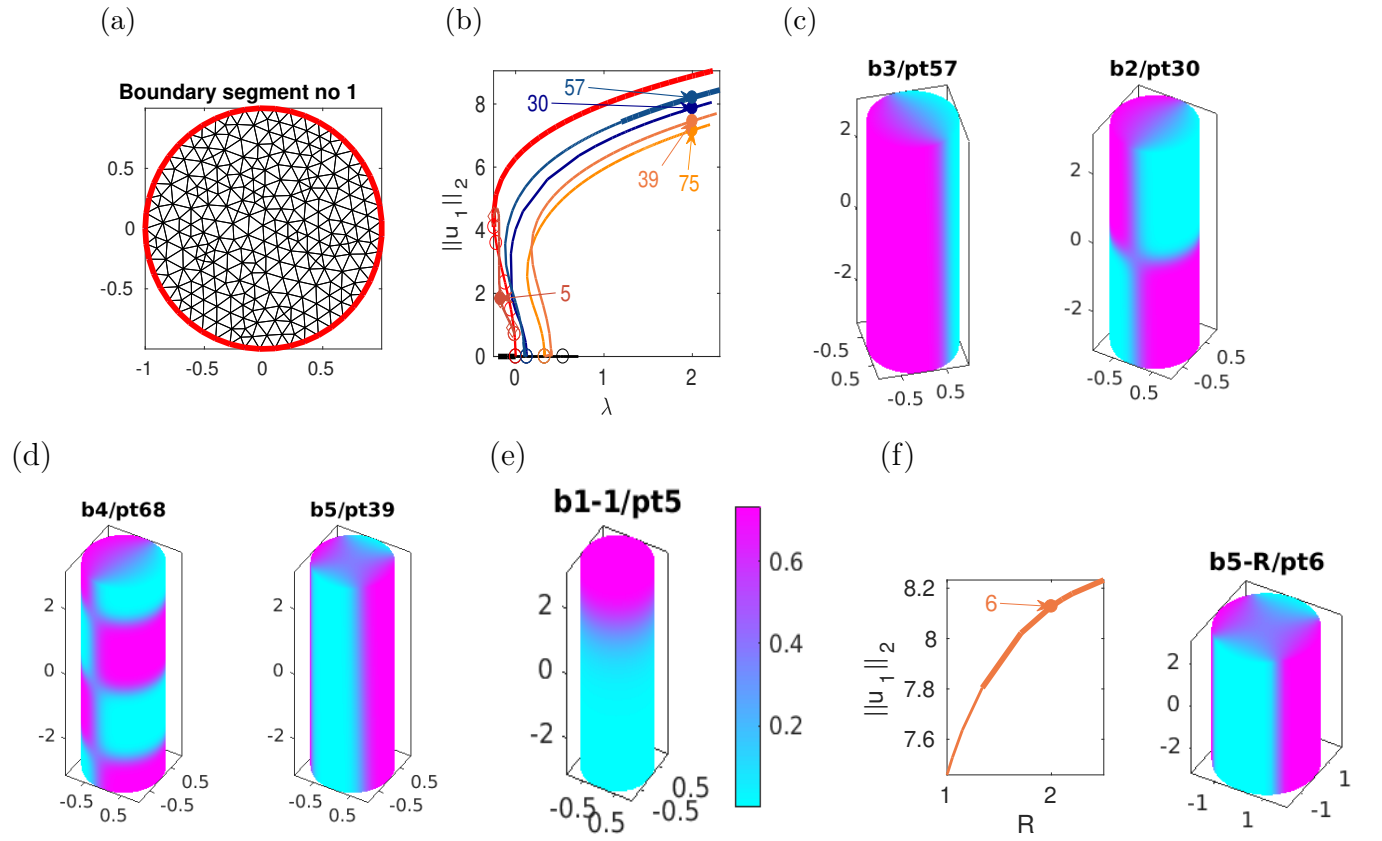


Figure 44: Results for (103), $(c, \gamma, l_y) = (0.1, 1, \pi)$ throughout, discretization with 4800 (cylinder Ω_1) + 500 (top lid Ω_2) triangles. (a) shows the mesh of the top disk, and its boundary, plotted via `[x2,i2]=p.p2.grid.bdseg(1)`. (b) shows a basic bifurcation diagram, with five primary branches (`b1` (red), `b2` (light blue), `b3` (dark blue), `b4` (light yellow), `b5` (dark yellow)) and one secondary branch `b1-1` (brown). (c–e) show associated solution plots, where in (c,d) the color scale is roughly -1.3 to 1.3. (f) shows an example continuation in R.

ated with respect to `u1` and `u2` and accordingly be put into `sGjac.m`.

7. In the script file `cmds.m` we start with continuation of the trivial branch $u = (u_1, u_2) = (0, 0)$ in λ. The first bifurcation at $\lambda = 0$ is simple, and yields a spatially constant branch $(u_1, u_2) = (\alpha, \alpha)$, which becomes stable at the fold. There are a number of BPs up to the fold, and we follow one of these, see `b1-1` in Fig. 44(e). For $(R, l_y) = (\pi, \pi)$, after the first bifurcation there are many close together bifurcation points $(\lambda_j, 0)$ with $\lambda_j > 0$ on the trivial branch, and we use the same trick as in `schnaktor`. I.e., we simple localize some of the BPs using `bifcheck=2`, then use `cswibra` with `aux.besw=0` to only compute nearby kernel vectors, which we then use as (approximate) predictors. If the solutions depend on x on Ω_1, then we need to set a phase condition as in (35), see `qf`, `qfder`. In these continuations we switch off bifurcation detection, but note that a large number of further bifurcations occur. In particular, `b3` (with angular wave number 1) stabilizes at a certain amplitude, i.e., near $\lambda = 1$.
8. At the end of `cmds.m` we also do one continuation in R, which exemplarily shows that many patterned solutions stabilize at large R, as expected from the standard AC equation [RU19].

```
function p=oosetfemops(p) % AC on cyl with lid; coupled problem, 2 fields u1,u2
gr=p.pdeo.grid; fem=p.pdeo.fem; n=p.np;
[Kphi,M,~]=fem.assema(gr,[ones(n,1);0*ones(n,1)],1,1);
[Kz,~,~]=fem.assema(gr,[0*ones(n,1); ones(n,1)],0,1);
p.nu=p.nu1; % set nu to nu1 (dim of first field) for filltrafo
```

Table 8: Main scripts and functions in `accyl/`; at the bottom we list local modifications of library functions to use `p.nu1` instead of the standard `p.nu`.

script/function	purpose, remarks
cmds	main script, essentially yielding Fig. 44.
acinit	initialization, here setting up *two* PDE objects `pde` and `p2`, containing meshes etc for the two domains.
oosetfemops	set two sets of FEM matrices `M,K` etc, and coupling matrices.
sG,sGjac	rhs and Jacobian
qf,qfder	phase condition (and Jac) for rotational invariance
userplot	plotting cylinders as in Fig. 44, called by `plotsol` due to `p.plot.pstyle=-1`
box2per, stanbra	mods of respective library functions `box2per`

```
p.mat.Kphi=filltrafo(p,Kphi); p.mat.Kz=filltrafo(p,Kz);
p.mat.M0=M; p.mat.M1=filltrafo(p,M);
Dphi=convection(fem,gr,[1;0]); p.mat.Dphi=filltrafo(p,Dphi); % for PC
p.nu=p.nu1+p.np2; % reset nu to total number of unknowns
gr=p.p2.grid; fem=p.p2.fem; % assemble matrices for 2nd field
[K,M,~]=fem.assema(gr,1,1,1); p.mat.K2=K; p.mat.M2=M;
% assemble full mass M needed for spcalc
p.mat.M=[p.mat.M1 sparse(p.nu1,p.np2); sparse(p.np2,p.nu1) M];
```

Listing 22: `oosetfemops.m` from `pftut/accyl` with some non standard steps to deal with the different dimensions (and properties) of the two fields `u1` and `u2`. See 5. in the above comments.

6.7 Bulk–surface coupling in a model for cell polarization: demo `cpol`

In [CEKM+19], a model for so-called cell polarization is set up and studied analytically and numerically by time–integration using a bulk–surface FEM. The model consists of (fast) linear diffusion of inactive GTPase (variable w) in the 3D bulk Ω of the cell (the cytosol), coupled a reaction–diffusion equation for the active GTPase (variable u) on the surface $\Gamma = \partial\Omega$ (the cell membrane). After nondimensionalzation it reads

$$\partial_t u = \varepsilon\Delta_\Gamma u + \frac{1}{\varepsilon} f(u,w), \quad x \in \Gamma, \tag{106a}$$

$$\partial_t w = \frac{1}{\varepsilon}\Delta w, \qquad x \in \Omega, \tag{106b}$$

$$\partial_n w = -f(u,w), \qquad x \in \Gamma, \tag{106c}$$

where $f(u,w) := (k_0 + \frac{\gamma u^2}{1+u^2})w - u$, with $k_0, \gamma > 0$ and $0 < \varepsilon = \sqrt{D_u/D_w} \ll 1$ a small parameter such that ε^2 is the ratio of the diffusion constants of u and w. In the limit $\varepsilon \to 0$, a scalar equation for u can be formally derived, but here we shall study the full coupled problem for small but finite ε, choosing $\Omega = B_1(0)$. Cell polarization in this and similar models (see Remark 6.3) roughly means that u concentrates in some part of Γ, and w concentrates in some part of Ω.

Importantly, (106) conserves the mass

$$m = \int_\Gamma u \,\mathrm{d}\Gamma + \int_\Omega w \,\mathrm{d}x, \tag{107}$$

and m turns out to be a convenient bifurcation parameter for steady states of (106), like in the

Cahn–Hilliard equation in §5.2. We thus write the steady state problem for (106) as

$$0 = \varepsilon\Delta_\Gamma u + \frac{1}{\varepsilon} f(u,w) + \lambda_m + s\partial_\phi u, \qquad x \in \Gamma, \tag{108a}$$

$$0 = \frac{1}{\varepsilon}\Delta w + \lambda_m, \qquad x \in \Omega, \tag{108b}$$

$$\partial_n w = -f(u,w), \qquad x \in \Gamma, \tag{108c}$$

where λ_m acts as a Lagrange multiplier for the mass–conservation, and we also introduced a rotational wave–speed s to deal with the rotational invariance (in x_1 =azimuth) of the sphere. In contrast to the CH problem in §5.2, (106) is not variational, and thus the λ_m in (108) are somewhat formal Lagrange multipliers corresponding to the functional $\lambda_m(\int_\Gamma u\,\mathrm{d}\Gamma + \int_\Omega w\,\mathrm{d}x - m)$. Nevertheless, like in the KS equation they work efficiently to fulfill (107), and $|\lambda_m| < 10^{-7}$ for all our continuations below.

Provided that $8k_0 < \gamma$ the reaction $f(u,\overline{w})$ shows bistability for $\overline{w} \in (w_1, w_2)$, see Fig. 45, i.e., $f(u,\overline{w})$ has three zeros u, which together with a sufficiently small ε is the crucial ingredient for pattern formation (i.e., cell polarization) for (106).

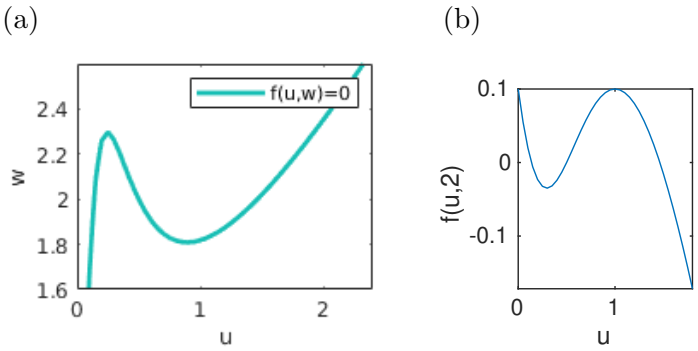


Figure 45: Bistability for f with $k_0 = 0.05$ and $\gamma = 1$, for which $w_1 \approx 1.8$ and $w_2 \approx 2.3$.

Remark 6.3. Problems similar to (106) are also studied in, e.g., [RR14, GKRR16, MC16, NRV19]. In [RR14], (fast) linear diffusion for w in the bulk Ω is coupled to a RD *system* for species u, v on the surface $\Gamma = \partial\Omega$, and a number of interesting results are derived analytically and numerically, indicating that also there cell polarization is triggered by D_w being much larger than $D_u \approx D_v$, i.e., $0 < \varepsilon^2 = D_w/D_u \ll 1$. In particular, [NRV19] contains a detailed asymptotic analysis of this problem, reducing the system to a scalar obstacle problem in the limit $\varepsilon \to 0$, and obtaining conditions for cell polarization to occur. In [GKRR16] a Cahn–Hilliard like phase–field model is studied, and [MC16] coupleds a RD system for $w = (w_1, w_2)$ in the bulk to a RD system for (u, v) on the surface, focussing on patterns generated by a Turing instability in Ω or on Γ.

The numerical analysis in [RR14, GKRR16, MC16] (and also in [CEKM$^+$19]) then proceeds by time–integration, where [RR14, GKRR16] use a diffusive interface approach to approximate the surface diffusion, while [MC16, CEKM$^+$19] constructs compatible meshes for Ω and Γ. However, no (numerical or analytical) bifurcation analysis seems to exist for any of these systems. ⌋

In Fig. 46 we show a simple bifurcation diagram for (106) with (107), base parameters

$$(R, k_0, \gamma, \varepsilon) = (1, 0.1, 1, 0.1), \tag{109}$$

using m as the primary bifurcation parameter, and λ_m as the Lagrange parameter for $n_q = 1$ due to (107). Table 9 comments on the files for the implementation. Concerning the spatial discretizations,

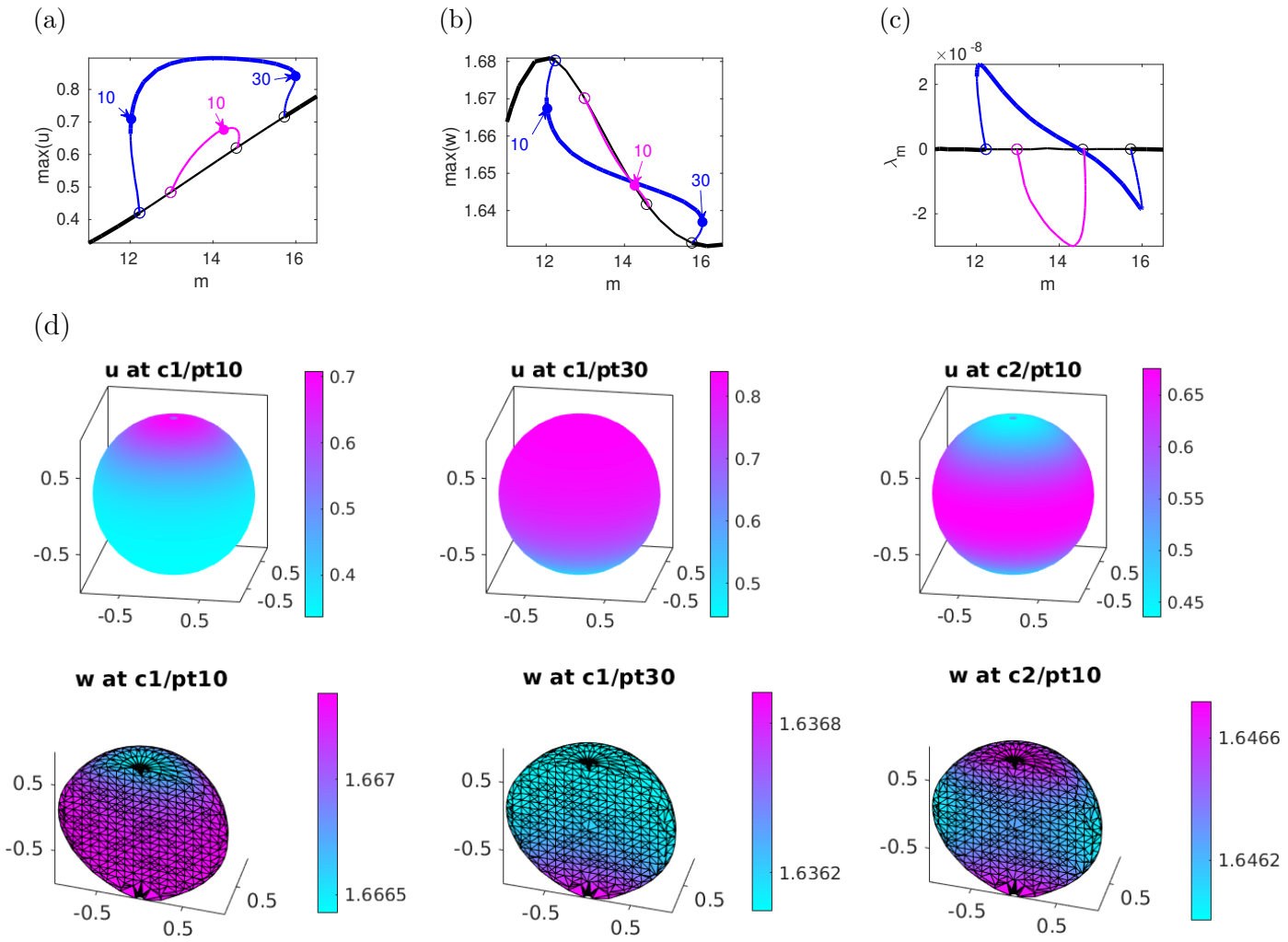


Figure 46: Results for (106); $(R, k_0, \gamma, \varepsilon) = (1, 0.1, 1, 0.1)$. (a)–(c) Bifurcation diagrams, homogeneous (black), primary (blue) and secondary (magenta) patterns; (c) shows the Lagrange multiplier λ_m for mass conservation. (d) Sample solutions, where the cut-away plots (`pstyle=4`) of w are for the cutting plane $y = 0.01$.

we proceed similarly to [MC16, CEKM+19]: like in §6.6 for the case of two surfaces, we generate two compatible meshes and FEM spaces, one for Ω and one for Γ. As we restrict to $\Omega = B_1(0)$ and hence $\Gamma = S^1$, we reuse the `sppdeo` for Γ such that we have a natural Laplace Beltrami operator Δ_Γ on the preimage of Γ. Then we generate the mesh for Ω such that the boundary points for Ω are the mesh points for Γ (plus two mesh points for the north pole and the south pole, respectively). This can be generalized to other domains Ω, and is a convenient method as long as we have a good parametrization of Γ (or of patches of Γ) and associated LBOs Δ_Γ on the preimages. Moreover, on suitable domains it easily generalizes to models as in [RR14, GKRR16, MC16] consisting of RD systems in the bulk and/or on the surface.

Thus, in `cpinit` we start with `sppdeo` to generate a 2D pde-object `p.pdeo` with mesh (x_i, y_i) of `p.nus` points for the preimage

$$Q = (-\pi, \pi) \times (-\pi/2 + \delta, \pi - \delta) = \phi^{-1}(\Gamma)$$

of the sphere Γ without the poles $N = (0, 0, 1)$ and $S = (0, 0, -1)$, cf. §6.2 and Fig. 35(a). Then we map

the grid-points from Q to Γ, extend these by N and S, then extend these further by points in $B_{1-h_0/2}(0)$ using `distmesh`, and thus generate a PDE object `p.p2` for Ω, where h_0 is a parameter for the mesh–width in the 3D mesh. The first `p.nus+2` points in the mesh for Ω then are $(\phi(x_i, y_i)_{i=1,\mathtt{p.nus}}, N, S)$, and we refer to `cpinit.m`, `oosetfemops.m` and `sGcp.m` for details and comments how to further set up and use the coupling matrices for the Robin BCs 108(c). We need to locally modify some library functions to deal with the two meshes, and in `userplot` we essentially shift `p.p2` to `p.pdeo` to enable the `cutplots` in Fig. 46(e).

Table 9: Scripts and functions in `cpol`; at the bottom are local modifications of library functions.

script/function	purpose, remarks
cpcmds1	main script, essentially yielding Fig. 46. `cpcmds2` runs on a coarser mesh, shows the structure of Jacobian, and explains the use of the rotational phase condition.
cpinit	initialization, setting up *two* PDE objects `pde` and `p2`.
sphere2ballpdeo	convenience function to generate a mesh for Ω compatible with the one for Γ; based on `hudistmesh`
oosetfemops	set two sets of FEM matrices `M,K` etc, and coupling matrices.
cpsG,cpsGjac,cpbra	rhs and Jacobian, branch-output
qfm,qfmder	aux.eqn and derivative for mass conservation (107)
qf2,qf2der	for augmenting qfm, qfmder by the rotational phase condition on certain branches
userplot	producing plots as in Fig. 46, i.e., u on Γ and $w \in \Omega$ using a cutplot, see also `mapogrid`
box2per, getGupde	small modifications of respective library functions to deal with the two meshes present here, for instance the special sparsity structure of Jacobians in getGupde

The results in Fig. 46 illustrate some main features of (106), extending results from time integration from [CEKM$^+$19]. There is a unique homogeneous steady state $U_{\rm hom}(m)$ (other parameters fixed), which is stable at small and large m. For sufficiently small ε, increasing m from low values we find a subcritical pitchfork at $m = m_1 \approx 12.23$ to a polarized state oriented along the N–S–axis (blue branch in Fig. 46(a). The kernel at bifurcation is three–dimensional including two other orientations, but the $N - S$–orientation can be continued without further phase condition due to the surface mesh.[12] This polarized state becomes stable after a fold at $m \approx 12.01$, looses stability at a second fold at $m = m_1 \approx 15.99$, and then returns to $U_{\rm hom}$ in another subcritical pitchfork at $\tilde m_1 \approx 15.72$. For the parameters (109), in particular $\varepsilon = 0.1$, there are two more bifurcation points $m_2 < m_3$ on $U_{\rm hom}$ close to $m = 12.98$: At m_2 the kernel is three-dimensional, and Fig. 46(f) illustrates the bifurcating branch with N–S orientation, while at m_3 we have a doube bifurcation point with a genuine angular dependence, and these bifurcating branches again return to $U_{\rm hom}$ at $\tilde m_3 < \tilde m_2$ near $m = 14.59$.

Similarly, for smaller ε we find several more branches bifurcating at some $m_i(\varepsilon)$ with $m_1(\varepsilon) < m_i(\varepsilon) < \tilde m_i(\varepsilon) < \tilde m_1(\varepsilon)$ with higher wave numbers, which again form loops with between $m_i(\varepsilon)$ and $\tilde m_i(\varepsilon)$. However, they are all unstable, and hence we content ourselves with Fig. 46.

[12] See `cpcmds2` for using `cswibra` to obtain and continue the same branch in different orientations.

7 Demos `bruosc` and `bruosc-tpf`: Oscillating Turing patterns

So far we restricted to steady patterns. We now consider, rather closely following [YZE04], oscillating Turing patterns in the Brusselator model

$$\partial_t u_1 = a - (b+1)u_1 + u_1^2 u_2 + \alpha\cos(2\pi\beta t) + D_u\Delta u_1, \tag{110a}$$
$$\partial_t u_2 = bu_1 - u_1^2 u_2 + D_v\Delta u_2, \tag{110b}$$

with parameters $a, b, D_u, D_v, \alpha, \beta$, where $\alpha\cos(2\pi\beta t)$ describes a time–periodic forcing. First, however, we focus on the autonomous case $\alpha = 0$, in which (110) has the spatially homogeneous steady solution

$$u^* = (u_1, u_2)^* \equiv (a, b/a). \tag{111}$$

(a) Hopf and Turing lines (b) Dispersion relations; left to right corresponding to labels 1 to 4 in (a).

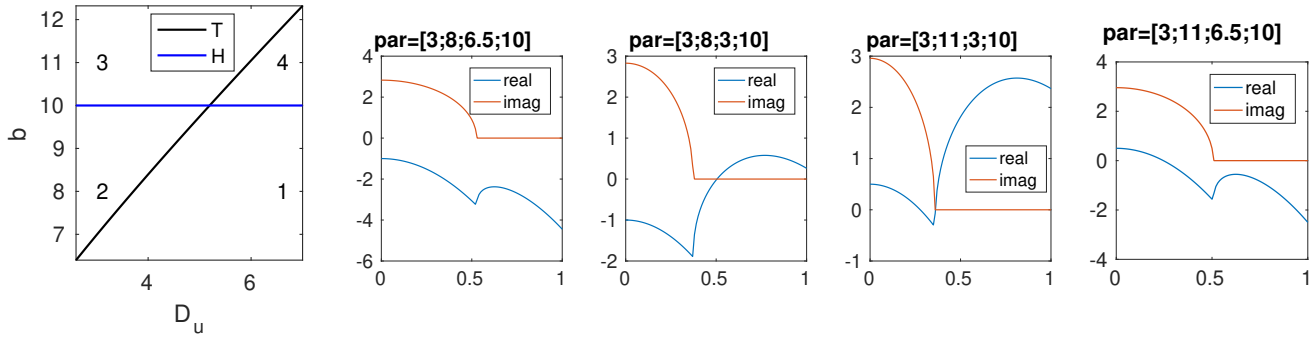


Figure 47: Phase diagram for fixed $a = 3$ and $D_v = 10$, and sketches of dispersion relations $\mu(k;\mathtt{par})$ with parameters $\mathtt{par} = (a, b, D_u, D_v)$ as indicated and wave number k horizontally.

The dispersion relation for the linearization of (110) around (111) can be studied analytically. For instance, fixing a and D_v we obtain a phase diagram in the D_u, b plane as shown in Fig. 47(a), where the Hopf and Turing lines are given by

$$b^H = 1 + a^2, \quad b^T = (1 + a\sqrt{D_u/D_v})^2, \tag{112}$$

respectively. In Fig. 47(b), the dispersion plots are obtained from `spufu` as in Fig. 19, and the Turing line in (a) can also be obtained from branch point continuation as in §5, and the Hopf line from a Hopf point continuation which works via an analogous extended system, see [Uec20a]. These commands are in `bruosc/auxcmds1`, with the pertinent extended Jacobians in `bpjac.m` and `hpjac.m`, as usual.

If for instance we cross the Turing line by decreasing d_u from sector 1 to 2, then we find a Turing bifurcation with wave number $k_c = \sqrt{a/\sqrt{D_u D_v}}$, resp. wave vector k with $|k| = k_c$. In this sense we have a completely analogous situation as for the SH equation in §3 or the Schnakenberg system (which is obviously quite related to (110)) in §4, and obtain analogous steady patterns. See, e.g., [UW20] for (localized) BCCs in (110), similar to §3.5. Moreover, a three–component version of (110) (but without periodic forcing) was already used as a demo problem for Hopf bifurcations and Turing–Hopf bifurcations in [Uec19a, Uec20a], following [YDZE02].

In §7.1 we focus on oscillating (time–periodic) solutions for (110) with $\alpha = 0$ as we cross the Hopf line from sector 1 to 4 in Fig. 47(a). The primary bifurcation then is to a spatially homogeneous Hopf orbit, and oscillating spatial patterns are afterwards created in (temporal) period doubling bifurcations

from the homogeneous Hopf orbit. In §7.2 we explain a trick how to add the time periodic forcing ($\alpha > 0$) to (110), which will yield subcritical oscillating Turing patterns. Additionally, we remark that [UW20] considers (localized) 3D Turing patterns for (110) with $\alpha = 0$, and that [Uec20a] also contains some examples of an alternate setup for non–autonomous problems.

7.1 The autonomous case

Hopf bifurcation means the bifurcation of time periodic orbits, Hopf orbits in short, from steady solutions, due to complex conjugate eigenvalues crossing the imaginary axis with non-zero imaginary parts and non–zero speeds. The stability of a Hopf–orbit is determined by its Floquet multipliers. There always is the trivial multiplier $\gamma = 1$, and if all other multipliers are inside the unit disk, then the orbit is (orbitally) stable. See [Uec19a, Uec20a] and the references therein for basics of Hopf bifurcations and in particular the implementation in `pde2path`.

Bifurcations from Hopf orbits can occur if multipliers cross the unit circle, and currently `pde2path` can deal with

- Hopf pitchforks or transcritical bifurcations, associated to a multiplier going through 1;
- Period doubling (PD) bifurcations, associated to a multiplier going through -1.

The case of complex conjugate multipliers crossing the unit circle elsewhere is called Neimark-Sacker case, and numerically expensive even in low dimensional ODEs. A multiplier also passes through 1 at non–degenerate folds of Hopf-orbits, but like for folds of steady states we do not take special care of this.

Table 10: Scripts and functions in `bruosc`.

script/function	purpose, remarks
cmds1d,cmds2d	scripts for 1D and 2D, yielding Figs. 48 and 49.
auxcmds1	script for Fig. 47, including Hopf and branch point continuation
bruinit, oosetfemops	initialization, and setting of FEM matrices
sG,nodalf,sGjac,njac	encode G and $\partial_u G$
bpjac, hpjac	Jacobians for BP and HP continuation
spufu, myhoplot	mod of stanufu for the disp. relations in Fig. 47, and convenience mod of hoplot

Table 10 lists the files in `bruosc`. The setup of `sG, sGjac`, etc is as usual, and we can concentrate on the main scripts `cmds1d` and `cmds2d`. As already said, the basic setups and algorithms to deal with Hopf bifurcations and Hopf orbits are described in [Uec19a, Uec20a]. One key point is the use of `p.sw.bifcheck=2`, and often also the initialization of spectral shifts as guesses for the crossing of Hopf eigenvalues. In `pde2path`, this can be done with `initeig`, see Cell1 of `cmds1d.m` in Listing 23. Moreover, if one expects Hopf bifurcations we recommend `hobra` as the standard setting for branch output. For steady solutions, this is analogous to `stanbra`, but after a `hoswibra` to a Hopf orbit it takes the data from `p.hopf.y` (the periodic orbits). The default norm in `hobra` thus is

$$\|u\| := \begin{cases} \|u\|_{L^2(\Omega\times(0,T))}/\sqrt{T|\Omega|} & \text{if } T > 0 \text{ (genuine Hopf orbit)}, \\ \|u\|_{L^2(\Omega)}/\sqrt{|\Omega|} & \text{if } T = 0 \text{ (steady solution)}, \end{cases} \tag{113}$$

where T is the time–period and $|\Omega|$ the volume of the domain.

In Cell 1 of `cmds1d` we initialize (110) in sector 1 in Fig. 47(a), and then increase b, which yields

a Hopf bifurcation point (HBP) at $b = 10$. The branch switching *to* the bifurcating spatially homogeneous Hopf branch `1dh1` works by `hoswibra`, followed by `cont` as usual. For the Hopf orbits we use `lssbel` (bordered elimination) as the linear system solver, which typically is more efficient than the standard solver `lss` (basically `Matlab`'s \), see [UW17, Uec19a]. On `1dh1` we find a sequence of PD bifurcations, and the branch switching *from* Hopf branches is implemented in `poswibra` (periodic orbit swibra). Subsequently (Cell 4) we plot the BD and example solutions to obtain Fig. 48.

```
%% C1: 2-compo brusselator on 1D domain, increase b to cross Hopf line
p=[]; lx=2*pi/0.7; nx=50; a=3; b=8; du=6; dv=10;
par=[a b du dv]; p=bruinit(p,lx,nx,par); p=setfn(p,'hom1Db');
p=initeig(p,10); p.nc.neig=[4, 4]; % set estimates for Hopf evals crossing
p.nc.ilam=2; p.sol.ds=0.2; p=findbif(p,2); p=cont(p,10);
%% C2: Hopf bif (hom), use lssbel for bordered system solution
figure(2); clf; aux=[]; aux.tl=40; p=hoswibra('hom1Db','hpt1',0.1,4,'1dh1',aux);
p.hopf.bisec=10; p.nc.dsmax=0.2; p=setbel(p,2,1e-4,5,@lss);
p.plot.bpcmp=8; p=cont(p,50);
%% C3: PD bifs to osc. Turing pattern
ds=0.5; aux=[]; aux.sw=-1; p=poswibra('1dh1','bpt1','pd1',ds,aux);
p.sw.bifcheck=1; p.hopf.fltol=1e-3; p.nc.tol=1e-6; p=cont(p,30);
p=poswibra('pd1','bpt2','pd2',ds,aux);  p=cont(p,40);
```

Listing 23: `bruosc/cmds1d`, first 3 cells, the remainder dealing with plotting. See text for comments.

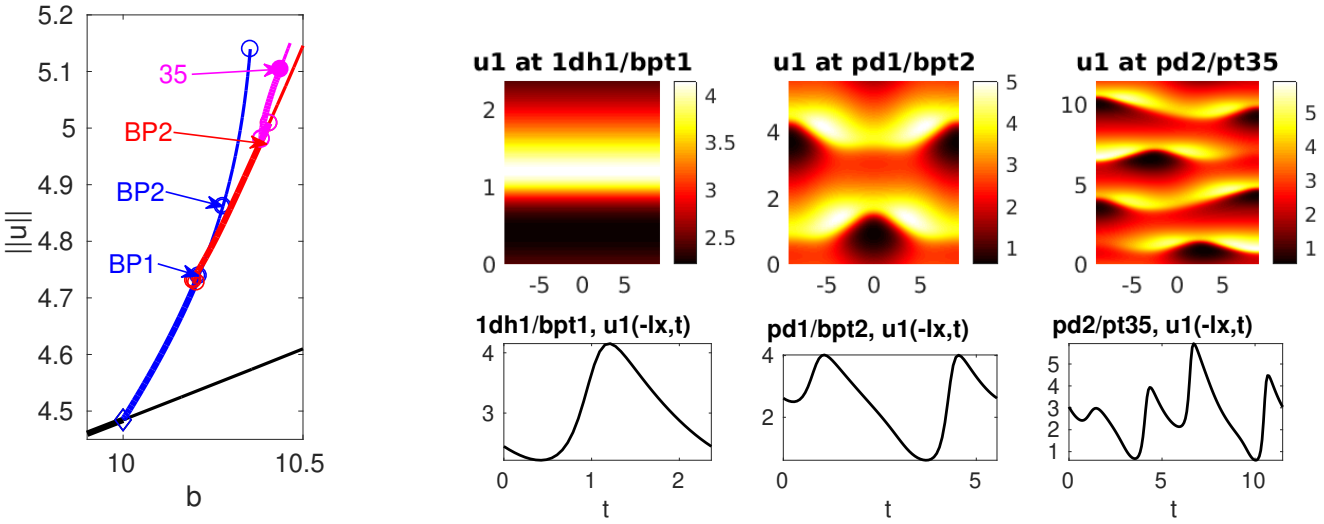
The left to right plots in Fig. 48(b) show example solutions from a sequence of period–doublings which occurs for increasing b. We used $n_p = 51$ spacial points (1D), and $n_t = 30$ temporal points on the primary Hopf orbit, and hence $n_t = 59$ and $n_t = 117$ after the period doublings. The first plot in (c) shows the multipliers at the well localized (see Remark 7.1) first PD point from `1dh1`. The second and third plots in (c) show the multipliers at the rather poorly localized 2nd BPs from `1dh1` and `pd1` (the first BP on `pd1` is a fold). The fourth plot in (c), shortly after `pd2` looses stability, indicates that this loss of stability is due to a Neimark–Sacker bifurcation.

Remark 7.1. The localization of BPs from periodic orbits currently works by a simple bisection for the change of the number of unstable multipliers. This is expensive, and the multipliers often have a rather sensitive dependence on the parameters, but our experience is that typically we do not need a very accurate localization of the BPs for successful branch-switching, and thus we content ourselves with, e.g., the results from Fig. 48. However, see also Cell 3 of Listing (24) for a more careful localization of PD points in 2D. ⌋

Figure 49 shows analogous results for (110) over the 2D square domain $\Omega = (-l_x, l_x)^2$, $l_x = \sqrt{2}\pi/0.7$, with a spatial criss-cross mesh of $n_p = 221$ discretization points. The factor $\sqrt{2}$ is as in Fig. 11 to have squares as the first oscillating Turing mode. The script `cmds2d` works quite analogous to `cmds1d`, with the main difference that now some of the PD branch points have multiplicity two, defined as in the steady case as the dimension of the critical eigenspace. We do not yet have algorithms (like `q(c)swibra` for the steady case) for the systematic treatment of HBPs or POBPs of higher multiplicity. However, similar to `gentau`, the user may select coefficients to choose initial states for the `poswibra` predictor (and hence for the predictor as a function of t) 'by hand', here by passing coefficients for the kernel vectors via `aux.coeff`.

```
p=[]; lx=sqrt(2)*pi/0.7; ly=1*lx; nx=10; a=3; b=8; du=6; dv=10;
par=[a b du dv al om]; sw.sym=2; p=bruinit(p,[lx ly],nx,par,sw);
p=initeig(p,10); p.nc.neig=[4, 4]; p.file.smod=5; p=setfn(p,'hom2Db');
p.nc.ilam=2; p.sol.ds=0.2; p=findbif(p,2); p=cont(p,10);
%% C2: Hopf bif (hom)
figure(2); clf; aux.tl=30; p=hoswibra('hom2Db','hpt1',0.1,4,'2dh1',aux);
```

(a) period doubling BD (b) solution plots, blue branch, red branch, magenta branch (left to right)



(c) Selected Floquet spectra

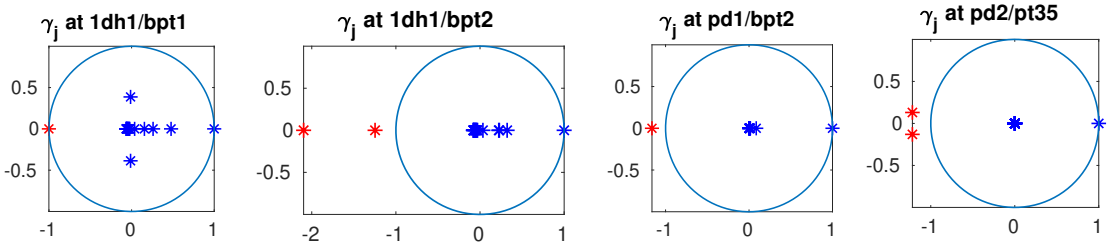


Figure 48: Period doubling bifurcations to oscillating Turing patterns in (110) in 1D, $(a, D_u, D_v) = (3, 6, 10)$, $\Omega = (-l_x, l_x)$, $l_x = 2\pi/0.7$, with Neumann BCs. (a) BD of $u^* = (a, b/a)$ (steady state, black branch), primary Hopf branch (blue), first period doubling (red), secondary period doubling (magenta). The norm in (a) is (113). (b) space (horizontal)–time (vertical) solution plots of u_1 (top row), and associated time–series at left boundary (bottom row). (c) Leading 20 Floquet multipliers at selected solutions.

```
p.hopf.bisec=2; p.nc.dsmax=0.3; p.nc.tol=1e-6; pause;
p.hopf.fltol=1e-2; bw=1; beltol=1e-6; belimax=5; % low accuracy for speed
p=setbel(p,bw,beltol,belimax,@lss); p=cont(p,40);
%% C3: reload and carefully find first 2 period doublings (PDs)
p=loadp('2dh1','pt10');p.sol.ds=0.05;p.nc.dsmax=0.05;p.hopf.bisec=5;p=cont(p);
%% C4: PD bif to primary osc squares
ds=0.5; aux=[]; aux.sw=-1; p=poswibra('2dh1','bpt1','2dpd1',ds,aux);
p.nc.dsmax=0.5; p.hopf.bisec=2; p.nc.tol=1e-6; pause; p=cont(p,30);
%% C5: PD bif at 2nd PDpt (double), kernel=2 squares, compose stripes by hand
aux.sw=-1; aux.coeff=[1 1]; % PD to stripes
p=poswibra('2dh1','bpt2','2dpd2a',ds,aux); p.nc.dsmax=0.5; p=cont(p,30);
aux.sw=-1; aux.coeff=[1 0]; % PD to square
p=poswibra('2dh1','bpt2','2dpd2b',ds,aux); p.nc.dsmax=0.5; p=cont(p,30);
```

Listing 24: `bruosc/cmds2d`, first 5 Cells. After following the primary Hopf branch with large `ds` in C2, in C3 we reload a point and decrease `ds` to localize the first 2 BPs reasonably well. In C4 we deal with the primary PD bifurcation to squares, and in C5 with PD bifurcations at the second BP, which is of multiplicity two. The remainder of `cmds2d.m` deals with plotting.

(a) period doubling BD (b) solution plots, from red, dark blue and magenta branch (top–down)

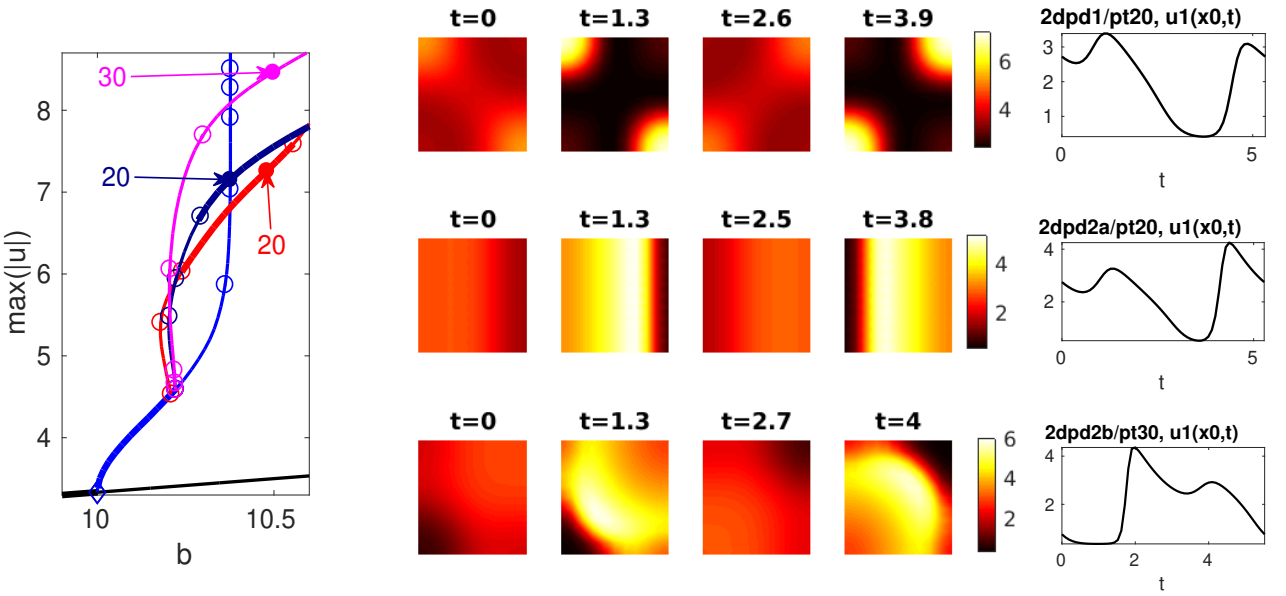


Figure 49: Period doubling to stable squares and other osc. patterns for (110) over $\Omega = (-l_x, l_x)^2$, $l_x = \sqrt{2}\pi/0.7$. (a) BD of $u^* = (a, b/a)$ (steady state, black branch), primary Hopf branch (blue), squares (red), and two branches at the 2nd PD from the blue branch. The stripes branch (dark blue) corresponds to `pd1` from Fig. 48 but with a longer spatial period, while the magenta branch has squares of larger wavelength, which however are unstable.

7.2 Time periodic forcing

Another interesting result from [YZE04] is that stable oscillating Turing patterns can occur under (spatially homogeneous) time periodic forcing $\alpha > 0$ in the range where both, Turing and Hopf modes are subcritical, i.e., the steady state u^* is (exponentially) stable. As an example we consider the point $(D_u, b) = (6, 9.5)$, choose $\beta = 0.42$ as in [YZE04], motivated by $\mathrm{Im}\mu(0) = 2\pi\beta \approx 2.65$ for $(a, b, D_u, D_v) = (3, 10, 6, 10)$, and increase α. The first task is to implement the time–periodic forcing. We do this by augmenting G by a (nonlinear) oscillator (in complex notation) $\dot v = (\delta + 2\pi\beta\mathrm{i})v - |v|^2 v$, which in polar coordinates $v(t) = r(t)\mathrm{e}^{\mathrm{i}\phi(t)}$ for $\delta > 0$ yields the periodic orbit $r = \sqrt{\delta}, \phi = 2\pi\beta t$. The full system thus reads

$$\begin{aligned}
\partial_t u_1 &= a - (b+1)u_1 + u_1^2 u_2 + v_1 + D_u \Delta u_1,\\
\partial_t u_2 &= b u_1 - u_1^2 u_2 + D_v \Delta u_2,\\
\partial_t v_1 &= \delta v_1 - 2\pi\beta v_2 - (v_1^2 + v_2^2)v_1,\\
\partial_t v_2 &= \delta v_2 + 2\pi\beta v_1 - (v_1^2 + v_2^2)v_2,
\end{aligned} \tag{114}$$

i.e., we have the forcing amplitude $\alpha = \sqrt{\delta}$ for $\delta \geq 0$. To put (114) into `pde2path` we proceed similar to §5.2: After creating the FEM mesh and nodal values for u we add two 'virtual nodal values', i.e., set $n_u = n_u + 2$ and append $v = (v_1, v_2)$ to `u`. The pertinent commands in `bruinit.m` are `p.nu=p.nu+2; p.u=[u;v;0;0;par']`. In `oosetfemops` we then extend the left hand side mass matrix `M` accordingly, and we append the v equations in `sGpf`, see Listing 25. Additionally, we need some minor modifications for generating the branch data and plotting (replacing back n_u by $n_u - 2$), and then we can proceed as before.

```
function p=oosetfemops(p) % for brusselator with time periodic forcing
```

```
[p.mat.K,M,~]=p.pdeo.fem.assema(p.pdeo.grid,1,1,1);
p.mat.Ms=M; p.mat.M0=kron([[1,0];[0,1]],M); %
p.mat.M=[[p.mat.M0 zeros(2*p.np,2)]; % extend FEM-M by oscillator M
         [zeros(2,2*p.np) speye(2)]];
```

```
function f=nodalfpf(p,u) % for brusselator with time periodic forcing
u1=u(1:p.np); u2=u(p.np+1:2*p.np); par=u(p.nu+1:end); a=par(1); b=par(2);
v1=u(p.nu-1); % oscillator-v, used for time-per.-forcing
f1=a-(b+1)*u1+u1.^2.*u2+v1; f2=b*u1-u1.^2.*u2; f=[f1; f2];
```

```
function r=sGpf(p,u) % for brusselator with time periodic forcing
f=nodalfpf(p,u); par=u(p.nu+1:end); % a,b,d_u, d_v,al,om
du=par(3); dv=par(4); del=par(5); om=2*pi*par(6);
K=kron([[du,0];[0,dv]],p.mat.K);
v1=u(p.nu-1); v2=u(p.nu);
ru=K*u(1:p.nu-2)-p.mat.M0*f;
va=v1^2+v2^2; rv1=-del*v1+om*v2+va*v1; rv2=-om*v1-del*v2+va*v2; % the osc.eqns
r=[ru;rv1;rv2];
```

Listing 25: `bruosc-tpf/oosetfemops, nodalpf` and `sGpf`. With $n_u = n_u + 2$ (from `bruinit`), the PDE components u_1, u_2 are now at `u(1:p.nu-2)`, and the auxiliary oscillator components v at `u(p.nu-1:p.nu)`.

.

Listing 26 shows the main commands from the script `cmds1d`, and Fig. 50 a selection of results. The first PD bifurcation from the blue branch is subcritical to an oscillating Turing pattern, which becomes stable after the fold. There are further PD bifurcations, but the bifurcating branches are unstable.

```
%% C1: brusselator, 1D, with per.forcing (amplitude sqrt(del) for del>0), init
p=[]; lx=3*pi/0.7; nx=100; a=3; b=9.5; du=6; dv=10; om=0.42; del=-0.1;
par=[a b du dv del om]; p=bruinit(p,lx,nx,par); p.sw.verb=0;
%% C2: increase del to find primary Hopf
p=setfn(p,'hom1D'); p.nc.ilam=5; p.sol.ds=0.1; p=cont(p,3);
%% C3: follow Hopf bif in oscillator (which also forces u)
para=4; ds=0.1; figure(2); clf; aux.tl=30; aux.dlam=0;
p=hoswibra('hom1D','hpt1',0.1,4,'1dh1',aux); p.hopf.bisec=5; p.nc.dsmax=0.15;
p=setbel(p,2,1e-4,5,@lss); p.nc.tol=1e-6;  p.hopf.fltol=1e-3; p=cont(p,50);
%% C4: PD bifs to subcritical osc. Turing pattern
aux.sw=-1; p=poswibra('1dh1','bpt1','pd1',0.5,aux); p=cont(p,40);
```

Listing 26: `cmds1d` (first 4 cells). In C2/C3 we increase δ and follow the oscill. branch for $\delta > 0$ to switch on the periodic forcing and increase its amplitude. We then find secondary period–doubling bifurcations to oscillatory Turing patterns.

References

[AGH+05] M. Alber, T. Glimm, H. G. E. Hentschel, B. Kazmierczak, and S. A. Newman. Stability of n-dimensional patterns in a generalized Turing system: implications for biological pattern formation. *Nonlinearity*, 18(1):125–138, 2005.

[ALB+10] D. Avitabile, D.J.B. Lloyd, J. Burke, E. Knobloch, and B. Sandstede. To snake or not to snake in the planar Swift-Hohenberg equation. *SIAM J. Appl. Dyn. Syst.*, 9(3):704–733, 2010.

[BGUY17] S. Bier, N. Gavish, H. Uecker, and A. Yochelis. Mean field approach to first and second order phase transitions in ionic liquids. *PRE*, 95:060201, 2017.

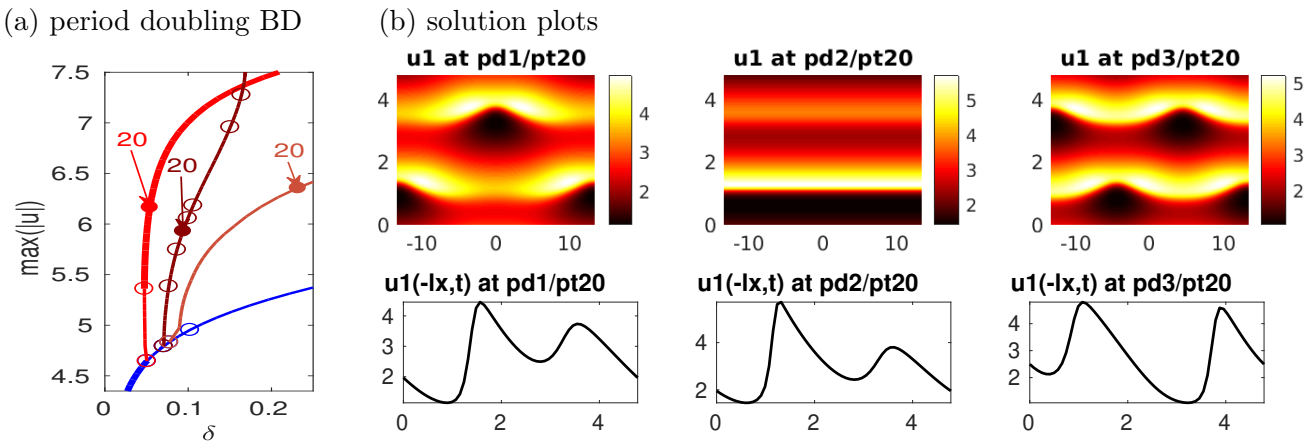


Figure 50: BD under periodic forcing with amplitude $\alpha = \sqrt{\delta}$, $\beta = 0.42$ fixed, $(a, b, D_u, D_v) = (3, 9.5, 6, 10)$. Subcritical oscillating Turing patterns under time-periodic forcing via period doubling bifurcations from the 'natural' forced branch.

[BK06] J. Burke and E. Knobloch. Localized states in the generalized Swift-Hohenberg equation. *Phys. Rev. E*, 73:056211, 2006.

[BKL+09] M. Beck, J. Knobloch, D.J.B. Lloyd, B. Sandstede, and T. Wagenknecht. Snakes, ladders, and isolas of localized patterns. *SIAM J. Math. Anal.*, 41(3):936–972, 2009.

[Bus75] F. Busse. Patterns of convection in spherical shells. *J. Fluid Mech.*, 72:67–85, 1975.

[Cal04] T. K. Callahan. Turing patterns with O(3) symmetry. *Phys. D*, 188(1-2):65–91, 2004.

[CEKM+19] D. Cusseddu, L. Edelstein-Keshet, J. A. Mackenzie, S. Portet, and A. Madzvamuse. A coupled bulk-surface model for cell polarisation. *J. Theoret. Biol.*, 481:119–135, 2019.

[CH93] M.C. Cross and P.C. Hohenberg. Pattern formation outside equilibrium. *Rev. Mod. Phys.*, 65:854–1190, 1993.

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

[CK99] T. K. Callahan and E. Knobloch. Pattern formation in three-dimensional reaction-diffusion systems. *Phys. D*, 132(3):339–362, 1999.

[CK01] T. K. Callahan and E. Knobloch. Long-wavelength instabilities of three-dimensional patterns. *Phys. Rev. E*, 64:036214, 2001.

[CKF18] E.G. Charalampidis, P.G. Kevrekidis, and P.E. Farrell. Computing stationary solutions of the two-dimensional Gross–Pitaevskii equation with deflated continuation. *Communications in Nonlinear Science and Numerical Simulation*, 54:482–499, 2018.

[CKP19] A. Christlieb, N. Kraitzman, and K. Promislow. Competition and complexity in amphiphilic polymer morphology. *Phys. D*, 400:132144, 20, 2019.

[CLM90] P. Chossat, R. Lauterbach, and I. Melbourne. Steady-state bifurcation with O(3)-symmetry. *Arch. Rational Mech. Anal.*, 113(4):313–376, 1990.

[Cra91] J. D. Crawford. Surface waves in nonsquare containers with square symmetry. *PRL*, 67(4):441–445, 1991.

[DF20] Q. Du and X. Feng. The phase field method for geometric moving interfaces and their numerical approximations. In *Handbook of numerical analysis*, volume 21, pages 425–508. Elsevier, 2020.

[DHPW14] A. Doelman, G. Hayrapetyan, K. Promislow, and B. Wetton. Meander and pearling of single-curvature bilayer interfaces in the functionalized Cahn-Hilliard equation. *SIAM J. Math. Anal.*, 46(6):3640–3677, 2014.

[dRGGHP03] H. del Rio Guerra, C. E. Garza-Hume, and P. Padilla. Geodesics, soap bubbles and pattern formation in Riemannian surfaces. *J. Geom. Anal.*, 13(4):595–604, 2003.

[DRUW14] T. Dohnal, J.D.M. Rademacher, H. Uecker, and D. Wetzel. pde2path 2.0. In H. Ecker, A. Steindl, and S. Jakubek, editors, *ENOC 2014 - Proceedings of 8th European Nonlinear Dynamics Conference, ISBN: 978-3-200-03433-4*, 2014.

[DU17] T. Dohnal and H. Uecker. Periodic boundary conditions in pde2path, 2017.

[dW17] H. de Witt. Fold and branch point continuation in pde2path – a tutorial for systems, 2017.

[dW19] H. de Witt. Beyond all order asymptotics for homoclinic snaking in a Schnakenberg system. *Nonlinearity*, 32:2667–2693, 2019.

[Eck65] W. Eckhaus. *Studies in Non–Linear Stability Theory.* Springer Tracts in Nat. Phil. Vol.6, 1965.

[EGU+19] S. Engelnkemper, S. V. Gurevich, H. Uecker, D. Wetzel, and U. Thiele. Continuation for thin film hydrodynamics and related scalar problems. In *Computational Modeling of Bifurcations and Instabilities in Fluid Mechanics*, pages 459–501. Springer, 2019.

[Ell89] C. M. Elliott. The Cahn-Hilliard model for the kinetics of phase separation. In *Mathematical models for phase change problems (Óbidos, 1988)*, volume 88 of *Internat. Ser. Numer. Math.*, pages 35–73. Birkhäuser, Basel, 1989.

[Erm91] B. Ermentrout. Stripes or spots? Nonlinear effects in bifurcation of reaction-diffusion equations on the square. *Proc. R. Soc. Lond., Ser. A*, 434(1891):413–417, 1991.

[FBB16] P. E. Farrell, C. H. L. Beentjes, and A. Birkisson. The computation of disconnected bifurcation diagrams, arxiv:1603.00809, 2016.

[FBF15] P. E. Farrell, Á. Birkisson, and S. W. Funke. Deflation techniques for finding distinct solutions of nonlinear partial differential equations. *SIAM J. Sci. Comput.*, 37(4):A2026–A2045, 2015.

[FCS07] W. J. Firth, L. Columbo, and A. J. Scroggie. Proposed resolution of theory-experiment discrepancy in homoclinic snaking. *PRL*, 99, 2007.

[GHP03] C. E. Garza-Hume and P. Padilla. Closed geodesics on oval surfaces and pattern formation. *Comm. Anal. Geom.*, 11(2):223–233, 2003.

[GHY11] N. Gavish, K. Hayrapetyan, G.and Promislow, and Li Yang. Curvature driven flow of bi–layer interfaces. *Phys. D*, 240:675–693, 2011.

[GKRR16] H. Garcke, J. Kampmann, A. Rätz, and M. Röger. A coupled surface-Cahn-Hilliard bulk-diffusion system modeling lipid raft formation in cell membranes. *Math. Models Methods Appl. Sci.*, 26(6):1149–1189, 2016.

[GM88] M. E. Gurtin and H. Matano. On the structure of equilibrium phase transitions within the gradient theory of fluids. *Quart. Appl. Math.*, 46(2):301–317, 1988.

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

[GS08] R. Glowinski and D. C. Sorensen. Computing the eigenvalues of the Laplace-Beltrami operator on the surface of a torus: a numerical approach. In *Partial differential equations*, volume 16 of *Comput. Methods Appl. Sci.*, pages 225–232. Springer, Dordrecht, 2008.

[Gua18] M. A. M. Guaraco. Min-max for phase transitions and the existence of embedded minimal hypersurfaces. *J. Differential Geom.*, 108(1):91–133, 2018.

[HK09] S. M. Houghton and E. Knobloch. Homoclinic snaking in bounded domains. *Phys. Rev. E*, 80:026210, 2009.

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

[HSO07] D. Ueyama H. Shoji, K. Yamada and T. Ohta. Turing patterns in three dimensions. *Phys. Rev. E*, 75:046212, 2007.

[IG84] E. Ihrig and M. Golubitsky. Pattern selection with O(3) symmetry. *Phys. D*, 13(1-2):1–33, 1984.

[JLTW16] A. Jamieson-Lane, P. H. Trinh, and M. J. Ward. Localized spot patterns on the sphere for reaction-diffusion systems: theory and open problems. In *Mathematical and computational approaches in advancing modern science and engineering*, pages 641–651. Springer, 2016.

[KBFVG18] A. L. Krause, A. M. Burton, N. T. Fadai, and R. A. Van Gorder. Emergent structures in reaction-advection-diffusion systems on a sphere. *Phys. Rev. E*, 97(4):042215, 13, 2018.

[KC13] G. Kozyreff and S.J. Chapman. Analytical results for front pinning between an hexagonal pattern and a uniform state in pattern-formation systems. *PRL*, 111(5):054501, 2013.

[KEVG18] A. L. Krause, M. Ellis, , and R. A. Van Gorder. Influence of curvature, growth, and anisotropy on the evolution of Turing patterns on growing manifolds. *Bull. Math. Biology*, pages s11538–018–0535–y, 2018.

[Kie12] H. Kielhöfer. *Bifurcation theory*, volume 156 of *Applied Mathematical Sciences*. Springer, New York, second edition, 2012. An introduction with applications to partial differential equations.

[KL72] H. B. Keller and W. F. Langford. Iterations, perturbations and multiplicities for nonlinear bifurcation problems. *Arch. Rational Mech. Anal.*, 48:83–108, 1972.

[Kno08] E. Knobloch. Spatially localized structures in dissipative systems: open problem. *Nonlinearity*, 21:T45–T60, 2008.

[KT18] C. Kuehn and S. Throm. Validity of amplitude equations for nonlocal nonlinearities. *J. Math. Phys.*, 59(7):071510, 17, 2018.

[Kue15] C. Kuehn. Efficient gluing of numerical continuation and a multiple solution method for elliptic PDEs. *Appl. Math. Comput.*, 266:656–674, 2015.

[KUW19] E. Knobloch, H. Uecker, and D. Wetzel. Defect–like structures and localized patterns in the cubic–quintic–septic Swift–Hohenberg equation. *PRE*, 100(1):012204, 2019.

[LBFS17] D. Lacitignola, B. Bozzini, M. Frittelli, and I. Sgura. Turing pattern formation on the sphere for a morphochemical reaction-diffusion model for electrodeposition. *Commun. Nonlinear Sci. Numer. Simul.*, 48:484–508, 2017.

[LVE09] M. Leda, V. K. Vanag, and I. R. Epstein. Instability of a three-dimensional localized spot. *PRE*, 80:066204, 2009.

[LZ01] Yongxin Li and Jianxin Zhou. A minimax method for finding multiple critical points and its applications to semilinear PDEs. *SIAM J. Sci. Comput.*, 23(3):840–865, 2001.

[LZ02] Yongxin Li and Jianxin Zhou. Convergence results of a local minimax method for finding multiple critical points. *SIAM J. Sci. Comput.*, 24(3):865–885, 2002.

[Mat03] P. C. Matthews. Transcritical bifurcations with O(3) symmetry. *Nonlinearity*, 16(4):1449–1471, 2003.

[Mat04] P. C. Matthews. Pattern formation on a sphere. In *Dynamics and bifurcation of patterns in dissipative systems*, volume 12 of *World Sci. Ser. Nonlinear Sci. Ser. B Spec. Theme Issues Proc.*, pages 102–123. World Sci. Publ., Hackensack, NJ, 2004.

[MC16] A. Madzvamuse and A. Chung. The bulk-surface finite element method for reaction–diffusion systems on stationary volumes. *Finnite Elements in Analysis and Design*, 108:9–21, 2016.

[MD14] D. Morgan and J.H.P. Dawes. The Swift–Hohenberg equation with a nonlocal nonlinearity. *Physica D*, 270:60–80, 2014.

[Mei89] Zhen Mei. A numerical approximation for the simple bifurcation problems. *Numer. Funct. Anal. Optim.*, 10(3-4):383–400, 1989.

[Mei00] Zhen Mei. *Numerical bifurcation analysis for reaction-diffusion equations*. Springer, 2000.

[Mie02] A. Mielke. The Ginzburg-Landau equation in its role as a modulation equation. In *Handbook of dynamical systems, Vol. 2*, pages 759–834. North-Holland, 2002.

[Mir19] Alain Miranville. *The Cahn-Hilliard equation*, volume 95 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2019. Recent advances and applications.

[Mod87] L. Modica. The gradient theory of phase transitions and the minimal interface criterion. *Arch. Rational Mech. Anal.*, 98(2):123–142, 1987.

[Moo80] G. Moore. The numerical treatment of nontrivial bifurcation points. *Numer. Funct. Anal. Optim.*, 2(6):441–472 (1981), 1980.

[Mur89] J. D. Murray. *Mathematical biology*. Biomathematics. Springer-Verlag, Berlin, 1989.

[NLCAS17] M. Nunez-Lopez, G. Chacon-Acosta, and J.A. Santiago. Diffusion-driven instability on a curved surface: Spherical case revisited. *Braz J Phys*, (47):231–238, 2017.

[NRV19] B. Niethammer, M. Röger, and J. Velazquez. A bulk-surface reaction–diffusion system for cell polarization. Preprint, 2019.

[Pac12] Fr. Pacard. The role of minimal surfaces in the study of the Allen-Cahn equation. In *Geometric analysis: partial differential equations and surfaces*, volume 570 of *Contemp. Math.*, pages 137–163. Amer. Math. Soc., Providence, RI, 2012.

[Pis06] L.M. Pismen. *Patterns and interfaces in dissipative dynamics*. Springer, 2006.

[PSGnP+04] R. G. Plaza, F. Sánchez-Garduño, P. Padilla, R. A. Barrio, and P. K. Maini. The effect of growth and curvature on pattern formation. *J. Dynam. Differential Equations*, 16(4):1093–1121, 2004.

[PTVF07] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007.

[PW17] K. Promislow and Q. Wu. Existence, bifurcation, and geometric evolution of quasi-bilayers in the multicomponent functionalized cahn–hilliard equation. *J. mathem. biol.*, 75:443–489, 2017.

[RR14] A. Rätz and M. Röger. Symmetry breaking in a bulk-surface reaction-diffusion model for signalling networks. *Nonlinearity*, 27(8):1805–1827, 2014.

[RU17] J.D.M. Rademacher and H. Uecker. Symmetries, freezing, and Hopf bifurcations of modulated traveling waves in pde2path, 2017.

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

[SH77] J. Swift and P.C. Hohenberg. Hydrodynamic fluctuations at the convective instability. *Physical Review A*, 15(1):319–328, 1977.

[Sie18] E. Siero. Nonlocal grazing in patterned ecosystems. *Journal of Theoretical Biology*, 436:64–71, 2018.

[SLT+15] N. Stoop, R. Lagrange, D. Terwagne, P. M. Reis, and J. Dunkel. Curvature-induced symmetry breaking determines elastic surface patterns. *Nature materials*, (14):337–342, 2015.

[SU17] G. Schneider and H. Uecker. *Nonlinear PDE – a dynamical systems approach*, volume 182 of *Graduate Studies Mathematics*. AMS, 2017.

[Ton05] Y. Tonegawa. On stable critical points for a singular perturbation problem. *Comm. Anal. Geom.*, 13(2):439–459, 2005.

[TT19] J. C. Tzou and L. Tzou. Spot patterns of the Schnakenberg reaction-diffusion system on a curved torus, 2019. Preprint.

[Tur54] A. Turing. The Turing Digital Archive, `www.turingarchive.org/`, AMT/C/10, 1954.

[Tur92] A. Turing. Morphogen theory of phyllotaxis, Parts I - III. In P.T. Saunders, editor, *Collected Works of A.M. Turing: Morphogenesis.*, pages 41–118. Elsevier, Amsterdam, 1992.

[Uec16] H. Uecker. Optimal harvesting and spatial patterns in a semi arid vegetation system. *Natural Resource Modelling*, 29(2):229–258, 2016.

[Uec19a] H. Uecker. Hopf bifurcation and time periodic orbits with pde2path – algorithms and applications. *Comm. in Comp. Phys*, 25(3):812–852, 2019.

[Uec19b] H. Uecker. Multiple bifurcation points in pde2path, 2019.

[Uec19c] H. Uecker. Using `trullekrul` in `pde2path` – anisotropic mesh–adaptation for some Allen–Cahn models in 2D and 3D, Preprint, arXiv 1912.11130 , 2019.

[Uec20a] H. Uecker. User guide on Hopf bifurcation and time periodic orbits with pde2path, 2020.

[Uec20b] H. Uecker. `www.staff.uni-oldenburg.de/hannes.uecker/pde2path`, 2020.

[UW14] H. Uecker and D. Wetzel. Numerical results for snaking of patterns over patterns in some 2D Selkov-Schnakenberg Reaction-Diffusion systems. *SIADS*, 13(1):94–128, 2014.

[UW17] H. Uecker and D. Wetzel. The pde2path linear system solvers – a tutorial, 2017.

[UW19] H. Uecker and D. Wetzel. The `ampsys` tool of `pde2path`, arxiv 1906.10622, 2019.

[UW20] H. Uecker and D. Wetzel. Snaking branches of planar BCC fronts in the 3D Brusselator. *Phys. D*, 406:132383, 2020.

[UWR14] H. Uecker, D. Wetzel, and J.D.M. Rademacher. pde2path – a Matlab package for continuation and bifurcation in 2D elliptic systems. *NMTMA*, 7:58–106, 2014.

[VAB99] C. Varea, J. L. Aragón, and R. A. Barrio. Turing patterns on a sphere. *PRE*, 60:4588, 1999.

[Wet16] D. Wetzel. Pattern analysis in a benthic bacteria-nutrient system. *Math. Biosci. Eng.*, 13(2):303–332, 2016.

[YDZE02] L. Yang, M. Dolnik, A. M. Zhabotinsky, and I. R. Epstein. Pattern formation arising from interactions between Turing and wave instabilities. *J. Chem. Phys.*, 117(15):7259–7265, 2002.

[YZE04] L. Yang, A. M. Zhabotinsky, and I. R. Epstein. Stable squares and other oscillatory Turing patterns in a reaction–diffusion model. *PRL*, 92(19):198303–1–4, 2004.

[ZUFM17] Y. Zelnik, H. Uecker, U. Feudel, and E. Meron. Desertification by front propagation? *Journal of Theoretical Biology*, (418):27–35, 2017.