

Pattern formation with `pde2path` – a tutorial

Hannes Uecker

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

June 28, 2018

Abstract

Using the demo directories `sh` (Swift–Hohenberg (SH) equation), `schnakpat` (Schnakenberg reaction diffusion system), `hexex` (a scalar problem on a hexagon) and `gcs` (a SH equation with global coupling), we explain some `pde2path` setups for pattern formation in 1D, 2D and 3D. A focus is on new `pde2path` functions for branch switching at steady bifurcation points of higher multiplicity, but we also review general concepts and results of pattern formation, including localized patterns and homoclinic snaking, again in 1D, 2D and 3D. Inter alia, the demo `sh` also illustrates how to rewrite the (4th order) Swift–Hohenberg equation as a 2-component 2nd-order system in a consistent way, and `schnakpat` simplifies and unifies previous results in a simple and concise way.

Contents

1	Introduction	1
2	Some theory: pattern formation in the Swift-Hohenberg equation	2
2.1	1D	3
2.2	2D.	4
2.3	3D	8
3	Tutorial examples	9
3.1	Demo <code>sh</code>	9
3.2	Remarks on choices of 2D and 3D meshes	18
3.3	Problems with ‘too many solutions and branching points’, warnings, tips and tricks .	20
3.4	Demo <code>schnakpat</code>	25
3.5	Higher degeneracy: Demo <code>hexex</code>	29
3.6	Demo <code>gcs</code> : global coupling, with customized linear system and eigenvalue solvers . .	30

1 Introduction

The `Matlab` bifurcation and continuation package `pde2path` [UWR14, Uec18c] can be used to study solution branches and bifurcations in pattern forming systems (PFS), in particular reaction diffusion systems of the form

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

where $\Omega \subset \mathbb{R}^d$ is a bounded domain, $d = 1, 2, 3$ (1D, 2D and 3D case, respectively), $D \in \mathbb{R}^{N \times N}$ a positive (semi-)definite diffusion matrix, $\Delta = \partial_{x_1}^2 + \dots + \partial_{x_d}^2$, 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 (BC). See, e.g., [UWR14, §4.2] and [UW14, Uec16, Wet16, BGUY17, ZUFM17, Uec18a] for examples, mostly related to pattern formation and Turing bifurcations [Mur89].

However, so far `pde2path` only dealt with *simple* bifurcation points (BPs), where exactly two solution branches intersect, although in applications (discrete) symmetries of the domain often enforce higher multiplicities of BPs. For instance, for Turing bifurcations over square domains with Neumann BC 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

$$m = \dim N(G_u(u_0, \lambda_0)) \quad (2)$$

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. 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 behaviour at the (artificially) simple BPs is in general quite different from that at the (original) 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 multiple BPs, aimed particularly at pattern formation in $d \geq 2$ space dimensions, have been recently implemented in `pde2path` [Uec18b]. Here we take a somewhat wider perspective and review in a tutorial style some general ideas of applying `pde2path` to PFS in 1D, 2D and 3D, thus complementing [Uec18b]. 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 a toy problem. In §3 we then explain the `pde2path` demos `sh` (the SH equation), `schnakpat` (the Schnakenberg reaction diffusion system), `hexex` (a scalar problem on a hexagonal domain with higher order indeterminacy), and `gcsh` (the SH equation with a global coupling which hence requires some customized linear system and eigenvalue solvers). In particular, `schnakpat` simplifies and unifies in a concise way many of the results from [UWR14, §4.2] and [UW14]. We focus on 2D results, but also give outlooks on 3D patterns, and along the way also comment on the choice of meshes in 2D and 3D (§3.2), and on tips and tricks (§3.3, including time–integration) how to deal with problems with very many solutions.

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

As our first example we consider the (quadratic-cubic) Swift-Hohenberg (SH) equation

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

with instability parameter $\lambda \in \mathbb{R}$, second parameter $\nu \in \mathbb{R}$, and boundary conditions (BC) $\partial_n u|_{\partial\Omega} = \partial_n(\Delta u)|_{\partial\Omega} = 0$. 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

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

where either $\Omega = \mathbb{R}^d$ or Ω a bounded domain and as above we assume the Neumann BC $\partial_x u|_{\partial\Omega} = \partial_x \Delta u|_{\partial\Omega} = 0$. 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

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

then the Hamiltonian, written as a function of u ,

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

is conserved, i.e., $\frac{d}{dx}H(u(x)) = 0$. This can be used to discuss the location (in parameter space) of localized patterns, see §3.1. A similar Hamiltonian also exist in 2D, see, e.g., [ALB⁺10].

For all $\lambda \in \mathbb{R}$, (3) has the spatially homogenous 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 $v(x, t) = e^{ik \cdot x + \mu(k)t}$, $k \in \mathbb{R}^d$, where

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

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 BC.

Remark 2.1. 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.,

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

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 $k_{2,3} = 1 \pm 1/(2l)$, $k_{4,5} = 1 \pm 1/l, \dots$, 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 $l \rightarrow \infty$ 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., [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

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

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 preceeding 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 BC enforce $\text{Im}(A_j) = 0$ 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 BC or homogeneous Dirichlet BC.

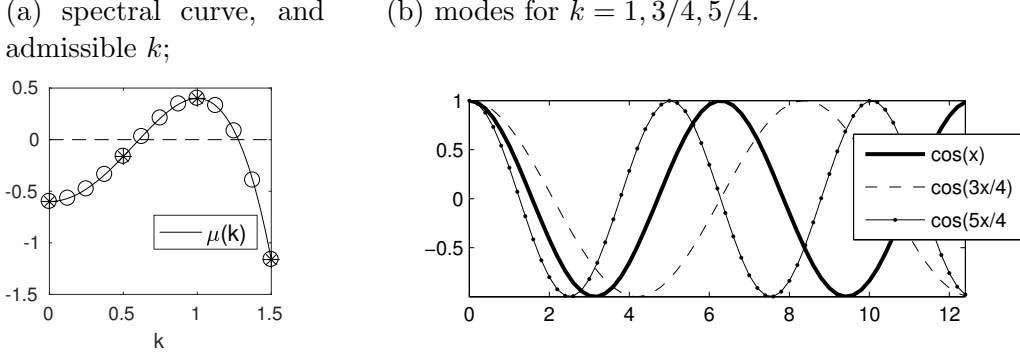


Figure 1: (a) Dispersion relation with admissible k for $\Omega = (0, \pi)$ (*) and $\Omega = (0, 4\pi)$ (o). (b) First 3 modes for $\Omega = (0, 8\pi)$.

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

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

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 §3.1 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 §3.1.

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 Ω =square (rectangular dual grid).

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.

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 $e_{m,n} = e^{i(mx+ny)}$, and make the ansatz

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

where again c.c. stands for the complex conjugate since we look for real solutions. The BC 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 BC, where

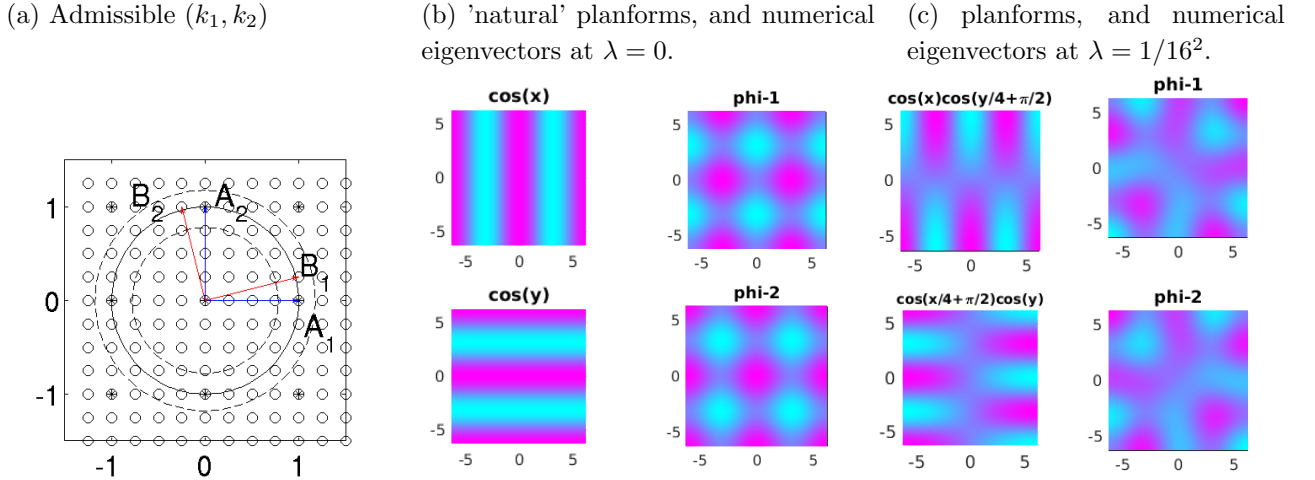


Figure 2: Spectral situation for the SH equation over square domains. (a) Admissible wave vectors k for $\Omega = (-\pi/2, \pi/2)^2$ (*) and $\Omega = (-2\pi, 2\pi)^2$ (o), respectively, with Neumann BC. The amplitudes $A_{1,2}$ and $B_{1,2}$ are used in the amplitude equations below. (b,c) Kernels at 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 .

$\text{Im}A_k$ 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

$$\begin{aligned}
u^2 &= \varepsilon^2 \left[A_1^2 e_{2,0} + A_2^2 e_{0,2} + 2(|A_1|^2 + |A_2|^2) e_0 + 2A_1 A_2 e_{1,1} + 2A_{-1} A_2 e_{-1,1} \right] \\
&\quad + 2\varepsilon^3 \left[A_0 (A_1 e_1 + A_2 e_2) + (A_{2,0} A_{-1} + A_{1,1} A_{-2} + A_{1,-1} A_2) e_1 + (A_{0,2} A_{-2} + A_{1,1} A_{-1} + A_{-1,1} A_1) e_2 \right] \\
&\quad + \text{c.c.} + \text{h.o.t.}, \\
u^3 &= 3\varepsilon^3 \left[(|A_1|^2 + 2|A_2|^2) A_1 e_1 + (2|A_1|^2 + |A_2|^2) A_2 e_2 \right] + \text{c.c.} + \text{h.o.t.}, \\
\partial_t u &= \varepsilon^3 (\dot{A}_1 e_1 + \dot{A}_2 e_2) + \text{c.c.} + \text{h.o.t.},
\end{aligned}$$

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

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

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 [Uec18b] 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

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

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

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

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 behaviour 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 §3.1.

$$\begin{array}{c|cccc} I_\nu & [0, \nu_1) & (\nu_1, \nu_2) & (\nu_2, \nu_3) & (\nu_3, \infty) \\ \text{sector} & \text{I} & \text{II} & \text{III} & \text{IV} \end{array} \quad (13)$$

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 $u = \varepsilon(B_1 e^{i(x+y/4)} + B_2 e^{i(-x/4+y)}) + \text{c.c.} + \text{h.o.t.}$, 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 $\Omega = \text{rectangle}$, with 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 BC 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

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

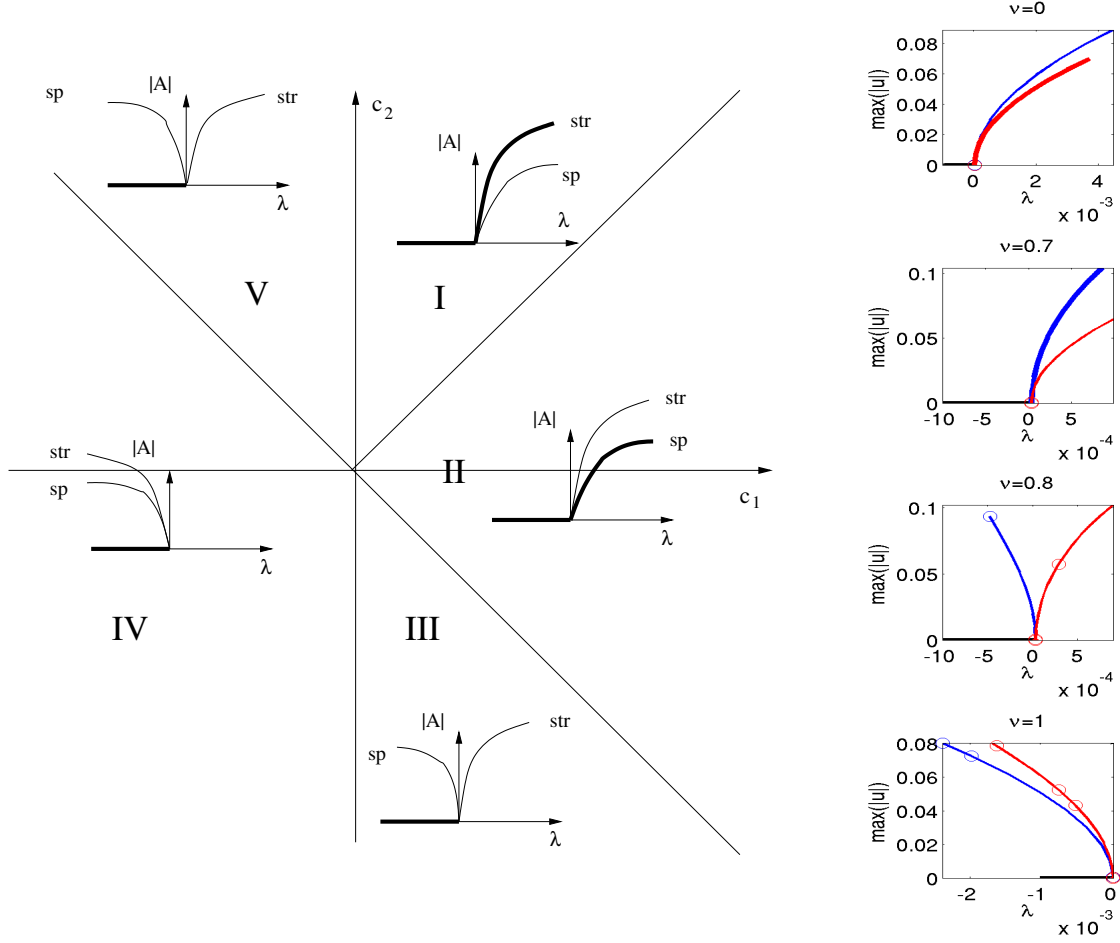


Figure 3: Left: Bifurcation (and stability) diagram for (11) in the c_1 - c_2 plane. Right: corresponding numerical bifurcation diagrams for (3) in dependence of ν , stripes=red, spots=blue, thick lines=stable solutions, thin lines=unstable solutions.

and the pertinent symmetry considerations yield the amplitude equations (in complex notation, although again for Neumann BC 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)

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

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

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

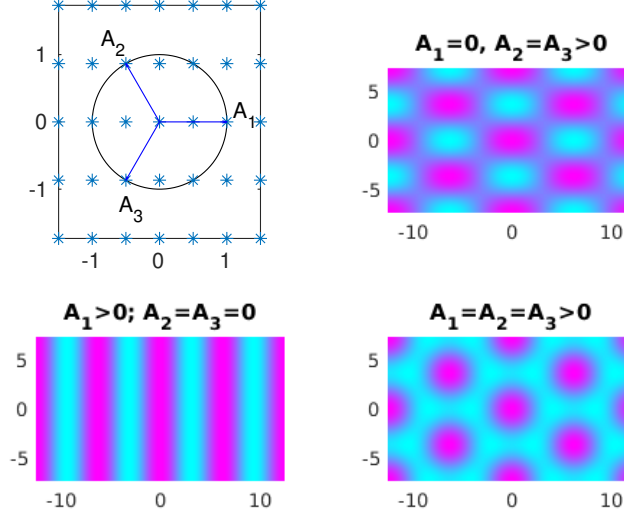


Figure 4: Admissible wave vectors for the domain $\Omega = (-\pi, \pi) \times (-\pi/\sqrt{3}, \pi/\sqrt{3})$, and associated planforms at bifurcation, for clarity plotted over $\Omega = (-4\pi, 4\pi) \times (-4\pi/\sqrt{3}, 4\pi/\sqrt{3})$.

$$\text{pathwork quilt: } A_1 = 0, \quad A_2 = A_3 = \pm \sqrt{\frac{\mu}{c_1 + c_2}} \quad (\text{only if } \gamma = 0).$$

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

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

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. 6 in §3.1.3

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 behaviour 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 BC, while Dirichlet or Neumann BC 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

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

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 $e_{lnm} = \exp(i(lx_1 + nx_2 + mx_3))$. 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

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

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 $\mu = 0, (A_1, A_2, A_3) = s(1, \pm 1, \pm 1), s \in \mathbb{R}$, 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 §3.1.5 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 Tutorial examples

In §3.1 and §3.4 we consider the two main example problems, the quadratic-cubic SH equation (3), and, as a proper RD system the Schnakenberg system (see (32) below), which has been used as a `pde2path` model problem before, see in particular [UWR14, §4.2] and [UW14, dW17, dW18]. For the Schnakenberg system, besides illustrating the usage of `q(c)swibra` we also explain a trick for conveniently computing the dispersion relation, and put all this in a demo directory to also get the basic results from [UW14] in a simple and updated way. Additional to 2D and 3D domains, where naturally $m \geq 2$, we also treat the 1D cases, because the general numerical setup is independent of the spatial dimension, and the 1D cases give fast results.

3.1 Demo sh

Besides the connection to the analytical results from §2, the SH equation gives an opportunity to show how to rewrite this fourths order equation as a 2-component 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

$$\mathcal{M} \frac{d}{dt} \mathbf{u} = -G(\mathbf{u}) \quad (19)$$

for the unknown nodal values $u \in \mathbb{R}^{n_u}$, 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

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

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

$$\mathcal{M}\dot{u} = -(\mathcal{K}u - F(u)), \quad (21)$$

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

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 [Uec18a] 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 [RU18, Remark 1.1] for further comments.]

See Table 1 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 1: Main scripts and functions in `/demos/sh`.

script/function	purpose, remarks
<code>cmds1d</code> , <code>cmds1dhplot</code>	scripts for 1D, essentially yielding Fig. 5.
<code>cmds2dsq</code>	script for 2D square domain, essentially yielding Fig. 2.
<code>cmds2dhex</code>	script for 2D rectangular domain for hexagons, essentially yielding Fig. 6.
<code>cmds2dhexfro</code>	script for localized hex patterns on 2D long rectangular domain, see Fig. 7
<code>cmds2dhexb</code>	similar to <code>cmds2dhex</code> , but on domain twice as large; meant to illustrate tips and tricks, for problems with ‘too many’ patterns, essentially yielding Fig. 11.
<code>cmds2dtint</code>	script for obtaining patterns from initial guesses and time integration, see Fig. 12.
<code>cmds3dSC</code>	script for simple cube (SC) 3D patterns, see Fig. 8.
<code>cmds3dBCC</code>	script for body centered cube (BCC) 3D patterns, see Fig. 9.
<code>cmdsBCClong</code>	script for localized BCC patterns obtained from initial guesses, see Fig. 13.
<code>shinit</code>	initialization
<code>oosetfemops</code>	set FEM matrices (stiffness K , and two mass matrices M , M_0)
<code>sG,nodalf,sGjac</code>	encodes G with ‘nonlinearity’ in <code>nodalf</code> , and Jacobian
<code>spjac</code>	Jacobian for fold continuation
<code>shbra1d</code>	modification of <code>stanbra</code> for putting the Hamiltonian H on the branch
<code>geth</code>	function to compute the Hamiltonian for the spatial dynamics formulation
<code>e2rs</code>	<code>Element2RefineSelection</code> function, used for mesh adaption, here just ad-hoc
<code>hfplot,spl,spplots</code>	convenience functions to plot solutions, Fourier transforms, and planforms

¹See, e.g., [RU18, Uec18a] for details on the general classes of systems of PDEs that `pde2path` can treat.

```

1 function p=ooetfemops(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: `ooetfemops.m`, `nodalf.m` and `sG.m` from `demos/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 empty as we implement the 2nd equation from (20) via \mathcal{K} .

3.1.1 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 [Kno08, BKL⁺09, HK09, ALB⁺10, KC13]. Since the SH equation has the spatial Hamiltonian (5), i.e., $H(u) = \partial_x u \partial_x^3 u - \frac{1}{2}(\partial_x^2 u)^2 + (\partial_x u)^2 + \frac{1}{2}(1 - \lambda)u^2 - F(u)$, $F(u) = \int_0^u f(v) dv$, and since $H(0) = 0$, a front between $u = 0$ and a periodic pattern must connect to a pattern $u_{\text{per}} = u_{\text{per}}(\lambda)$ with $H(u_{\text{per}}(\lambda), \lambda) = 0$. 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 $H(u_{\text{per}}(\lambda), \lambda)$ 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 underresolved 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.

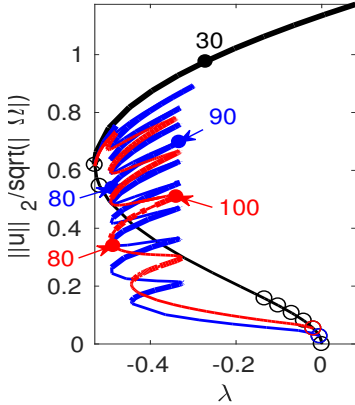
Regarding the implementation, the main additional function is `geth` to compute H , which is then put on the branch for plotting in `shbra1d`. Additional to the above results, at the end of `cmds1d` we continue the fold for illustration of fold continuation in a system, cf. [dW17], with a straightforward implementation of `spjac`. Moreover, in §3.6 (demo `gcsh`) we add a global coupling to (3) and illustrate how this modifies the branches in Fig. 5.

3.1.2 Intermezzo: branch switching at BPs of higher multiplicity

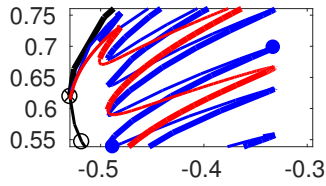
As discussed in §2, for ‘natural’ (i.e. highly symmetric) choices of domains in 2D and 3D, bifurcation points (u_0, λ_0) on homogeneous branches often have a multiplicity

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

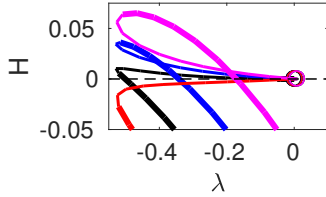
(a1) Primary branch (black) and two snaking branches of loc. patterns



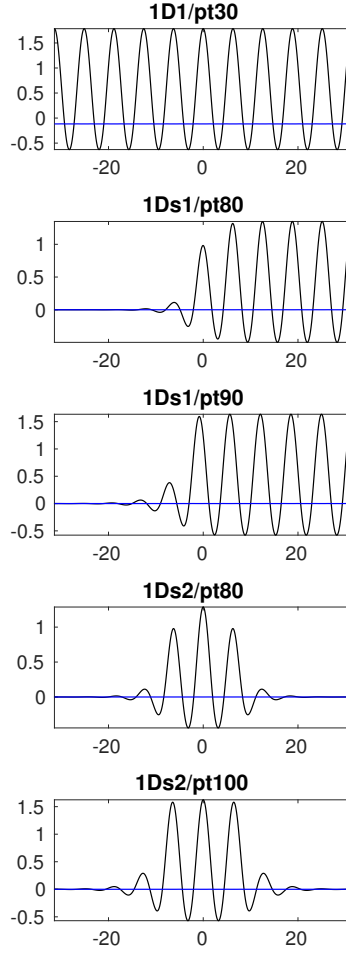
(a2) blow-up from (a1)



(a3) H on the first four periodic branches, $\Omega = (-20\pi, 20\pi)$



(b) solution plots, including H



(c) $|\mathcal{F}(u)(k)|$ from (b)

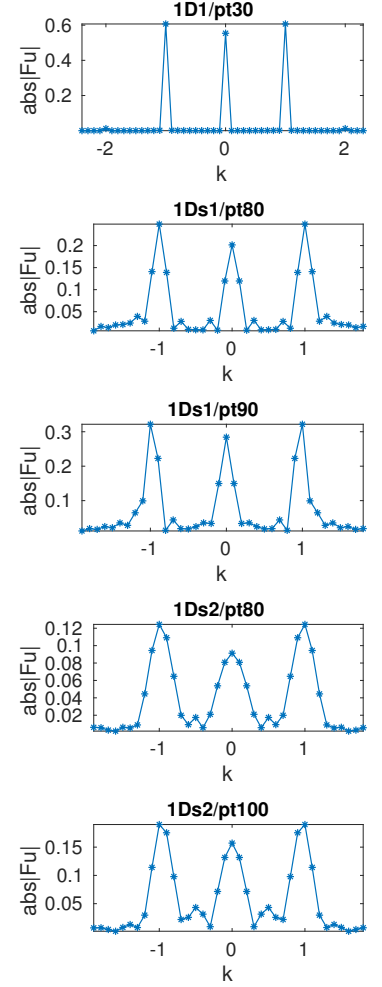


Figure 5: (a1) Subcritical bifurcation of primary periodic patterns (black branch) in the SH equation with $\nu = 2$, $\Omega = (-10\pi, 10\pi)$ and secondary bifurcations of snaking branches of a front (blue, 1Ds1) and a localized pattern (red 1Ds2). (a2) blow-up from (a1) showing how the snaking branches 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.

Here we briefly review the algorithm for branch switching at multiple bifurcation points from [Uec18b]. Let (u_0, λ_0) be a bifurcation point of multiplicity $m \geq 2$, and let

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

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 $\langle \phi_0, \psi_j \rangle = 0$, $j = 1, \dots, m$. The ansatz

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

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

$$B(\alpha_0, \alpha) = 0 \in \mathbb{R}^m, \quad (25)$$

$$B_i(\alpha_0, \alpha) = \sum_{j=1}^m \sum_{k=1}^m a_{ijk} \alpha_j \alpha_k + 2 \sum_{j=1}^m b_{ij} \alpha_j \alpha_0 + c_i \alpha_0^2, \quad 1 \leq i \leq m,$$

$$a_{ijk} = \langle \psi_i, G_{uu}^0[\phi_j, \phi_k] \rangle, \quad b_{ij} = \langle \psi_i, G_{uu}^0[\phi_0, \phi_j] + G_{u\lambda}^0 \phi_j \rangle, \quad c_i = \langle \psi_i, G_{uu}^0[\phi_0, \phi_0] + 2G_{u\lambda}^0 \phi_0 + G_{\lambda\lambda}^0 \rangle.$$

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. Wlog 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 $\alpha_i \in \{0, \pm 1\}, i = 1, \dots, m$.

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 [Uec18b, Algorithm 2.1].

The case $m \geq 2$ is more difficult, and the QBE (25) may (and typically will) not yield (all) bifurcating branches, but only those that are 2-determined, see [Uec18b]. Pitchfork bifurcations are at best 3-determined, and in this case we use the ansatz

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

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

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

of m cubic bifurcation equations (CBE), see [Uec18b]. 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=qswwibra(dir,bpt)` (`p0=cswwibra(dir,bpt)`) attempt to solve the QBE (CBE), and store the computed tangents in the fields `p0.mat.qtau` (`p0.mat.ctau`), respectively, and store the kernel vectors ϕ_1, \dots, ϕ_m in `p0.mat.ker`. Additionally, `cswwibra` 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 $\tau = \sum_i \gamma_i \text{p.mat.ker}(i)$ via `p=gentau(p0,ga)` where the sum runs from 1 to `length(ga)`. In Algorithm 3.1 we summarize the approach, in Table 2 we collect auxiliary arguments for fine tuning of `qswwibra` and `cswwibra`, and otherwise refer to [Uec18b] for further mathematical comments.

3.1.3 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=qswwibra(dir,pt,aux)` and `p0=cswwibra(dir,pt,aux)`

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 2 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`).

can and sometimes must be fine tuned via the auxiliary argument `aux`, which can have the fields from Table 2.

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`.

The script `cmds2dhex.m` considers (3) over $\Omega = (-l_x, l_x) \times (-l_y, l_y)$, $l_x = 2\pi, l_y = 2\pi/\sqrt{3}$. This small domain is 4 times the minimal domain $\Omega = (0, l_x) \times (0, l_y)$ allowing hexagon patterns. In Fig. 6(a) we have $\nu = 0$ and hence at $\lambda = 0$ have pitchforks of stripes (stable), pq (stable) and hex (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.

```

%% init and zero-branch
lx=2*pi; nx=round(8*lx); ly=lx; ndim=2; lam=-0.001; nu=0; par=[lam; nu];
3 p=shinit(p,nx,lx,ly,ndim,par); p=setfn(p,'2D0'); 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
8 % 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
13 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;

```

Table 2: 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`.

field	purpose, remarks
<code>soltol</code>	tolerance to solve the QBE or the CBE, i.e., $ F(\alpha) < \text{soltol}$, default 10^{-20} .
<code>isotol</code>	tolerance to view a solution α of the QBE or CBE as isolated if $ \det \partial_\alpha F(\alpha) > \text{isotol}$, default 10^{-10} .
<code>mu2</code>	to override <code>p.nc.mu2</code> where an eigenvalue μ is considered to be zero if $ \mu < \text{mu2}$.
<code>m</code>	to explicitly give the dimension of the kernel, instead of $m = \#\{\mu : \mu < \text{mu2}\}$
<code>al0v</code>	to override $\alpha_0=0.001$ for <code>qswibra</code> ; can be a vector $\alpha_0=(\alpha_0(1), \dots, \alpha_0(j))$
<code>bet0</code>	to override $\beta_0=0.001$ for <code>cswibra</code> ; see [Uec18b] for the (scaling) purposes of α_0 and β_0 .
<code>alc</code>	to override the initial guesses for the Newton loop for α ; again see [Uec18b]
<code>ral</code>	use random initial guesses for α if <code>ral=1</code>
<code>ds</code>	to override the steplength selection <code>p.nc.dsmax/10</code> ; also used for computing the quadratic predictor (u, λ) from (26).
<code>hasker</code>	for a subsequent call to <code>qswibra</code> or <code>cswibra</code> in the syntax, e.g., <code>p0=cswibra(p0,aux)</code> ; if <code>hasker=1</code> , then the kernel is taken from <code>p0.mat.ker</code> and not recomputed; useful for experimenting with parameters, and for instance used in <code>qcswibra</code> .
<code>keep1ss</code>	if 1, then the linear system solver <code>p0.fuha.1ss</code> is used for solving the linear systems inside <code>cswibra</code> . Otherwise, and as default setting, <code>1sslu</code> is used.

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

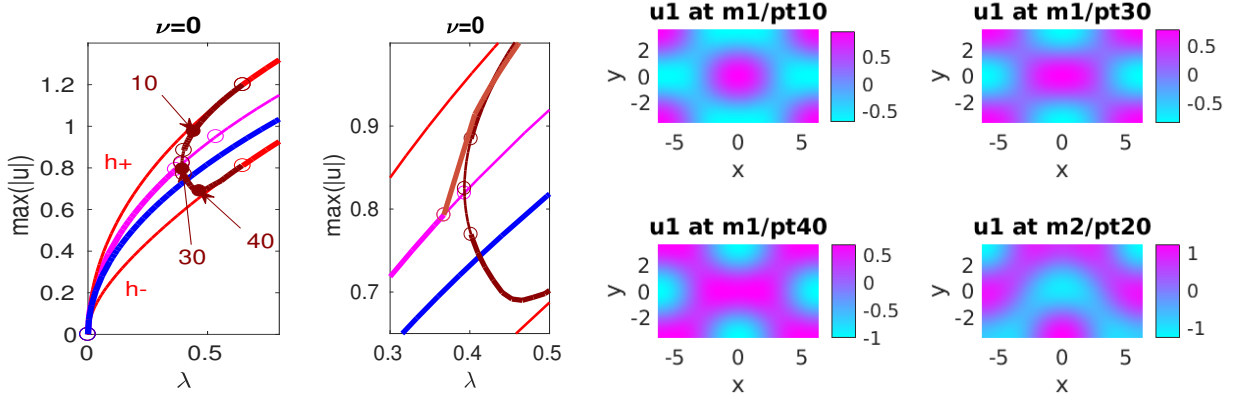
Listing 2: (Selection from) `sh/cmds2dsq.m`. Script for the 2D SH equation on the square domain $\Omega = (-2\pi, 2\pi)^2$, which yields a double branch points at $\lambda = 0$. `cswibra` in line 5 finds 4 bifurcation directions, falling into the two classes of spots and stripes. Thus, in lines 10,11 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. 2), plotting, and the 2nd bifurcation point, cf. Remark 2.3.

For $\nu = 1.3$ in (b), the hex bifurcate transcritically and the up-hex 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-hex become stable at $\lambda = \lambda_3 \approx 0.46$, and the branch bifurcating there connects to (shifted) up hex at $\lambda \approx 1.8$. Even on this relatively small domain there are many additional bifurcation on the hex and stripes branches. 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 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 serious problems. See §3.3.

(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

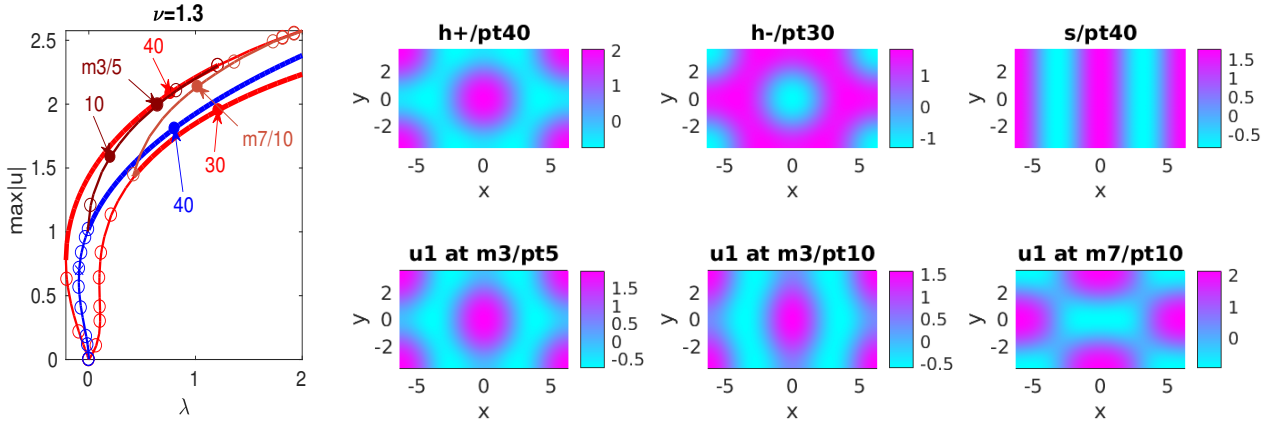


Figure 6: Example results from `cmds2dhex.m`. Bifurcation diagrams and example plots SH over a small rectangular domain permitting hex solutions. For $\nu = 0$ in (a), hex (unstable), str and pq (stable) bifurcate in supercritical pitchforks. The (up and down) hex become stable at $\lambda = \lambda_1 \approx 0.645$, and these points are connected by a mixed mode branch m1, which 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.

3.1.4 Planar fronts between hexagons and 0

Localized patterns similar to Fig. 5 can also occur in 2D. 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 §3.3, in §3.4, and in, e.g., [UW14], and here we restrict to planar fronts between hexagons and $u \equiv 0$, and 1D-localized hex-patches. Figure 7 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.

```

1 % 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.np
  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);
2 % hex from qswibra, needs low isotol due to poor grid
  aux=[]; aux.isotol=1e-16; p0=qswibra('2Dhex8','bpt1',aux); p0.sw.bifcheck=1;

```

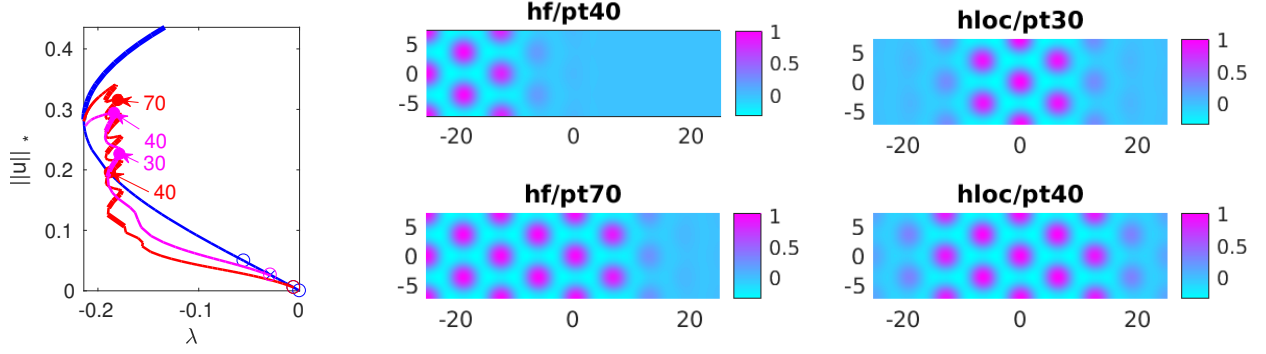



Figure 7: Results from `cmds2dhexfro` for the SH over a long rectangular domain, $\nu = 1.3$. Primary branch of (up) hexagons (blue), and snaking branches of fronts (red, hf) between hexagons and $u = 0$ and of localized hex (magenta, hloc).

```

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
11 p=pmcont(p,50);
    %% 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;

```

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.

3.1.5 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, see, e.g., [HSO07, LVE09]. Here we first restrict to a simple 3D analogon of §2.2.1, namely $\Omega = (-\pi, \pi)^3$, again with homogeneous Neumann BC $\partial_n u = \partial_n \Delta u = 0$ on $\partial\Omega$. At the first bifurcation point $\lambda = 0$ we then have a three dimensional kernel $N(G_u) = \text{span}\{\cos(x), \cos(y), \cos(z)\}$, 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 8 shows some results from `cmds3dSC.m`. In (a) we give the three numerically obtained kernel vectors ϕ_1, ϕ_2, ϕ_3 , given by three clean lamellas. This, however, strongly depends on the chosen mesh, see §3.2 for further remarks, and Fig. 9 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. 9 we consider solutions on a cube allowing BCC (body-centered cubic) branches. The proper BCC lattice corresponds to $n=6$ wave vectors

$$k_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, k_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, k_3 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, k_4 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, k_5 = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}, k_6 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \quad (28)$$

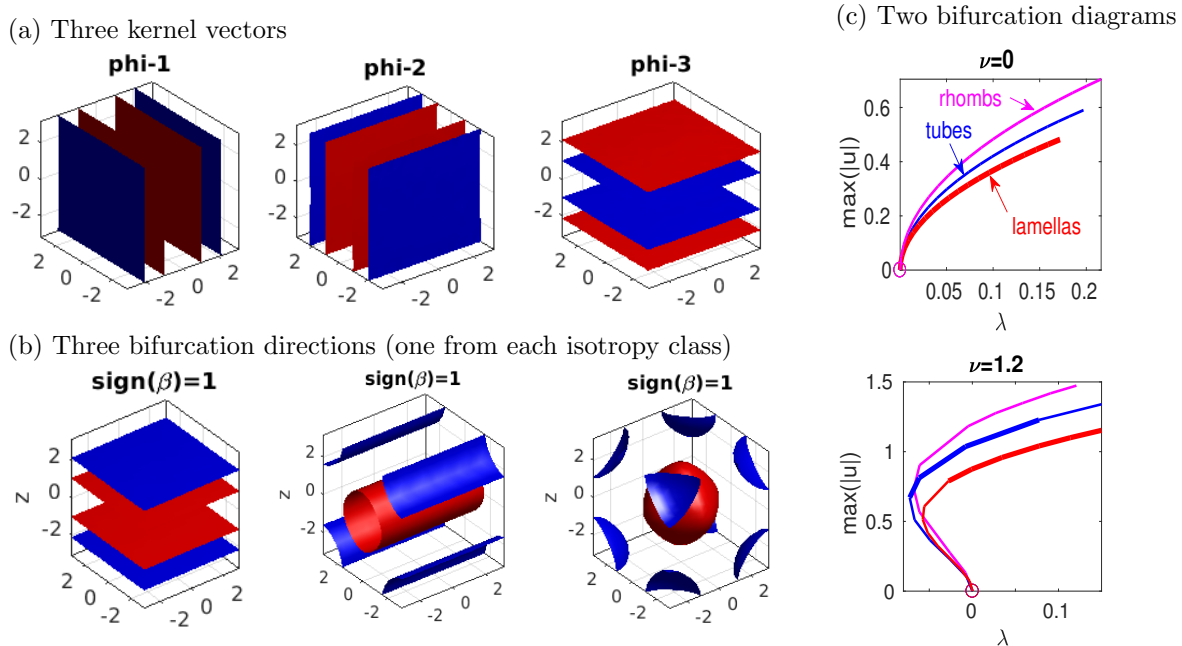


Figure 8: Selected results from demo `sh/cmds3dcube`. Primary bifurcations at $\lambda = 0$ in (3) over the cube $\Omega = (-\pi, \pi)^3$ with homogeneous Neumann BC, pseudo criss-cross mesh of $n_p = 6006$ points and $n_t = 33000$ tetrahedral elements, see §3.2 for comments on the meshing. (a) Isosurface 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. $\text{sign}(\beta) = 1$ 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.

leading to a six-dimensional amplitude systems for amplitudes A_1, \dots, 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 BC, only few of the solutions of the amplitude system can be realized, namely:

- Tubes, or more precisely square prisms, corresponding to $A_1 = A_4 \in \mathbb{R}$, $A_2, A_3, A_5, A_6 = 0$, 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, $A_1 = \dots = A_6 \in \mathbb{R}$, which correspond to equal amplitude superpositions of tubes.

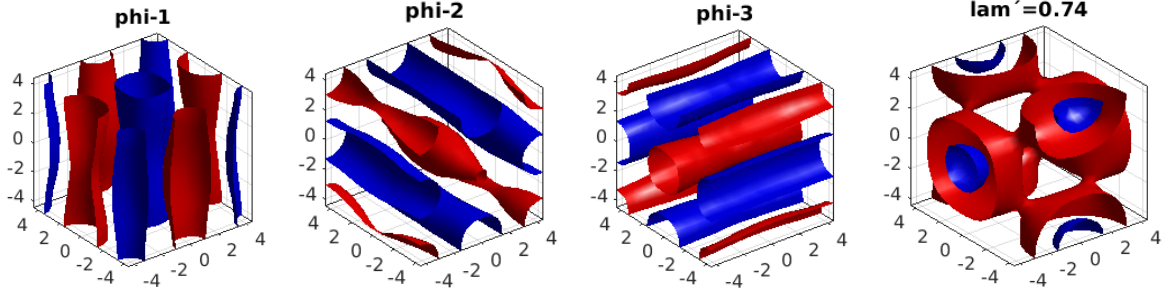
The three kernel vectors at the primary bifurcation are computed as distorted tubes. The BCC branch bifurcates transcritically, and bifurcation directions are found by `qswibra`. We call the branch to the left (right) hot (cold) balls as they have maxima (minima) in the centers. The hot balls become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

Figures 8 and 9 are just intended as first illustrations of 3D pattern formation with `pde2path`. For instance, by extending the domain from Fig. 9 in one direction we can now produce snaking branches of 'localized hot balls', see §3.3. 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 initial guesses.

3.2 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,

(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

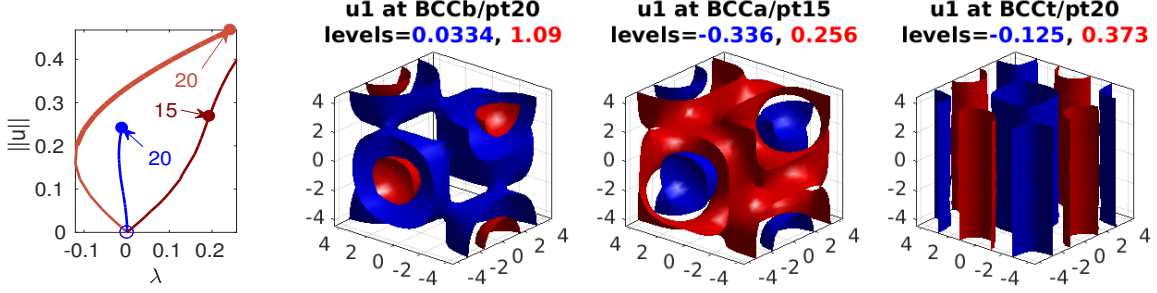


Figure 9: (3) on a “BCC lattice cube” $\Omega = (-\sqrt{2}\pi, \sqrt{2}\pi)^3$, $\nu = 1$. (a) kernel vectors (distorted square prisms) and bifurcation direction for a BCC obtained from `qswibra`. (b) BD, ‘hot’ and ‘cold’ BCCs and square prism example solutions. $n_p = 10351$, $n_t = 57024$ tetrahedra, see `cmds3dBCC.m` for details.

see Fig. 10(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. 10(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 Delaunay triangulation then yields meshes of type (c), which are at least reflection symmetric.

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. 8 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. 10, 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 Delaunay 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 10(f) shows one of the three distorted lamella kernel vectors obtained for the same settings as in Fig. 8, 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. 8(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, \dots , the struct `sw` as an auxiliary argument. Currently, this can have two fields, namely

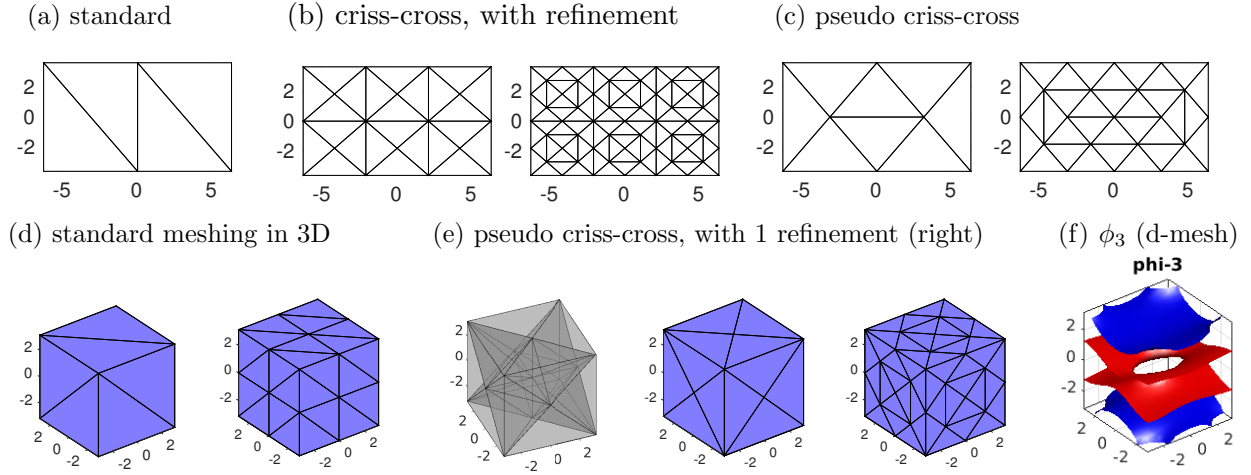


Figure 10: (a-c) 2D meshes on $\Omega = (-l_x, l_x) \times (-l_y, l_y)$, $l_x = 2\pi, l_y = 2\pi/\sqrt{3}$, starting from a meshgrid with $n_x = 3, n_y = 2$. (a) Standard meshing ($n_p = 6$) in 2D destroys reflection and rotation symmetry. (b) criss-cross via “refine-longest”, $n_p = 18$, and with one 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. 8(a).

- If `sw.sym=1`, then meshes of type (c) (2D) and (e) (3D) from Fig. 10 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.

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.

3.3 Problems with ‘too many solutions and branching points’, warnings, tips and tricks

3.3.1 General remarks

In §3.1.3 (2D) and §3.1.5 (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. 6, see Fig. 11, 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. 11(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).

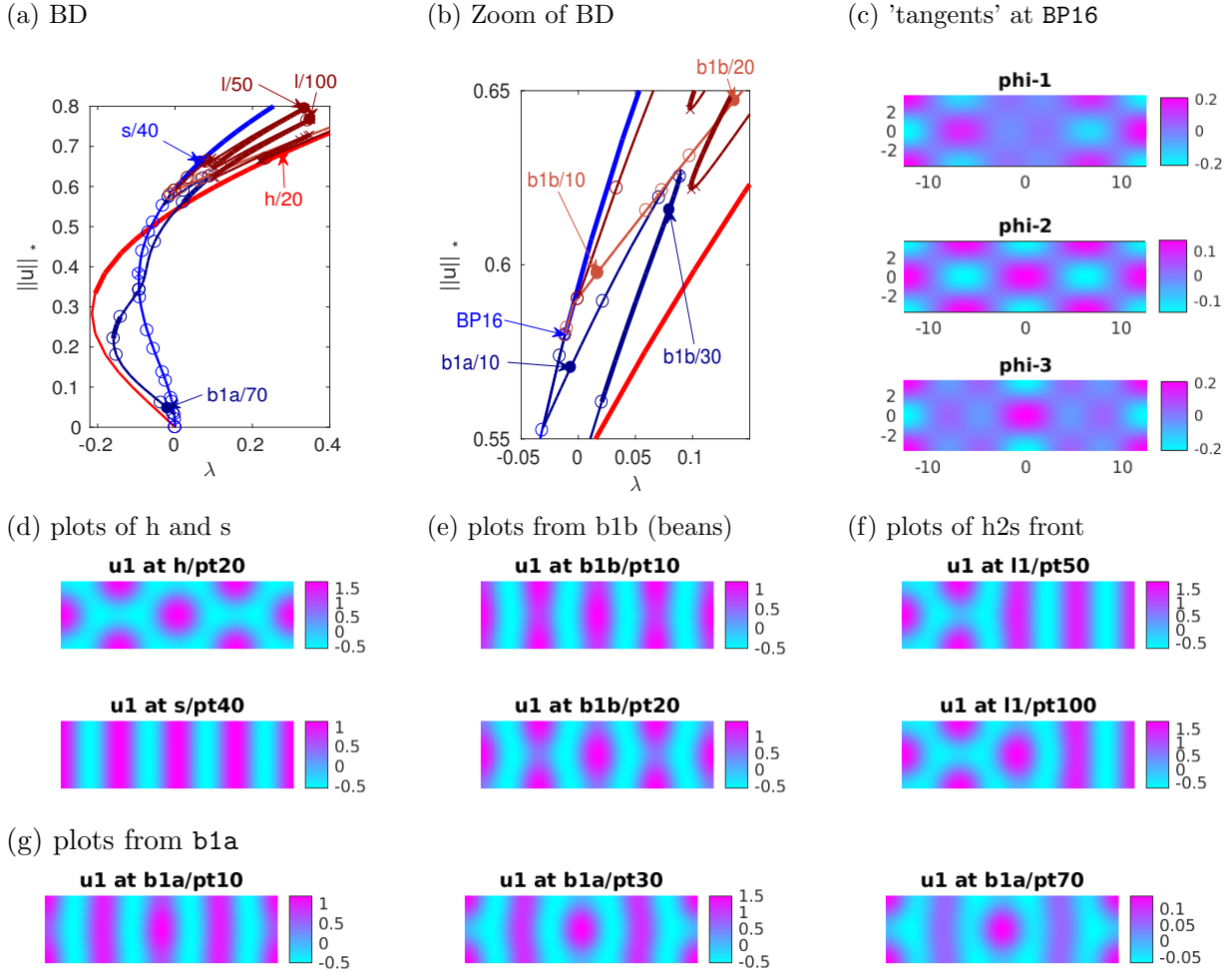


Figure 11: (3) on $\Omega = (-2l_x, 2l_x) \times (-l_y, l_y)$, $l_x = 2\pi$, $l_y = 2\pi/\sqrt{3}$, $\nu = 1.3$, example results from `cmds2dhexb.m`. See text and `cmds2dhexb.m` for details.

In particular, the **b1b** branch are the beans (light brown (e)), from which a snaking branch (dark brown (f)) of fronts between hex 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.

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 behaviour 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 behaviour 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 11 branch, after which the branch continues qualitatively as before. Finally, consider the remarks on mesh-symmetry from §3.2.

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.

3.3.2 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. 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^* . A simple example is given in `cmds2dtint.m`, with some results plotted in Fig. 12. Here we use same domain as in Fig. 11 and aim to directly obtain a 'hex2stripes' front as in Fig. 11(f). For this we use initial guesses of the form

$$u_1(x, y) = \begin{cases} \cos(x) + \cos(x/2) \cos(\sqrt{3}y/2) & x \leq 0 \\ \alpha \cos(x) & x \geq 0 \end{cases}, \quad u_2(x, y) \equiv 0, \quad (29)$$

$\alpha=2$. 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. 11(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 (29) with $\alpha=4$ in Fig. 11(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 [DRUW14], which essentially only need the struct `p` as main inputs. The main time-steppers are `tint`, `tintx` (general purpose, where the `x` stands for more comprehensive output such as time-series of the residuals $\text{res}(t) = \|G(u(t))\|_\infty$), 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. 11(c), and a subsequent Newton loop takes us to the desired **solution 1**.

```
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);
  huclean(p); 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
6 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
11 %% 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);
16 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);
21 %% 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);
26 plotsol(p,1,1,2); pause; clf(2); p=pmcont(p,20);

```

Listing 4: sh/cmds2dtint.m. Obtaining solutions from initial guesses, possibly combined with some time-integration (lines 17-21).

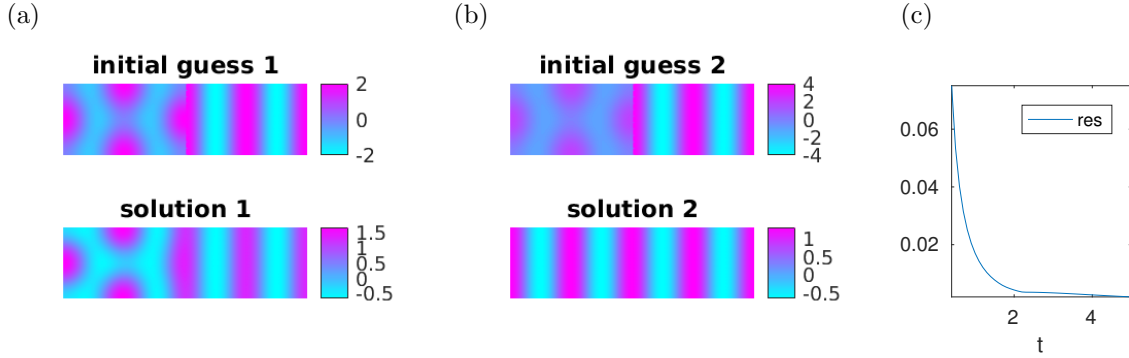


Figure 12: Obtaining solutions from guesses, if necessary including time-integration. (a) A 'reasonable' initial guess for a hex-to-stripes front, yielding the desired solution directly from a Newton loop, $\nu = 1.3$, $\lambda = 0.2$. (b) A 'bad' initial guess for a hex-to-stripes front; here the Newton 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.

In Fig. 13 we proceed similarly in 3D. Here, as in Fig. 5 (1D) and in Fig. 7 (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. 13(b).² Then we let $\Omega = (-l_x, l_x)^2 \times (-8l_x, 8l_x)$ and $\lambda = -0.3$, and try the guess

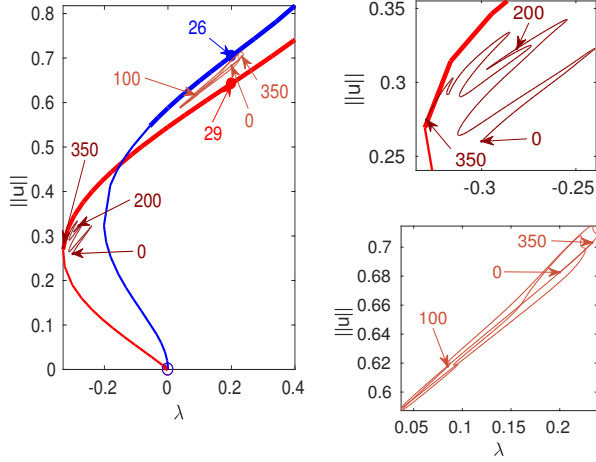
$$u_1(x, y, z) = \begin{cases} 0.4\text{Re}\left[\sum_{j=1}^6 \exp(ik_j \cdot (x, y, z))\right], & z \geq 0 \\ 0 & z \leq 0 \end{cases}, \quad u_2(x, y, z) \equiv 0, \quad (30)$$

with k_j from (28) 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

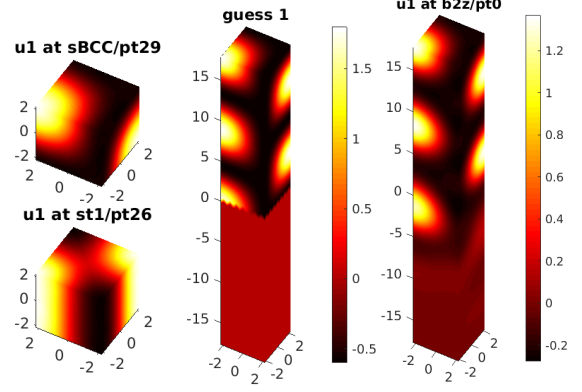
²For speed and convenience, the red and blue BCC and tubes branches in the left panel of (a) are from the minimal domain $\Omega = (-l_x, l_x)^3$, including the stability, but we can obtain the same branches on $\Omega = (-l_x, l_x)^2 \times (-8l_x, 8l_x)$, with the same stability for the BCCs and almost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

spectral computations, and instead here remark that this snaking branch consists of alternating stable and unstable segments, as expected.

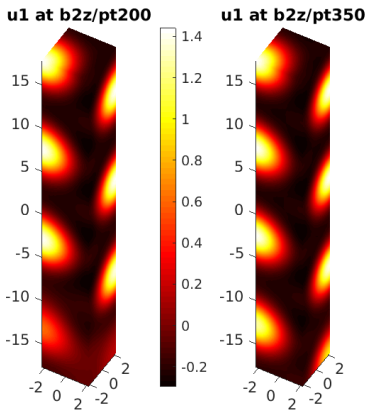
(a) BD and zooms



(b) BCC, tube, guess for b2z front, and 1st solution



(c) two more solutions on b2z branch



(d) guess for bcc-to-tubes (b2t) solution, and three sol. on b2t branch.

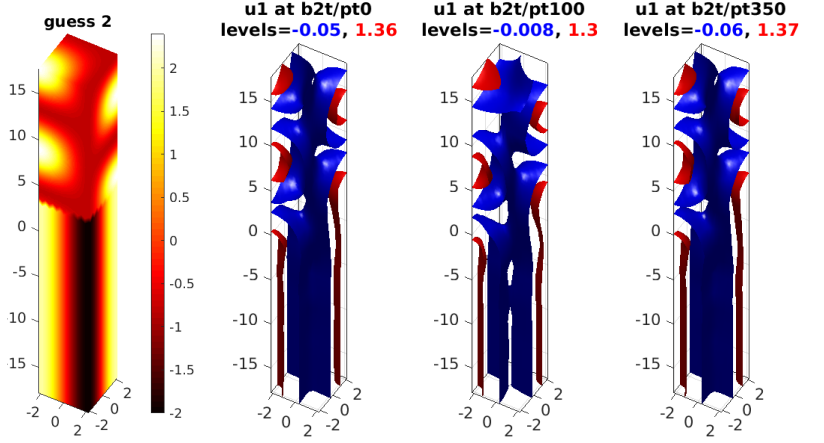


Figure 13: Results from `sh/cmdsBCClong.m` for (3) on $\Omega = (-l_x, l_x)^2 \times (-l_z, l_z)$, $l_x = \pi/\sqrt{2}$, $l_z = 8l_x$, $\nu = 1.5$. (a) BD of BCCs (red), tubes (blue), and b2z front branch (dark brown) and b2t isola (light brown). (b) BCC and tube plotted over small domain; guess for 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.

As an additional example for the multitude of patterns on this long domain, similar to Fig. 12 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

$$u_1(x, y, z) = \begin{cases} 0.4\text{Re}\left[\sum_{j=1}^6 \exp(ik_j \cdot (x, y, z))\right], & z \geq \pi \\ \cos((x+y)/\sqrt{2}) + \cos((x-y)/\sqrt{2}), & z \leq \pi \end{cases}, \quad u_2(x, y, z) \equiv 0, \quad (31)$$

see the first plot in Fig. 13(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.

3.4 Demo schnakpat

In the demo **schnakpat** we consider the modified Schnakenberg reaction diffusion system

$$\partial_t U = D\Delta U + F(U), \quad U = \begin{pmatrix} u \\ v \end{pmatrix}, \quad F(U) = \begin{pmatrix} -u+u^2v \\ \lambda-u^2v \end{pmatrix} + \sigma \left(u - \frac{1}{v}\right)^2 \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad (32)$$

with diffusion matrix $D = \begin{pmatrix} 1 & 0 \\ 0 & d \end{pmatrix}$ and parameters $\lambda > 0$ and $\sigma \in \mathbb{R}$. In suitable parameter regimes, (32) shows Turing bifurcations from the homogeneous branch $(u, v) = (\lambda, 1/\lambda)$, and, in a nutshell, we may may more or less expect all phenomena explained for the SH equation in the previous sections also in (32). 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. (32) 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 tutorial demo which besides illustrating the use of **qswibra** and **seltau** also recovers some of the results from [UW14]. Table 3 gives an overview of the involved files. We fix $d = 60$ throughout, and focus on five tasks, namely:

- explain a trick to let **pde2path** display the dispersion relation for homogeneous states;
- compute a basic bifurcation diagram of steady spatially periodic 1D patterns, including snaking branches of localized patterns;
- generate a basic bifurcation diagram of steady spatially periodic 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.

Table 3: 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. 14–16.
cmds2dsq	compute a basic bifurcation over a square, primary bifurcations are pitchforks.
cmds2da	compute a basic bifurcation over a small rectangle, Fig. 17.
cmds2db	compute a branch of spots embedded in stripes over a long 2D domain, Fig. 18.
cmds3dSC, cmds3dBCC	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)

3.4.1 1D: computing the dispersion relation, basic branches, and snaking

For (32) (with $d = 60$ fixed) we in principle know the 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 C1 of **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. 14. **spufu.m** is a modification of the **pde2path** library function **stanufu**, see Listing 7, and should be easily adaptable to any RD system.

```
% 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(nu+1)]; % short Hom-state vector
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; k1=length(kv); muv=zeros(2,k1); % provide wave-nrs and mem
```

```

25 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
30 end
    figure(10); clf; plot(kv, real(muv(1,:)), kv, imag(muv(1,:))); % plot leading Eval

```

Listing 5: (Selection from) `schnakpat/spufu.m`. Modification (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 $\partial_{\bar{u}} f$ 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"
4 p.sol.ds=-0.1; p=setfn(p,'dummy'); p=cont(p,20); % continue just 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=250;
p=schnakinit(p,lx,nx,par); p=setfn(p,'h1D');
p=findbif(p,6); % many bif-points close to each other, use findbif
9 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);

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

```

Listing 6: (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.

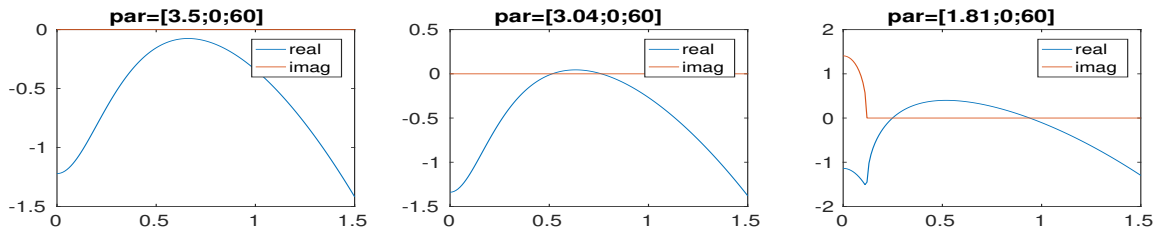


Figure 14: Preparatory step (not strictly necessary) for (32): plotting the dispersion for different λ .

Listing 6 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. 15. We 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 unphysical 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.

In the remainder of `cmds1d` we follow the folds on T1, T3 and T6 as functions of σ , see Fig. 16. 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 $\sigma > \sigma_0$.

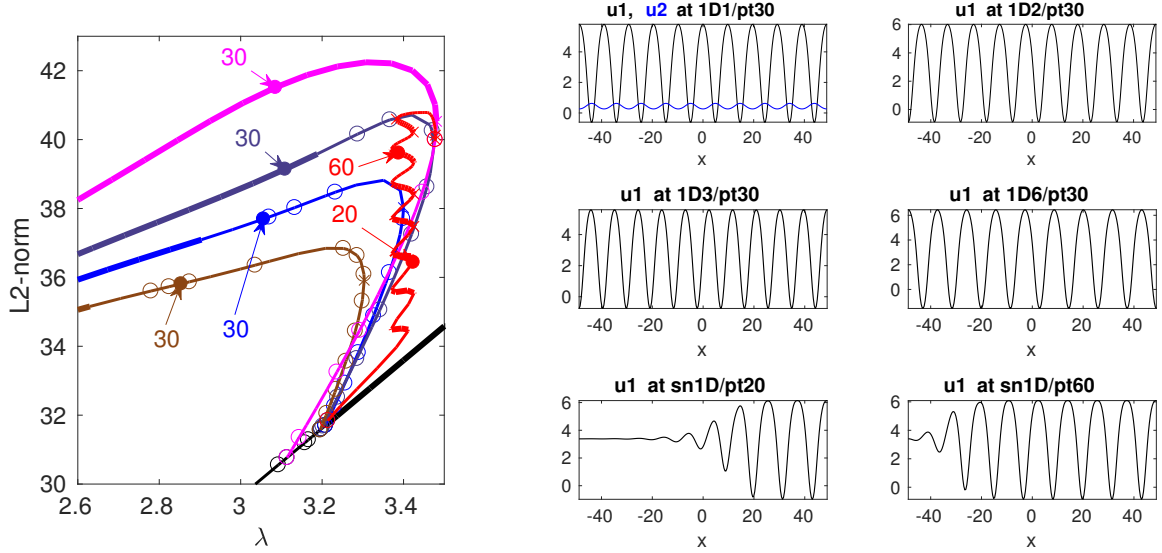


Figure 15: (32), $(\sigma, d) = (-0.6, 60)$, $l_x = 10\pi/kc \approx 48.8132$. Turing branches T1 (blue), T2 (dark blue), T3 (brown) and T6 (magenta), and a snaking branch of a front bifurcating from T1 but reconnecting to T6. In the solution plot of 1D1/pt30 we use the setting `plotsol('1D1','pt30',1,[1 2], 'c1','k','b')`; to plot both components.

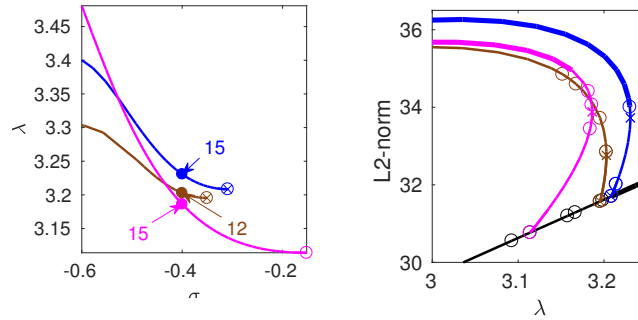


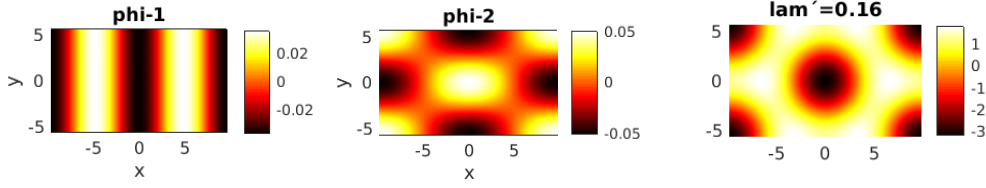
Figure 16: Continuation of folds on T1, T3 and T6 in σ , and behavior of branches at $\sigma = -0.4$.

3.4.2 2D: basic bifurcation diagram, and snaking branches of localized patterns

To compute the 'standard' bifurcation diagram of stripes and hexagons in Fig. 17 we first let $\sigma = 0$ and proceed similarly as for the SH stripes and hexagons in Fig. 6: 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 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. 17. 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`

```
% 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);
5 %% hex via qswibra, continue in both directions
```

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



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

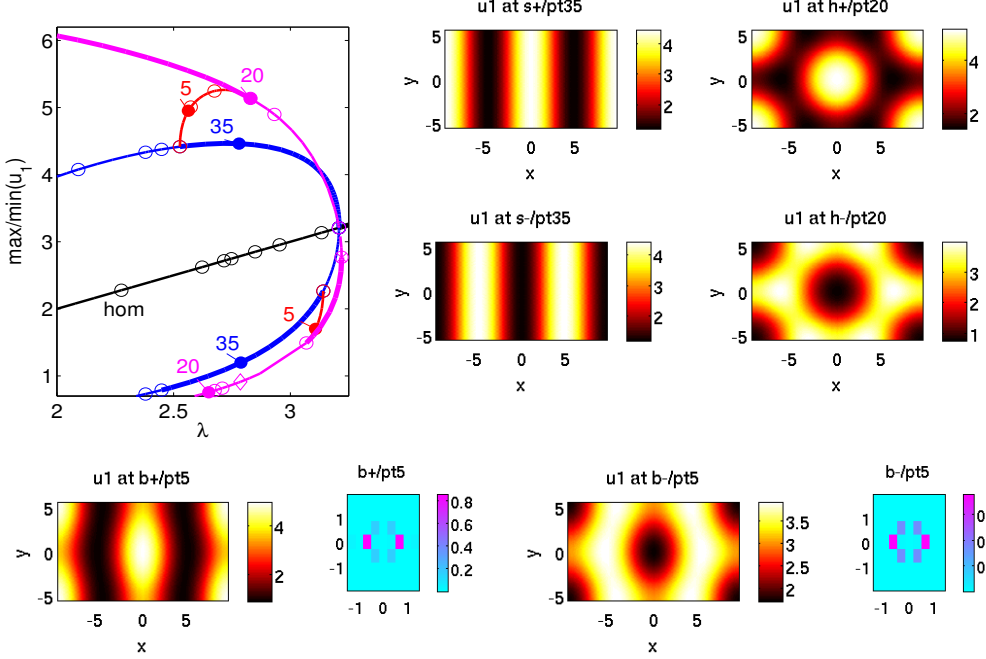


Figure 17: Results from `schnakpat/cmds2da.m` for (32) on a small rectangular domain $\Omega = (-l_x, l_x) \times (-l_y, l_y)$, $l_x = \pi/k_c$, $l_y = l_x/\sqrt{3}$, corresponding to a hexagonal dual lattice. (a) kernel at the first BP, and 'hex' bifurcation direction obtained from `qswibra`. (b) Bifurcation diagram and example solution plots; stripes (blue), hexagons (magenta), and mixed modes or beans (red), including Fourier plots.

```
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
10 p=gentau(p0,1,'s+',2); p.sol.ds=-0.05; p=pmcont(p,20);
```

Listing 7: (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`.

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. 18 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.huha.outfu=@schnakbra`. Here we append the (normalized) L^8 norm $ds\|u\|_8 = \left(\frac{1}{|\Omega|} \int_{\Omega} u^8 dx\right)^{1/8}$ 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.

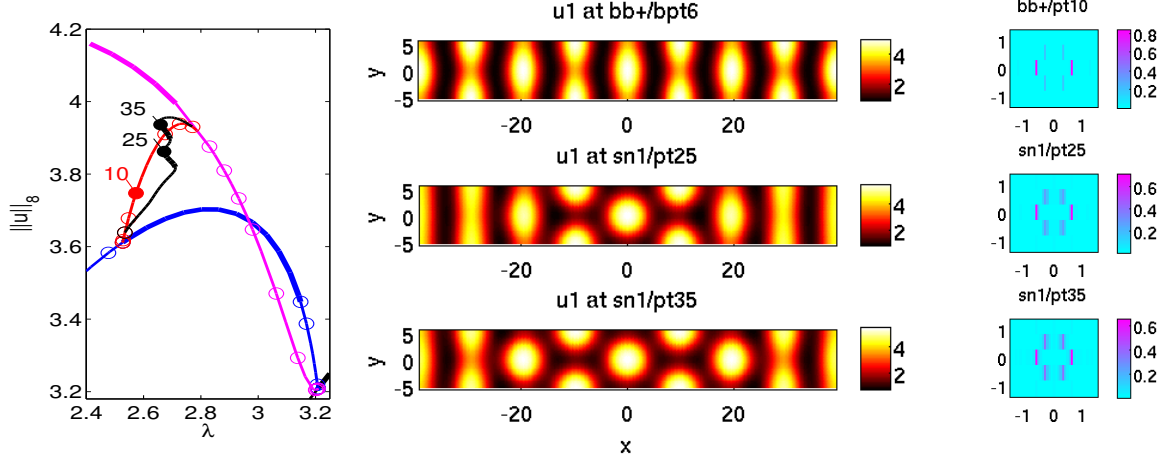


Figure 18: (32) on a long rectangular domain $\Omega = (-l_x, l_x) \times (-l_y, l_y)$, $l_x = 8\pi/k_c$, $l_y = \pi/(\sqrt{3}k_c)$. On the bean branch (red) between stripes (blue) and spots (magenta), there are bifurcation points leading to snaking branches of localized patterns. Right column: Fourier spectrum of $u_1 - \langle u_1 \rangle$.

3.4.3 3D

The analogs of Fig. 8 and Fig. 9 are computed in `cmds3DSC.m` and `cmds3DBCC.m`, and we only remark that:

- For (32) 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 `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.

3.5 Higher degeneracy: Demo hexex

As already said, while `qswibra` and `cswibra` work robustly for most of the example problems we considered, they are not failsafe: the underlying QBE and CBE 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 2, 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

$$G(u, \lambda) := \Delta u + \lambda(u + u^3) = 0 \quad (33)$$

on a hexagon with unit side-lengths and Dirichlet BC. 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), \dots . See Fig. 19(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. [Uec18b, 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 **demos/hexex** is fairly standard, and thus we refrain from detailed comments. The only nonobvious 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 8.

```

classdef hexpdeo < pde % hexpdeo (classdef, modification of stanpdeo2D)
methods(Access = public)
    function o=hexpdeo(hmax) % constructor
        o.grid=grid2D; s3=sqrt(3);
25      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
30 end

```

Listing 8: **hexex/hexpdeo.m**. Using **grid.freeGeometry** to generate a hexagonal domain and coarse mesh, then do some uniform refinement.

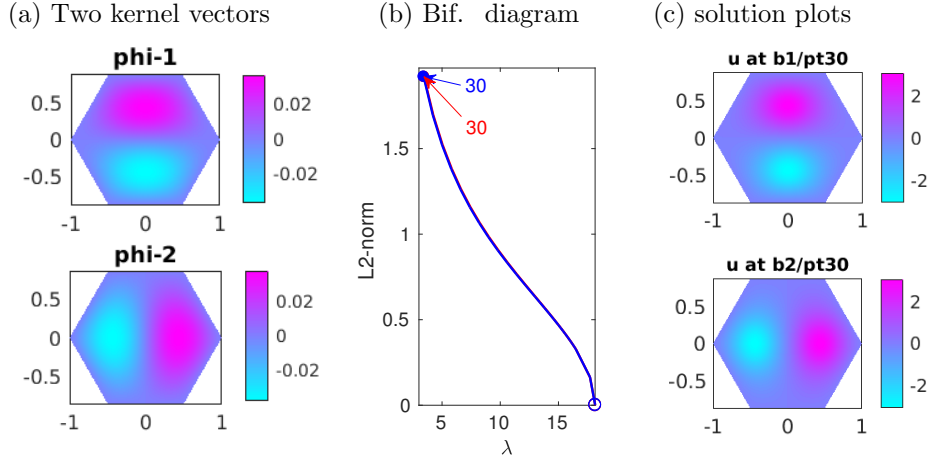


Figure 19: Results from the demo **hexex** for bifurcations at the second BP for (33).

3.6 Demo gcsh: global coupling, with customized linear system and eigenvalue solvers

Interesting phenomena in pattern formation for RD systems (or SH type of equations) can occur under additional nonlocal or global coupling [FCS07, MD14, KT17, Sie18]. Here we explain a setup such that equations of type $M\partial_t u = -G(u)$, respectively the steady version $G(u, \lambda) = 0$, $u : \Omega \rightarrow \mathbb{R}^N$, can augmented by *global* coupling in the fairly general form

$$0 = G(u) + f_{\text{nl}}(u, a), \quad a = \langle h(u) \rangle, \quad (34)$$

where $f_{\text{nl}} : \mathbb{R}^{N+1} \rightarrow \mathbb{R}^N$ and $h : \mathbb{R}^N \rightarrow \mathbb{R}$ are general functions, and $\langle v \rangle = \frac{1}{|\Omega|} \int v(x) dx$ denotes a global average. See Remark 3.2 for comments on *non-local* (but also non-global) coupling. Naturally, f_{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 (34) yields full Jacobians, i.e.,

$$\frac{d}{du} f_{\text{nl}}(u, a)v = \partial_u f_{\text{nl}}(u, a)v + \partial_a f_{\text{nl}}(u, a) \langle h_u(u)v \rangle. \quad (35)$$

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 coupling³. Moreover, we also provide customized interfaces to `eigs`, 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

$$\partial_t u = -(1 + \Delta)^2 u + \lambda u + \nu u^2 - u^3 - \gamma \|u\|^2 u, \quad (36)$$

with parameters $\nu, \gamma \in \mathbb{R}$, $\|u\|^2 := \frac{1}{|\Omega|} \int u^2(x) dx$, and (again) Neumann BC $\partial_n u = \partial_n \Delta u = 0$, extending (3), and for instance considered in [FCS07]. For (36) with $\gamma > 0$, the nonlocal term simply acts as a reduction of the instability parameter λ : steady solutions $u(x; \lambda, \gamma)$ of (36) correspond to steady solutions $u(x; \lambda - \gamma \|u\|^2, 0)$ of (36) with $\gamma = 0$. In particular, for instance the branches of periodic and localized solutions from Figs. 5 and 7 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. 20 for some exemplary results obtained from numerical continuation of (36).

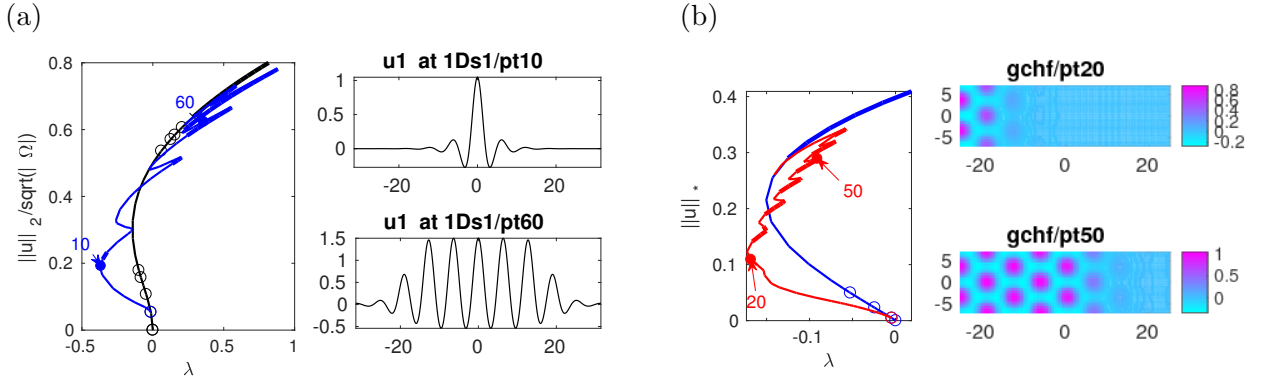


Figure 20: Slanted snaking for the globally coupled 2-3 SH equation (36). (a) $\Omega = (-10\pi, 10\pi)$, $\nu = 2, \gamma = 2$, compare to Fig. 5. (b) $\Omega = (-l_x, l_x) \times (-l_y, l_y)$, $l_x = 8\pi, l_y = 4\pi/\sqrt{3}$, $\nu = 1.3, \gamma = 1$, compare to Fig. 7.

Here we are mainly interested in the efficient implementation of (34) in `pde2path`, and use (36) only as an example, essentially ignoring the scaling relation to (3). Again setting $(u_1, u_2) = (u, \Delta u)$, (36) becomes,

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \partial_t \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} -\Delta u_2 - 2u_2 - (1 - \lambda)u_1 + f(u_1) - \gamma \|u_1\|^2 u_1 \\ -\Delta u_1 + u_2 \end{pmatrix}, \quad (37)$$

$f(u) = \nu u^2 - u^3$, which is of the form (34) with $f_{\text{nl}}(u, a) = \begin{pmatrix} -\gamma a u_1 \\ 0 \end{pmatrix}$ and $h(u) = u_1^2$. On the FEM level we obtain

$$\mathcal{M} \dot{u} = -(\mathcal{K}u - F(u) - \mathcal{M}f_{\text{nl}}(u, a)), \quad (38)$$

³this has also been generalized in the demo `acsuite/acgc`

$\mathcal{M}, \mathcal{K}, F$ as in (21), where $f_{\text{nl}}(u, a)$ now means the vector $f_{\text{nl}}(u, a) = -\gamma a(u_{1,1}, \dots, u_{1,n_p}, 0, \dots, 0)^T$, and where $a = \langle h(u) \rangle$ is evaluated as

$$\langle h(u) \rangle = a_{\text{av}} h(u), \quad a_{\text{av}} = \frac{1}{|\Omega|} \text{sum}(\mathcal{M}), \quad h(u) = (h(u_{1,1}), \dots, h(u_{1,n_p}), 0, \dots, 0). \quad (39)$$

Since here $\mathcal{M} = \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix}$, the last n_p entries of a_{av} are zero anyway, but in a general setup we would rather recommend $a_{\text{av}} = \frac{1}{|\Omega|} (\text{sum}(M), \dots, \text{sum}(M))$ (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 $A_{\text{nl}} = \partial_a f_{\text{nl}}(u, a) \langle h_u(u) \cdot \rangle$ of the Jacobian from (35) becomes

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

which again illustrates that A_{nl} is a full matrix but has but rank 1. The \mathcal{M} in $(\mathcal{M} \partial_a f_{\text{nl}})$ comes from the \mathcal{M} multiplying f_{nl} in (38), while the \mathcal{M} in a_{jac} is as in (39).

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

$$z = K^{-1}b + \alpha(K^{-1}c)(a^T K^{-1}b), \quad \alpha = \frac{1}{1 - a^T K^{-1}c}. \quad (40)$$

Thus, the idea is as follows: in `sGjac` we separately assemble the sparse part $G_u(u) + \partial_u f_{\text{nl}}$ of G_u corresponding to K in (40), and the vectors $c = \partial_a f_{\text{nl}}$ and a_{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 $a = a_{\text{jac}}$, these are passed via the global `pde2path` struct `p2pglob`. Table 4 lists the pertinent files from demo `gcsh`, and some functions from `libs` which have not yet been documented otherwise, and Listings 9–11 show the essential modifications compared to the demo `sh`.

Table 4: 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
<code>cmds1d</code> , <code>cmds2dhexfro</code> <code>shinit</code>	scripts for 1D and 2D examples, see Fig. 8 initialization, sets <code>p.fuha.lss=@gclss</code> ; <code>p.fuha.blss=@gcblls</code> ; as linear system solvers
<code>oosetfemops</code> <code>sG,nodalf,sGjac</code>	set FEM matrices, also stores the 'averaging vector' <code>p.avec</code> encodes G with 'nonlinearity' in <code>nodalf</code> , and Jacobian; <code>sG</code> also sets the <code>global cvec</code> , and <code>sGjac</code> the <code>global avjvec</code> needed by the linear system solvers.
<code>shbra1d</code> <code>fnl</code> , <code>hfu</code> <code>fnljac</code> , <code>hjacc</code> , <code>fnl_a</code>	modification of <code>stanbra</code> for putting the normalized L^2 on the branch functions f_{nl} and h for global coupling from (36) Jacobians of f_{nl} and h , and $\partial_a f_{\text{nl}}$.
<code>gclss</code> <code>gcblls</code>	implements (40), with vectors $c = \text{cvec}$ and $a = \text{avec}$ passed in <code>p2pglob</code> implements the version of (40) for the bordered systems of arclength continuation; here a and c are augmented by a single 0
<code>gclsseigs</code>	version of (40) for <code>eigs</code> (inverse vector iteration); uses <code>lsslueigs</code> to solve $Az=b$, where A is LU -prefactored due to many repeated solves with the same A .
<code>gcacfun</code> <code>lsslueigs</code>	interface routine for <code>eigs</code> which contains the actual call to <code>gclsseigs</code> version of <code>lsslu</code> which stores LU factorizations as <code>globals</code> .

```
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=@gcblls; % Sherman-Morrison versions
p.sw.runpar=0; % switch off parfor in pmnewtonloop (clashes with global vars)
5 p.sw.eigssol=1; % use Sherman-Morrison in eigs (gclsseigs)
```

Listing 9: `gcsh/shinit.m` (lines 1-6). The differences to `sh/shinit.m` are in lines 3-5.


```

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

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

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

```

Listing 10: `gcsh/sG.m`, `gcsh/fnl.m` and `gcsh/hfu.m`, which are straightforward, as are the Jacobians and `fnl_a`. `p.avevec` is precomputed in `oosetfemops`.

```

function Gu=sGjac(p,u) % jac for SH with GC
global p2pglob; % p2pglob.avevec, cvec computed here and used in, e.g., gclss
hj=hjac(p,u); p2pglob.avevec=sum(p.mat.M*spdiags(hj,0,p.nu,p.nu))./p.0m;
p2pglob.cvec=p.mat.M*fnl_a(p,u);
5 [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 11: `gcsh/sGjac.m`. In `sG`, the term `fnl` is first added to the ‘nonlinearity’ f , and then multiplied by $\mathcal{M}=\mathbf{p.mat.M}$. 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 (36) are as for (3), see `gcsh/cmds1d.m` and `gcsh/cmds2dhexfro.m`, and the computations for (36) run almost as fast as for (3). In 3D (or, more generally, for large n_u) it turns out that combining `gclssseigs` with iterative linear system solvers yields further speed advantages, but this will be described elsewhere.

Remark 3.2. More general nonlocal couplings are often given as $f_{\text{nl}}(u)(x) = \int_{\Omega} \kappa(x, \xi) h(u(\xi)) d\xi$, with a kernel $\kappa : \Omega \times \Omega \rightarrow \mathbb{R}$, often of the form $\kappa(x, \xi) = k(x - \xi)$, specifically with Gaussians $k_{\text{Gauss}}(y) = \alpha_1 e^{-\|y\|^2/\alpha_2}$. 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_{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. \square

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.
- [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.
- [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(5), 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.
- [DRUW14] T. Dohnal, J. Rademacher, H. Uecker, and D. Wetzel. pde2path - V2: faster FEM and periodic domains, 2014.
- [dW17] H. de Witt. Fold and branch point continuation in pde2path – a tutorial for systems, 2017.
- [dW18] H. de Witt. Beyond all order asymptotics for localized patterns and homoclinic snaking in a Schnakenberg model, Preprint, 2018.
- [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.
- [FCS07] W. J. Firth, L. Columbo, and A. J. Scroggie. Proposed resolution of theory-experiment discrepancy in homoclinic snaking. *PRL*, 99, 2007.
- [GS02] M. Golubitsky and I. Stewart. *The symmetry perspective*. Birkhäuser, Basel, 2002.
- [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.
- [KC13] G. Kozyreff and S.J. Chapman. Analytical results for front pinning between an hexagonal pattern and a uniform state in pattern-formation systems. *Phys. Rev. Letters*, 111(5):054501, 2013.
- [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.
- [KT17] C. Kühn and S. Throm. Validity of amplitude equations for non-local non-linearities, 2017.
- [LVE09] M. Leda, V. K. Vanag, and I. R. Epstein. Instability of a three-dimensional localized spot. *PRE*, 80:066204, 2009.
- [MD14] D. Morgan and J.H.P. Dawes. The Swift–Hohenberg equation with a nonlocal nonlinearity. *Physica D*, 270:60–80, 2014.
- [Mei00] Z. Mei. *Numerical bifurcation analysis for reaction-diffusion equations*. Springer, 2000.
- [Mur89] J. D. Murray. *Mathematical Biology*. Springer, Berlin, 1989.
- [Pis06] L.M. Pismen. *Patterns and interfaces in dissipative dynamics*. Springer, 2006.

- [PTVF07] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007.
- [RU18] J. Rademacher and H. Uecker. The OOPDE setting of pde2path – a tutorial via some Allen-Cahn models, 2018.
- [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.
- [SU17] G. Schneider and H. Uecker. *Nonlinear PDE – a dynamical systems approach*, volume 182 of *Graduate Studies Mathematics*. AMS, 2017.
- [Uec16] H. Uecker. Optimal harvesting and spatial patterns in a semi arid vegetation system. *Natural Resource Modelling*, 29(2):229–258, 2016.
- [Uec18a] H. Uecker. Hopf bifurcation and time periodic orbits with pde2path – algorithms and applications, *Comm. in Comp. Phys*, to appear, 2018.
- [Uec18b] H. Uecker. Multiple bifurcation points in pde2path, 2018.
- [Uec18c] H. Uecker. www.staff.uni-oldenburg.de/hannes.uecker/pde2path, 2018.
- [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.
- [UWR14] H. Uecker, D. Wetzel, and J. Rademacher. pde2path – a Matlab package for continuation and bifurcation in 2D elliptic systems. *NMTMA*, 7:58–106, 2014.
- [Wet16] D. Wetzel. Pattern analysis in a benthic bacteria-nutrient system. *Math. Biosci. Eng.*, 13(2):303–332, 2016.
- [ZUFM17] Y. Zelnik, H. Uecker, U. Feudel, and E. Meron. Desertification by front propagation? *Journal of Theoretical Biology*, pages 27–35, 2017.