The ampsys tool of pde2path

Hannes Uecker[*], Daniel Wetzel[†]

June 3, 2019

Abstract

The computation of coefficients of amplitude systems for Turing bifurcations is a straightforward but sometimes elaborate task, in particular for 2D or 3D wave vector lattices. The Matlab tool ampsys automates such computations for two classes of problems, namely scalar equations of Swift–Hohenberg (SH) type and generalizations, and reaction–diffusion systems with an arbitrary number of components. The tool is designed to require minimal user input, and for a number of cases can also deal with symbolic computations. After a brief review of the setup of amplitude systems we explain the tool by a number of 1D, 2D and 3D examples over various wave vector lattices.

1 Introduction

The Turing bifurcation in a pattern forming system close to onset is usually described by (systems of) amplitude equations (AEs), also called Landau equations. These are ODEs for the amplitudes of the critical modes, and their derivation, based on center–manifold reduction or Liapunov-Schmidt reduction, is essentially a mechanical task, but may become elaborate if the bifurcation is of higher multiplicity, e.g., due to symmetries of the domain in higher space dimensions. Such symmetries and the associated AEs have been classified and analyzed in detail, see [GS02, Hoy06] and the references therein. Typical exampes include, e.g., wave-vector lattices of square and hexagonal type in two space dimensions (2D), see Fig. 1, and simple cubes (SCs), face centered cubes (FCCs) and body centered cubes (BCCs) in 3D.

Given a bifurcation problem as above, the tool ampsys, included in pde2path [Uec19c], can be used to compute the coefficients of the AEs with minimal user input. We proceed by example, and illustrate the usage of ampsys to compute the AEs for Swift–Hohenberg (SH) type scalar equations, and for reaction–diffusion systems (RDS), over 1D, 2D and 3D domains corresponding to various wave-vector lattices. The class of SH type equations is of the form

$$\partial_t u = Lu + \lambda u + c_2 u^2 + c_3 u^3, \quad Lu = -(1+\Delta)^2 u \ \text{ (or similar)}, \tag{1}$$

where $u = u(x,t) \in \mathbb{R}$, $t \geq 0$, $x \in \mathbb{R}^d$, where $\lambda \in \mathbb{R}$ is the bifurcation parameter, and where c_2, c_3 can either be real coefficients, or operators such as $(\partial_{x_1} + \ldots + \partial_{x_d})u^2$, thus including Kuramoto-Sivashinsky (KS) type of equations. The RD systems are of the form

$$u_t = D\Delta u + f(u), \tag{2}$$

where $u \in \mathbb{R}^N$ ($N \geq 2$ components), $D \in \mathbb{R}^{N\times N}$ is a diffusion matrix, and $f : \mathbb{R}^N \to \mathbb{R}^N$.

Essentially, the user has to provide (for the RD class)

[*]Institut für Mathematik, Universität Oldenburg, D-26128 Oldenburg, Germany; hannes.uecker@uol.de
[†]danieldwetzel@gmail.com

- The diffusion matrix D and a function handle for f.
- a (spatially homogeneous) steady state u^*, and the parameter value(s) where the Turing bifurcation occurs;
- the critical wave number, and a choice of a wave-vector lattice for the amplitudes.

For the SH class, the setup is even simpler and the user has to provide

- the symbol of the linear operator L in Fourier space such as $\hat{L}\hat{u} = -(1-|k|^2)$, and the coefficients c_2 and c_3.

In both cases, this data is to be put into a matlab struct, for simplicity in the following called `p` as in problem. Then calling `[Q,C,c1,phi]=ampsys(p)` returns

- the coefficients (or selected coefficients) up to third order in the AEs for the given lattice, and (optionally) the critical eigenvector.

Setup: `ampsys` is included in the numerical continuation and bifurcation package `pde2path`[UWR14, Uec19c], with the demo files in `demos/asdemos`. In this case, the `Matlab` path to `ampsys` is already set together with the path to all other `pde2path` library functions via `setpde2path` [dWDR+18]. However, `ampsys`can also be downloaded from [Uec19c] as a standalone tool, in which case the path is set by calling `setastool` in the `ampsys` root directory, with the demos in the subdirectory `asdemos`.

Remark 1.1. Panels (a) and (b) of Fig. 1 show the square and hexagonal lattices (first three 'layers'). For such lattice problems, the justification of the amplitude equations we compute follows from center–manifold reduction. (c) illustrates the quasi-lattice generated by $\vec{k} = \pm e^{ij\pi/4}, j = 0, \ldots, 3$. For this,the nonlinearity generates wave vectors arbitrary close to the critical circle (the full lattice is dense in $\mathbb{R}^2$), and thus the justification of amplitude equations (or the right truncation order) becomes a small divisor-problem. See, e.g., [IR10] and the references therein. In `ampsys`, we compute the amplitude equations to third order, and thus the distinction between lattices and quasi-lattices plays no role. ⌋

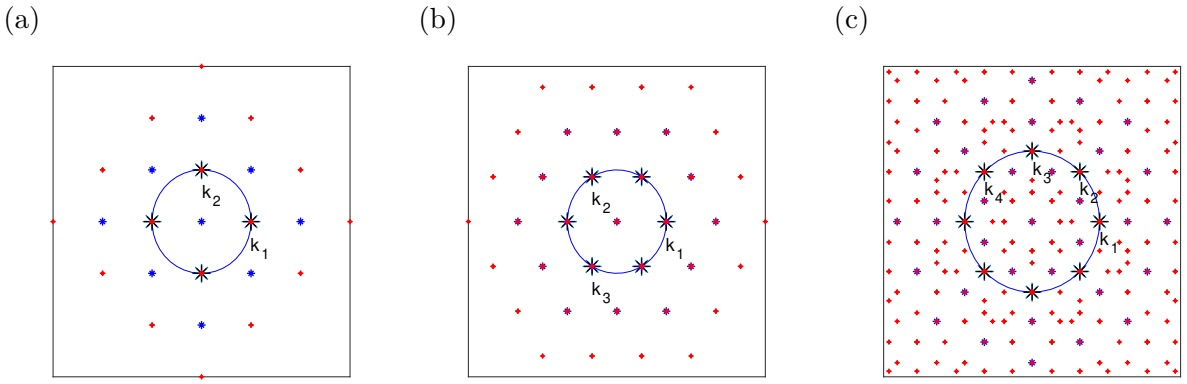

Figure 1: First three layers (wave vectors generated by terms up to cubic order) for (a) Square lattice; (b) hexagonal lattice; (c) 8-fold quasilattice. The thick black stars are the basic wave vectors (on the critical circle), the blue (red) dots are generated by quadratic (cubic) interactions.

Acknowledgment. The work of DW was supported by the DFG under Grant No. 264671738.

2 Some amplitude systems on simple lattices as analytic examples

We briefly review the amplitude formalism (AF); users familiar with Turing bifurcations and the AF, and mainly interested in the setup and use of `ampsys`, can safely skip this section. Moreover, we restrict

to just the formal derivation of the AEs, and except for a few remarks refer to [Hoy06, SU17, Uec19b] and the references therein for justification and further conclusions, for instance on special solutions of the AEs. See also, e.g., [UW14, Wet18, UW19] for comparisons of the solutions constructed via the AF with (numerical) solutions of the associated full pattern forming systems.

2.1 The quadratic-cubic Swift–Hohenberg equation

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

$$\partial_t u = -(1+\Delta)^2 u + \lambda u + c_1 u^2 + c_3 u^3, \quad u = u(x,t) \in \mathbb{R},\ x \in \Omega, \tag{3}$$

where $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$ (1D, 2D and 3D case, respectively), with instability parameter $\lambda \in \mathbb{R}$, nonlinearity parameters $c_{2,3} \in \mathbb{R}$, and, if $\Omega \neq \mathbb{R}^d$, boundary conditions (BC), for instance of the form $\partial_n u|_{\partial\Omega} = \partial_n(\Delta u)|_{\partial\Omega} = 0$. The original (cubic) SH model [SH77] corresponds to $c_2 = 0$ and $c_3 = -1$, while the case $f(u) = \tilde{c}_3 u^3 - u^5$ instead of $f(u) = c_2 u^2 - u^3$ 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 systems [CH93, Pis06, SU17].

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 $v(x,t) = \mathrm{e}^{\mathrm{i}k\cdot x + \mu(k)t}$, $k \in \mathbb{R}^d$, where

$$\mu(k,\lambda) = -(1-|k|^2)^2 + \lambda, \quad |k|^2 := k_1^2 + \ldots + k_d^2. \tag{4}$$

Thus, $u^* \equiv 0$ is asymptotically stable for $\lambda < 0$, and unstable for $\lambda > 0$ with respect to periodic waves with wave vector k with $|k| = k_c = 1$.

Remark 2.1. Besides the 'classical' SH equation (3), with dispersion relation (4), we can also consider equations for which the linearization shows simultaneous instabilities at different $|k|$, e.g.,

$$Lu = -(1+\Delta)^2(1+q^{-2}\Delta)^2 u + \lambda u, \tag{5}$$

where wlog $q > 1$, with dispersion relation

$$\mu(k,\lambda) = -(1-|k|^2)^2(1-q^{-2}|k|^2)^2 + \lambda. \tag{6}$$

Similar problems are often used as toy models for quasicristals, cf., e.g., [SAKR16], and, in 2D and 3D, allow multitudes of interesting wave vector interactions and associated (quasi) patterns. In 1D, we essentially only have to distinguish the cases $q \in \{2,3\}$ (resonant case) or $q \notin \{2,3\}$. We come back to this in §3.2. ⌋

2.1.1 1D

For solutions of (3) in 1D we make the ansatz

$$u(t,x) = \varepsilon\Psi_A(t,x) := \varepsilon A_1(T)e_1 + \varepsilon^2\left[\frac{1}{2}A_0(T)e_0 + A_2(T)e_2\right] + \text{c.c.} + \text{h.o.t}, \quad e_j = \mathrm{e}^{\mathrm{i}jx}, \tag{7}$$

where the amplitude scaling ε is introduced as $\varepsilon^2 = |\lambda - \lambda_c| = |\lambda|$ (since $\lambda_c = 0$), such that ε^2 is the distance from criticality. In general (with λ a generic name for a bifurcation parameter) we shall use the expansion

$$\mu(\lambda) = \mu'(\lambda_c)(\lambda - \lambda_c) + \mathcal{O}(|\lambda - \lambda_c|^2) =: c_1(\lambda - \lambda_c) + \mathcal{O}(|\lambda - \lambda_c|^2) \tag{8}$$

for the critical eigenvalue. For (3) this just gives $c_1 = 1$. The amplitudes $A_j = A_j(T) \in \mathbb{C}$ in (7) depend on the slow time $T = \varepsilon^2 t$, c.c. stands for the complex conjugate of the preceeding terms to

obtain real valued u, and h.o.t denotes higher order terms which are not relevant for the present computation. The c.c. of, e.g., $A_1 e_1$ is also conveniently written as $A_{-1}e_{-1}$.

The *residual* of a given ansatz is defined as $\mathrm{Res}(u) = -\partial_t u + Lu + f(u)$, and the goal is to choose the ansatz such that the residual formally becomes small. Plugging (7) into (3) we first obtain the $\mathcal{O}(\varepsilon^2)$ terms

$$\mathrm{Res}(u) = \varepsilon^2 \left(-A_0 e_0 - 9A_2 e_2 + c_2(2|A_1|^2 e_0 + 2A_2^2 e_2) + \text{c.c.}\right) + \mathcal{O}(\varepsilon^3).$$

Importantly, we can solve at the modes e_0 and e_2 to obtain

$$A_0 = 2c_2|A_1|^2 \text{ and } A_2 = \frac{1}{9}c_2 A_1^2. \tag{9}$$

Then collecting terms at $\mathcal{O}(\varepsilon^3 e_1)$ yields the AE

$$\frac{\mathrm{d}}{\mathrm{d}T}A_1 = A_1(c_1\lambda\varepsilon^{-2} + c_{31}|A_1|^2) \text{ with } c_1 = \mu'(0) = 1 \text{ and } c_{31} = 3c_3 + \frac{38}{9}c_2^2. \tag{10}$$

(10) predicts the bifurcation of 'stripes' $|A_1| = \sqrt{\lambda/c_{31}}$, where $\lambda < 0$ for $c_{31} < 0$ (subcritical bifurcation), or $\lambda > 0$ ($c_{31} > 0$, supercritical case). The phase of A_1 is free, and determined by the BC for (3). After deriving (10) we can either simply set $\varepsilon = 1$, or rescale $\lambda = \varepsilon^2\tilde{\lambda}$, to obtain an amplitude equation independent of ε.

An ε–scaling as in (7) is not always possible in a consistent way (see below), but if it is, then it is useful as the subsequent derivation of expressions for A_0, A_2 and $\frac{\mathrm{d}}{\mathrm{d}T}A_1$ is just a matter of sorting wrt the modes e_j and powers of ε. If we omit the ε scaling, then we essentially need to sort wrt the modes e_j and powers of A_1, but identification of "equal powers" may be somewhat ambiguous, see below.

2.1.2 2D

In 2D, the most prominent wave vector lattices are (cf. Fig. 1),

- squares, given by, for instance the two wave vectors $k_1 = (1, 0)$ and $k_2 = (0, 1)$;
- hexagons, given by, e.g., $k_1 = (1, 0)$, $k_2 = \frac{1}{2}(-1, \sqrt{3})$ and $k_3 = \frac{1}{2}(-1/2, -\sqrt{3})$.

The crucial difference between the two is that for squares the quadratic interaction of critical modes only gives stable modes, i.e., modes off the critical circle $|k| = 1$ such that quadratic terms can be removed from the residual as in (9). The hexagon lattice supports *quadratic resonances*, e.g., $k_1 = -k_2 - k_3$. As a consequence, quadratic terms can in general not be removed from the residual, but must be kept in the amplitude equations. The same distinction will appear in 3D between, e.g., simple cube (SC) lattices and body centered cube (BCC) lattices.

Squares. We let $e_{m,n} = \mathrm{e}^{\mathrm{i}(mx+ny)}$, and make the ansatz

$$u = \varepsilon(A_1 e_{1,0} + A_2 e_{0,1}) + \varepsilon^2 \left[\frac{1}{2}A_0 + A_{1,1}e_{1,1} + A_{-1,1}e_{-1,1} + A_{2,0}e_{2,0} + A_{0,2}e_{0,2}\right] + \text{c.c.} + \text{h.o.t}, \tag{11}$$

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

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

$$A_0 = 2c_2(|A_1|^2 + |A_2|^2), \quad A_{2,0} = \frac{c_2}{9}|A_1|^2, \quad A_{0,2} = \frac{c_2}{9}|A_2|^2, \quad A_{1,1} = 2c_2 A_1 A_2, \quad A_{-1,1} = 2c_2 A_{-1} A_2,$$

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

$$\frac{\mathrm{d}}{\mathrm{d}T}\begin{pmatrix}A_1\\A_2\end{pmatrix} = \begin{pmatrix}A_1(c_1\lambda\varepsilon^{-2} + c_{31}|A_1|^2 + c_{32}|A_2|^2)\\A_2(c_1\lambda\varepsilon^{-2} + c_{31}|A_2|^2 + c_{32}|A_1|^2)\end{pmatrix}, \; c_1 = 1, \; c_{31} = 3c_3 + \frac{38}{9}c_2^2, \; c_{32} = 6c_3 + 12c_2^2. \tag{12}$$

Note that the nonlinear coefficients in the second equation are obtained by flipping A_1 and A_2, while the linear coefficient c_1 is the same for all modes, due to the rotational invariance of Δ.

Hexagons. On a hexagonal grid, a natural ansatz is

$$u(x,t) = A_1(t)e_1 + A_2(t)e_2(t) + A_3(t)e_3 + \text{c.c.} + \text{h.o.t}, \tag{13}$$

where $e_j = \mathrm{e}^{\mathrm{i}k_j\cdot x}$, but where the ε–scaling used before is omitted. The reason is that a consistent ε–scaling leading to AEs at third order is only possible if the quadratic terms (c_2u^2 in (3)) are small, i.e., $c_2 = \mathcal{O}(\varepsilon)$. Plugging (13) into (3) we obtain for instance the term $2c_2\overline{A_2}\,\overline{A_3}$ at e_1, i.e., in the equation for A_1, and since $L(|k_c|)$ is not invertible we can no longer remove it. Thus, we need to keep it, and altogether the amplitude system to third order reads

$$\begin{aligned}
\dot{A}_1 &= c_1\lambda A_1 + c_{21}\overline{A_2A_3} + c_{31}|A_1|^2A_1 + c_{32}(|A_2|^2+|A_3|^2)A_1,\\
\dot{A}_2 &= c_1\lambda A_2 + c_{21}\overline{A_1A_3} + c_{31}|A_2|^2A_3 + c_{32}(|A_1|^2+|A_3|^2)A_2,\\
\dot{A}_3 &= c_1\lambda A_3 + c_{21}\overline{A_1A_2} + c_{31}|A_3|^2A_3 + c_{32}(|A_1|^2+|A_2|^2)A_3,
\end{aligned} \tag{14}$$

where

$$c_1 = 1, \quad c_{21} = 2c_2 + \mathcal{O}(|\lambda c_2|), \quad c_{31} = 3c_3 + \mathcal{O}(|\lambda| + c_2^2), \quad c_{32} = 6c_3 + \mathcal{O}(|\lambda| + c_2^2). \tag{15}$$

In (15), λ is not scaled, but we treat $|\lambda|$ and in particular c_2 as small, to have a *consistent* expansion. However, if we just keep all terms up to third order in A_j, then

$$c_{31} = 3c_3 + \frac{38}{9}c_2^2, \quad c_{32} = 6c_3 + 9c_2^2. \tag{16}$$

In §3 we recover the formulas (12), and (15), and the *inconsistent* version (16), and their 3D analogs, with `ampsys`. For this we provide the switch `p.cons`(istency), where `p.cons=0` (default setting) corresponds to the case (16), while `cons=1` yields $c_{31} = 3c_3$ and $c_{32} = 6c_3$.

2.2 The Brusselator

As an example of an RD system we consider the Brusselator [PL68]

$$\begin{aligned}
\partial_t u &= a - (b+1)u + u^2v + D_1\Delta u,\\
\partial_t v &= bu - u^2v + D_2\Delta v,
\end{aligned} \tag{17}$$

where $u = u(x,t)$ and $v = v(x,t)$ correspond to chemical concentrations of an activator and inhibitor, respectively, D_1 and D_2 are their diffusivities, and A and B are parameters. Homogeneous steady states are given by $u^* = a$ and $v^* = b/a$. We set

$$D_1 = 1 \text{ and } D_2 = (a/R)^2, \tag{18}$$

with $R > 0$ a convenient 'unfolding' parameter (see below), and consider b as the primary bifurcation parameter. $U^* = (u^*, v^*)$ is stable for

$$B < B_c = (1+R)^2, \tag{19}$$

where a Turing bifurcation occurs with critical wavenumber $k_c = \sqrt{R/D_1}$.

Taylor expanding $f = (a-(b+1)u+u^2v, au-u^2v)^T$ at U^* to third order and setting $w = (\tilde{u},\tilde{v}) = (u,v) - (u^*,v^*)$ yields

$$\partial_t w = L(\Delta)w + B(w,w) + C(w,w,w), \tag{20}$$

where $L(\Delta) = J_f + \left(\begin{smallmatrix} \Delta & 0 \\ 0 & D_2\Delta \end{smallmatrix}\right)$, with $J_f = J_f(w^*)$ the Jacobian at w^*, and B and C are symmetric bilinear and trilinear forms, respectively. For $p, q, r \in \mathbb{R}^2$ they have the form

$$\begin{aligned} B(p,q) &= \frac{1}{2} f_{uv}(w^*)(p_1 q_2 + p_2 q_1) + \frac{1}{2}(f_{uu}(w^*) p_1 q_1 + f_{vv}(w^*) p_2 q_2), \\ C(p,q,r) &= \frac{1}{6}(f_{uuu}(w^*) p_1 q_1 r_1 + f_{vvv}(w^*) p_2 q_2 r_2) \\ &\quad + \frac{1}{6}\big(f_{uuv}(w^*)(p_1 q_1 r_2 + r_1 p_1 q_2 + q_1 r_1 p_2) + f_{uvv}(w^*)(p_1 q_2 r_2 + r_1 p_2 q_2 + q_1 r_2 p_2)\big). \end{aligned}$$

As a compromise between overly simple and too elaborate computations, we again choose the 2D square lattice $k_1 = k_c(1,0)$, $k_2 = k_c(0,1)$ of critical wave vectors, and make the ansatz

$$w = \varepsilon \sum_{j=1}^{2} A_j e_j \phi + \varepsilon \sum_{j=-2}^{-1} A_j e_j \phi, \tag{21}$$

where again $\varepsilon > 0$ is defined via $\varepsilon^2 := b - b_c$ (distance from criticality), $e_j = e^{\mathrm{i} k_j \cdot x}$, $k_1 = k_c(1\ 0)^T$, $k_2 = k_c(0\ 1)^T$, $k_{-1} = -k_1$, $k_{-2} = -k_2$, and ϕ is the eigenvector of $\hat{L}(k_c)$ corresponding to the zero-eigenvalue $\mu(b_c, k_c)$. By default we normalize ϕ by $\phi_1 = 1$. Letting $I = \{-2. -1, 1, 2\}$ we obtain

$$B(w,w) = \varepsilon^2 \sum_{i \in I} \sum_{j \in I} A_i A_j e_i e_j B(\phi, \phi), \tag{22}$$

and to remove the quadratic terms (22) from the residual, we extend the ansatz (21) to

$$w = \varepsilon \sum_{j \in I} A_j e_j \phi + \varepsilon^2 \sum_{i \in I} \sum_{j \in I} d_{ij} A_i A_j e_i e_j. \tag{23}$$

We determine d_{ij} by substituting (23) into (20) and collecting terms at $\mathcal{O}(\varepsilon^2)$, i.e.,

$$\begin{aligned} &d_{ij} = -\hat{L}^{-1}(2k_c) B(\phi,\phi) \text{ for } i = j, \\ &d_{ij} = -\hat{L}^{-1}(0) B(\phi,\phi) \quad \text{for } (i,j) \in \{(1,-1),(-1,1),(2,-2),(-2,2)\}, \\ &d_{ij} = -\hat{L}^{-1}(\sqrt{2}k_c) B(\phi,\phi) \text{ for } (i,j) \in \{(\pm 1, \pm 2), (\pm 2, \pm 1)\}. \end{aligned} \tag{24}$$

To remove terms of order $\varepsilon^3 e_i$ from the residual, we extend the ansatz (23) to

$$w = \varepsilon \sum_{j \in I} A_j e_j \phi + \varepsilon^2 \sum_{i \in I} \sum_{j \in I} d_{ij} A_i A_j e_i e_j + \varepsilon^3 \sum_{i \in I} \phi_{3i} e_i. \tag{25}$$

Substituting (25) into (20) and sorting with respect to $\varepsilon^3 e_1,\ \varepsilon^3 e_2,\ \varepsilon^3 e_3,\ \varepsilon^3 e_4$, yields

$$\begin{aligned} \varepsilon^3 e_1: \quad -\hat{L}(k_c)\phi_{31} &= -\partial_T A_1 + \tilde{c}_1 (b - b_c)\varepsilon^{-2} A_1 + \tilde{c}_2 A_1 A_1 A_{-1} + \tilde{c}_3 A_1 A_2 A_{-2}, \\ \varepsilon^3 e_2: \quad -\hat{L}(k_c)\phi_{32} &= -\partial_T A_2 + \tilde{c}_1 (b - b_c)\varepsilon^{-2} A_2 + \tilde{c}_2 A_2 A_1 A_{-1} + \tilde{c}_3 A_2 A_2 A_{-2}, \end{aligned} \tag{26}$$

and similar for A_{-1}, A_{-2} with

$$\begin{aligned} \tilde{c}_1 &= \partial_b \mu(b_c)\phi, \\ \tilde{c}_2 &= 3C(\phi,\phi,\phi) + 2\left(B(\phi, d_{11}) + B(\phi, d_{1-1}) + B(\phi, d_{-11})\right) \\ &= 3C(\phi,\phi,\phi) + 2B(\phi, -\hat{L}^{-1}(2k_c)B(\phi,\phi)) + 4B(\phi, -\hat{L}^{-1}(0)B(\phi,\phi)), \\ \tilde{c}_3 &= 6C(\phi,\phi,\phi) + 2\left(B(\phi, d_{2-2}) + B(\phi, d_{-22}) + B(\phi, d_{1-2}) + B(\phi, d_{-21}) + B(\phi, d_{12}) + B(\phi, d_{21})\right) \\ &= 6C(\phi,\phi,\phi) + 4B(\phi, -\hat{L}^{-1}(0)B(\phi,\phi)) + 8B(\phi, -\hat{L}^{-1}(\sqrt{2}k_c)B(\phi,\phi)). \end{aligned} \tag{27}$$

By the Fredholm alternative there exists a solution for (26) if the right hand sides of (26) are in $\ker(\hat{L}(k_c)^H)^\perp$. Thus, let ψ be the adjoint eigenvector of $\hat{L}(k_c)$ to the zero-eigenvalue, i.e., $\hat{L}(k_c)^H\psi = 0$, normalized such that $\langle\phi,\psi\rangle = 1$. The scalar products of (26) with ψ then yield

$$\begin{aligned}\partial_T A_1 &= c_1(b-b_c)\varepsilon^{-2}A_1 + c_2 A_1 A_1 A_{-1} + c_3 A_1 A_2 A_{-2},\\ \partial_T A_2 &= c_1(b-b_c)\varepsilon^{-2}A_2 + c_2 A_2 A_2 A_{-2} + c_3 A_2 A_1 A_{-1},\end{aligned} \tag{28}$$

with

$$c_i = \langle\tilde{c}_i,\psi\rangle. \tag{29}$$

Using $A_1 = \overline{A}_{-1}$, $A_2 = \overline{A}_{-2}$, and returning to unscaled amplitudes A_1, A_2 and renaming $c_2 = c_{31}$, $c_3 = c_{32}$, we may write this as

$$\begin{aligned}\partial_T A_1 &= c_1(b-b_c)A_1 + c_{31} A_1|A_1|^2 + c_{32} A_1|A_2|^2,\\ \partial_T A_2 &= c_1(b-b_c)A_2 + c_{31} A_2|A_2|^2 + c_{32} A_2|A_1|^2.\end{aligned} \tag{30}$$

Remark 2.2. a) (30) shows the structure of the amplitude equations, which is completely as in (12), while the coefficients need to be computed from (24), (27) and (29). This is, essentially, what `ampsys` does, for any choice of wave vector lattices, including 3D cases. For the Brusselator, this has been done analytically for a number of lattices. For instance, for the square lattice we obtain

$$c_1 = \frac{a^2}{(1+R)(a^2-R^2)}, \quad c_{31} = \frac{-8+38R+5R^2-8R^3}{9R(a^2-R^2)}, \quad c_{32} = 2c_{31}, \tag{31}$$

b) For the hexagonal lattice, if we substitute the ansatz corresponding to (13) into B, we obtain terms of the form

$$A_{-2}A_{-3}e_{-2}e_{-3}B(\phi,\phi) = A_{-2}A_{-3}e_1 B(\phi,\phi). \tag{32}$$

Since $\hat{L}(k_c)$ is not invertible, we cannot remove such terms from the residual and proceed as in (14), i.e., keep them. We then obtain the amplitude system (14) with $\lambda = b - b_c$, and, analytically, up to third order, but inconsistently in the sense of (16), c_1, c_{31} as in (31), and [VdWDB92]

$$c_{21} = \frac{2a(1+R)(1-R)}{a^2-R^2}, \quad c_{32} = \frac{3-5R+7R^2-5R^3}{a^2R(1+R)}. \tag{33}$$

Thus, $c_{21} = 0$ for $R = 1$, and we should expect (14) to be valid only for small $|R-1|$. See also [CK97, CK99] for 3D cases.

c) In summary, given the user data D, u^*, the parameters for the Turing bifurcation, the critical wave number and the choice of wave vector lattice, and the function f, `ampsys` proceeds as follows:

- Expand f to third order around u^*.
- Compute the critical eigenvector ϕ (normalized to $\phi_1 = 1$), $c_1 = \partial_\lambda\mu_c(\lambda_c, k_c)$, and the adjoint critical eigenvector ψ (normalized to $\langle\phi,\psi\rangle = 1$).
- Check if there are quadratic resonances.
  - If no, then compute the terms d_{ij} as in (24), and from these the cubic coefficients as in (27) and (29) (for more complicated lattices, there will be many more coefficients to be computed, and to be returned in adequate form, see §3).
  - If yes, then also compute (and return) the quadratic coefficients. Moreover, here we need to decide if we want a *consistent* expansion or not, where as in (15) consistent means that we do not add quadratic corrections to the ansatz for computing the third order terms, because the quadratic corrections are (assumed to be) small and hence formally do not show up in the cubic terms. On the other hand, an inconsistent expansion such as (16) may be more accurate. In `ampsys`, this choice is made by a switch `p.cons`, and the default value 0 means the inconsistent but more standard choice.

For the scalar (SH or KS like) case, the procedure is essentially the same, with $\phi = \psi = 1$. The setup to apply this algorithmn, and the results, are explained in §3 by a number of examples. ⌋

3 The demos

3.1 The Swift-Hohenberg equation, demo `SH`

As first example we consider (3), and thus in the demo directory `SH` set up `L` as in Listing 2. Furthermore, `ampsys` needs information about c_2, c_3, and the wave vectors. In the following we explain this for four cases, namely 1D, 2D (square and hex) and a 3D SC. We mainly use symbolic computations, i.e., compute the coeffients in the amplitude equations as functions of c_2, c_3. The script for all demos `SH/cmds.m` is in cell mode, which means that the user can and should run cells interactively one-by-one.

Remark 3.1. The general calling syntax of `ampsys` is `[Q,C,c1,phi]=ampsys(p)`, where `p` contains the problem description. For the class of SH equations (scalar) in this section, always `phi=1`, and for `c1` we just return the dummy `c1=0`. The idea is that for SH type equations, the user can and should always compute `c1` herself. Thus, for SH type equations we can as well just call `[Q,C]=ampsys(p)`.

```
function l=L(k,p); l=-(1-norm(k)^2)^2; end
```

Listing 1: `SH/L.m`, encoding the Fourier transform of $L = -(1+\Delta)^2$.

```
%% C1:  SH 1D, numerical coefficients
p=[]; p.c2=0.1; p.c3=-1; % cleaning p, setting numerical values for parameters
p.k=1; [Q,C]=ampsys(p); % setting kc, and calling ampsys
%% C2:  SH 1D, symbolic coefficients
p.sb=1; syms c2 c3; p.c2=c2; p.c3=c3;  % symbolic parameters
p.k=1; [Q,C]=ampsys(p); % setting kc, and calling ampsys
%% C3:  SH 2D, square lattice, symbolic nonlinearity coeff, coeff for both eqns
kc=1; type=21; p.k=wavevec(kc,type); p.eqnr=[1 2]; [Q,C]=ampsys(p);
%% C4:  SH 2D, hexagon lattice, symbolic nonlinearity coeff, cons=1
kc=1; type=22; p.k=wavevec(kc,type); p.cons=1; p.eqnr=1; [Q,C]=ampsys(p);
pause; p.eqnr=[2 3]; [Q,C,phi]=ampsys(p); % give coeff for other eqns
%% C5:  SH 2D, hexagon lattice, symbolic nonlinearity coeff, cons=0
p.cons=0; kc=1; type=22; p.k=wavevec(kc,type); p.eqnr=1; [Q,C]=ampsys(p);
%% C6:  SH 3D, BCC lattice, symbolic, cons=0;
p.cons=0; kc=1; type=33; p.k=wavevec(kc,type); p.eqnr=1; [Q,C]=ampsys(p);
```

Listing 2: script `SH/cmdsSH.m`, organized in `Matlab` cells, i.e., to be run cell-by-cell.

3.1.1 1D

Numerical values for parameters. In Cell 1 of `cmdsSH.m` we set `p.c2=0.1, p.c3=-1`, and, in 1D, `p.k=1`, because `ampsys` automatically uses $-k$ as well, i.e., the ansatz is $u = \varepsilon A_1 e^{ix} + \varepsilon A_{-1} e^{-ix}$. The output of `[Q,C]=ampsys(p)` is

$$\texttt{Q} = [], \qquad \text{and} \qquad \texttt{C} = (1\ 1\ -1\ -2.958).$$

The last entry of `C` is the coefficient c_{31}, while the preceeding entries are the indices of the A_j, which here means that the cubic term in the first equation is $-2.958 A_1 A_1 A_{-1}$, i.e.,

$$\partial_T A_1 = A_1 - 2.958 A_1 A_1 A_{-1}.$$

Since u is real, $A_{-1} = \overline{A}_1$, and hence $\partial_T A_1 = A_1 - 2.958 A_1 |A_1|^2$.

Symbolic parameters. In Cell 2 we switch to symbolic parameters, and obtain, in agreement with (10), `Q` $= []$ and `C` $= (1, 1, -1, 3c_3 + 38c_2^2/9)$.

3.1.2 2D

Square lattice. To set the wave-vector lattice for the square lattice, we can use `p.k=[1,0; 0,1]`. More conveniently, we can use the function `p.k=wavevec(kc,type)`, which provides the most common lattices, see Table 1. The output of `ampsys` for the squares reads (slightly cleaning up the `Matlab`

Table 1: Using `k=wavevec(kc,type)`

type	wave vectors	lattice type
1	$k = \texttt{kc}$	1D
21	$k = \texttt{kc}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$	square (SQ)
22	$k = \texttt{kc}\begin{pmatrix}1 & -0.5 & -0.5\\ 0 & \sqrt{3}/2 & -\sqrt{3}/2\end{pmatrix}$	hexagonal
31	$k = \texttt{kc}\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$	simple cubic (SC)
32	$k = \frac{\texttt{kc}}{\sqrt{3}}\begin{pmatrix}1 & 1 & -1 & -1\\ 1 & -1 & 1 & -1\\ 1 & -1 & -1 & 1\end{pmatrix}$	face-centered cubic (FCC)
33	$k = \frac{\texttt{kc}}{\sqrt{2}}\begin{pmatrix}1 & 0 & 1 & 1 & 0 & -1\\ 1 & 1 & 0 & -1 & 1 & 0\\ 0 & 1 & 1 & 0 & -1 & 1\end{pmatrix}$	body-centered cubic (BCC)

output for `C`)

$$\texttt{Q=[]} \text{ (no quadratic terms) and } \texttt{C=}\begin{pmatrix}1 & 1 & -1 & 3c_3 + 38c_2^2/9\\ 1 & 2 & -2 & 6c_3 + 12c_2^2\end{pmatrix}.$$

Thus, the cubic terms in the first equation are $(3c_3 + 38c_2^2/9)A_1^2A_{-1} + (6c_3 + 12c_2^2)A_1A_2A_{-2})$, i.e.,

$$\partial_T A_1 = c_1A_1 + (3c_3 + 38c_2^2/9)|A_1|^2A_1 + (6c_3 + 12c_2^2)|A_2|^2A_1,$$

in agreement with the first equation in (12). To see the coefficients in both equations (which follow from symmetry), in C3 we let `p.eqnr=[1 2]` and call `[Q,C]=ampsys(p)`. Then `Q` $= []$ as before, and

$$\texttt{C} = \begin{pmatrix}1 & 1 & -1 & 38c_2^2/9 + 3c_3 & 1 & 2 & -1 & 12c_2^2 + 6c_3\\ 1 & 2 & -2 & 12c_2^2 + 6c_3 & 2 & 2 & -2 & 38c_2^2/9 + 3c_3\end{pmatrix}.$$

The coefficients for $\frac{\mathrm{d}}{\mathrm{d}T}A_1$ are in the first 2×3 block, and those for $\frac{\mathrm{d}}{\mathrm{d}T}A_2$ in the second 2×3 block, i.e., the second equation reads

$$\partial_T A_2 = c_1A_1 + (6c_3 + 12c_2^2)|A_1|^2A_2 + (3c_3 + 38c_2^2/9)|A_2|^2A_2. \tag{34}$$

Hexagon lattice. We let `p.k=wavevec(1,22)` and `p.cons=1` (the consistent choice) and obtain

$$\texttt{Q} = (-2, -3, c_2) \text{ and } \texttt{C} = \begin{pmatrix}1 & 1 & -1 & 3c_3\\ 1 & 2 & -2 & 6c_3\\ 1 & 3 & -3 & 6c_3\end{pmatrix},$$

meaning that in agreement with (15) the quadratic and cubic terms in the first equation are $c_2\overline{A_2A_3}$ and $3c_3|A_1|^2A_1+6c_3(|A_2|^2+|A_3|^2)A_1$, respectively. Note that c_2 is considered to be small and hence omitted in the expressions in `C`. On the other hand, for `p.cons=0` we recover (16), i.e.,

$$\texttt{Q} = (-2,-3,c_2) \text{ and } \texttt{C} = \begin{pmatrix} 1 & 1 & -1 & 3c_3+38c_2^2/9 \\ 1 & 2 & -2 & 6c_3+9c_2^2 \\ 1 & 3 & -3 & 6c_3+9c_2^2 \end{pmatrix}.$$

3.1.3 3D BCC

On the BCC lattice, `[Q,C]=ampsys(p)` (with default setting `p.cons=0`) yields

$$\texttt{Q} = \begin{pmatrix} 2 & -6 & 2c_2 \\ 3 & 5 & 2c_2 \end{pmatrix}, \quad \texttt{C} = \begin{pmatrix} 1 & 1 & -1 & 3c_3+38c_2^2/9 \\ 1 & 2 & -2 & 6c_3+9c_2^2 \\ 1 & 3 & -3 & 6c_3+9c_2^2 \\ 1 & 4 & -4 & 6c_3+12c_2^2 \\ 1 & 5 & -5 & 6c_3+9c_2^2 \\ 1 & 6 & -6 & 6c_3+9c_2^2 \\ 2 & 4 & 5 & 6c_3+12c_2^2 \\ 3 & -4 & -6 & 6c_3+12c_2^2 \end{pmatrix}. \tag{35}$$

Thus, the first amplitude equation is given by

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}T}A_1 =& \lambda A_1 + q(A_2\overline{A}_6 + A_3A_5) + c_{31}|A_1|^2A_1 + c_{32}(|A_2|^2+|A_3|^2+|A_5|^2+|A_6|^2) \\ &+ c_{33}|A_4|^2A_1 + c_{34}(A_2A_4A_5 + A_3\overline{A}_4\overline{A}_6), \end{aligned} \tag{36}$$

with $q=2c_2$, $c_{31}=3c_3+38c_2^2/9$, $c_{32}=6c_3+9c_2^2$, $c_{33}=6c_3+9c_2^2$ and $c_{34}=6c_3+12c_2^2$. The form (36) is the *general* form on a BCC, following from symmetry, which is why we did not group together c_{32} and c_{34}, which in general are not equal.

If we enforce consistency via `p.cons=1`, then $c_{31}=3c_3$, and always (i.e., independent of the model)

$$c_{32} = c_{33} = c_{34} = 2c_{31}. \tag{37}$$

3.2 Two length scales pattern formation, demo `qc`

To illustrate the flexibility of `ampsys` for scalar equations, we consider the 8th order equation

$$\partial_t u = Lu + \lambda u + c_2u^2 + c_3u^3, \quad Lu = -(1+\Delta)^2(1+q^{-2}\Delta)u, \tag{38}$$

with dispersion relation $\mu(|k|,\lambda) = -(1-|k|^2)^2(1-q^{-2}|k|^2)^2+\lambda$. We have $\mu(|k|,0)=0$ simultaneously at $|k|=1$ and $|k|=q^2$. Hence, for instance in 1D, the ansatz for bifurcating (quasi–)periodic patterns reads

$$u(x,t) = A_1e_1 + A_qe_q + \text{c.c.} + \text{h.o.t}, \quad e_j = \mathrm{e}^{\mathrm{i}jx}, \tag{39}$$

and in 2D or 3D there are plenty of possibilities to choose wave vectors sets describing different (quasi)patterns. Here we restrict to amplitude equations for two cases, namely: 1D, and 2D with a hexagonal lattice for $|k|=1$ and a square lattice for $|k|=q$.

For the amplitude equations we must distinguish between the resonant case $q \in \{2,3\}$, where, e.g., quadratic ($q=2$) or cubic ($q=3$) interactions of $|k|=1$ modes may directly map to $|k|=q$ modes, and the non–resonant case $q \notin \{2,3\}$, where such direct couplings do not occur. Moreover, the

coupling between modes belonging to wave vectors with different $|k|$ is no longer symmetric, such that, given the coefficients in the first equation, the coefficients in the other equations no longer follow from simple symmetries. Thus it is useful and convenient to tell `ampsys` to compute coefficients for different equations in the amplitude system via `p.eqnr`, which, e.g., yields the full system via `p.eqnr=1:m` with `m` the number of modes in the lattice. Listing 3 shows the implementation of $\hat{L}$ for (38), and Listing 4 the script file. In any case, recall from Remark 1.1 that the amplitude equations derived here are third order truncations of an in general very complicated small divisor problem.

```
function l=L(k,p)
q=p.q; l=-(1-norm(k)^2)^2*(1-(norm(k)/q)^2)^2;
end
```

Listing 3: `qc/L.m`, passing the parameter q via the problem struct `p`.

```
%% C1:  qc 1D, non-resonant case, give coefficients for both eqns
q=1.5; p.k=[1 q]; p.q=q; % store second critical wave-nr, needed in L
p.sb=1; syms c2 c3; p.c2=c2; p.c3=c3; p.eqnr=[1 2];
[Q,C]=ampsys(p);
%% C2: 1D, resonant case, with cons=1
q=2; p.k=[1 q]; p.q=q; p.cons=1; [Q,C]=ampsys(p);
%% C3: 1D, resonant case, with cons=0
p.cons=0; [Q,C]=ampsys(p);
%% C4: 2D, mix of hex and square lattice, resonant
p.q=2; p.k=[wavevec(1,22) wavevec(p.q,21)];
p.eqnr=[1 4]; [Q,C]=ampsys(p);  % return coeff for first and 4th equation
```

Listing 4: `qc/cmdsqc.m`.

3.2.1 1D

As a warmup example for the non–resonant case let $q = 1.5$, see C1. Then setting `p.eqnr=[1,2]` and calling `[Q,C]=ampsys(p)` yields $Q = []$, and

$$C = \begin{pmatrix} 1 & 1 & -1 & 214c_2^2/49 + 3c_3 & 1 & 2 & -1 & 2557c_2^2/196 + 6c_3 \\ 1 & 2 & -2 & 2557c_2^2/196 + 6c_3 & 2 & 2 & -2 & 1153c_2^2/288 + 3c_3 \end{pmatrix}.$$

Hence, the amplitudes A_1 and A_q from (39) fulfill

$$\begin{aligned} \partial_T A_1 = &\ c_1 A_1 + (c_{31}|A_1|^2 + c_{32}|A_2|^2)A_1, \\ \partial_T A_2 = &\ c_1 A_1 + (c_{32}|A_1|^2 + c_{33}|A_2|^2)A_2, \end{aligned}$$

with $c_{31} = 3c_3 + 214c_2^2/49$, $c_{32} = 6c_3 + 2557c_2^2/196$ and $c_{33} = 3c_3 + 1153c_2^2/288$.

On the other hand, for, e.g., $q = 2$ and `p.cons=1` we obtain

$$\mathtt{Q} = (2, -1, 2c_2, 1, 1, c_2), \quad \mathtt{C} = \begin{pmatrix} 1 & 1 & -1 & 3c_3 & 1 & 2 & -1 & 6c_3 \\ 1 & 2 & -2 & 6c_3 & 2 & 2 & -2 & 3c_3 \end{pmatrix},$$

from which we can again write down the system for $\partial_T A_1, \partial_T A_2$. For `p.cons=0` we obtain Q as before, and

$$\mathtt{C} = \begin{pmatrix} 1 & 1 & -1 & 6c_2^2 + 3c_3 & 1 & 2 & -1 & 201c_2^2/25 + 6c_3 \\ 1 & 2 & -2 & 201c_2^2/25 + 6c_3 & 2 & 2 & -2 & 8102c_2^2/2025 + 3c_3 \end{pmatrix}.$$

All these formulas can be checked by hand in a straightforward although already a bit lengthy way.

3.2.2 2D

In Cell 4 of `cmdsqc` we set up a lattice as indicated in Fig. 2, and let `ampsys` return the coefficients in the 1st and 4th equation to obtain

$$\mathtt{Q} = \begin{pmatrix} 4 & -1 & 2c_2 & 1 & 1 & c_2 \\ -2 & -3 & 2c_2 & 0 & 0 & 0 \end{pmatrix}$$

and

$$\mathtt{C} = \begin{pmatrix} 1 & 1 & -1 & 6c_2^2+3c_3 & 1 & 4 & -1 & 201c_2^2/25+6c_3 \\ 1 & 2 & -2 & 24c_2^2+6c_3 & 1 & -2 & -3 & 36c_2^2+6c_3 \\ 1 & 3 & -3 & 24c_2^2+6c_3 & 2 & 4 & -2 & 1636c_2^2/81+6c_3 \\ 1 & 4 & -4 & 201c_2^2/25+6c_3 & 3 & 4 & -3 & 1636c_2^2/81+6c_3 \\ 1 & 5 & -5 & 12c_2^2+6c_3 & 4 & 4 & -4 & 8102c_2^2/2025+3c_3 \\ 2 & 3 & 4 & 36c_2^2+6c_3 & 4 & 5 & -5 & 204c_2^2/49+6c_3 \end{pmatrix}.$$

There are two quadratic terms mapping on $k_1 = (1,0)$ but only one mapping to $k_4 = (2,0)$, and therefore the second row of the second block of `Q` is filled with zeros.

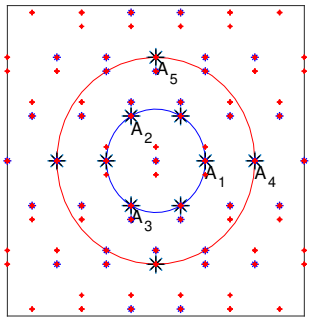

Figure 2: Critical wave vectors (black stars), and first three 'layers' of the generated quasi-lattice.

3.3 A damped Kuramoto-Shivashinsky type of equation, demo `KS`

The demo KS deals with damped/driven Kuramoto-Shivashinsky (KS) [KY76, Siv88] type of equations, e.g., in 1D,

$$\partial_t u = -(1+\partial_x^2)^2 u + \lambda u + c_2 \partial_x (u^2), \tag{40}$$

i.e., a SH equation with a convective nonlinearity, which gives another important class of pattern forming systems. Generalizing to $\mathbb{R}^d$, in Fourier space, $(\partial_{x_1}+\ldots+\partial_{x_d})(u^2)$ becomes $\hat{f}(u) = \hat{q}(k)\mathcal{F}(u^2)(k)$, with the symbol $\hat{q}(k) = \mathrm{i}(k_1+\ldots+k_d)$. To deal with such problems, `ampsys` has the optional field `p.qs`, where the user can put a function handle to the desired symbol for quadratic terms. For instance, for $\hat{q}(k) = \mathrm{i}(k_1+\ldots+k_d)$ we define `function q=qs1(k); q=sum(k)*1i; end`. Then calling `p.sb=1; p.qs=@qs1; syms c2; p.c2=c2; p.c3=0; [Q,C]=ampsys(p);` (see script `KS/cmdsKS.m`) in 1D we obtain `Q=[]` (no quadratic interaction in 1D), and

$$\mathtt{C} = (1,1,-1,-4c_2^2/9), \quad \text{i.e. } \frac{\mathrm{d}}{\mathrm{d}T}A_1 = A_1 - \frac{4}{9}c_2^2|A_1|^2 A_1,$$

which can be quickly checked by hand.

Another canonical form of the nonlinearity, in particular in higher space dimensions, is $c_2|\nabla u|^2$, where $|\nabla u|^2 := (\partial_{x_1} u)^2 + \ldots + (\partial_{x_d} u)^2$. In this case $\hat{q}(k) = -|k|^2$, and setting `p.qs=@qs2` with `function q=qs2(k); q=-norm(k)^2; end` and computing on a 3D FCC we obtain `Q=[]` again, and

$$\mathtt{C} = \begin{pmatrix} 1 & 1 & -1 & 8c_2^2/9 \\ 1 & 2 & -2 & 1296c_2^2/25 \\ 1 & 3 & -3 & 1296c_2^2/25 \\ 1 & 4 & -4 & 1296c_2^2/25 \\ -2 & -3 & -4 & 144c_2^2 \end{pmatrix}.$$

As usual, the coefficients in the equations for $\frac{\mathrm{d}}{\mathrm{d}T} A_j$, $j = 2, 3, 4$ can be obtained from symmetry, or `ampsys` with suitable `p.eqnr`.

3.4 The Brusselator model, demo `brussel`

As an RD example we consider the Brusselator (17) (`asdemos/brussel`), over square (2D) and BCC (3D) lattices. Listing 5 shows the straightforward implementation of the 'nonlinearity' f, which is independent of the lattice, while Listing 6 shows the commands for computing the AEs over different lattices, and some comparison to the analytic formulas (31).

```
function w=f(u,p) % nonlinearity (everything except diffusion) for the Brusselator
% u is the vector of unknowns, everything else (e.g., parameters) passed via p
a=p.par(1); b=p.par(2);  % extract parameters
f1=a-(b+1)*u(1)+u(1)^2*u(2); f2=b*u(1)-u(1)^2*u(2);  w=[f1;f2];
```

Listing 5: `brussel/f.m`, see comments for explanation.

```
%% standard Brusselator, C1: Preparations, clear (possible) previous setups of p
% and set parameters;  to be run as prep. for Cells 2-5 below
p=[];
a=2; R=0.9; bc=(1+R)^2; b=bc; D1=1; D2=(a/R)^2; % parameters
uh=[a; b/a]; kc=sqrt(R/D1); % derived data (u*, crit.wave number kc)
p.par=[a,b]; p.uh=uh; p.D=[D1 0; 0 D2]; % put data into p
p.bifpar=2; p.del=1e-3; bp=b+p.del; p.uhp=[a; bp/a]; % settings for mu'(b_c,k_c):
% bifpar is b, p.uhp is u* at b+del, p.del is used for FD for mu'
%% C2: 2D square lattice, with comparison to analytic formulas for lam and c31
p.k=wavevec(kc,21); [Q,C,c1]=ampsys(p);  % compute ampsys over squares
t0=(a^2-R^2)/(a^2*(1+R)); lam=1/bc;
c3a=(8-38*R-5*R^2+8*R^3)/(9*R*(a^2-R^2));
fprintf('(c1,c31,lam,c3a)=(%g, %g, %g, %g)\n', c1,C(1,end),lam/t0,c3a);
%% C3: 3D BCC, with p.cons=1, yielding c32=c33=c34=2*c31
p.cons=1; p.k=wavevec(kc,33); [Q,C,c1]=ampsys(p); % BCC
q=Q(1,end); c31=C(1,end); fprintf('(c1,q,c31)=(%g %g %g)\n', c1,q,c31);
t0=(a^2-R^2)/(a^2*(1+R)); lam=1/bc; c1a=lam/t0;
qa=2*a*(1-R)/(a^2-R^2); c3a=(8-38*R-5*R^2+8*R^3)/(9*R*(a^2-R^2));
fprintf('(c1a,qa,c3a)=(%g %g %g)\n', c1a,qa,c3a); % analytic formulas:
%% C4: BCC, with cons=0, yielding c31 as in formula, and different c32,c33,c34
p.cons=0; p.k=wavevec(kc,33); [Q,C,c1]=ampsys(p); % BCC
q=Q(1,end); c31=C(1,end); c32=C(2,end); c33=C(4,end); c34=C(7,end);
fprintf('(c3a1,c3a)=(%g %g)\n', c31,c3a);
fprintf('(c1,q,c31,c32,c33,c34)=(%g %g %g %g %g %g)\n', c1,q,c31,c32,c33,c34);
%% C5: same as C5 with R=0.75, showing more sign.deviations from c33=2*c31 etc
R=0.4; bc=(1+R)^2; b=bc; D1=1; D2=(a/R)^2; % update parameters
uh=[a;b/a];kc=sqrt(R/D1);p.par=[a,b]; p.uh=uh; p.D=[D1 0;0 D2]; % put data into p
bp=b+p.del; p.uhp=[a; bp/a]; p.k=wavevec(kc,33);
p.cons=0; [Q,C,c1]=ampsys(p); % BCC
```

Listing 6: `brussel/cmdsBr.m`, script for (17) over various lattices.

The first cell in Listing 6 contains preparations for subsequent calls to `ampsys`. Here, the idea of the cell mode is that the user can run (just) the first cell to set/change parameters, and then choose the lattice and call `ampsys` for the new parameters. This is our typical operational mode. Thus, in lines 5 and 6 we set up the parameters as needed, namely the problem parameters a, b (using the auxiliary parameter R to set b and D_2) including the diffusion coefficients, and the homogeneous steady state U^*. In line 7 we put this data into `p`, and in line 8 we provide the data needed to compute $\mu'(k_c) = \partial_b \mu(k_c)$. We set `p.bifpar=2` as we take b as the bifurcation parameter.

3.4.1 Squares.

In Cell 2 of Listing 6 we set a 2D square lattice, and call `ampsys` to compute the coefficients c_1, c_{31}, c_{32} as in (30). The output is `Q=[]` (no quadratic resonances on the square lattice), `c1=0.66`, and

$$\texttt{C} = \begin{pmatrix} 1 & 1 & -1 & -0.945 \\ 1 & 2 & -2 & -1.462 \end{pmatrix}, \tag{41}$$

i.e. $c_{31} = -0.945$ and $c_{32} = -1.462$, in the notation from (30). This agrees with (31), and this also holds for other values a, R chosen in line 5.

3.4.2 The BCC lattice

In Cell 3 of Listing 6 we consider the BCC lattice, where by symmetry the general form of the first amplitude equation is (36), now with $\lambda = c_1(B - B_c)$. For c_1 and c_{31} (with `p.cons=0`) we naturally have the formulas (31) again, and a_{12} has been computed in [VdWDB92] to

$$a_{12} = \frac{4A(1-R)}{A^2-R^2}, \tag{42}$$

which shows that $|R-1|$ should be small. Moreover, in the limit $R \to 1$ we obtain

$$c_{32}/c_{31} \to 2, c_{33}/c_{31} \to 2, c_{34}/c_{31} \to 2 \text{ as } R \to 1, \tag{43}$$

and $c_{31} \to -1$ for the choice $A = 2$, fixed in Cell 1. For `p.cons=1` we always have

$$c_{32} = c_{33} = c_{34} = 2c_{31} \text{ also for } R \neq 1. \tag{44}$$

Using `ampsys`, we can check the formulas (42) for c_1, a_{12} and c_{31}, and compute $c_{32}, \ldots, c_{34}$ to check (43), respectively quantify the deviations from the limit. In C3 we run `ampsys` with `p.cons=1`. The output is $c_1 = 0.66$, $\texttt{Q} = \begin{pmatrix} 2 & -6 & 0.125 \\ 3 & 5 & 0.125 \end{pmatrix}$ and

$$\texttt{C} = \begin{pmatrix} 1 & 1 & -1 & -0.846 \\ 1 & 2 & -2 & -1.693 \\ 1 & 3 & -3 & -1.693 \\ \vdots & \vdots & \vdots & \vdots \\ 3 & -4 & -6 & -1.693 \end{pmatrix},$$

yielding the correct values for c_1, a_{12}, and (44) holds, but the value for c_{31} naturally differs from

$c_{31} \approx -0.945$ from (31). On the other hand, using `p.cons=0` in C4 yields c_1, Q as before, and

$$\mathtt{C} = \begin{pmatrix} 1 & 1 & -1 & -0.945 \\ 1 & 2 & -2 & -1.999 \\ 1 & 3 & -3 & -1.999 \\ 1 & 4 & -4 & -1.462 \\ 1 & 5 & -5 & -1.999 \\ 1 & 6 & -6 & -1.999 \\ 2 & 4 & 5 & -2.041 \\ 3 & -4 & -6 & -2.041 \end{pmatrix}.$$

Thus, c_{31} agrees with (31), but there is a deviation from (44) in particular for c_{33} (4th row of `C`). This gets worse for smaller R in C5, as expected.

3.5 An extended Brusselator as a three component system, demo `ExtBrus`.

As a 3-component example in `ampsys` consider the extended Brusselator [YDZE02]

$$\begin{aligned} \dot{u}_1 &= D_1\Delta u_1 + a - (1+b)u_1 + u_1^2u_2 - cu_1 + du_3, \\ \dot{u}_2 &= D_2\Delta u_2 + bu_1 - u_1^2u_2, \\ \dot{u}_3 &= D_3\Delta u_3 + cu_1 - du_3, \end{aligned} \tag{45}$$

which also serves as a tutorial example in [Uec19a], see also [Uec18]. A homogeneous steady state is given by $(u_1, u_2, u_3) = (a, b/a, ac/d)$, and for fixed $(D_1, D_2, D_3) = (0.01, 0.1, 1)$ this undergoes Hopf–, wave– or Turing bifurcations as the parameters (a, b, c, d) vary. For convenience, in `ExtBrus` we include a short script `plotev` to plot the dispersion relation. Fixing $(a, c, d) = (1.08, 1, 1)$, the first instability is a Turing instability at $b_c \approx 3.057$, with $k_c \approx 6.83$. Listing 7 shows the script file to compute the Landau coefficients for 2 simple cases, giving, e.g., $c_1 = 0.893$, $Q = []$ and $C = \begin{pmatrix} 1 & 1 & -1 & -1.098 \\ 1 & 2 & -2 & 92.877 \end{pmatrix}$ on the square.

```
%% preparations
p=[]; Dx=0.01; Dy=0.1; Dz=1; p.D=[Dx 0 0;  0 Dy 0;  0 0 Dz];
a=1.08; b=3.057; p.par=[a b]; p.uh=[a; b/a; a];
p.del=del; p.uhp=[a; (b+p.del)/a; a]; p.bifpar=2; kc=6.83;
%% calling ampsys, 1D, and 2D square
p.k=wavevec(kc,1); [Q,C,c1,phi]=ampsys(p); pause
p.k=wavevec(kc,21); [Q,C,c1,phi]=ampsys(p);
```

Listing 7: `ExtBrus/cmds.m`, script for (45) in 1D, and over 2D square lattice.

References

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

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

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

[dWDR+18] H. de Witt, T. Dohnal, J.D.M. Rademacher, H. Uecker, and D. Wetzel. pde2path - Quickstart guide and reference card, 2018.

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

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

[IR10] G. Iooss and A. M. Rucklidge. On the existence of quasipattern solutions of the Swift-Hohenberg equation. *J. Nonlinear Sci.*, 20(3):361–394, 2010.

[KY76] Y. Kuramoto and T. Yamada. Pattern formation in oscillatory chemical reactions. *Progress of theoretical physics*, 56(3):724–740, sep 1976.

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

[PL68] I. Prigogine and R. Lefever. Symmetry Breaking Instabilities in Dissipative Systems. II. *J. Chem. Phys*, 48(4):1695–1700, 1968.

[SAKR16] P. Subramanian, A.J. Archer, E. Knobloch, and A.M. Rucklidge. Three-dimensional icosahedral phase field quasicrystal. *Phys. Rev. Lett.*, 117:075501, 2016.

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

[Siv88] G. Sivashinsky. Nonlinear analysis of hydrodynamic instability in laminar flames—I. Derivation of basic equations. *Acta Astronautica*, pages 459 – 488, 1988.

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

[Uec18] H. Uecker. User guide on Hopf bifurcation and time periodic orbits with pde2path, 2018. Available at [Uec19c].

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

[Uec19b] H. Uecker. Pattern formation with pde2path – a tutorial, 2019.

[Uec19c] H. Uecker. `www.staff.uni-oldenburg.de/hannes.uecker/pde2path`, 2019.

[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.

[UW19] H. Uecker and D. Wetzel. Snaking branches of planar BCC fronts in the 3D Brusselator. *preprint*, 2019.

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

[VdWDB92] J. Verdasca, A. de Wit, G. Dewel, and P. Borckmans. Reentrant hexagonal Turing structures. *Phys. Lett. A*, 168(194):194–198, 1992.

[Wet18] D. Wetzel. Tristability between stripes, up-hexagons, and down-hexagons and snaking bifurcation branches of spatial connections between up- and down-hexagons. *Phys. Rev. E*, 97(062221), 2018.

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