Hopf bifurcation and time periodic orbits with `pde2path` – a tutorial via reaction–diffusion systems and distributed optimal control

Hannes Uecker

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

July 24, 2016

Abstract

We describe how to use the `Matlab` continuation and bifurcation package `pde2path` for Hopf bifurcations and the continuation of branches of periodic orbits, including the computation of Floquet multipliers, in systems of PDEs in 1, 2, and 3 spatial dimensions. The setup is explained first by three reaction diffusion examples, namely a complex Ginzburg–Landau equation as model problem, a reaction diffusion system on a disk featuring interesting rotational waves including stable (anti) spirals bifurcating out of the trivial solution, and an extended Brusselator system with interaction of Turing and Turing–Hopf bifurcations. Finally we consider a system from distributed optimal control, which is ill-posed as an initial value problem, and thus needs a particularly stable method for computing Floquet multipliers, for which we use a periodic Schur decomposition. The package (library and demos) can be downloaded at `www.staff.uni-oldenburg.de/hannes.uecker/pde2path`.

MSC: 35J47, 35B22, 37M20
Keywords: Hopf bifurcation, periodic orbit continuation, Floquet multipliers, partial differential equations, finite element method, reaction–diffusion, distributed optimal control

Contents

1 Introduction

The package `pde2path` [UWR14, DRUW14] has been developed as a continuation/bifurcation package for stationary problems of the form

$$G(u,\lambda) := -\nabla\cdot(c\otimes\nabla u) + au - b\otimes\nabla u - f = 0. \tag{1}$$

Here $u = u(x) \in \mathbb{R}^N$, $x \in \Omega$ with $\Omega \subset \mathbb{R}^2$ some bounded domain, $\lambda \in \mathbb{R}^p$ is a parameter (vector), and $c \in \mathbb{R}^{N\times N\times 2\times 2}$, $b \in \mathbb{R}^{N\times N\times 2}$, $a \in \mathbb{R}^{N\times N}$ and $f \in \mathbb{R}^N$ can depend on $x, u, \nabla u$, and parameters.[1] The boundary conditions (BC) are of "generalized Neumann" form, i.e.,

$$\mathbf{n}\cdot(c\otimes\nabla u) + qu = g, \tag{2}$$

where $\mathbf{n}$ is the outer normal and again $q \in \mathbb{R}^{N\times N}$ and $g \in \mathbb{R}^N$ may depend on x, u, ∇u and parameters. These BC include zero flux BC, and a "stiff spring" approximation of Dirichlet BC via large prefactors in q and g, and over suitable (rectangular) domains, periodic BC are also supported. Moreover, there are interfaces to couple (1) with additional equations, such as mass conservation, or phase conditions for considering co-moving frames, and to set up extended systems, for instance for branch point continuation.

`pde2path` has been applied to various research problems, e.g., patterns in 2D reaction diffusion systems [UW14, Küh15b, Küh15a, SDE+15, Wet16], some problems in fluid dynamics and nonlinear optics [ZHKR15, DU16] and, with the add-on package `p2poc`, in optimal control [GU16, Uec16a]. We have now revised `pde2path` to also work efficiently for $\Omega \subset \mathbb{R}^d$ with $d = 1, 3$. This extension is based on replacing `Matlab`'s `pdetoolbox` by the FEM implementation `OOPDE` [Prü16]. This can also be used in 2D, which makes `pde2path` independent of the `pdetoolbox`, and with unified user interfaces in 1D, 2D and 3D. Moreover, we are adding new features, and the main purpose of this note is to explain new `pde2path`–functions, collected in a library `hopf`, to treat Hopf (or Poincaré–Andronov–Hopf) bifurcations and the continuation of time–periodic orbits for systems of the form

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

with G from (1) and BC from (2). Of course, adding the time dimension makes computations more expensive, such that here we focus on 1D and 2D, and only give one 3D example to illustrate that all user interfaces are essentially dimension independent. Thus, this manual also serves as an introduction to the new `pde2path OOPDE` setup, although details on 3D computations (with focus on the stationary case) will appear elsewhere.

For general introductions to and reviews of (numerical) continuation and bifurcation we recommend [Gov00, Kuz04, Doe07, Sey10], and [Mei00], which has a focus on reaction–diffusion systems. The treatment of large scale problems, typically from the spatial discretization of PDEs, including the computation and continuation of time periodic orbits, has for instance been discussed in [LRSC98, TB00, LR00], and has recently been reviewed in [DWC+14]. There, the focus has been on matrix–free methods where the periodic orbits are computed by a shooting method, which can

[1] We have $[\nabla\cdot(c\otimes\nabla u)]_i := \sum_{j=1}^N [\partial_x c_{ij11}\partial_x + \partial_x c_{ij12}\partial_y + \partial_y c_{ij21}\partial_x + \partial_y c_{ij22}\partial_y]u_j$ (i^{th} component), and similarly $[au]_i = \sum_{j=1}^N a_{ij}u_j$, $[b\otimes\nabla u]_i := \sum_{j=1}^N [b_{ij1}\partial_x + b_{ij2}\partial_y]u_j$, and $f = (f_1,\ldots,f_N)$ as a column vector.

conveniently be implemented if a time–stepper for the given problem is available. In many cases, shooting methods can also be used to investigate the bifurcations from periodic orbits, and to trace bifurcation curves in parameter space, by computing the Floquet multipliers of the periodic orbits. In this direction, see in particular [BT10, SGN13, WIJ13, NS15] for impressive results in fluid problems.

Here we proceed by a collocation (in time) method for the continuation of periodic orbits. With respect to computation time and in particular memory requirements such methods are often more demanding than (matrix free) shooting methods. However, one reason for the efficiency of shooting methods in the works cited above is that there the problems considered are strongly dissipative, with only very few of the eigenvalues for the linearized evolution near the imaginary axis. We also treat such problems, and show that up to moderately large scale they can efficiently be treated by collocation methods as well. However, another class of problems we deal with are canonical systems obtained from distributed optimal control problems with infinite time horizons. Such problems are ill-posed as initial value problems, which seems quite problematic for genuine shooting methods.

We also compute the Floquet multipliers for periodic orbits. For this, a direct approach is to explicitly construct the monodromy matrix from the Jacobian used in the collocation solver for the periodic orbit. We find that this works well for dissipative problems, but completely fails for the ill–posed optimal control problems, and thus we also provide a method based on a periodic Schur decomposition, which can handle this situation. Currently, our Floquet computations can be used to assess the stability of periodic orbits, and for *detection* of possible bifurcations from them. However, we do not (yet) provide tools for *localization* of, or branch switching at, such bifurcation points, which is work in progress.

To explain the setup and usage of our `hopf` library we use four example problems, with the `Matlab` files included in the package download at [Uec16b] as demo directories. The first is a cubic–quintic complex Ginzburg–Landau (cGL) equation, which we consider over 1D, 2D, and 3D cuboids with homogeneous Neumann and Dirichlet BC, such that we can explicitly calculate all Hopf bifurcation points (HBP) from the trivial branch. For periodic BC we also have the Hopf branches explicitly, which altogether makes the cGL equation a nice toy problem to validate and benchmark our routines. Next we consider a reaction diffusion system from [GKS00] on a circular domain and with somewhat non-standard Robin BC, which lead to rotating waves, and in particular to the bifurcation of (anti) spiral waves out of the trivial solution. Our third example is a Brusselator system from [YDZE02], which shows interesting interactions between Turing branches and Turing–Hopf branches. As a non–dissipative example we treat the canonical system for a simple control problem of "optimal pollution". This is still of the form (1), but is ill–posed as an initial value problem, since it includes "backward diffusion". Nevertheless, we continue steady states, and obtain Hopf bifurcations and branches of periodic orbits.

Many of the numerical results on periodic orbits in PDE in the literature, again see [DWC$^+$14] for a review, are based on custom made codes, which in turn very reasonably rely on large scale packages, but which sometimes do not seem easy to access and modify for non–expert users. Although in some of our research applications we consider problems with on the order of 10^5 unknowns in space, `pde2path` is not primarily intended for large scale problems. Instead, the goal of `pde2path` continues to be a general and easy to use (and modify and extend) toolbox to investigate bifurcations in PDEs of the (rather large) class given by (3). With our `hopf` library we provide some basic functionality for Hopf bifurcations and continuation of periodic orbits for such PDEs over 1D, 2D, and 3D domains, where at least the 1D cases and simple 2D cases are sufficiently fast to use `pde2path` as a quick (i.e., interactive) tool for studying interesting problems. The user interfaces reuse the standard `pde2path` setup (with the addition of `OOPDE`), and no new user functions are necessary. Due to higher computational costs in 2+1D, in 3D, or even 3+1D, compared to the 2D case from [UWR14], in this tutorial we work with quite coarse meshes, but give a number of comments on how to adaptively generate and work with finer meshes.

In §2 we review some basics of the Hopf bifurcation, of periodic orbit continuation and multiplier computations, and their numerical treatment in `pde2path`. In §3 we review the `pde2path` setup, data structures and help system, and in §4 we present the examples. A brief summary and outlook are given in §5, and in the Appendices we collect information on the pertinent new `pde2path` fields, functions and switches, and some implementation details. For comments, questions, and bugs, please mail to `hannes.uecker@uni-oldenburg.de`.

Acknowledgment. Many thanks to Francesca Mazzia for providing TOM [MT04], which was essential help for setting up the `hopf` library; to Uwe Prüfert for providing `OOPDE`; to Tomas Dohnal, Jens Rademacher and Daniel Wetzel for some testing of the Hopf examples; to Daniel Kressner for `pqzschur`; to Arnd Scheel for helpful comments on the system in §4.3; and to Dieter Grass for the cooperation on distributed optimal control problems, which was one of my main motivations to implement the `hopf` library.

2 Hopf bifurcation and periodic orbit continuation in `pde2path`

Our description of the algorithms is based on the spatial FEM discretization of (3), which, with a slight abuse of notation, we write as

$$M\dot{u}(t) = -G(u(t), \lambda), \tag{4}$$

where $M \in \mathbb{R}^{n_u \times n_u}$ is the mass matrix, $n_u = Nn_p$ is the number of unknowns (degrees of freedom DoF), where n_p is the number of mesh-points, and, for each t,

$$u(t) = (u_{1,1}, \dots, u_{1,n_p}, u_{2,1}, \dots, u_{N,1}, \dots u_{N,n_p})(t) \in \mathbb{R}^{n_u},$$

and similarly $G : \mathbb{R}^{n_u} \times \mathbb{R}^p \to \mathbb{R}^{n_u}$. As in [UWR14, DRUW14] we assume that the problem is described by the `Matlab` struct `p`, with its various subfields such as `p.nc, p.sw` and `p.fuha` for the numerical controls, switches, and function handles. We use the generic name λ for the parameter vector, *and* the *active* continuation parameter, again see [DRUW14] for details. When in the following we discuss eigenvalues μ and eigenvectors ϕ of the linearization

$$M\dot{v} = -\partial_u G(u, \lambda)v \tag{5}$$

of (4) around some (stationary) solution of (4), or simply eigenvalues of $\partial_u G = \partial_u G(u, \lambda)$, we always mean the generalized eigenvalue problem

$$\mu M\phi = \partial_u G\phi. \tag{6}$$

Thus eigenvalues of $\partial_u G$ with *negative* real parts give dynamical *instability* of u.

Remark 2.1. For, e.g., the continuation of travelling waves in translationally invariant problems, the PDE (3) is typically transformed to a moving frame $\xi = x - \gamma(t)$, with BC that respect the translational invariance, and where $\dot{\gamma}$ is an unknown wave speed, which yields an additional term $\dot{\gamma}\partial_x u$ on the rhs of (3). This then requires an additional equation, a phase condition, for instance of the form $q(u) = \langle \partial_x \tilde{u}, \tilde{u} - u\rangle \stackrel{!}{=} 0$, where $\tilde{u}$ is a reference wave (e.g. $\tilde{u} = u_{\rm old}$, where $u_{\rm old}$ is from a previous continuation step), and $\langle u, v\rangle = \int_\Omega uv \,\mathrm{d}x$. Together we obtain a differential–algebraic system instead of (4), and similarly for other constraints such as mass conservation, see [DRUW14, §2.4,§2.5] for examples. More generally, see for instance [BT07] for equations with continuous symmetries and the associated "freezing method". Hopf bifurcations can occur in such systems, see e.g. the Hopf bifurcations from traveling ($\dot{\gamma} \neq 0$) or standing ($\dot{\gamma} = 0$) fronts and pulses in [HM94, GAP06, BT07, GF13], but are more difficult to treat numerically than the case without constraints. Thus, here we restrict to the simpler pure PDE problems of the form (3), and hence (4) on the spatially discretized level. ⌋

2.1 Branch and Hopf point detection and localization

Hopf bifurcation is the bifurcation of a branch of time periodic orbits from a branch $\lambda \mapsto u(\cdot, \lambda)$ of stationary solutions of (3), or numerically (4). This generically occurs if at some $\lambda = \lambda_H$ a pair of simple complex conjugate eigenvalues $\mu_j(\lambda) = \overline{\mu}_{j+1}(\lambda)$ of $G_u = \partial_u G(u, \lambda)$ crosses the imaginary axis with nonzero imaginary part and nonzero speed, i.e.,

$$\mu_j(\lambda_H) = \overline{\mu}_j(\lambda_H) = \mathrm{i}\omega \neq 0, \quad \text{and } \mathrm{Re}\mu_j'(\lambda_H) \neq 0. \tag{7}$$

Thus, the first issue is to define a suitable test function ψ_H to numerically detect (7). Additionally, we also want to detect real eigenvalues crossing the imaginary axis, i.e.,

$$\mu_j(\lambda_{\mathrm{BP}}) = 0, \quad \text{and } \mathrm{Re}\mu_j'(\lambda_{\mathrm{BP}}) \neq 0. \tag{8}$$

A fast and simple method for (8) is to monitor sign changes of the product

$$\psi(\lambda) = \prod_{i=1,\ldots,n_u} \mu_i(\lambda) = \det(G_u) \tag{9}$$

of all eigenvalues, which even for large n_u can be done quickly via the LU factorization of G_u, respectively the extended matrix in arclength continuation, see [UWR14, §2.1]. This so far has been the standard setting in `pde2path`, but the drawback of (9) is that the sign of ψ only changes if an odd number of real eigenvalues crosses 0.

Unfortunately, there is no general method for (7) which can be used for large n_u. For small systems, one option would be

$$\psi_H(\lambda) = \prod_i (\mu_i(\lambda) + \mu_{i+1}(\lambda)), \tag{10}$$

where we assume the eigenvalues to be sorted by their real parts. However, this, unlike (9) requires the actual computation of all eigenvalues, which is not feasible for large n_u. Another option are dyadic products, [Kuz04, Chapter 10], which again is not feasible for large n_u.

If, on the other hand, (3) is a dissipative problem, then we may try to just compute n_{eig} eigenvalues of G_u of smallest modulus, which, for moderate n_{eig} can be done efficiently, and to count the number of these eigenvalues which are in the left complex half plane, and from this detect both (7) and (8). The main issue then is to choose n_{eig}, which unfortunately is highly problem dependent, and for a given problem may need to be chosen large again.

The method presented in [GS96] uses complex analysis, namely the winding number $W(g(i\omega), 0, \infty)$ of $g(z) = c^T(G_u - zM)^{-1}b$, which is the Schur complement of the bordered sytem $\begin{pmatrix} G_u - zM & b \\ c^T & 0 \end{pmatrix}$ with (some choices of) $b, c \in \mathbb{R}^{n_u}$. We have

$$g(z) = \frac{N(z)}{\det(G_u - zM)}, \text{ where } W(g(i\omega), 0\infty) = \pi(Z_l - Z_r + P_r - P_l)/2, \tag{11}$$

where $Z_{l,r}, P_{l,r}$ are the zeros and poles of $g(z)$ in the left and right complex half planes, respectively, and where N is a polynomial in z which depends on b, c. Since $\det(G_u - zM)$ does not depend on b, c, using some clever evaluation [GS96] of (11) for some choices of b, c one can count the poles of g, i.e. the eigenvalues of G_u in the left half plane.

Here we combine the idea of counting small eigenvalues with suitable spectral shifts $\mathrm{i}\omega_{1,2,\ldots}$. To estimate these shifts, given a current solution (u, λ) we follow [GS96] and compute

$$[0, \omega_{\max}] \ni \omega \mapsto g(u, \lambda, \mathrm{i}\omega; b) := b^T(G_u - \mathrm{i}\omega)^{-1}b, \tag{12}$$

for one or several suitably chosen $b \in \mathbb{R}^{n_u}$. Generically, g will be large for $\mathrm{i}\omega$ near some complex eigenvalue $\mu = \mu_r + \mathrm{i}\mu_i$ with small μ_r, and thus we may consider this $\mathrm{i}\omega$ as a *guess* for a Hopf eigenvalue during the next continuation steps. To accurately compute g from (12) we again use ideas from [GS96] to refine the ω discretization (and actually compute the winding number). Then, after each continuation step we compute a few eigenvalues near $0, \omega_1, \ldots$. We can reset the shifts ω_i after a number of continuation steps by evaluating (12) again, and instead via (12) the user can also set the ω_i by hand.[2]

Of course, the idea is mainly heuristic, and, in this simple form, may miss some bifurcation points (BPs, in the sense of (8)) and Hopf bifurcation points (HBPs, in the sense of (7)), and can and typically will detect false BPs and HBPs, see Fig. 1, which illustrates two ways in which the algorithm can fail.[3] However, some of these failures can be detected and/or corrected, see Remark 2.2, and in practice we found the algorithm to work remarkably well in our examples, with a rather small numbers of eigenvalues computed near 0 and $\mathrm{i}\omega_1$, and in general to be more robust than the theoretically more sound usage of (11).[4] See Table 1 for an overview of the new bifurcation detection setup in `pde2path`, flagged by `p.sw.bifcheck`, and §4 for examples.

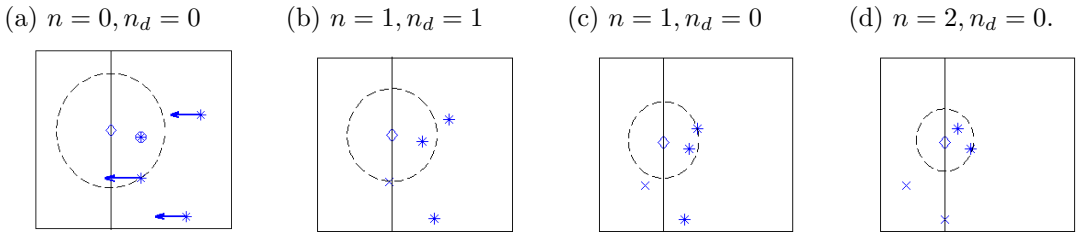


Figure 1: Sketch of the idea, and typical failures, of detecting Hopf points by counting eigenvalues with negative real parts near some shift $\mathrm{i}\omega_1$, marked by $\diamond$. Here, for illustration we use `neig=2`, i.e., use the 2 eigenvalues closest to $\mathrm{i}\omega_1$ for bifurcation detection, and show 4 eigenvalues near $\mathrm{i}\omega_1$, stable ones with $*$ and unstable ones with $\times$. n is the total number of negative eigenvalues, and n_d the number of detected ones. From (a) to (d) we assume that some parameter λ varies, and the shown eigenvalues depend continuously on λ; for better illustration we assume that the eigenvalue circled in (a) stays fixed. The dashed circle has radius $|\mu(\lambda) - \mathrm{i}\omega_1|$ with μ the second closest eigenvalue to $\mathrm{i}\omega_1$. From (a) to (b) we correctly detect a HBP. From (b) to (c) we incorrectly find a HBP, as the only unstable eigenvalue wanders out of the pertinent circle. From (c) to (d) we miss a HBP, as the guess $\mathrm{i}\omega_1$ is too far off. The failure (b) to (c) can be detected in the localization by requiring that at the end the real part of the eigenvalue closest to the imaginary axis is sufficiently small. The failure from (c) to (d) can be resolved by either (i) computing more eigenvalues close to $\mathrm{i}\omega_1$, or (ii) by updating $\mathrm{i}\omega_1$ using (12).

After detection of a (possible) BP or a (possible) HBP, or of several of these along a branch between s_0 and $s_0 + \mathrm{ds}$, it remains to locate the BP or HBP. Again, there are various methods to do this, using, e.g., suitably extended systems [Gov00]. However, so far we restrict to a simple bisection, which works well and sufficiently fast in our examples.[5]

[2] In principle, instead of using (12) we could also compute the guesses ω_i by computing eigenvalues of $G_u(u, \lambda)$ at a given (u, λ); however, this may itself either require a priori information on the pertinent ω_i (for shifting), or we may again need to compute a large number of eigenvalues of G_u. Thus we find (12) more simple, efficient and elegant.

[3] A third typical kind of failure is that during a continuation step a number m of eigenvalues crosses the imaginary axis close to $\mathrm{i}\omega_1$, and simultaneously m already unstable eigenvalues leave the pertinent circle to the left due to a decreasing real part. The only remedy for this is to decrease the step–length ds. Clearly, a too large ds can miss bifurcations even if we could compute *all* eigenvalues, for instance if along the true branch eigenvalues cross back and forth.

[4] However, if additionally to bifurcations one is interested in the stability of (stationary) solutions, then the numbers of eigenvalues should not be chosen too small; otherwise the situation in Fig. 1(c,d) may easily occur, i.e., undetected negative eigenvalues.

[5] The only extended systems we deal with so far are those for continuation of stationary branch points and of fold

Table 1: Setting of `p.sw.bifcheck`

bifcheck	method, comments
0	Just continuation, no detection of bifurcation at all.
1	Use (9); fast, reliable, but only detects an odd number of eigenvalues crossing the imaginary axis, hence no Hopf.
2	Compute `p.nc.neig(j)` eigenvalues nearest to the shifts `p.nc.eigref(j)`, and for each $j = 0, 1, \ldots, j_{\max}$ count the number of those with negative real–part. General, can detect (7) and (8), but can falsely detect bifurcation points.

A located BP or HBP is saved as `p.file.dir/bpt[bpcount]` or `p.file.dir/hpt[hpcount]`, respectively, where bpcount and hpcount are counters, and BPs will moreover be indicated by `p.sol.ptype=1`, while HBPs have by `p.sol.ptype=3`. As before, BPs are indicated by a $\circ$ in bifurcation diagrams, while HBPs have $\diamond$.

Remark 2.2. To avoid unnecessary bisections and false BPs and HBPs we proceed as follows. After detection of a BP or HBP *candidate* with shift ω_j, we check if the eigenvalue μ closest to $\mathrm{i}\omega_j$ has $|\mathrm{Re}\mu| \le \mu_1$, with default $\mu_1 =$ `p.nc.mu1` $= 0.01$. If not, then we assume that this was a false alarm. Similarly, *after completing* a bisection we check if the eigenvalue μ *then* closest to ω_j has $|\mathrm{Re}\mu| < \mu_2$, with default $\mu_2 =$ `p.nc.mu2` $=$ `0.0001`, and only then accept the computed point as a BP (if $\omega_j = 0$) or HBP (if $\omega_j > 0$). In our examples, about 50% of the candidates enter the bisection, and of these about 10% are rejected afterwards, and hardly any false BPs or HBs are saved to disk. This seems to be a reasonable compromise between speed *and* not missing BPs and HBPs *and* avoiding false ones, but of course the values of μ_1, μ_2 may be highly problem dependent. Thus, some trial and error may be advisable, and if `p.sw.verb>1`, then the eigenvalues computed near 0 and the shifts $\mathrm{i}\omega_j$ are plotted for inspection, which might be helpful to estimate reasonable values for μ_1, μ_2. ⌋

2.2 Branch switching

Branch switching at a BP works as usual by computing an initial guess from the normal form of the stationary bifurcation, see [UWR14, §2.1]. Similarly, to switch to a Hopf branch of time periodic solutions we compute an initial guess from an approximation of the normal form

$$\dot{w} = \mu(\lambda)w + c_1(\lambda)|w|^2 w, \tag{13}$$

of the bifurcation equation on the center manifold associated to $(\lambda, \mu) = (\lambda_H, \mathrm{i}\omega_H)$. Thus we use

$$\mu(\lambda) = \mu_r(\lambda) + \mathrm{i}\mu_i(\lambda) = \mu_r'(\lambda_H)(\lambda - \lambda_H) + \mathrm{i}(\omega_H + \mathcal{O}(\lambda - \lambda_H)) + \mathcal{O}((\lambda - \lambda_H)^2), \tag{14}$$

and with $w = r\mathrm{e}^{\mathrm{i}\omega_H t}$ replace (13) by

$$0 = r\Big[\mu_r'(\lambda_H)(\lambda - \lambda_H) + c_1(\lambda_H)|r|^2\Big]. \tag{15}$$

Following [Kuz04], $c_1 = c_1(\lambda_H) \in \mathbb{R}$ is related to the first Lyapunov coefficient l_1 by $c_1(\lambda_H) = \omega_H l_1$, and we use the formulas from [Kuz04, p531-536] for the numerical computation of l_1. Setting $\lambda = s\varepsilon^2$ with $s = \pm 1$ we then have a nontrivial solution

$$r = \varepsilon\alpha, \quad \alpha = \sqrt{-s\mu'(\lambda_H)/c_1(\lambda_H)} \tag{16}$$

points, see [DRUW14, §2.1.4]. These can as well be used for branch point and fold point localization, although we did not yet elaborate on this; see `acfold_cmds.m` in `demos/acfold` for examples.

of (15) for $s = -\mathrm{sign}(\mu'(\lambda_H)/c_1)$, and thus take

$$\lambda = \lambda_H + s\varepsilon^2, \quad u(t) = u_0 + 2\varepsilon\alpha \mathrm{Re}(\mathrm{e}^{-\mathrm{i}\omega_H t}\Psi), \tag{17}$$

as an initial guess for a periodic solution of (4) with period near $2\pi/\omega$. The approximation (17) of the bifurcating solution in the center eigenspace, also called linear predictor, is usually accurate enough, and is the standard setting in the routine `p=hoswibra(dir,pt,ds,para,aux)`, where `dir` and `pt` determine the Hopf point previously saved to file, `ds` corresponds to the step length ε in (17), where `para=3` or `para=4` distinguishes between natural (by λ) vs arclength parametrization of the bifurcating branch, see §2.3, and where `aux` may be used to pass additional arguments, see App. A. The coefficients $s = \pm 1$ and α in (17) are computed, and ε is then chosen in such a way that the initial step length is `ds` in the norm (21) below.

2.3 The continuation of branches of periodic orbits

2.3.1 General setting

The continuation of the Hopf branch is, as usual, a predictor–corrector method, and for the corrector we offer, essentially, two different methods, namely natural (`p.sw.para=3`) and arclength (`p.sw.para=4`) continuation. For both, we reuse the standard `pde2path` settings for assembling G in (3) and Jacobians, such that the user does not have to provide new functions. In any case, first we rescale $t = Tt$ in (4) to obtain

$$M\dot{u} = -TG(u,\lambda), \quad u(\cdot,0) = u(\cdot,1), \tag{18}$$

with unknown period T, but with initial guess $T = 2\pi/\omega$ at bifurcation.

2.3.2 Arclength parametrization

We start with the arclength setting, which is more general and more robust, although our natural continuation has other advantages such as error control and adaptive mesh refinement for the time discretization, see below. We add the phase condition

$$\phi := \int_0^1 \langle u(t), M\dot{u}_0(t)\rangle \, \mathrm{d}t \stackrel{!}{=} 0, \tag{19}$$

where $\langle\cdot,\cdot\rangle$ is the scalar product in $\mathbb{R}^{n_u}$ and $\dot{u}_0(t)$ is from the previous continuation step, and we add the step length condition

$$\psi := \xi_{\mathrm{H}} \sum_{j=1}^{m} \big\langle u(t_j) - u_0(t_j), u_0'(t_j)\big\rangle + (1-\xi_{\mathrm{H}})\big[w_T(T-T_0)T_0' + (1-w_T)(\lambda-\lambda_0)\lambda_0'\big] - \mathrm{ds} \stackrel{!}{=} 0, \tag{20}$$

where again T_0, λ_0 are from the previous step, ds is the step–length, $' = \frac{\mathrm{d}}{\mathrm{d}s}$ denotes differentiation with respect to arclength, ξ_{H} and w_T denote weights for the u and T components of the unknown solution, and $t_0 = 0 < t_1 < \ldots < t_m = 1$ is the temporal discretization. Thus, the step length is ds in the weighted norm

$$\|(u,T,\lambda)\|_\xi = \sqrt{\xi_{\mathrm{H}} \sum_{j=1}^{m} \|u(t_j)\|_2^2 + (1-\xi_{\mathrm{H}})\big[w_T T^2 + (1-w_T)\lambda^2\big]}. \tag{21}$$

Even if ξ_{H} is similar to the (average) mesh–width in t, then the term $\xi_{\mathrm{H}} \sum_j \|u(t_j)\|_2$ is only a crude approximation of the "natural length" $\int_0^1 \|u(t)\|_2 \, \mathrm{d}t$. However, the choice of the norm is

somewhat arbitrary, and we found (21) most convenient. Typically we choose $w_T = 1/2$ such that T and λ have the same weight in the arclength (20). A possible choice for ξ_{H} to weight the number mn_u of components of u is

$$\xi_{\mathrm{H}} = \frac{1}{mn_u}. \tag{22}$$

However, in practice we choose $\xi_{\mathrm{H}} = \frac{10}{mn_u}$, or even larger (by another factor 10), since at the Hopf bifurcation the branches are "vertical" ($\|u-u_0\| = \mathcal{O}(\sqrt{|\lambda-\lambda_0|})$, cf. (17)), and ξ_{H} tunes the search direction in the extended Newton loop between "horizontal" (large ξ_{H}) and "vertical" (small ξ_{H}). See [UWR14, §2.1] for the analogous role of ξ for stationary problems.

The integral in (19) is discretized as a simple Riemann sum, such that the derivative of ϕ with respect to u is, with $\tilde{u}_0(t) = M\dot{u}_0(t)$,

$$\partial_u\phi = (h_1\tilde{u}(t_1)_1, \ldots, h_1\tilde{u}(t_1)_{n_u}, h_2\tilde{u}(t_2)_1, \ldots, h_2\tilde{u}(t_2)_{n_u}, \ \ldots, h_{l-1}\tilde{u}(t_{m-1})_{n_u}, 0, \ldots, 0), \tag{23}$$

n_u zeros at the end, where $h_l = t_{l+1}-t_l$ is the mesh–size in the time discretization. Similarly, denoting the tangent along the branch as

$$\tau = (\tau_u, \tau_T, \tau_\lambda), \quad \tau_u \in \mathbb{R}^{1\times mn_u} \text{ (row vector as in (23))}, \quad \tau_T,\ \tau_\lambda \in \mathbb{R}, \tag{24}$$

we can rewrite ψ in (20) as

$$\psi = \xi_{\mathrm{H}}\tau_u(u-u_0) + (1-\xi_{\mathrm{H}})(w_T\tau_T(T-T_0) + (1-w_T)\tau_\lambda(\lambda-\lambda_0)) - \mathrm{ds}. \tag{25}$$

Setting $U = (u, T, \lambda)$, and writing (18) as $\mathcal{G}(u, T, \lambda) = 0$, in each continuation step we thus need to solve

$$H(U) := \begin{pmatrix} \mathcal{G}(U) \\ \phi(u) \\ \psi(U) \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \in \mathbb{R}^{mn_u+2}, \tag{26}$$

for which we use Newton's method, i.e.,

$$U^{j+1}=U^j-\mathcal{A}(U^j)^{-1}H(U^j), \quad \mathcal{A}=\begin{pmatrix} \partial_u\mathcal{G} & \partial_T\mathcal{G} & \partial_\lambda\mathcal{G} \\ \partial_u\phi & 0 & 0 \\ \xi_{\mathrm{H}}\tau_u & (1-\xi_{\mathrm{H}})w_T\tau_T & (1-\xi_{\mathrm{H}})(1-w_T)\tau_\lambda \end{pmatrix}. \tag{27}$$

After convergence of U^j to U, i.e., $\|H(U)\| \leq$"tolerance" in some suitable norm, the next tangent τ_1 with preserved orientation $\langle\tau_0, \tau_1\rangle > 0$ can be calculated as usual from

$$\mathcal{A}(U)\tau_1 = (0, 0, 1)^T. \tag{28}$$

It remains to discretize in time and assemble $\mathcal{G}$ in (18) and the Jacobian $\partial_u\mathcal{G}$. For this we use (modifications of) routines from TOM [MT04], which assumes the unknowns in the form

$$u = (u_1, \ldots, u_m) = (u(t_1), u(t_2), \ldots, u(t_m)), \quad (m \text{ time slices}), \tag{29}$$

Then, using the TOM $k = 1$ setting, we have, for $j = 1, \ldots, m-1$, the implicit backwards time differences

$$(\mathcal{G}(u))_j = -h_{j-1}^{-1}M(u_j - u_{j-1}) - \frac{1}{2}T(G(u_j) + G(u_{j-1})), \tag{30}$$

where $u_0 := u_{m-1}$, and additionally the periodicity condition

$$G_m(u) = u_m - u_1. \tag{31}$$

The Jacobian is $\partial_u \mathcal{G} = A_1$, where we set, as it reappears below for the Floquet multipliers,

$$A_\gamma = \begin{pmatrix} M_1 & 0 & 0 & 0 & \dots & -H_1 & 0 \\ -H_2 & M_2 & 0 & 0 & \dots & 0 & 0 \\ 0 & -H_3 & M_3 & 0 & \dots & 0 & 0 \\ \vdots & \dots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & \dots & \ddots & \ddots & 0 & 0 \\ 0 & \dots & \dots & 0 & -H_{m-1} & M_{m-1} & 0 \\ -\gamma\, I & 0 & \dots & \dots & \dots & 0 & I \end{pmatrix}, \tag{32}$$

where $M_j = -h_{j-1}^{-1} M - \frac{1}{2} T G_u(u_j)$, $H_j = -h_{j-1}^{-1} M + \frac{1}{2} T G_u(u_{j-1})$, and I is the $n_u \times n_u$ identity matrix. The derivatives $\partial_T \mathcal{G}, \partial_\lambda \mathcal{G}$ in (27) are cheap from numerical differentiation, and $\partial_u \phi$ and τ do not change during Newton loops, and are easily taken care of anyway.

Remark 2.3. $\mathcal{A} \in \mathbb{R}^{(mn_u+2)\times(mn_u+2)}$ in (27), (28) consists of $A = A_1 = \mathcal{G}_u \in \mathbb{R}^{mn_u \times mn_u}$, which is large but sparse, and borders of widths 2, i.e., symbolically,

$$\mathcal{A} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}, \text{ with large and sparse } A, \text{ with } C^T, B \in \mathbb{R}^{mn_u \times 2}, \text{ and } D \in \mathbb{R}^{2\times 2}.$$

There are various methods to solve bordered systems of the form

$$\mathcal{A}x = b, \qquad b = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{33}$$

see, e.g., [Gov00]. Here we use the very simple scheme

$$V = A^{-1}B, x_1 = A^{-1}f, \tilde{D} = D - CV, y_1 = g - Cx_1, y_2 = \tilde{D}^{-1}y_1, x_2 = x_1 - Vy_2, x = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}. \tag{34}$$

The big advantage of such bordered schemes is that solving systems such as $Ax_1 = f$ (where we either pre-factor A for repeated solves, or use a preconditioned iterative method) is usually much cheaper due to the structure of A than solving $\mathcal{A}x = b$ (either by factoring $\mathcal{A}$, or by an iterative method with some preconditioning of $\mathcal{A}$).[6]

The scheme (34) is implemented in the function `mbel`. It may suffer from some instabilities, but often these can be corrected by a simple iteration: If $\|r\|$ with $r = \mathcal{A}x - b$ is too large, then we solve $\mathcal{A}\hat{x} = r$ (again by (34), which is cheap) and update $x = x - \hat{x}$, until $\|r\| \le$ "tolerance". We sometimes obtain poor solutions of (33) for $b = (0, 0, 1)^T$ from (28), but they typically can be improved by a few iterations. Altogether we found the scheme (34) to work well in our problems, with a typical speedup of up to 50 compared to the direct solution of $\mathcal{A}x = b$. Again, see [Gov00] for alternative schemes and detailed discussion.

For the solutions of $AV = B$ and $Ax_1 = f$ in (34) we give the option to use a preconditioned iterative solver from `ilupack` [Bol11]. For this, the user may set `p.hopf.ilss=1` before calling `p=cont(p)`, see the examples in §4. The (initial) computation of the ilu-preconditioner, the calls to the iterative solvers, and the update of the preconditioner when needed, are handled in `mbel`. The number of continuation steps along a (nontrivial) before a new preconditioner is needed can be quite large, and often the iterative solvers give a significant speedup. However, this of course is strongly problem dependent. Thus, the iterative option in `mbel` should rather be seen as somewhat experimental, and the default settings as a template for building problem adapted solvers; see [Bol11] for details on controlling the `ilupack` behavior. ⌋

[6] The special structure of A from (32) can also be exploited to solve $Ax = f$ in such a way that subsequently the Floquet multipliers, see §2.4, can easily be computed. See [Lus01] for comments on the related algorithms used in AUTO.

2.3.3 Natural parametrization

By keeping λ fixed during correction we cannot pass around folds, but on the other hand can take advantage of further useful features of TOM. TOM requires separated boundary conditions, and thus we use a standard trick and introduce, in the notation (29), auxiliary variables $\tilde{u} = (\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_m)$ and additional (dummy) ODEs $\dot{\tilde{u}}_l = 0$. Then setting the boundary conditions

$$u_1 - \tilde{u}_1 = 0, \quad u_m - \tilde{u}_m = 0 \tag{35}$$

corresponds to periodic boundary conditions for u. Moreover, we add the auxiliary equation $\dot{T} = 0$, and set up the phase condition

$$\phi = \langle u(0), M\dot{u}_0(0) \rangle \stackrel{!}{=} 0. \tag{36}$$

as an additional boundary condition. Thus, the complete system to be solved is

$$\begin{pmatrix} M\dot{u} \\ \dot{\tilde{u}} \\ \dot{T} \end{pmatrix} = \begin{pmatrix} -TG(u) \\ 0 \\ 0 \end{pmatrix}, \tag{37}$$

together with (35) and (36). We may then pass an initial guess (from a predictor) at a new λ to TOM, and let TOM solve for $(u, \tilde{u})$ and T. The main advantage is that this comes with error control and adaptive mesh refinement for the temporal discretization. See §4 for examples.

2.4 Floquet multipliers

The (in)stability of – and possible bifurcations from – a periodic orbit u_H are analyzed via the Floquet multipliers γ. These are obtained from finding nontrivial solutions (v, γ) of the variational boundary value problem

$$M\dot{v}(t) = -T\partial_u G(u(t))v(t), \tag{38}$$

$$v(1) = \gamma v(0). \tag{39}$$

Equivalently, the multipliers γ are the eigenvalues of the monodromy matrix $\mathcal{M}(u_0) = \partial_u \Phi(u_0, T)$, where $\Phi(u_0, t)$ is the solution of the initial value problem (4) with $u(0) = u_0$ from u_H. Thus, $\mathcal{M}(u_0)$ depends on u_0, but the multipliers γ do not. By translational invariance, there always is the trivial multiplier $\gamma_1 = 1$. $\mathcal{M}(u_0)$ is the linearization of the Poincaré map $\Pi(\cdot; u_0)$ around u_0, which maps a point $\tilde{u}_0$ in a hyperplane Σ through u_0 and transversal to u_H to its first return to Σ. Therefore, a necessary conditions for the bifurcation from a branch $\lambda \mapsto u_H(\cdot, \lambda)$ of periodic orbits is that at some $(u_H(\cdot, \lambda_0), \lambda_0)$, additional to the trivial multiplier $\gamma_1 = 1$ there is a second multiplier γ_2 (or a complex conjugate pair $\gamma_{2,3}$) with $|\gamma_2| = 1$, which generically leads to the following bifurcations (see, e.g., [Sey10, Chapter 7] or [Kuz04] for more details):

(i) $\gamma_2 = 1$, yields a fold of the periodic orbit, or a transcritical or pitchfork bifurcation of periodic orbits.

(ii) $\gamma_2 = -1$, yields a period–doubling bifurcation, i.e., the bifurcation of periodic orbits $\tilde{u}(\cdot; \lambda)$ with approximately double the period, $\tilde{u}(\tilde{T}; \lambda) = \tilde{u}(0; \lambda)$, $\tilde{T}(\lambda) \approx 2T(\lambda)$ for λ near λ_0.

(iii) $\gamma_{2,3} = \mathrm{e}^{\pm \mathrm{i}\vartheta}$, $\vartheta \neq 0, \pi$, yields a torus (or Naimark–Sacker) bifurcation, i.e., the bifurcation of periodic orbits $\tilde{u}(\cdot, \lambda)$ with two "periods" $T(\lambda)$ and $\tilde{T}(\lambda)$; if $T(\lambda)/\tilde{T}(\lambda) \neq \mathbb{Q}$, then $\mathbb{R} \ni t \mapsto \tilde{u}(t)$ is dense in certain tori.

Here we are first of all interested in the computation of the multipliers. Using the same discretization for v as for u, it follows that γ and $v = (v_1, \ldots, v_m)$ have to satisfy the matrix eigenvalue problem

$$A_\gamma v = 0, \tag{40}$$

where now γ in (32) is free. From this special structure it is easy to see, that $\mathcal{M}(u_{j_0})$ can be obtained from certain products involving the M_j and the H_j, for instance

$$\mathcal{M}(u_{m-1}) = M_{m-1}^{-1} H_{m-1} \cdots M_1^{-1} H_1. \tag{41}$$

Thus, a simple way to compute the γ_j is to compute the product (41) and subsequently (a number of) the eigenvalues of $\mathcal{M}(u_{m-1})$. We implemented this in a function `floq.m`, and using

$$\mathrm{err}_{\gamma_1} := |\gamma_1 - 1| \tag{42}$$

as a measure of accuracy we find that this works fast and accurately for our dissipative examples. Typically $\mathrm{err}_{\gamma_1} < 10^{-10}$, although at larger amplitudes of u_H, and if there are large multipliers, this may go down to $\mathrm{err}_{\gamma_1} \sim 10^{-8}$, which is the (default) tolerance we require for the computation of u_H itself. Thus we give a warning if $\mathrm{err}_{\gamma_1} >$ `p.hopf.fltol`, with default setting `p.hopf.fltol`$=10^{-7}$. However, for the optimal control example in §4.5, where we naturally have multipliers γ_j with $|\gamma_j| > 10^{20}$ and larger[7], `floq` completely fails to compute any meaningful multipliers.

Indeed, in, e.g., [FJ91, Lus01], it is discussed that methods based directly on (41)

- may give considerable numerical errors, in particular if there are both, very small and very large multipliers γ_j;
- discard much useful information, for instance eigenvectors of $\mathcal{M}(u_l)$, $l \neq m-1$, which are useful for branch switching.

As an alternative, [Lus01] suggests to use a periodic Schur decomposition [BGVD92] to compute the multipliers (and subsequently pertinent eigenvectors), and gives examples that in certain cases this gives much better accuracy, which as above is defined by the distance of the (numerical) trivial multiplier to 1. See also [Kre01, Kre06] for similar ideas and results.

We thus provide a function `floqps`, which, based on `pqzschur.m` and `percomplex.f` from [Kre01], computes a periodic Schur decomposition of the matrices involved in (41), from which we immediately obtain the multipliers, see Remark 2.4(d). For large n_u and m, `floqps` gets rather slow, and thus for now we use it in two ways. First, to validate (by example) `floq`, and second to compute the multipliers when `floq` fails, in particular for our OC example.

In summary, for small to medium sized *dissipative* problems, i.e., $n_u * m < 10000$, say, computing (a number of) multipliers with `floq`, flagged by `p.hopf.flcheck=1`, is a matter of a seconds, and only takes a fraction of the time needed to compute the orbit itself. For the *ill-posed* OC problems we have to use `floqps`, flagged by `p.hopf.flcheck=2`, which is slower and for medium sized problems can be as slow as the computation of u_H. In any case, because we do not yet consider the localization of the bifurcations (a)–(c) from periodic orbits (this is work in progress[8]), for efficiency we give the option to switch off the simultaneous computation of multipliers during continuation of periodic orbits by setting `p.hopf.flcheck=0`. As a simple alternative we provide `floqap` and `floqpsap` for the *a posteriori* computation of multipliers.

Remark 2.4. (a) To save the stability information on the branch (if `p.hopf.flcheck`> 0) we define

$$\mathrm{ind}(u_H) = \text{number of multipliers } \gamma \text{ with } |\gamma| > 1 \text{ (numerically: } |\gamma| > 1 + \texttt{fltol}), \tag{43}$$

such that unstable orbits are characterized by $\mathrm{ind}(u_H) > 0$, and put $\mathrm{ind}(u_H)$ in row three of `p.branch`, i.e., `p.branch`$(3, \texttt{count}) = \mathrm{ind}(u_H)$. This is consistent with the case of steady states, for

[7] I.e., $|\gamma_{n_u}| \to \infty$ as $n_u \to \infty$, although the orbits may still be stable in the sense of optimal control, see §4.5

[8] In contrast to bifurcation from stationary solutions, the *localization* of the bifurcations via bisection becomes too slow (except for simple 1D cases). Instead, the bifurcations will be localized via properly defined extended systems, depending on the type (i)–(iii) of the (expected) bifurcation, which will also allow to efficiently compute curves of bifurcation points from periodic orbits.

which `p.branch(3,count)` contains the number of unstable eigenvalues of $\partial_u G$. Thus, a change in $\mathrm{ind}(u_H)$ signals a possible bifurcation, and via

$$\gamma_{\mathrm{cand}} := \mathrm{argmin}\{|\gamma_j| : |\gamma_j| > 1\}$$

we also get an approximation for the critial multiplier, and thus a classification of the possible bifurcation in the sense (i)-(iii). For `p.hopf.flcheck=0` we set `p.branch(3,count)=-1`.

(b) In `floq` we compute $n_+ =$ `p.hopf.nfloq` multipliers $\gamma_2, \ldots, \gamma_{n_+}$ of largest modulus (recall that we reserve γ_1 for the trivial multiplier), with $|\gamma_2| \geq |\gamma_3| \geq \ldots \geq |\gamma_{n_+}|$, and count how many of these have $|\gamma_j| > 1$, which gives $\mathrm{ind}(u_H)$ if we make sure that $|\gamma_{n_+}| < 1$. For dissipative systems, typically $1 < n_+ \ll n_u$, and the large multipliers of $\mathcal{M}$ can be computed efficiently and reliably by vector iteration. However, it does happen that some of the small multipliers do not converge, in which case we also give a warning, and recommend to check the results with `floqps`.

(c) The idea of the periodic Schur decomposition is as follows. Given two collections $(A_i), (B_i)$, $i = 1, \ldots, m$, of matrices $A_i, B_i \in \mathbb{C}^{n\times n}$, `pqzschur` computes $Q_i, Z_i, \tilde{A}_i, \tilde{B}_i \in \mathbb{C}^{n\times n}$ such that $\tilde{A}_i, \tilde{B}_i$ are upper triangular, Q_i, Z_i are orthogonal, and

$$\begin{array}{ll} A_1 = Q_1 \tilde{A}_1 Z_m^H, & B_1 = Q_1 \tilde{B}_1 Z_1^H \\ A_2 = Q_2 \tilde{A}_2 Z_1^H, & B_2 = Q_2 \tilde{B}_1 Z_2^H \\ \ldots, & \ldots \\ A_m = Q_m \tilde{A}_m Z_{m-1}^H, & B_m = Q_m \tilde{B}_m Z_m^H. \end{array}$$

Consequently, for the product $\mathcal{M} = B_m^{-1} A_m \cdots B_1^{-1} A_1$ we have

$$\mathcal{M} = Z_m \tilde{B}_m^{-1} \tilde{A}_m \cdots \tilde{B}_1^{-1} \tilde{A}_1 Z_m^H,$$

and similar for products with other orderings of the factors. In particular, the eigenvalues of $\mathcal{M}$ are given by the products $d_i = \prod_{j=1}^{m} \tilde{a}_{ii}^{(j)} / \tilde{b}_{ii}^{(j)}$, and, moreover, the associated eigenvectors can also be extracted from the Q_j, Z_j, see [Kre06] for further comments.

(d) Alternatively to using Floquet multipliers, we can assess the stability of the periodic orbits by using the time–integration routines from `pde2path`, which moreover has the advantage of giving information about the evolution of perturbations of unstable solutions; see §4 for examples, where in all cases perturbations of unstable periodic orbits lead to convergence to some other (stable) periodic orbit. ⌋

3 Download, installation, help, and data structures

The package and demos, and older `pde2path` manuals and further information can be downloaded at [Uec16b]. The file `pde2path.tar.gz` (or `pde2path.zip`) unpacks to the root-directory `pde2path`, which contains the directory tree shown in Fig. 2(a), where `demos` contains the stationary `pde2path` demos, `hopfdemos` contains the demos described here, `html` contains help, `libs` contains the `pde2path` libraries, `ocdemos` contains the optimal control demos described in [Uec15], `pqzschur` contains the periodic Schur decomposition, which has to be mexed (see README in `pqzschur` for further instructions), and `OOPDElightNA` is a "light" version of `OOPDE` [Prü16], with No Abstract classes for compatibility with older `Matlab` versions.

To get started, in `Matlab` change into the `pde2path` directory and run `setpde2path`, which also makes available the help system. Calling `p2phelp` yields the main help menu shown in Fig. 2(b). The first two topics are short thematic overviews of the data structures and main functions in `pde2path`, while clicking `p2plib`, ..., `tom` yields complete alphabetic function overviews of these

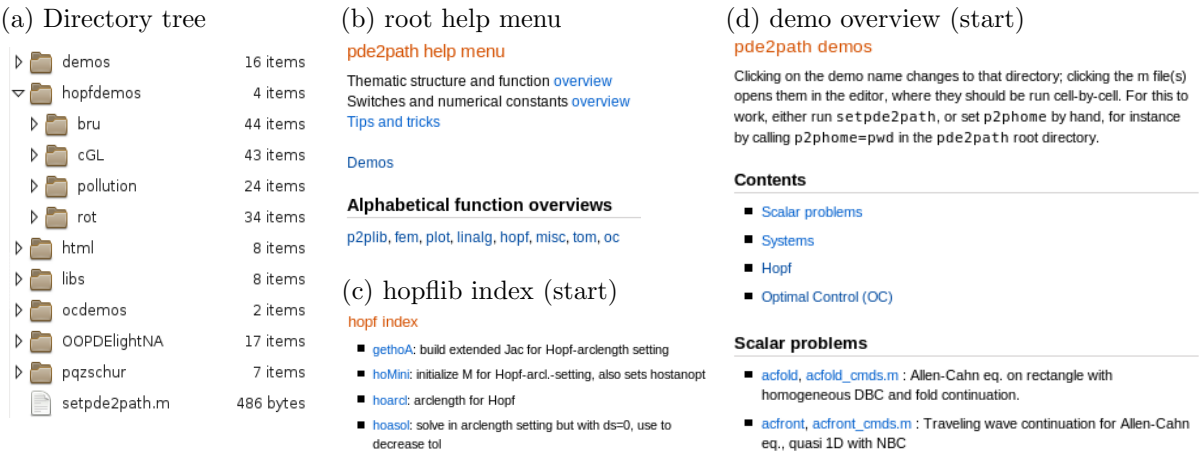


Figure 2: Directory tree, Root help menu, and starting parts of `hopflib` index and demo overview.

`pde2path` libraries, with a short description of each function, which can then be clicked for documentation.[9] Similarly, clicking on `demos` opens the demo overview in (d), with brief descriptions of and pertinent links to the demo directories and the basic script files.

As already said, see [UWR14] and in particular [DRUW14] for the basic organization of the `pde2path`-struct `p` describing a given problem, which however for convenience we briefly recall in Table 2, where the last two lines contain new fields described here. Both are supplementary in the following sense: `p.pdeo` is only needed/used if the user chooses the `OOPDE` setup described in §4.1, while `p.hopf` is not needed/used for stationary problems, but initialized by `hoswibra`, i.e., by branch switching at a HBP.

Table 2: Fields in structure `p`; see [DRUW14] for a review of the contents of (and standard settings in) these fields, except for `p.pdeo`, `p.hopf`, described in §4.1 and §A, respectively. See also the html help or `stanparam.m` for further information.

field	purpose	field	purpose
fuha	**fu**nction **ha**ndles, e.g., fuha.G, ...	nc	**n**umerical **c**ontrols, e.g., nc.tol, ...
sw	**sw**itches such as sw.bifcheck,...	sol	values/fields calculated at runtime
eqn	tensors c, a, b for the sfem=1 setup	mesh	mesh data (if the `pdetoolbox` is used)
plot	switches and controls for plotting	file	switches etc for file output
time	timing information	pm	pmcont switches
fsol	switches for the `fsolve` interface	nu,np	# PDE unknowns, # meshpoints
u,tau	solution and tangent	branch	branch data
usrlam	vector of user set target values for the primary parameter, default usrlam=[];		
mat	problem matrices, in general data that is not saved to file		
pdeo	OOPDE data if OOPDE is used, see [Prü16] for documentation		
hopf	Hopf data, initialized in `hoswibra`		

To get started with the `hopfdemos`,

- change into one of the `hopfdemos` directories, and open `*cmds.m`; here * stands for `cGL, rot, bru` or `pollution`, and the spatial dimension, where we recommend to start with `cGL1dcmds`;
- execute `*cmds.m` cell by cell, and compare to the descriptions given in §4.2–4.5.

To set up your own problem, copy the demo directory which seems closest to your problem to a new directory, then modify the functions and scripts in it. See also the various (stationary) demos in `demos`, described in [UWR14, DRUW14], for further tips and tricks regarding domains, BC, and the general setup of the pertinent G from (3).

[9] Of course, help on any `pde2path` function `foo` is also given by typing `help foo` or `doc foo`, but in practice we find the alphabetic library overviews such as in Fig. 2(c) most convenient.

To use `ilupack`, which we found useful in a number of demos, mex its `Matlab` interface, put `ilupacks` */mex directory in the `Matlab` path, and before calling `cont` for a Hopf branch set `p.hopf.ilss=1`. See also `mbel.m` in directory `hopflib/` concerning the standard settings for `ilupack` used. The settings concerning `droptol, maxit` etc. are often problem dependent, and thus should be modified by copying `mbel.m` to your problem directory and modifying it there.

4 Four examples

4.1 OOPDE, and general remarks

Before presenting the demos we briefly comment on the new `OOPDE` setting in `pde2path`. `OOPDE` (object oriented PDE) [Prü16] is a FEM package in `Matlab`. Inter alia, it provides, in 1D, 2D and 3D, most of the functionality that the `pdetoolbox` provides in 2D, and with similar basic interfaces. Thus, we use it to transfer the functionality of `pde2path` to 1D and 3D, and to also make `pde2path` independent of the `pdetoolbox` in 2D. A major difference, however, is the object oriented (OO) setup of `OOPDE`, which has advantages such as tighter control of data access by the user and natural reuse resp. overload of methods by inheritance, although currently we hardly use the OO aspects of `OOPDE`.

Our basic strategy for using `OOPDE` is as follows: There are three templates for creating pde–objects, namely the subclasses `stanpdeo1D,stanpdeo2D, stanpdeo3D` of the `OOPDE` class `pde`. These only set up simple domains (interval, rectangle, cuboid, respectively), the grids (intervals, triangles, tetrahedra) and the finite elements (piece-wise linear continuous). Thus, calling, e.g., `p.pdeo=stanpdeo1D(lx,2*lx/nx)`, we have `pdeo` as a pde object in `p`, i.e., the 1D domain $\Omega = (-l_x, l_x)$ with a mesh of width $2l_x/n_x$, and, by default, linear Lagrange elements associated to it. This is already enough to, e.g., call `[K,M,dummyF]=p.pdeo.fem.assema(p.pdeo.grid,1,1,0)` to assemble the (one component) mass matrix M and stiffness matrix K (such that $M^{-1}K$ corresponds to the Neumann Laplacian), and there are very useful tools to set up boundary conditions as well. After some post processing to, e.g., create the needed matrices for systems of PDEs, we save these system matrices in `p.mat.K, p.mat.M`; see `oosetfemops` in the examples below. After this, we can implement all functions necessary to describe the system in a standard `pde2path` way. This way we only use a small subset of the `OOPDE` capabilities, but on the other hand stay close to the original `pde2path` layout.

The basic setup of all four demos (`cGL, rot, bru, pollution`) is similar. Each demo directory contains:

- Function files named `*init.m` for setting up the geometry and the basic `pde2path` data, where `*` stands for the problem, i.e., `cGL, rot, bru` or `pollution`.
- Main script files, such as `cGL*dcmds.m` where * stands for the space dimension.
- Function files `sG.m` and `sGjac.m` for setting up the rhs of the equation and its Jacobian. Here, somewhat different from the setup in [UWR14], we omit special names for these functions, i.e., they are just files `sG.m, sGjac.m` in each problem directory. Of course, one can still give individual names such as, e.g., `cGLsG.m`, and then set `p.fuha.sG=@cGLsG`. Moreover, we decided to *not* introduce a new function handle for the nonlinearity f in (1), which is needed in the computation of the coefficient α in the normal form (17); instead, we assume that f is encoded as `nodalf.m` in the problem directory, and of course we then also call `nodalf` in `sG.m`.[10]

[10] This assumes the semilinear setting, flagged by `p.sw.sfem=1` (`pdetoolbox`) or `p.sw.sfem=-1` (`OOPDE`), i.e., $\partial_t u = c\Delta u + au + b \otimes \nabla u + f(u)$, where $c, a \in \mathbb{R}^{N\times N}$ and $b \in \mathbb{R}^{N\times N\times 2}$ are fixed tensors, f is the only nonlinearity in (1), and the BC are linear as well. The computation of α is not yet supported for quasilinear problems, and thus for these the user should provide a guess, see the description of `hoswibra.m` in §A. However, so far all our examples are semilinear.

- `oosetfemops.m` for setting up the system matrices, except for the demo `rot` where we use the old `pdetoolbox` setup.
- A few auxiliary functions, for instance small modifications of the basic plotting routine `hoplot.m` from the `hopf` library, which we found convenient to have problem adjusted a posteriori plots, while during continuation we typically use the standard `hoplot`.
- Some auxiliary scripts `auxcmds.m`, which contain commands, for instance for creating movies or for mesh–refinement, which here we do not discuss in detail, but which we find convenient for illustrating either some mathematical aspects of the models, or some further `pde2path` capabilities, and which we hope the user will find useful as well.
- For the demo `pollution` (which can also be seen as a optimal control demo) we additionally have the functions `polljcf.m`, which implements the objective function, and `pollbra.m`, which combines output from the standard Hopf output `hobra.m` and the standard OC output `ocbra.m`.

In the following, we mostly focus on explaining the output of the main script files (i.e., the relevant plots), and on relating them to some mathematical background of the equations. Thus, for comments on implementation details, in particular for setting up the equations and Jacobians in `sG.m` and `sGjac.m` we mostly refer to the m-files, and to [UWR14, DRUW14] and [Uec16b]. However, for our first example we also give some comments in Appendix B. The script files `*cmds` are in cell mode, and we recommend to also execute them cell–by–cell to see the effect of the individual commands.

In all examples, the meshes are chosen rather coarse, to quickly get familiar with the software. We did check for all examples that these coarse meshes give reliable results by running the same simulations on finer meshes, without qualitative changes. In some simulations (demo `rot`, and `cGL` in 3D and `bru` in 2D) we additionally switch off the simultaneous computation of Floquet multipliers, i.e., set `p.hopf.flcheck=0`, and instead compute the multipliers a posteriori at selected points on branches. Computing the multipliers simultaneously is possible as well, but may be slow. Nevertheless, even with the coarse meshes some commands, e.g., the continuation of Hopf branches in 3+1D, may take several minutes, so for `cGL` and `bru` we recommend to start with the 1D problems, where the numerics only take a few seconds. All computational times given in the following are from an i5 laptop with Linux Mint 17 and `Matlab` 2013a.

4.2 A complex Ginzburg–Landau equation: Demo `cGL`

We consider the cubic-quintic complex Ginzburg–Landau equation

$$\partial_t u = \Delta u + (r + \mathrm{i}\nu)u - (c_3 + \mathrm{i}\mu)|u|^2 u - c_5|u|^4 u, \quad u = u(t,x) \in \mathbb{C}, \tag{44}$$

with real parameters r, ν, c_3, μ, c_5. Equations of this type are canonical models in physics, and are often derived as amplitude equations for more complicated pattern forming systems [AK02, Mie02]. Using real variables u_1, u_2 with $u = u_1 + \mathrm{i}u_2$, (44) can be written as a real 2–component system of the form (3), i.e.,

$$\partial_t \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} \Delta + r & -\nu \\ \nu & \Delta + r \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} - (u_1^2 + u_2^2) \begin{pmatrix} c_3 u_1 - \mu u_2 \\ \mu u_1 + c_3 u_2 \end{pmatrix} - c_5 (u_1^2 + u_2^2)^2 \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}. \tag{45}$$

We set

$$c_3 = -1, c_5 = 1, \nu = 1, \mu = 0.1, \tag{46}$$

and use r as the main bifurcation parameter. Considering (45) on, e.g., a (generalized) rectangle

$$\Omega = (-l_1\pi, l_1\pi) \times \cdots \times (-l_d\pi, l_d\pi) \tag{47}$$

with homogeneous Dirichlet BC or Neumann BC, or with periodic BC, we can explicitly calculate all Hopf bifurcation points from the trivial branch $u = 0$, and, for periodic BC, the bifurcating time periodic branches. For this let

$$u(x,t) = a\mathrm{e}^{\mathrm{i}(\omega t - k\cdot x)}, \text{ with wave vector } k = (k_1, \dots, k_d), k_j \in \frac{1}{2l_j}\mathbb{Z}, \tag{48}$$

and temporal period $2\pi/\omega$, which yields

$$|a|^2{=}|a(k,r)|^2{=}-\frac{c_3}{2c_5} \pm \sqrt{\frac{c_3^2}{4c_5^2} + r - |k|^2}, \quad \omega{=}\omega(k,r){=}\nu - \mu|a|^2, \quad |k|^2{=}k_1^2 + \ldots + k_d^2. \tag{49}$$

Note that ω and hence the period $T = 2\pi/\omega$ depend on $|a|$, that $u(t,x)$ on each branch is a single harmonic in x and t, and that the phase of a is free. Using (46) we obtain subcritical Hopf bifurcations of solutions (48) at

$$r = |k|^2, \text{ with folds at } r = |k|^2 - \frac{1}{4}. \tag{50}$$

Moreover, for these orbits we can also compute the Floquet multipliers explicitly. For instance, restricting to $k = 0$ in (48), and also to the invariant subspace of spatially independent perturbations, in polar-coordinates $\tilde{u}(t) = \tilde{a}(t)\mathrm{e}^{\mathrm{i}\tilde{\phi}(t)}$ we obtain the (here autonomous) linearized ODEs

$$\frac{\mathrm{d}}{\mathrm{d}t}\tilde{a} = h(r)\tilde{a}, \quad \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\phi} = -2\mu a\tilde{a}, \text{ where } h(r) = r + 3a^2 - 5a^4. \tag{51}$$

The solution is $\tilde{a}(T) = \mathrm{e}^{h(r)T}\tilde{a}(0)$, $\tilde{\phi}(T) = \tilde{\phi}(0) + \frac{a}{h(r)}(\mathrm{e}^{h(r)T} - 1)\tilde{a}(0)$, and therefore the analytic monodromy matrix (in the $k{=}0$ subspace) is $\mathcal{M}_{k=0} = \begin{pmatrix} \mathrm{e}^{h(r)T} & 0 \\ \frac{a}{h(r)}(\mathrm{e}^{h(r)T} - 1) & 1 \end{pmatrix}$ with multipliers $\gamma_1{=}1$ and $\gamma_2{=}\mathrm{e}^{h(r)T}$.

Thus, (45) makes a nice toy problem to validate and benchmark our routines, where, as periodic BC are somewhat nonphysical, and to avoid translational invariance, cf. Remark 2.1, we use Neumann and Dirichlet BC. For these we still have the formula $r = |k|^2$ for the HBPs, although we lose the explicit branches, except the spatially homogeneous branch for $k = 0$ with Neumann BC.

Remark 4.1. An immediate consequence of the BC is that the solutions (48) have nodal sets, i.e., fixed subsets of Ω on which $u(x,t) = 0$ for all t, or equivalently $(u_1, u_2) = (0,0)$. For simplicity restricting to 1D and Dirichlet BC, the general solution can be written as

$$u(x,t) = \sum_{n\in\mathbb{N}} a_n(t)\sin(nx/l), \quad a_n \in \mathbb{C}. \tag{52}$$

Then $\sin(nx/l) = 0$ on $|x| = lm/n$, $0 \le m \le n$, and the bifurcation at, e.g., $r = n_0/2l$ is into the invariant subspace

$$u(x,t) = \sum_{n\in\{n_0,3n_0,5n_0,\ldots\}} a_n(t)\sin(nx/l),$$

i.e., the nodal structure is determined at bifurcation. The same conclusions hold for any spatial dimension d, and Neumann boundary conditions, and therefore the bifurcation for (45) on cuboids (47) with Neumann or Dirichlet BC is always to *standing* (i.e., oscillatory) patterns. In §4.3 we will consider a similar problem on a circle, where suitable boundary conditions lead to the bifurcation of rotating patterns. ⌋

The cGL demo directory consists, as noted above, of some function files to set up and describe (45), and some script files to run the simulations. As functions we have

- `cGLinit.m`, which (depending on the spatial dimension) sets up the domain, mesh, boundary conditions, and sets `p.fuha.sG=@sG` and `p.fuha.sGjac=@sGjac`;
- `sG.m` and `sGjac`, which encode (45) and the associated Jacobian of G;
- `oosetfemops.m`, which uses `OOPDE` s `assema` and `assemb` to set the system matrices;
- the auxiliary function `plotana` which plots the analytic branches (49).

Then we have three script files, `cGL*dcmds.m`, where *=1,2,3 stands for the spatial dimension, and an `auxcmds.m` script file. The three files `cGL*dcmds.m` are very similar, i.e., only differ in filenames for output and some plotting commands, but the basic procedure is always the same:

1. call `cGLinit`, then `cont` to find the HBPs from the trivial branch $u \equiv 0$;
2. compute branches of periodic orbits by calling `hoswibra` and `cont` again, including (in 1D and 2D, but for efficiency not in 3D) the multipliers, then plotting.
3. additionally assess the stability of periodic solutions u_H, and in case of instability get the evolution of perturbatons, by using `hotintxs` to time integrate (45) with initial condition $u_H(\cdot, 0)$.

There are no real eigenvalues of $\partial_u G$ on the trivial branch $u = 0$ in this example. Thus, for the HBP detection we can safely use `bifcheck=2` with `neig=10`, and postpone to §4.4 and §4.5 the discussion of problems which require preparatory calls to `initeig` to first estimate possible values for ω_1 (which here would be known a–priori as $\omega_1 = 1$). As already said, here we do not repeat all details about, e.g., translating (45) into `sG.m`, but give some brief remarks in Appendix B.

In 1D we use Neumann BC, and $n_x = 30$ spatial, and (without mesh-refinement) $m = 20$ temporal discretization points. Just for illustration, in `cGL1dcmds.m` we compute the first two branches using the `para=4` (arclength) setting from the start, while for the third branch we first do 5 steps with `para=3` (nat.param.), where `tomsol` refines the starting t–mesh of 20 points to 40 points.[11] This produces the plots in Fig. 3, where the norm in (a) is

$$\|u\|_* := \|u\|_{L^2(\Omega\times(0,T),\mathbb{R}^N)}/\sqrt{T|\Omega|}, \tag{53}$$

which is our default for plotting of Hopf branches. Additionally we remark that during the continuation `hoplot` also plots the time–series $t \mapsto u_1(x_0, t), u_2(x_0, t)$ for some mesh point x_0, selected by the index `p.hopf.x0i`, which is set in `cGLinit` (see also Fig. 4). The simulations run in less than 10 seconds per branch, but the rather coarse meshes lead to some inaccuracies. For instance, the first three HBPs, which analytically are at $r = 0, 1/4, 1$, are obtained at $r = 6*10^{-5}, 0.2503, 1.0033$, and (b) also shows some visible errors in the period T. However, these numerical errors quickly decay if we increase n_x and m, and runtimes stay small.

On `b1`, initially there is one unstable multiplier γ_2, i.e., $\mathrm{ind}(u_H) = 1$, cf. (43), which passes through 1 to enter the unit circle at the fold. Its numerical value is 10^{-5} close to the analytical result from (51), and this error decreases upon refining the t–mesh. On b2 we start with $\mathrm{ind}(u_H) = 3$, and have $\mathrm{ind}(u_H) = 2$ after the fold. Near $r = 0.45$ another multiplier moves through 1 into the unit circle, such that afterwards we have $\mathrm{ind}(u_H) = 1$, with, for instance $\gamma_2 \approx 167$ at $r = 1$. Thus, we may expect a type (i) bifurcation (cf. p. 11) near $r = 0.45$, and similarly we can identify a number of possible bifurcation on b3 and other branches. The trivial multiplier γ_1 is 10^{-12} close to 1 in all these computations, using `floq`.

Switching to continuation in another parameter works just as for stationary problems by calling `p=hoswiparf(...)`. See the start of `cGL/auxcmds.m` for an example, and Fig. 4 for illustration.

Via Fig. 5 we briefly explain how to use time integration to assess the stability of periodic solutions, and in particular obtain the time evolution of perturbations of unstable orbits. The idea

[11] See also `cGL/auxcmds.m` for examples how to switch back and forth between `para=4` and `para=3` with the aim of error control and mesh refinement in t.

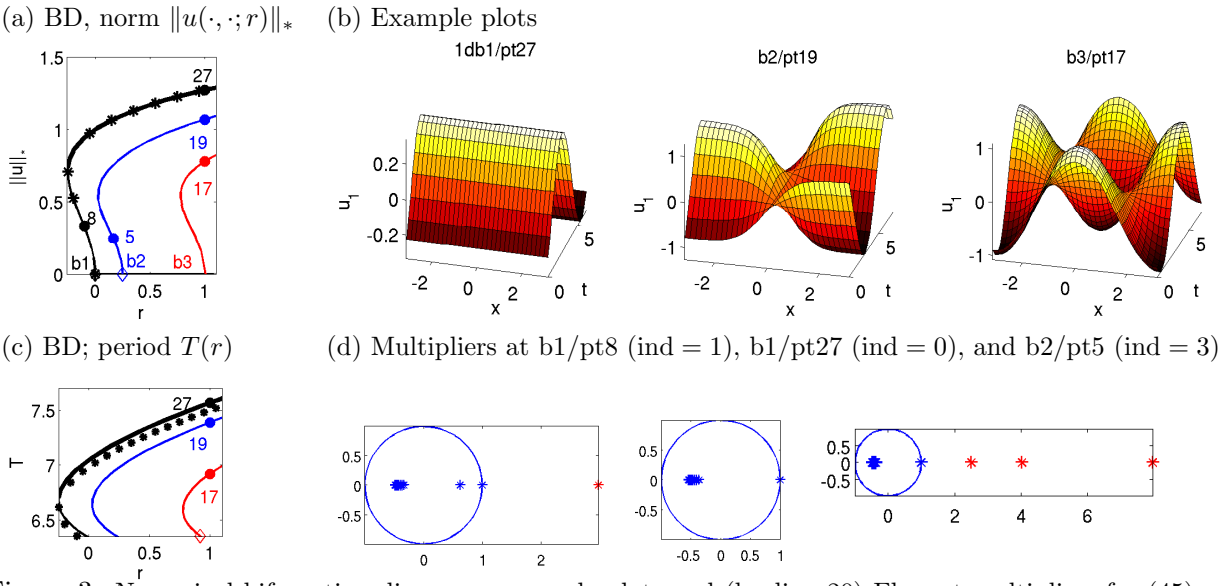


Figure 3: Numerical bifurcation diagrams, example plots and (leading 20) Floquet multipliers for (45) on the domain $\Omega = (-\pi, \pi)$ with Neumann BC, 30 grid–points in x. Parameters $(\nu, \mu, c_3, c_5) = (1, 0.1, -1, 1)$, hence bifurcations at (restricting to the first three branches) $r = 0$ ($k = 0$, spatially homogeneous branch, black), $r = 1/4$ ($k = 1/2$, blue) and $r = 1$ ($k = 1$, red), see (50). The black dots in (a), (b) are from the analytical solution (49) with $k = 0$. The thick part of the black line in (a),(b) indicates the only stable periodic solutions.

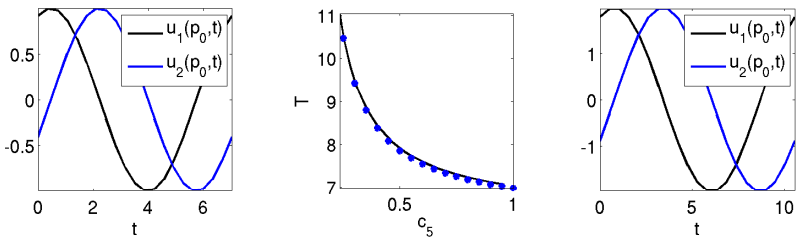


Figure 4: Continuing the solution b1/pt28 from Fig. 3(a,b) in c_5. (a), (c) show $t \mapsto (u_1, u_2)(x_0, t)$ for some user selected x_0 (here arbitrary). In accordance with (49) the temporal dependence stays a single harmonic, and only the amplitude and period change. The blue dots in (b) are the analytical results (49).

is to start time integration from some point on the periodic orbit, e.g. $u_0(\cdot) = u_H(\cdot, 0)$, and to monitor, inter-alia, $e(t) := \|u(t,\cdot) - u_0(\cdot)\|$, where by default $\|\cdot\| = \|\cdot\|_\infty$. Without approximation error for the computation of u_H (including the period T) and of $t \mapsto u(\cdot, t)$ we would have $e(nT) = 0$. In general, even if u_H is stable we cannot expect that, in particular due to errors in T which will accumulate with n, but nevertheless we usually can detect instability of u_H if at some t there is a qualitative change in the time–series of $e(t)$.[12] In Fig. 5(a), where we use the smaller amplitude periodic solution at $r = 0$ for the IC, this happens right from the start. Panel (b) illustrates the stability of the larger amplitude periodic solution at $r = 0$, while in (c) the instability of the solution on `h2` at $r = 1$ manifests around $t = 30$, with subsequent convergence to the (stable) spatially homogeneous periodic orbit.

Mainly to illustrate how to set up boundary conditions in `OOPDE`, in 2D we choose homogeneous

[12] The time integration `hotintxs` takes inter alia the number `npp` of time steps per period T as argument. Time integration is much faster than the BVP solver used to compute the periodic orbits, and thus `npp` can be chosen significantly larger than the number m of time-discretization points in the BVP solver. Thus, choosing `npp` $= 5m$ or `npp` $= 10m$ appears a reasonable practice.

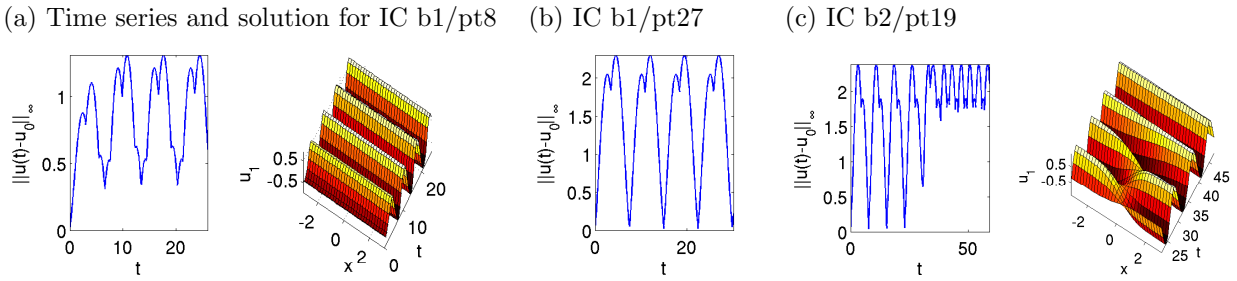


Figure 5: Stability experiments for (45) in 1D. (a) IC h1/pt8, time series of $\|u(\cdot,t)-u_0\|_\infty$ and $u_1(x,t)$, showing the convergence to the larger amplitude solution at the same r. (b) IC h1/pt27 from Fig. 3, where we plot $\|u(\cdot,t)-u_0\|_\infty$ for $t\in[0,4T]$, which shows stability of the periodic orbit, and a good agreement for the temporal period under time integration. (c) instability of b2/pt19 from Fig. 3, and again convergence to the solution on the b1 branch. Note that the time–stepping is much finer than the appearance of the solution plots, but we only save the solution (and hence plot) every 100th step, cf. footnote 12.

Dirichlet BC for u_1, u_2, see also the discussion of `oosetfemops.m` in Appendix B. Then the first two HBPs are at $r_1 = 5/4$ ($k = (1/2, 1)$, and $r_2 = 2$ ($k = (1, 1)$). Figure 6 shows some results obtained from `cGL2dcmds.m`, with a coarse mesh of 26×13 points, hence $n_u = 676$ spatial unknowns, yielding the numerical values $r_1 = 1.262$ and $r_2 = 2.033$. With $m = 10$ temporal discretization points, the computation of each Hopf branch then takes less than a minute. Again, the numerical HBPs converge to the exact values when decreasing the mesh width, but at the prize of longer computations for the Hopf branches. For the Floquet multipliers we obtain a similar picture as in 1D. The first branch has $\mathrm{ind}(u_H) = 1$ up to the fold, and $\mathrm{ind}(u_H) = 0$ afterwards, and on b2 $\mathrm{ind}(u_H)$ decreases from 3 to 2 at the fold and to 1 near $r = 7.2$. Panel (c) illustrates the 2D analogue of Fig. 5(c), i.e., the instability of the second Hopf branch and stability of the first.

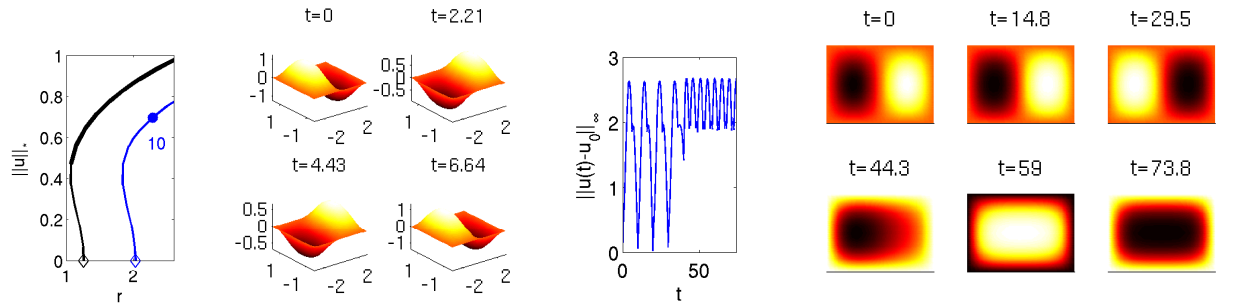


Figure 6: (a) Bifurcation diagrams of the first 2 Hopf branches for (45) in 2D. (b) Solution snapshot from b2/pt10, at $t = 0, \frac{3}{10}T, \frac{6}{10}T, \frac{9}{10}T$. (c) Time integration starting from (b) ($t = 0$), with convergence to the first Hopf branch.

To illustrate that exactly the same setup also works in 3D, in `cGL3dcmds.m` and Fig. 7 we consider (45) over $\Omega = (-\pi,\pi)\times(-\pi/2,\pi/2)\times(-\pi/4,\pi/4)$. Here we use a *very* coarse tetrahedral mesh of $n_p = 1386$ points, thus 2772 DoF in space. Analytically, the first 2 HBPs are $r_1 = 21/4$ ($k = (1/2, 1, 2)$) and $r_2 = 6$ ($k = (1, 1, 2)$, but with the coarse mesh we numerically obtain $r_0 = 5.7$ and $r_1 = 6.58$. Again, this can be greatly improved by, e.g., halving the spatial mesh width, but then the Hopf branches become very expensive. Using $m = 10$, the computation of the branches (with 10 continuation steps each) in Fig. 7 takes about 10 minutes, and a call of `floqap` to a posteriori compute the Floquet multipliers about 30 seconds. Again, on b1, $\mathrm{ind}(u_H) = 1$ up to fold and $\mathrm{ind}(u_H) = 0$ afterwards, while on b2 $\mathrm{ind}(u_H)$ decreases from 3 to 2 at the fold and to 1 at the end of the branch, and time integration from an IC from b2 yields convergence to a periodic solution from b1.

In 3D, the "slice plot" in Fig. 7(b), indicated by `p.plot.pstyle=1` should be used as a default setting, while the isolevels in (c) (via `p.plot.pstyle=2`) often require some fine tuning. Additionally we provide a "face plot" option `p.plot.pstyle=3`, which however is useless for Dirichlet BC. See `hoplot.m` for documentation of the plot options, and the ends of `cGLcmds2d.m` and `cGLcmds3d.m` for example plot calls, and on how to create movies.

(a) BD, $\|u\|_*$ and T (b) Example slice plot (c) Example isoplot

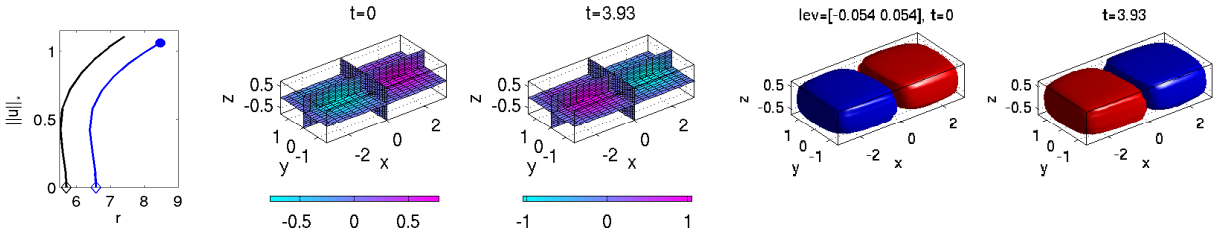


Figure 7: (a) Bifurcation diagram of first 2 Hopf branches for (45) in 3D. (b,c) Solution snapshots at $t = 0$ and $t = T/2$ for the blue dot in (a); slice-plot in (b), and isolevel plot in (c) with levels $0.525m_1 + 0.475m_2$ and $0.475m_1 + 0.525m_2$, where $m_1 = \min_{x,t} u_1(x,t)$ and $m_2 = \max_{x,t} u_1(x,t)$.

4.3 Rotating patterns on a disk: Demo `rot`

While the Hopf bifurcations presented in §4.2 have been to (standing) oscillatory patterns, cf. Remark 4.1, another interesting class is the Hopf bifurcation to rotating patterns, in particular to spiral waves. Such spirals are ubiquious in 2D reaction diffusion problems, see, e.g., [Pis06, CG09]. Over bounded domains, spiral waves are usually found numerically via time integration, with an $\mathcal{O}(1)$ amplitude, i.e., far from bifurcation. On the other hand, the bifurcation of spiral waves from a homogeneous solution is usually analyzed over all of $\mathbb{R}^2$, e.g., [Hag82, KH81, Sch98], where the spirals are relative equilibria, i.e., steady states in a comoving frame. Moreover, spiral waves often undergo secondary bifurcations such as drift, meandering and period doubling, see [Bar95, SSW99, SS07] and the references therein.

Here we study, on the unit disk, the bifurcation of spiral waves from the zero solution in a slight modification of a real two component reaction diffusion system from [GKS00], somewhat similar to the cGL, but with Robin BC. The system reads

$$\begin{aligned} \partial_t u &= d_1\Delta u + (0.5+r)u + v - (u^2+v^2)(u-\alpha v),\\ \partial_t v &= d_2\Delta v + rv - u - (u^2+v^2)(v+\alpha u), \end{aligned} \tag{54}$$

$$\partial_{\mathbf{n}} u + 10u = 0, \quad \partial_{\mathbf{n}} v + 0.01v = 0, \tag{55}$$

where $\mathbf{n}$ is the outer normal. First (§4.3.1) we follow [GKS00] and set $\alpha = 0$, $d_1 = 0.01$, $d_2 = 0.015$, and take r as the main bifurcation parameter. Then (§4.3.2) we set $\alpha = 1$, let

$$(d_1, d_2) = \delta(0.01, 0.015), \tag{56}$$

and also vary δ which corresponds to changing the domain size by $1/\sqrt{\delta}$.

Due to the BC (55), the eigenfunctions of the linearization around $(u,v) = (0,0)$ are build from Fourier Bessel functions

$$\phi(\rho,\vartheta,t) = \mathrm{Re}(\mathrm{e}^{\mathrm{i}(\omega t + m\vartheta)} J_m(q\rho)), \tag{57}$$

where (ρ,ϑ) are polar-coordinates, and with in general complex $q \in \mathbb{C}\setminus\mathbb{R}$. Then the modes are growing in ρ, which is a key idea of [GKS00] to find modes bifurcating from $(u,v) = (0,0)$ which resemble spiral waves near their core.

The implementation follows the general remarks from §4.1, and we only point out that the circle is easily set up in `circgeo.m`, that the BC (55) are encoded in `nbc.m`, and we have two script files, `rotcmds_a.m` for §4.3.1, and `rotcmds_b.m` for §4.3.2.

4.3.1 Bifurcations to rotational modes

The trvial homogeneous branch $(u, v) = (0, 0)$ is stable up to $r \approx -0.21$, and Fig. 8(a) shows the first 6 bifurcating branches h1,h2,..., h6, from left to right, while (b) shows the spatial modes for h1-h6 at bifurcation, with mode numbers $m = 0, 1, 2, 3, 2, 4$. We discretized (54), (55) with a mesh of 1272 points, hence $n_u = 2544$ DoF, and a coarse temporal discretization of 10 points, which yields about 2 minutes for the computation of each branch, with 10 points on each. Example plots of solutions on the last points on the branches are given in (e), with T near 2π for all branches.

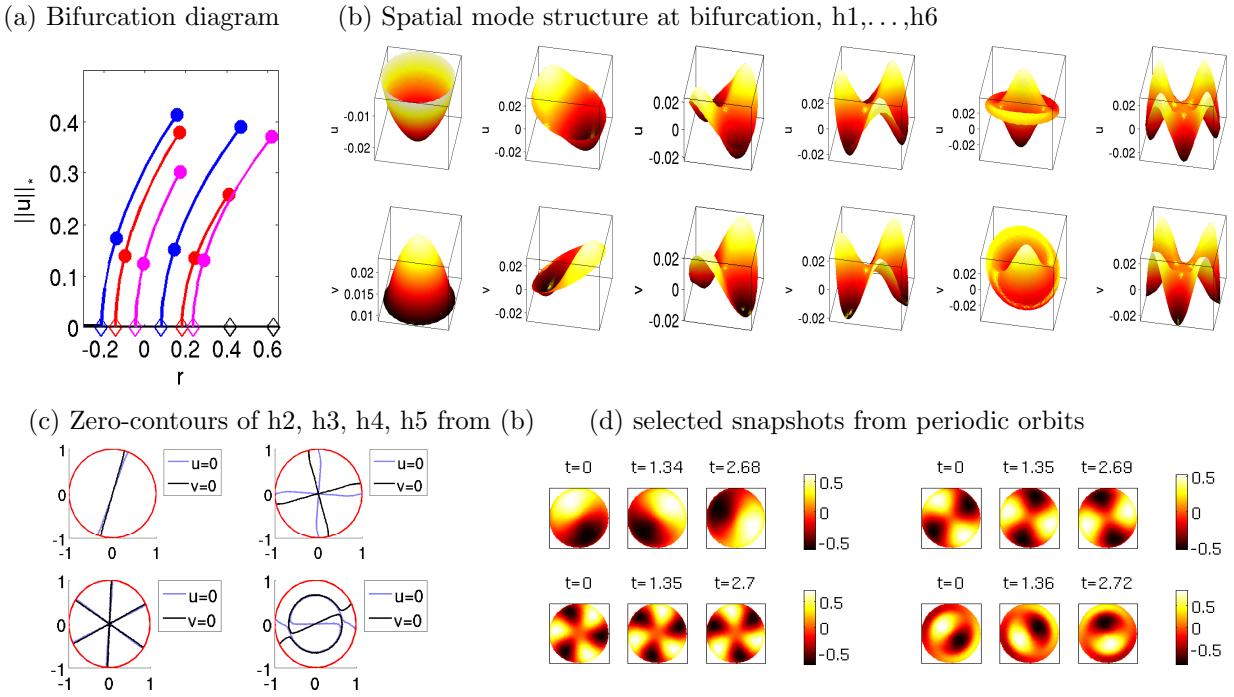


Figure 8: Basic bifurcation diagram (a) for (54), (55) with h1-h6 from left to right, 10 continuation steps for each. On each branch we mark the points 5 and 10. (b,c): information on initial mode structure on the first six bifurcating branches. (d) Example plots last points in h2, h3 (upper row) and h4, h5 (lower row). Snapshots of u at $t = 0, T_j/5$ and $2T_j/5$, with T_j the actual period.

The nontrivial solutions from Fig. 8(a),(d) are "rotations", except for the spatial $m = 0$ mode h1. To discuss this, we return to (c), which shows the nodal lines for the components u, v at bifurcation of h2 to h5 (vector Ψ in (17)). The pertinent observation is that h2 to h6 (not shown) do not have nodal lines, i.e., $u(x)v(x) \neq 0$ except at $x = 0$.[13] Thus, the branches h2 to h6 cannot consist of oscillatory patterns but must rotate. On the other hand, this rotation must involve higher order modes, and thus becomes more visible, i.e., *almost* (but never perfectly) rigid, at larger amplitude.

To assess the numerical accuracy, in Table 3 we compare the numerical values for the Hopf points and the temporal wave number ω with the values from [GKS00], who compute r_4, r_5, r_6, (and three more Hopf points) using semi analytical methods, and some numerics based on the

[13] The zero lines for h3 are close together, but not equal; for h1 we have $u(x, 0) < 0$ and $v(x, 0) > 0$ for all x.

`Matlab pdetoolbox` with fine meshes. Given our coarse mesh we find our results reasonably close, and again our values converge to the values from [GKS00] under mesh refinement.

Table 3: Comparison of HBPs with [GKS00] (starred values), and Floquet indizes at points on branches.

branch	h1	h2	h3	h4	h5	h6
r	-0.210	-0.141	-0.044	0.079	0.182	0.236
ω	0.957	0.967	0.965	0.961	0.953	0.957
r^*	NA	NA	NA	0.080	0.179	0.234
ω^*	NA	NA	NA	0.961	0.953	0.957
ind(u_H), pt5	0	2	6	12	16	20
ind(u_H), pt10	0	2	4	8	18	16

Using `floq` to compute the multipliers at a given u_H takes about 20s. Thus, for efficiency we switch it off in the demo, and use `floqap` to compute the spectra a posteriori. The last two rows of Table 3 give the Floquet indizes of points on the branches, where err_{γ_1} (cf. (42)) is around 10^{-10} for each computation. All branches except h1 are unstable, and the instability indizes increase from left to right, and also vary along the unstable branches. However, alltogether (54),(55) with $(\alpha, \delta) = (0, 1)$ does not appear to be very interesting from a dynamical and pattern forming point of view, as time–integration yields that for $r > r_0 = -0.21$ solutions to generic initial conditions converge to a periodic orbit from h1. Thus, we next choose $\alpha = 1$ to switch on a rotation also in the nonlinearity.

4.3.2 Spiral waves

For $(\alpha, \delta) = (1, 1)$ the linearization around $(u, v) = (0, 0)$ and thus also the Hopf bifurcation points $r_{h1}, \ldots, r_{h6}$ are as in §4.3.1. However, the nonlinear rotation yields a spiral wave structure on the branches s2,..., s6 bifurcating at these points, see Fig. 9(b), where we only give snapshots of $u(\cdot, 0)$, at $r = 1$ and at $r = 3$ for s2, and $r = 3$ for the remaining branches. On s2 , s3, s4, and s6 the solutions rotate almost rigidly in counterclockwise direction with the indicated period T, while on s5 we have a clockwise rotation. Thus, on s2, s3, s4 and s6 we have inwardly moving spirals, also called anti-spirals [VE01]. Moreover, again s1 is stable for all $r > r_{h1}$, but additionally s2 becomes stable for $r > r_1 \approx 1$, see Fig. 9(c), while s5 and the m–armed spirals with $m > 1$ on s3, s4, s6 are unstable, as should be expected [Hag82]; also note how the core becomes flatter with an increasing number of arms, again cf. [Hag82] and the references therein.

In Fig. 10(a) we first continue (u, v) from s2 at $r = 3$ in δ to $\delta = 0.1$, i.e., to domain radius $\sqrt{10}$ (branch s2d). As expected, with the growing domain the spirals become more pronounced (see the example plots in (c)). The solutions stay stable down to $\delta = \delta_1 \approx 0.15$, as illustrated in (b). In (c) we continue the solution from s2d/pt29 (with $\delta = 0.2$) again in r down to $r = r^*_{h2} \approx -0.22$, which is the associated Hopf bifurcation point over a circle of radius $\sqrt{5}$, see also the last plot in (c), which is very close to bifurcation. Now the 1-armed spiral like solution is stable also for rather small amplitude.

The model with $(r, \alpha, r) = (3, 1, 1)$ also appears to be quite rich dynamically. Besides solutions converging to s1 (not shown), the 1-armed spiral s2 has a significant domain of attraction (see Fig. 11(a) as an example), but there are also various at least meta-stable solutions, which consist of long-lived oscillations (with or without rotations), see Fig. 11(b) for an example. Panels (c,d) give just two examples for the dynamics for smaller δ, where in particular the instability of the 1-armed spiral for small δ is due to a bifurcation to a solution with a meandering spiral tip, again see, e.g., [Bar95] for a review of such phenomena.

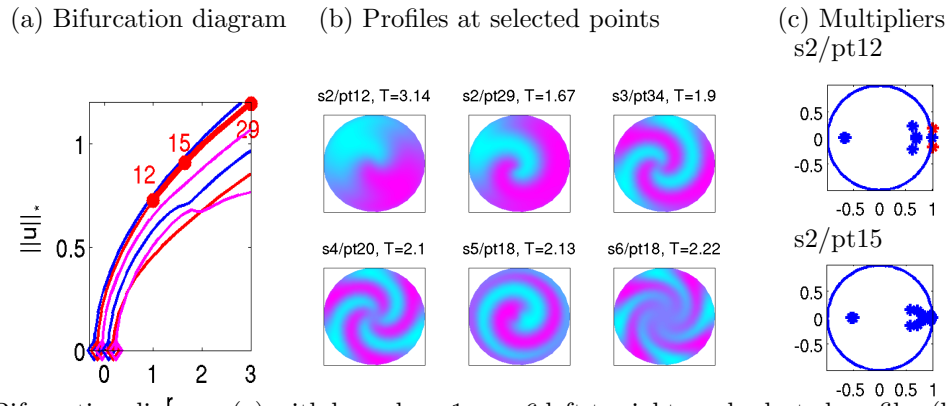


Figure 9: Bifurcation diagram (a) with branches s1,...,s6 left to right, and selected profiles (b) and Floquet spectra (c). The (non–rotational) branch s1 is stable for all r but plotted as a thin line (first blue line in (a)) for graphical reasons. The first two plots in (b) are both from s2, indicating the more pronounced spiral nature for larger r (on all branches); remaining plots all at $r = 3$. T in (b) indicating the period, which decreases in r and increases with number m of arms of the spirals.

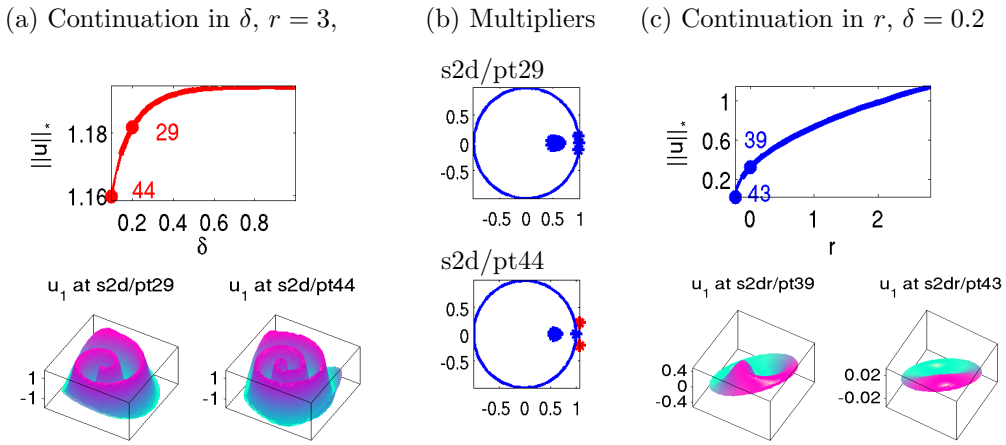


Figure 10: (a) Continuation of the one armed spiral in δ (inverse domain-size). Over a larger domain the spiral nature (of all spirals) is more visible. (b) Multipliers for points in (a). (b) Continuation of pt29 from (a) in r; over a larger domain the "one-armed spiral" is stable for lower amplitudes.

4.4 An extended Brusselator: Demo `bru`

As an example where there is an interesting interplay between stationary patterns and Hopf bifurcations, where there are typically many eigenvalues with small real parts, and where therefore detecting HBPs with `bifcheck=2` without first using `initeig` for setting a guess for a shift ω_1 is problematic, we consider an "extended Brusselator" problem from [YDZE02]. This is a three component reaction diffusion system of the form

$$\partial_t u = D_u \Delta u + f(u,v) - cu + dw, \quad \partial_t v = D_v \Delta v + g(u,v), \quad \partial_t w = D_w \Delta w + cu - dw, \tag{58}$$

where $f(u,v) = a - (1+b)u + u^2 v$, $g(u,v) = bu - u^2 v$, with kinetic parameters a, b, c, d and diffusion constants D_u, D_v, D_w. We consider (58) on rectangular domains in 1D and 2D, with homogeneous Neumann BC for all three components. The system has the trivial spatially homogeneous steady state

$$U_s = (u, v, w) := (a, b/a, ac/d),$$

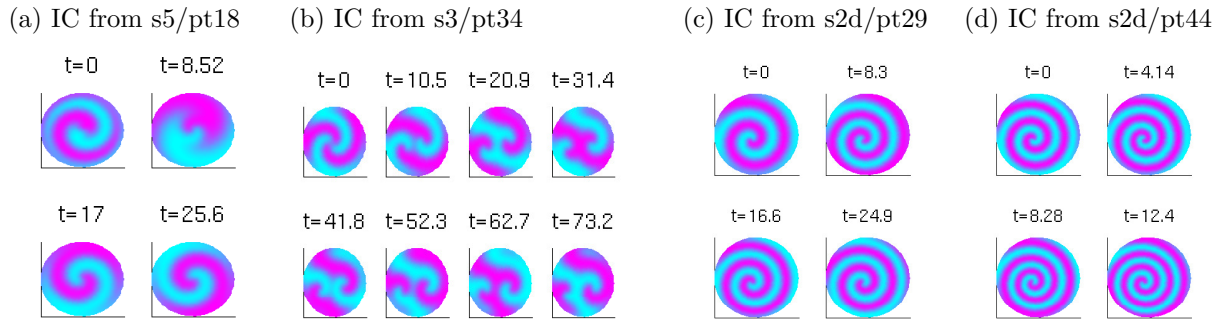


Figure 11: Time integration for (58). $(\alpha, \delta, r) = (1, 1, 3)$ in (a,b); starting from various IC the solution converges to the s2 branch (with (a) as an example), but other long time dynamics as in (b) also occur. (c) Stability of the one-armed spiral for $(\alpha, \delta, r) = (1, 0.2, 3)$. (d) Instability of the one-armed spiral for $(\alpha, \delta, r) = (1, 0.1, 3)$; the solution develops to a spiral with narrower arms and a meandering tip.

and in suitable parameter regimes it shows co-dimension 2 points between Hopf, Turing–Hopf (aka wave), and (stationary) Turing bifurcations from U_s. We follow [YDZE02] and fix the parameters

$$(c, d, D_u, D_v, D_w) = (1, 1, 0.01, 0.1, 1). \tag{59}$$

Figure 12(a) then shows a characterization of the pertinent instabilities of U_s in the a, b plane. U_s is stable in region I, and can loose stability by (a, b) crossing the Turing line, which yields the bifurcation of stationary Turing patterns, or the wave (or Turing–Hopf) line, which yields oscillatory Turing patterns. Moreover, there is the "Hopf line" which corresponds to Hopf–bifurcation with spatial wave number $k = 0$.

In the following we fix $a = 0.95$ and take b as the primary bifurcation parameter. Figure 12(b) illustrates the different instabilities from (a), i.e.: as we increase b from 2.75, we first cross the Turing–Hopf line, with first instability at critical spatial wave number $k_{\rm TH} \approx 0.7$, then the Hopf line, and finally the Turing line with critical wave number $k_{\rm T} \approx 6.4$. To investigate the bifurcating solutions (and some secondary bifurcations) with `pde2path`, we need to choose a domain $\Omega = (-l_x/2, l_x/2)$ (1D), where due to the Neumann BC l_x should be chosen as a (half integer) multiple of $\pi/k_{\rm TH}$. For simplicity we take the minimal choice $l_x = 0.5\pi/k_{\rm TH}$, which restricts the allowed wave numbers to multiples of $k_{\rm TH}$, as indicated by the black dots in Figure 12(b). Looking at the sequence of spectral plots for increasing b, we may then expect first the Turing–Hopf branch h1 with $k = k_{\rm TH}$, then a Hopf branch h2 with $k = 0$, then two Turing branches s1, s2 with $k = 6.3$ and $k = 7$, then a Turing–Hopf branch h3 with $k = 2k_{\rm TH}$, and so on, and this is what we obtain from the numerics, as illustrated in (c) and (d). Besides stationary secondary bifurcations we also get a rather large number of Hopf points on the Turing branches, and just as an example we plot the (Turing)Hopf branch s1h1 bifurcating from the first Hopf point on s1. The example plots in (d) illustrate that solutions on s1h1 look like a superposition of solutions on s1 and h1. Such solutions were already obtained in [YDZE02] from time integration, such that at least some these solutions also have some stability properties, see also [YE03] for similar phenomena. By following the models various bifurcations, this can be studied in a more systematic way.

In Fig. 13(a)-(d) we give some illustration that interesting bifurcations from the Hopf branches should occur in (58). It turns out that `h1` is always stable, and that (the spatially homogeneous branch) h2 is initially unstable with $\mathrm{ind}(u_H) = 2$, but close to pt5 on h2 we find a Neimark–Sacker bifurcation, after which solutions on h2 are stable. Similarly, solutions on h3 start with $\mathrm{ind}(u_H) = 5$, but after a Neimark–Sacker bifurcation, and a real multiplier going through 1 at $b \approx 3.35$ we find $\mathrm{ind}(u_H) = 2$, before $\mathrm{ind}(u_H)$ increases again for larger b. Also note that there are always many multipliers close to -1, but we did not find indications for period–doubling bifurcations. Finally, in Fig. 13(e)–(h) we illustrate the evolution of perturbations of s1h1/pt10. After a transient near

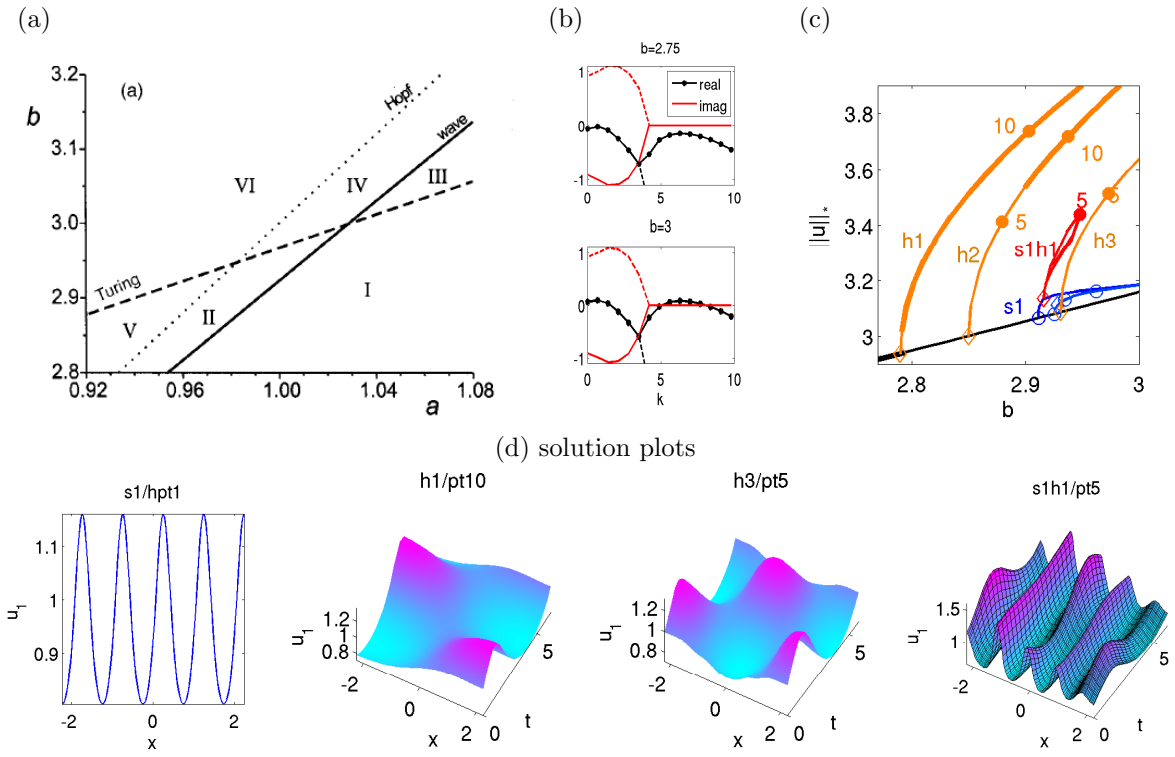


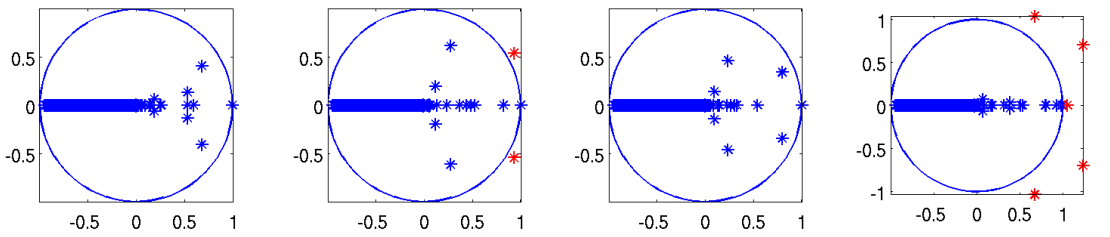
Figure 12: (a) Parameter plane with Hopf, Turing–Hopf (wave) and Turing instability lines for (58), reprinted with permission from [YDZE02], copyright 2002, AIP Publishing LLC. (b) Spectra for increasing b at $a = 0.95$. Contrary to the `pde2path` convention that due to $\partial_t u = -G(u)$ eigenvalues with *negative* real parts yield instabilities, here we directly plot the spectra of $-\partial_u G$, such that instability occurs for eigenvalues with positive real parts. The first instability (Turing–Hopf) occurs at $b \approx 2.794$, with $k_c \approx 0.7$. The admissible wave-numbers k on a domain $(-l_x, l_x)$ with $l_x = 0.5\pi/k_c$ are indicated by the dots. (c),(d): (partial) bifurcation diagram, and example plots on $\Omega = (-l_x, l_x)$.

`h3/pt5` (g) the solution converges to a solution from the primary Hopf branch `h1` (h), which however itself also shows some short wave structure at this relatively large distance from bifurcation.

In 1D we may still use `bifcheck=2` without preparation to detect (and localize) the Hopf bifurcations, i.e., by computing a number (here 20) of eigenvalues closest to zero. In 2D this is unfeasible, because even over rather small domains we obtain many wave vectors $k = (k_1, k_2)$ with modulus $|k| \in (5, 8)$, which give leading eigenvalues $\mu_1(k)$ with small $\mathrm{Re}\mu(k)$ and $\mathrm{Im}\mu(k) = 0$. This is illustrated in Fig. 14, which shows that for $\Omega = (-0.5\pi/k_{\mathrm{TH}}, 0.5\pi/k_{\mathrm{TH}})^2$ even for neig=200 (which is quite slow already) we do not even see any Hopf eigenvalues, which become "visible" at, e.g., neig=300. Thus, here we first call `initwn` and `initeig` to generate a guess for the Hopf bifurcation; subsequently `bifcheck=2` with `neig=[3 3]` runs fast and reliably.

Finally, in Fig. 15 we give an example of just four of the many branches which can be obtained for (58) in 2D, even over quite small domains. We use $\Omega = (-l_x, l_x) \times (-l_y, l_y)$, $l_x = \pi/2$, $l_y = \pi/8$, which means that admissible wave vectors are $(k_1, k_2) = (n, 4m)$, $n, m \in \mathbb{N}_0$. Consequently, no spatial structure in y direction occurs in the primary Hopf branches (cf. Fig. 12b), i.e., the first three are just analogous to those in Fig. 12 and occur at $b = 2.818$ (with $k = (1, 0)$), $b = 2.859$ (with $k = (0, 0)$, i.e., spatially homogeneous, and hence b independent of the domain) and $b = 3.202$ (with $k = (2, 0)$); see (b1) for an example plot on the first Hopf branch. The first stationary bifurcation (at $b = 2.912$) is now to a spotted branch `2ds1`, and stripe branches analogous to `s1` from Fig. 12 bifurcate at larger b. Moreover, while so far all branches were continued using `cont`, the continuation of the branch `2ds1` is problematic with `cont` as this leads to undesired branch switching (as usual

(a) h1/pt10, ind = 0 (b) h2/pt5, ind = 2 (c) h2/pt10, ind = 0 (d) h3/pt5, ind = 5



(e) "error" time series (f) initial evolution (g) transient near h3 (h) convergence to h1

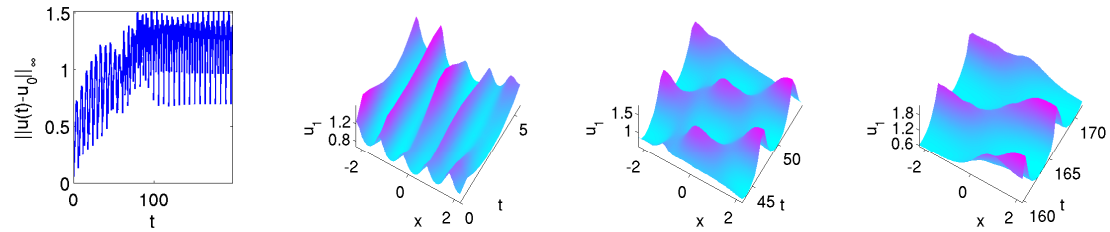


Figure 13: (a)-(d) A small sample of Floquet spectra of periodic orbits from Fig. 12 (200 largest multipliers computed via `floq`), illustrating that a Neimark–Sacker bifurcation should be expected near h2/pt5, and similar eigenvalue transitions occur on all other Hopf branches except h1. (e)-(h) Evolution of a perturbation of s1h1/pt10. After a rather long transient near h3 the solution converges to an orbit on h1.

for Turing branches in 2D, cf. [UWR14, §4]). Thus, in `br2dcmds.m` we use `pmcont` ([UWR14, §4.3]) to continue the `2ds1` branch.

Interestingly, after some stationary and Hopf bifurcations this branch becomes stable at $b = b_b \approx 2.785$, which illustrates that it is often worthwhile to follow unstable branches, as they may become stable, or stable branches may bifurcate off. In particular, here for $b < b_b$ we have a bistability of the trivial branch and `2ds1`, and hence (over somewhat larger domains) phenomena such as heteroclinics between solutions on the trivial branch and on `2ds1`, and associated snaking branches of localized spots. See [UW14] and the references therein for related results in various models.

However, here we are interested in Hopf bifurcations from `2ds1`, and Fig. 15(b2) shows an example plot from such a secondary Hopf branch. This is analogous to `s1h1` from Fig. 12, i.e., the solutions look like superpositions of the stationary pattern and solutions on the primary Hopf branch `h1`. Concerning the multipliers we find that $\mathrm{ind}(u_H) = 0$ on `2dh1`, and, e.g., $\mathrm{ind}(u_H) = 5$ at `2ds1h1/pt5`, where as in 1D (Fig. 14) there are multipliers suggesting Neimark–Sacker bifurcations. Figure 15 (c) illustrates the instability of the spotted Hopf solutions; the spots stay visible for about 4 periods, and subsequently the solution converges to a periodic orbit from the primary Hopf branch, as in Fig. 13.

Remark 4.2. As an example that besides the new functions from the `hopf` library we still have the full `pde2path` machinery available, in the script `auxcmds.m` we do some adaptive spatial mesh–refinement at the start of the (1D and 2D) Turing branches `s1` and `2ds1` and then continue.[14] The

[14] In the OOPDE setting used here, mesh-adaption so far is implemented in 1D and 2D. Moreover, it works somewhat differently and is slightly less general than in the `pdetoolbox` setting, as follows: its local error estimator is the same as in the `pdetoolbox`, and needs a function `[c,a,f]=eeG(p,u)` ("error estimate G") in the current directory, which returns the diffusion tensor `c`, the linear part `a` (usually `a=0`), and the nodal nonlinearity `f`. For `f` we can typically use the same `nodalf` which is also used in the normal form computation `hogetnf` or in setting up `sG`. See `eeG.m` in the `bru` demo directory. Note that there is no joint mesh adaption in t and x yet, and no mesh adaption in x on Hopf

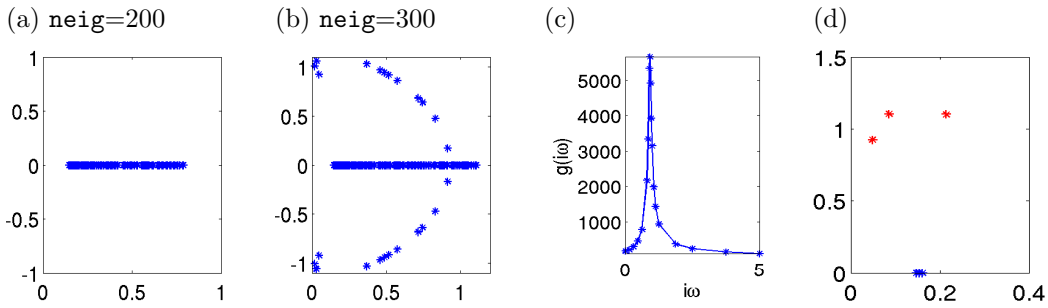


Figure 14: (a,b) `neig` eigenvalues of the linearization of (58) around U_s at $b = 2.75$, remaining parameters from (59); `bifcheck=2` with `neig`=200 but without preparation by `initeig` will not detect any Hopf points. (c) calling `initeig` yields a guess $\omega_1 = 0.9375$ for the ω value at Hopf bifurcation, and then using `bifcheck=2` with `neig=[3 3]` is reliable and fast: (d) shows the three eigenvalues closest to 0 in blue, and the three eigenvalues closest to $\mathrm{i}\omega_1$ in red.

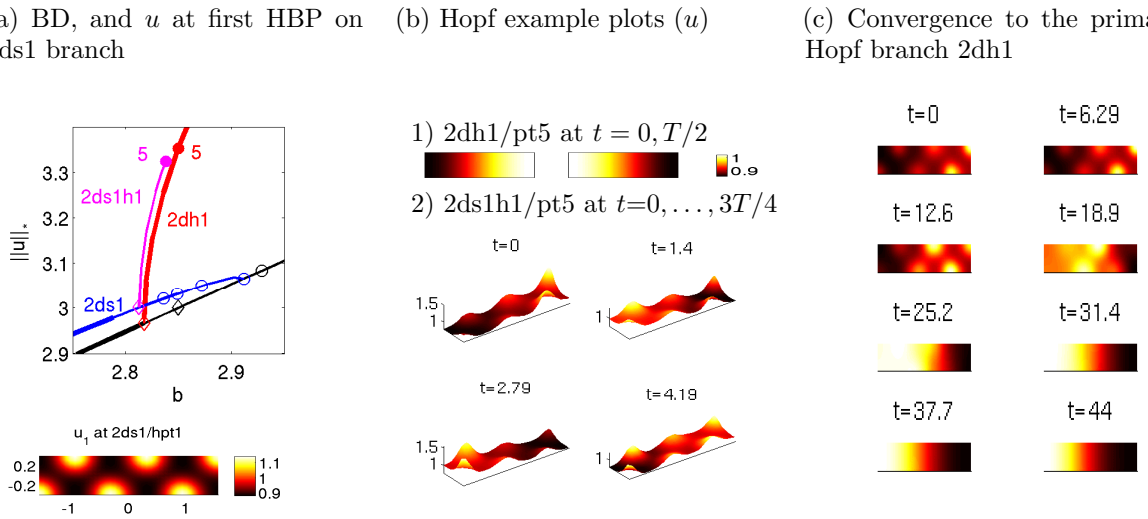


Figure 15: (a) Example bifurcations for (58) over a small 2D domain $\Omega = (-\pi/2, \pi/2) \times (-\pi/8, \pi/8)$, and example plots of u at 2nd Hopf point on the blue branch. (b) Example plots: solutions on primary Hopf branch (1), and on the secondary Hopf branch (2) (the amplitude at $t = 1.4$ and $t = 4.19$ is about 0.2). (c) Time integration with $u(\cdot, 0)$ from 2ds1h1/pt5, snapshots at $0, T, 2T, \ldots, 8T$.

further BPs and HBPs then obtained are very close to the BPs and HBPs on the coarser mesh, but the resolution of the bifurcating Hopf branches becomes considerably better, with a moderate increase of computation time, which in any case is faster than starting with a uniform spatial mesh yielding a comparable accuracy. ⌋

4.5 A canonical system from optimal control: Demo `pollution`

In [GU16, Uec16a], `pde2path` has been used to study so called canonical steady states and canonical paths for infinite time horizon distributed optimal control (OC) problems, see also [Uec15] for a short manual on OC computations with `pde2path`. As an example for such problems with Hopf bifurcations[15] we consider

$$V(v_0(\cdot)) := \max_{k(\cdot,\cdot)} J(v_0(\cdot), k(\cdot,\cdot)), \qquad J(v_0(\cdot), k(\cdot,\cdot)) := \int_0^\infty \mathrm{e}^{-\rho t} J_{ca}(v(t), k(t))\, \mathrm{d}t, \tag{60a}$$

orbits.

[15] which so far could not be found in the systems studied in [GU16, Uec16a]

where $J_{ca}(v(\cdot,t),k(\cdot,t)) = \dfrac{1}{|\Omega|}\displaystyle\int_\Omega J_c(v(x,t),k(x,t))\,\mathrm{d}x$ is the spatially averaged current value function, with

$$J_c(v,k) = pv_1 - \beta v_2 - C(k) \text{ the local current value}, \quad C(k) = k + \frac{1}{2\gamma}k^2, \tag{60b}$$

$\rho > 0$ is the discount rate (long-term investment rate), and where the state evolution is

$$\partial_t v_1 = -k + d_1\Delta v_1, \quad \partial_t v_2 = v_1 - \alpha(v_2) + d_2\Delta v_2, \tag{60c}$$

with Neumann BC $\partial_{\mathbf{n}} v = 0$ on $\partial\Omega$. Here,

- $v_1 = v_1(t,x)$ are the emissions of some firms,
- $v_2 = v_2(t,x)$ is the pollution stock,
- and the control $k = k(t,x)$ is the firms' abatement policies.

In J_c, pv_1 and βv_2 are the firms' value of emissions and costs of pollution, $C(k)$ are the costs for abatement, and $\alpha(v_2) = v_2(1-v_2)$ is the environment's recovery function. The discounted time integral in (60a) is typical for economic problems, where "profits now" weight more than mid or far future profits. Finally, the max in (60a) runs over all *admissible* controls k; this essentially means that $k \in L^\infty((0,\infty)\times\Omega,\mathbb{R})$, and we do not consider active control or state constraints.

The associated ODE OC problem (no x–dependence of v,k) was set up and analyzed in [TW96, Wir00]; in suitable parameter regimes it shows Hopf bifurcations of periodic orbits for the associated so called canonical (ODE) system. See also, e.g., [DF91, Wir96, GCF+08] for general results about the occurrence of Hopf bifurcations and optimal periodic solutions in ODE OC problems.

Setting $g_1(v,k) = (-k, v_1 - \alpha(v_2))^T$, and introducing the co–states (Lagrange multipliers)

$$\lambda : \Omega\times(0,\infty) \to \mathbb{R}^2$$

and the (local current value) Hamiltonian $\mathcal{H} = \mathcal{H}(v,\lambda,k) = J_c(v,k) + \langle\lambda, D\Delta v + g_1(v,k)\rangle$, by Pontryagin's Maximum Principle for $\tilde{\mathcal{H}} = \int_0^\infty \mathrm{e}^{-\rho t}\overline{\mathcal{H}}(t)\,\mathrm{d}t$ with $\overline{\mathcal{H}}(t) = \int_\Omega \mathcal{H}(v(x,t),\lambda(x,t),k(x,t))\,\mathrm{d}x$, an optimal solution (v,λ) has to solve the canonical system (first order necessary optimality conditions)

$$\partial_t v = \partial_\lambda \mathcal{H} = D\Delta v + g_1(v,k), \quad v|_{t=0} = v_0, \tag{61a}$$

$$\partial_t \lambda = \rho\lambda - \partial_v \mathcal{H} = \rho\lambda + g_2(v,k) - D\Delta\lambda, \tag{61b}$$

where $\partial_{\mathbf{n}}\lambda = 0$ on $\partial\Omega$. The control k fulfills $k = \mathrm{argmax}_{\tilde{k}}\,\mathcal{H}(v,\lambda,\tilde{k})$, and under suitable concavity assumptions on J_c and in the absence of control constraints is obtained from solving $\partial_k\mathcal{H}(v,\lambda,k)=0$, thus here

$$k = k(\lambda_1) = -(1+\lambda_1)/\gamma. \tag{62}$$

Note that (61) is ill–posed as an initial value problem due to the backward diffusion in the co–states λ. Thus it seems unlikely that periodic orbits for (61) can be obtained via shooting methods. For convenience we set $u(t,\cdot) := (v(t,\cdot),\lambda(t,\cdot)) : \Omega \to \mathbb{R}^4$, and write (61) as

$$\partial_t u = -G(u) := \mathcal{D}\Delta u + f(u), \tag{63}$$

where $\mathcal{D} = \mathrm{diag}(d_1,d_2,-d_1,-d_2)$, $f(u) = \Big(-k, v_1-\alpha(v_2), \rho\lambda_1 - p - \lambda_2, (\rho+\alpha'(v_2))\lambda_2 + \beta\Big)^T$. Besides the boundary condition $\partial_{\mathbf{n}} u = 0$ on $\partial\Omega$ and the initial condition $v|_{t=0} = v_0$ (only) for the states, we have the intertemporal transversality condition

$$\lim_{t\to\infty} \mathrm{e}^{-\rho t}\int_\Omega \langle v,\lambda\rangle\,\mathrm{d}x = 0. \tag{64}$$

A solution u of the canonical system (63) is called a *canonical path*, and a steady state of (63) (which automatically fulfills (64)) is called a *canonical steady state (CSS)*, and a first step for OC problems of type (60) is to study the CSS and canonical paths connecting to some CSS u^*. To find such connecting orbits to u^* we may choose a cut–off time T_1 and require that $u(\cdot, T_1)$ is in the stable manifold $W_s(u^*)$ of u^*, which we approximate by the associated stable eigenspace $E_s(u^*)$. If we consider (61) after spatial dicretization, then, since we have $n_u/2$ initial conditions, this requires that $\dim E_s(u^*) = n_u/2$. Defining the defect $d(u^*)$ of a CSS as

$$d(u^*) = \frac{n_u}{2} - \dim E_s(u^*), \tag{65}$$

it turns out (see [GU16, Appendix A]) that always $d(u^*) \geq 0$, and we call a u^* with $d(u^*) = 0$ a saddle–point CSS. See [GCF$^+$08, GU16] for more formal definitions, and further comments on the notions of optimal systems, the significance of the transversality condition (64), and the (mesh-independent) defect $d(u^*)$. For saddle point CSS u^* we can then compute canonical paths to u^*, and this has for instance been carried out for a vegetation problem in [Uec16a], with some surprising results, including the bifurcation of patterned optimal steady states.

A natural next step is to search for time–periodic solutions u_H of canonical systems, which obviously also fulfill (64). The natural generalization of (65) is

$$d(u_H) = \mathrm{ind}(u_H) - \frac{n_u}{2}. \tag{66}$$

In the (low–dimensional) ODE case, there then exist methods to compute connecting orbits to (saddle point) periodic orbits u_H with $d(u_H) = 0$, see [BPS01, GCF$^+$08], which require comprehensive information on the Floquet multipliers and the associated eigenspace of u_H. Our aim is to extend these methods to periodic orbits of PDE OC systems.

However, a detailed numerical analysis of (60) and similar PDE optimal control problems with Hopf bifurcations, and economic interpretation of the results, will appear elsewhere. Here we only illustrate that

- Hopf orbits can appear as candidates for optimal solutions in PDE OC problems of the form (60),
- and that the computation of multipliers via the periodic Schur decomposition can yield reasonable results, even when computation directly based on the product (41) completely fails.

For all parameter values, (63) has the spatially homogeneous CSS

$$u^* = (z_*(1-z_*), z_*, -1, -(p+\rho)), \quad \text{where} \quad z_* = \frac{1}{2}\left(1 + \rho - \frac{\beta}{p+\rho}\right).$$

We use similar parameter ranges as in [Wir00], namely

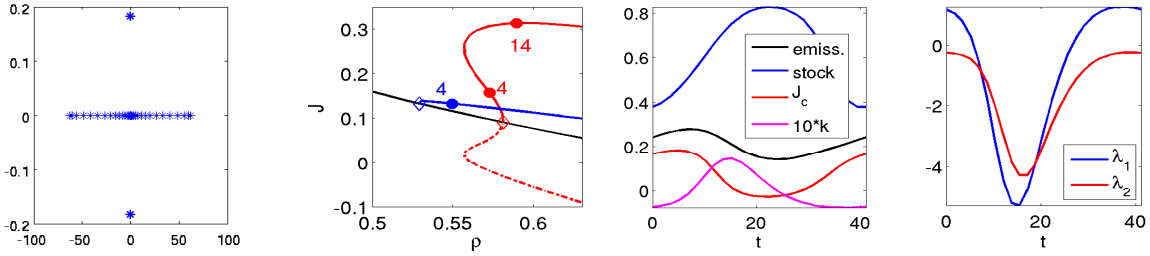
$$(p, \beta, \gamma) = (1, 0.2, 300), \text{ and } \rho \in [0.5, 0.65] \text{ as a continuation parameter,} \tag{67}$$

consider (63) over $\Omega = (-\pi/2, \pi/2)$, and set the diffusion constants to $d_1 = 0.001, d_2 = 0.2$.[16] In Figure 16 we give some basic results for (63) with a coarse spatial discretization of Ω by only $n_p = 17$ points (and thus $n_u = 68$). (a) shows the full spectrum of the linearization of (63) around u^* at $\rho = 0.5$, illustrating the ill-posedness of (63) as an initial value problem. (b) shows a basic bifurcation diagram. At $\rho = \rho_1 \approx 0.53$ there bifurcates a Hopf branch `h1` with spatial wave number

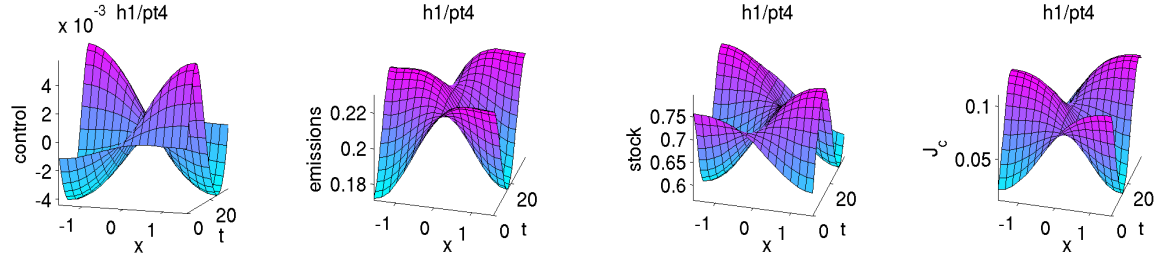
[16] The motivation for this choice is to have the first (for increasing ρ) Hopf bifurcation to a spatially patterned branch, and the second to a spatially uniform Hopf branch, because the former is more interesting. We use that the HBPs for the model (63) can be analyzed by a simple modification of [Wir00, Appendix A]. We find that for branches with spatial wavenumber $l \in \mathbb{N}$ the necessary condition for Hopf bifurcation, $K > 0$ from [Wir00, (A.5)], becomes $K = -(\alpha' + d_2 l^2)(\rho + \alpha' + d_2 l^2) - d_1 l^2(\rho + d_1 l^2) > 0$. Since $\alpha' = \alpha'(z_*) < 0$, a convenient way to first fulfill $K > 0$ for $l = 1$ is to choose $0 < d_1 \ll d_2 < 1$, such that for $l = 0, 1$ the factor $\rho + \alpha' + d_2 l^2$ is the crucial one.

$l = 1$, and at $\rho = \rho_2 \approx 0.58$ a spatially homogeneous ($l = 0$) Hopf branch `h2` bifurcates subcritically with a fold at $\rho = \rho_f \approx 0.55$. (c) shows the pertinent time series on h2/pt14. As should be expected, J_c is large when the pollution stock is low and emissions are high, and the pollution stock follows the emissions with some delay. The abatement investment k can be negative, and indeed must be for bifurcating periodic orbits as $k = 0$ for any CSS u^*. This, given the quadratic term in the costs $C(k) = k + \frac{1}{2\gamma}k^2$, might seem a bit odd, but as already said, here we refrain from a detailed model discussion.

(a) spectrum of $\partial_u G(u^*)$, $\rho = 0.5$ (b) bif. diagram (c) time series on h2/pt14 (spat. homogen. branch)



(d) example plots at h1/pt4



(f) the $\frac{n_u}{2}$ smallest γ_j at h2/pt4 (g) $|\gamma_j|$ for the $\frac{n_u}{2}$ largest γ_j at h2/pt4 (h) the $\frac{n_u}{2}$ smallest γ_j at h2/pt4 and at h2/pt14

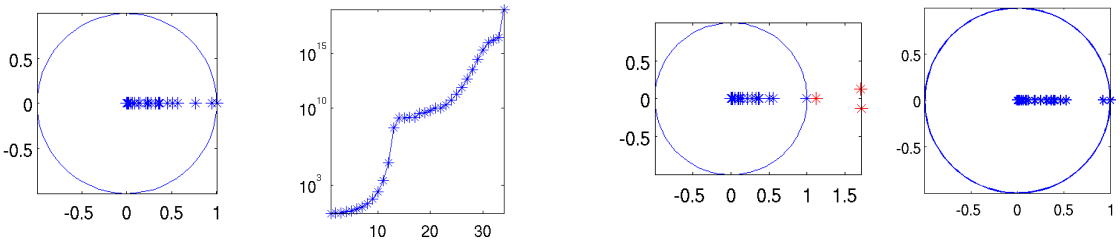


Figure 16: (a) full spectrum of the linearization of (63) around u^* at $\rho = 0.5$ on a coarse mesh with $n_p = 17$. (b) Bifurcation diagram, value J over ρ. Black: u^*; blue: `h1`, red: `h2`, $J(u^H; 0)$ (full line) and $J(u^H; T/2)$ (dashed line). (c) Time series of a spatially homogeneous solution, including current value J_c, control k, and co–states $\lambda_{1,2}$. (d,f,g) Example plots and and multipliers of u_H at `h1/pt4`, which shows that $\mathrm{ind}(u_H) = 0$. (h) multipliers at `h2/pt4`, which shows that $\mathrm{ind}(u_H) = 3$ at this point, while solutions on h2 become saddles after the fold.

Since ultimately we are interested in the values J of solutions of (63), in (b) we plot J over ρ. For the CSS u^* this is simply $J(u^*) = \frac{1}{\rho}J_{c,a}(u^*)$, but for the periodic orbits we have to take into account the phase, which is free for (63). If u_H is obtained for (63), then, for $\phi \in [0, T)$, we consider

$$J(u_H; \phi) := \int_0^\infty \mathrm{e}^{-\rho t} J_{c,a}(u_H(t + \phi))\,\mathrm{d}t = \frac{1}{1 - \mathrm{e}^{-\rho T}} \int_0^T \mathrm{e}^{-\rho t} J_{c,a}(u_H(t + \phi))\,\mathrm{d}t,$$

which in general may depend on the phase, and for `h2` in (c) we plot $J(u_H; \phi)$ for $\phi = 0$ (full red line) and $\phi = T/2$ (dashed red line). For the spatially periodic branch `h1`, $J_{c,a}(t)$ averages out in

x and hence $J(u_H;\phi)$ only weakly depends on ϕ. Thus, we first conclude that $\rho \in (\rho_1, \rho_f)$ the spatially patterned periodic orbits from `h1` give the highest J, while for $\rho \geq \rho_f$ this is obtained from `h2` with the correct phase. The example plots (c) at `h1/pt4` illustrate how the spatio-temporal dependence of k should be chosen, and the resulting behaviours of v and J_c.

It remains to compute the defects $d(u^*)$ of the CSS and $d(u_H)$ of periodic orbits on the bifurcating branches. For $d(u^*)$ we find that it starts with 0 at $\rho = 0.5$, and, as expected, increases by 2 at each Hopf point. On the Hopf branches we always have $n_+ \geq n_u/2$ unstable multipliers (computed with `floqps`, which yields $\mathrm{err}_{\gamma_1} < 10^{-8}$ for all computations, and hence we trust it), and the leading multipliers are very large, i.e., on the order of 10^{20}, even for the coarse space discretization. Thus, we may expect `floq` to fail, and indeed it does so completely. For instance, calling `floq` to compute all multipliers typically returns 10 and larger for the modulus of the *smallest* multiplier (which from `floqps` is on the order of 10^{-25}). For the plots we overload `floqpsap.m` in the directory `pollution` to give the logarithmic plots of the large multipliers.

On h1 we find $d(u_H) = 0$ up to pt4, see (e) for the $n_u/2$ smallest multipliers, and (f) for $|\gamma_j|$ for the large ones, which are mostly real, and $d(u_H) \geq 1$ for larger ρ. On `h2` we start with $d(u_H) = 3$, see (h), but $d(u_H) = 0$ after the fold until $\rho = \rho_1 \approx 0.6$, after which $d(u_H)$ increases again by multipliers going though 1. Since on `h1` we have that $J(u_H)$ is larger than $J(u^*)$, and since u_H is a saddle point up to `pt4`, we expect that these u_H are at least locally optimal, and similarly we expect u_H from `h2` after the fold until ρ_1 to be locally, and probably globally, optimal. However, as already said, for definite answers and, e.g., to characterize the domains of attractions, we need to compute canonical paths connecting to these periodic orbits, and this will be studied elsewhere.

5 Summary and outlook

With the `hopf` library we provide basic functionality for Hopf bifurcations and periodic orbit continuation for the class (3) of PDEs over 1D, 2D and 3D domains. The user interfaces reuse the standard `pde2path` setup, and no new user functions are necessary. For the detection of Hopf points we detect eigenvalues crossing the imaginary axis near guesses $\mathrm{i}\omega_j$, where the ω_j can either be set by the user (if such a priori information is available), or can be estimated via `initeig`, which is based on computing the function g from (12). An initial guess for a bifurcating periodic orbit is then obtained from the normal form (13), and the continuation of the periodic orbits is based on modifications of routines from TOM [MT04]. Floquet multipliers of periodic orbits can also be computed, and thus we can detect possible bifurcations from periodic orbits. We do not (yet) provide functionality for localizing these bifurcations, and consequently, no routines for branch switching at such bifurcation points.

Our `OOPDE` setup, with the goal of taking `pde2path` to 1D and 3D, is rather basic and does not yet provide the same flexibility concerning boundary conditions and fully nonlinear equations as the standard `pde2path` setup based on the `Matlab pdetoolbox`. Information for using more of `OOPDE` can be found at [Prü16]. Also, our usage of `ilupack` [Bol11] for the preconditioned iterative solution of linear systems is rather basic.

Floquet multipliers of periodic orbits can be computed using `floq` or `floqps`. The former is suitable for dissipative systems, and computes the `p.hopf.nfloq` largest multipliers of the explicit monodromy matrix $\mathcal{M}$ (41). This definitely fails for problems of the type considered in §4.5, and in general we recommend to monitor $\mathrm{err}_{\gamma_1} = |\gamma_1 - 1|$ to detect further possible inaccuracies. `floqps` is based on a periodic Schur decomposition of the factor matrices of $\mathcal{M}$. Therefore, for just computing the multipliers of $\mathcal{M}$ it is generally slower than `floq`, but it has distinct advantages: It can be used to efficiently compute eigenspaces at all time–slices and hence bifurcation information in case of critical multipliers, and, presently most importantly for us, it accurately (measured by err_{γ_1}) computes the multipliers also for ill posed evolution problems.

We explained the usage of the software using four example problems, where we believe that the second, third and fourth are close to interesting research problems. For instance, the demo `rot`, on the linear level based on [GKS00], seems to be the first work where the bifurcation of spiral waves out of zero has been studied numerically over a bounded domain, in a reaction diffusion system without very special boundary conditions. Further interesting problems will be, e.g., the bifurcation *from* Hopf branches in the demos `rot`) and `bru`. Thus, as one next step we plan to implemenent the necessary localization and branch switching routines, for which the demo `cGL` will again provide a good test case. The demo `pollution` gives a (very basic) illustration of the widely unexplored field of Hopf bifurcations and time periodic orbits in optimal control PDE problems. For this, as a next step we will implement routines to compute canonical paths connecting to periodic orbits.

Finally, an interesting field are Hopf bifurcations from travelling waves, or more generally in systems with continuous symmetries, see Remark 2.1. The treatment of these is also planned as a next step.

A `hopf` library overview

Our Hopf setup does not need any user setup additional to the functions such as `p.fuha.sG`, `p.fuha.sGjac` (or `p.fuha.G, p.fuha.Gjac`) already needed to describe stationary problems. The only changes of the core `p2p` library concern some queries whether we consider a Hopf problem, in which case basic routines such as `cont` call a Hopf version, i.e., `hocont`. HBPs are flagged by `p.sol.ptype=3`, while points on a Hopf branch have `p.sol.ptype=4`. The natural parametrization (§2.3.3) for periodic orbit continuation is flagged by `p.sw.para=3`, while `p.sw.para=4` flags arclength (§2.3.2).

Table 4: Entries in `p.hopf` (first block), and additions/modifications to `p.nc`.

field	purpose
y	for `p.sw.para=4`: unknowns in the form (29) ($n_u \times m$ matrix);
	for `p.sw.para=3`: y augmented by $\tilde{y}$ and T, λ ($(2n_u{+}2) \times m$ matrix).
y0d	for `p.sw.para=4`: $M\dot{u}_0$ for the phase condition (19) ($n_u \times m$ matrix);
	for `p.sw.para=3`: $M\dot{u}_0(0)$ for the phase condition (36) ($2n_u{+}2$ vector).
tau	tangent, see (24)
ysec	secant between two solutions (y_0, T_0, λ_0), (y_1, T_1, λ_1) for `p.sw.para=3`; $(2n_u{+}2) \times m$ matrix
t, T, lam	time discretization vector, current period and param.value
xi,wT	weights for the arclength, see (21)
x0i	index for plotting $t \mapsto u(\vec{x}$(`x0i`)$)$;
plot	aux. vars to control hoplot during hocont; see the description of `hoplot`; default plot=[]
wn	struct containing the winding number related settings for `initeig`
tom	struct containing TOM settings, including the mass matrix M
jac	switch to control assembly of $\partial_u \mathcal{G}$. jac=0: numerically (only recommended for testing); jac=1: via `hosjac`. Note that for `p.sw.jac=0` the local matrices $\partial_u G(u(t_j))$ are obtained via `numjac`, but this is still much faster than using `p.hopf.jac=0`.
flcheck	0 to switch off multiplier-computation during continuation, 1 to use `floq`, 2 to use `floqps`
nfloq	# of multipl. (of largest modulus) to compute (if flcheck=1)
fltol	tolerance for multiplier γ_1 (give warning if $\lvert\gamma_1 - 1\rvert >$`p.hopf.fltol`)
muv1,muv2	vectors of stable and unstable multipliers, respectively
p.nc.mu1	for bifcheck=2: start bisection if ineg changed, and $\lvert\mathrm{Re}(\mu)\rvert <$mu1, where μ is the pertinent
p.nc.mu2	eigenvalue; check that $\lvert\mathrm{Re}(\mu)\rvert <$mu2 at end of bisection, see Remark 2.2.
p.nc.eigref	now a vector (in general), as is p.nc.neig

The Hopf related variables are collected in the field `p.hopf`, see Table 4, while Table 5 summarizes the main functions from the `hopf` library. See also [DRUW14, App.A] for the general

organization of `p.fuha` (function handles), `p.nc` (numerical controls), `p.sw` (switches), `p.plot` (plot settings), `p.file` (file handling), `p.sol` (solution at runtime), etc in `p`. As Table 5 is only intended as an overview, we refer to the m-files, the `pde2path` help system, and the demo directories for the input and output arguments and further details on the used functions.

Table 5: Overview of main functions related to Hopf bifurcations and periodic orbits

name	purpose
hoswibra	branch switching at HBP, see (17), and comments below
hoplot	plot the data contained in hopf.y. Space-time plot in 1D; in 2D and 3D: snapshots at (roughly) $t = 0$, $t = T/4$, $t = T/2$ and $t = 3T/4$; see also hoplotf;
initeig	find guess for ω_1; see also initwn
floq	compute `p.hopf.nfloq` multipliers during continuation (`p.hopf.flcheck`=1)
floqps	use periodic Schur to compute (all) multipliers during continuation (flcheck=2)
floqap, floqpsap	a posteriori versions of floq and floqps, respectively
hobra	standard–setting for p.fuha.outfu (data on branch), template for adaption to a given problem
hostanufu	standard setting for screen printout, see also hostanheadfu
plotfloq	plot previously computed multipliers
hotintxs	time integrate (4) from the data contained in p.hopf and u0, with output of $\|u(t) - u_0\|_\infty$, and saving $u(t)$ to disk at specified values
tintplot*d	plot output of hotintxs; $x-t$–plots for *=1, else snapshots at specified times
initwn	init vectors for computation of g from (12)
hogetnf	compute initial guess dlam, al for the coefficients of bifurcating Hopf branch from the normal form (13)
hocont	main continuation routine; called by cont if p.sol.ptype>2
hostanparam	set standard parameters
hostanopt, hoMini	standard options for, and initialization of hopf.tom
hoinistep	perform 2 initial steps and compute secant, used if `p.sw.para=3`
honloopext,honloop	the arclength Newton loop (27), and the Newton loop with fixed λ
tomsol	use TOM to solve (37)
tomassemG	use TOM to assemble $\mathcal{G}$; see also `tomassem, tomassempbc`
gethoA	put together the extended Jacobian $\mathcal{A}$ from (27)
hopc	the phase condition ϕ from (19) and $\partial_u \phi$ from (23)
arc2tom, tom2arc	convert arclength data to tomsol data, e.g., to call tomsol for mesh adaptation. tom2arc to go back.
ulamcheckho	check for and compute solutions at user specified values in p.usrlam
hosrhs,hosrhsjac	interfaces to p.fuha.G and p.fuha.Gjac at fixed t, internal functions called by tomassempbc, together with hodummybc
horhs,hojac	similar to hosrhs, horhsjac, for (37), see also `hobc` and `hobcjac`

Besides `cont`, the functions `initeig`, `hoswibra`, `hoplot`, `floqap`, `floqpsap`, `floqplot`, `hotintxs`, and `tintplot*d` are most likely to be called directly by the user, and `hobra` and `hostanufu` are likely to be adapted by the user. The functions involving TOM, and those with rhs or jac in their name are basically described in §2. As usual, all functions in Table 5 can be most easily overloaded by copying them to the given problem directory and modifying them there.

In `p=hoswibra(dir,fname,ds,para,varargin)`, the auxiliary argument `aux=varargin{2}` (`varargin{1}` is the new directory) can for instance have the following fields:

- `aux.tl`=20: (initial, i.e., might be refined for `p.sw.para=3`) number of (equally spaced) meshpoints in $t \in [0, 1]$. For larger scale problems, i.e., with more than 2000 DoF in space, we recommend to at least initially reduce `tl` to 10, see, e.g., `cGL/cGL2dcmds.m`, `cGL/cGL3dcmds.m`, `rot/rotcmds.m` and `bru/bru2dcmds.m`.
- `aux.al, aux.dlam` (no preset): these can be used to pass a guess for α and s=dlam from (13) and thus circumvent `hogetnf`; useful for quasilinear problems, see footnote 10.

`hoplot(p,wnr,cnr,varargin)`, where `wnr` and `cnr` are the window number and component number, is the basic plotting routine for periodic orbits, contained in `p.hopf.y`. The auxiliary argument `aux=varargin` can contain a number of fields used to control its behavior. Examples are (with default values as indicated)

- `aux.lay`=[2 2]: sets the subplot-layout for the snapshots (in 2D and 3D)
- `aux.pind`=[]; set the indices, i.e., the times T*p.hopf.t(aux.pind), to be used for plotting; if `pind`=[], then the four indices 1, tl/4, tl/2, 3*tl/4 are used.
- `aux.xtics`=[]; set xtics, similar for `ytics` and `ztics`; see also `aux.cb`. (colorbar on/off)

This provides some flexibility for plotting snapshots of periodic orbits in 2D and 3D. However, most likely the user will adapt `hoplot` to the problem; see, e.g., the examples `hoplotrot` and `hoplotbru` in the demo directories `rot` and `bru`.

B Some implementation details

We exemplarily comment on how to set up and run (45), i.e.,

$$\partial_t \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} \Delta + r & -\nu \\ \nu & \Delta + r \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} - (u_1^2 + u_2^2) \begin{pmatrix} c_3 u_1 - \mu u_2 \\ \mu u_1 + c_3 u_2 \end{pmatrix} - c_5 (u_1^2 + u_2^2)^2 \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{68}$$

over 1D, 2D and 3D rectangles with homogeneous Neumann or Dirichlet BC in `pde2path`. In 1D, the initialization is (l3-6 of `cGL1dcmds.m`)

```
ndim=1; dir='hom1d'; p=[]; lx=pi; nx=30;
par=[-0.05; 1; 0.1; -1; 1]; % r   nu   mu    c3   c5
p=cGLinit(p,lx,nx,par,ndim);
```

where we use the domain size `lx`, the number `nx` of points in the spatial discretization, the base parameter set `par` from (46), and the space dimensions `ndim` as parameters. However, `ndim` only plays a role in l5–l14 of the init routine:

```
function p=cGLinit(p,lx,nx,par,ndim)
p=stanparam(p); screenlayout(p); % set standard parameters and screenlayout
p.nc.neq=2; p.nc.ilam=1; p.fuha.outfu=@hobra; % number of eq, cont-param, output
p.fuha.Gjac=@Gjac; p.fuha.sG=@sG;p.fuha.sGjac=@sGjac; p.sw.jac=1; % rhs and Jac
switch ndim % set domain and BC depending on ndim
    case 1;pde=stanpdeo1D(lx,2*lx/nx); p.vol=2*lx; p.hopf.x0i=1;
        bc=pde.grid.neumannBC('0');  % OOPDE BC syntax
    case 2; pde=stanpdeo2D(lx,lx/2,2*lx/nx); p.vol=2*lx^2; p.hopf.x0i=30;
        bc=pde.grid.dirichletBC('1','0');
    case 3; pde=stanpdeo3D(lx,lx/2,lx/4,2*lx/nx); p.vol=0.5*lx^3; p.hopf.x0i=200;
        bc=pde.grid.dirichletBC('1','0');
        p.plot.ng=20;  % settings for 3D plots (somewhat problem dependent)
        p.plot.levc={'blue','red'}; p.plot.lev=[-0.1 0.1]; p.plot.alpha=0.5;
end
pde.grid.makeBoundaryMatrix(bc); p.nc.sf=1e3; % OOPDE set up of BC for assemb
p.pdeo=pde; p.sw.sfem=-1; p.np=pde.grid.nPoints; p.nu=p.np*p.nc.neq;
p.sol.xi=1/p.nu;  p=setfemops(p); % setfemops calls oosetfemops in problem dir
u=0*ones(p.np,1); v=u; p.u=[u;v;par]; % initial guess (here trivial) and pars
p.usrlam=[-0.25 -0.2 -0.1 0 0.5  1 2 3];  % user-vals for output
p.file.smod=10; p.plot.cm=hot; p.plot.bpcmp=9; % saving, colormap, and branch-plot
```

In 1D, the choice of `x0i` (mesh point number for plot of time-series $t \mapsto u(x(\texttt{x0i}), t)$ is somewhat arbitrary due to the Neumann BC, but in 2D and 3D x(`x0i`) should not be on the boundary due to the Dirichlet BC. To set up the BC we use the `OOPDE` routines `pde.grid.neumannBC` and `pde.grid.dirichletBC` in l7, l9 and l11, respectively, such that the call of `pde.grid.makeBoundary Matrix` in l15 can take care of the rest, where the Dirichlet BC are implemented via a stiff spring approximation; see [Prü16] for details.

The crucial files to implement (68) and the BC are thus `oosetfemops.m, sG.m` and `sGjac.m`, which read

```
function p=oosetfemops(p) % in problem-dir since highly problem dependent
 [K,M,~]=p.pdeo.fem.assema(p.pdeo.grid,1,1,1); % stiffness/mass matrix (1 comp)
[Q,G,H,R]=p.pdeo.fem.assemb(p.pdeo.grid); % matrices for BC (empty for Neumann BC)
% augment K and M to 2-compos, and add BC matrices to K
sf=p.nc.sf; N=sparse(p.np, p.np);  % stiffness factor, and dummy N
p.mat.K=[[K+sf*(H'*H) N];[N K+sf*(H'*H)]]; p.mat.M=[[M N];[N M]];
end
```

```
function r=sG(p,u) % compute pde-part of residual
f=nodalf(p,u); r=p.mat.K*u(1:p.nu)-p.mat.M*f;
```

```
function Gu=sGjac(p,u) % compute pde-part of Jacobian
[f1u,f1v,f2u,f2v]=njac(p,u); n=p.np;
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;
end
```

In `oosetfemops.m` we first assemble the one–component Neumann Laplacian stiffness matrix K and the one–component mass matrix M, and then the one–component BC matrices Q, H, which implement the boundary conditions previously set in `cGLinit.m`, l7, 9, 13, respectively. Then in l7 we put together the true system matrices "by hand". So far we find this most convenient, but refer to [Prü16] for more sophisticated ways to set up systems and more complicated BC directly. Also note that here for the Dirichlet BC in 2D and 3D we use the typical stiff spring approximation with stiffness factor `sf`, and the matrix Q and the vectors G, R (again see [Prü16]) are not used.

The function `sG.m` is completely generic: it uses the *system* stiffness and mass matrices K and M, and to compute the nonlinear terms calls `nodalf.m`, which is also called in `hogetnf.m` for the computation of the normal form coefficients, and which reads

```
function f=nodalf(p,u) % the 'nonlinearity' (i.e., everything except diffusion)
% for cGL, directly defined via the nodal values of u1=Re(u) and u2=Im(u)
u1=u(1:p.np); u2=u(p.np+1:2*p.np); us=u1.^2+u2.^2; par=u(p.nu+1:end);
r=par(1); nu=par(2); mu=par(3); c3=par(4); c5=par(5);
f1=r*u1-nu*u2-us.*(c3*u1-mu*u2)-c5*us.^2.*u1;
f2=r*u2+nu*u1-us.*(c3*u2+mu*u1)-c5*us.^2.*u2;
f=[f1;f2];
end
```

After the extraction in l3 of the two components u_1 and u_2 and of the parameters from the internal solution vector `u`, the rhs from (68) is simply typed in. Similarly, `njac` from `sGjac.m`, l2, is the derivative of f, and thus easily typed in as well (see `njac.m`). All these files, i.e., `oosetfemops.m, sG.m, sGjac.m` and the nonlinearity `nodalf.m`, are completely dimension independent.

After the initialization `p=cGLinit(..)`, in `cGL1dcmds` we (re)set the output directory, the continuation step–length, and, most importantly, `bifcheck=2`, and then simply call `cont`:

```
p=setfn(p,dir); p.sol.ds=0.1; p.sw.bifcheck=2; p.nc.neig=10; p=cont(p,20);
```

This finds a number of (Hopf) bifurcation points from the trivial branch $u \equiv 0$. Then we call `hoswibra` (here for the first HP) and `cont` again:

```
para=4; ds=0.1; p=hoswibra('hom1d','hpt1',ds,para,'1db1'); % p.sw.verb=2;
p.fuha.blss=@mbel; p.nc.mbw=2; % p.hopf.ilss=1; % set to 0 if no ilupack
p=cont(p,10);
```

Uncommenting `p.sw.verb=2` (verbosity switch) in l1 gives more output, and `p.hopf.ilss=1` in l2 turns on the `ilupack` preconditioned iterative solver in `mbel`, see Remark 2.3.

Except for an exemplarily call of `hoswibra` with `para=3`, the remainder of the script file `cGL1dcmds.m` consists of plotting and stability check commands, and we refer to the script for comments. The 2D and 3D scripts `cGL2dcmds.m` and `cGL3dcmds.m` follow the same rules (with calls of `floqap` at the end of `cGL3dcmds.m`), and similarly do the scripts for the other demos. The main (implementational) difference of the `rot` demo to the others is that in `rot` we do not use `OOPDE`, such that the BC are set via `gnbc.m`, the system matrices are set directly via `setfemops.m` in a more convenient way, and there is no `oosetfemops.m`. As already said, each demo directory (except `pollution`) contains some auxiliary functions or scripts, for instance:

- `cGL/auxcmds.m` with examples of switching to a different parameter and of switching to natural parametrization for temporal mesh–refinement, and `cGL/plotana.m` used for plotting the analytical comparisons in Figs. 3 and 4;
- `rot/auxcmds.m` which contains commands to create a movie of the rotating patterns;
- `bru/auxcmds.m` which gives examples of how adaptive spatial mesh refinement can be used, cf. Remark 4.2.

References

[AK02] I. S. Aranson and L. Kramer. The world of the complex Ginzburg-Landau equation. *Rev. Modern Phys.*, 74(1):99–143, 2002.

[Bar95] D. Barkley. Spiral meandering. In *Chemical Waves and Patterns, edited by R. Kapral and K. Showalter*, page 163. Kluwer, 1995.

[BGVD92] A. Bojanczyk, G.H. Golub, and P. Van Dooren. The periodic Schur decomposition; algorithm and applications. In *Proc. SPIE Conference, Volume 1770*, pages 31–42. 1992.

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

[BPS01] W.J. Beyn, Th. Pampel, and W. Semmler. Dynamic optimization and Skiba sets in economic examples. *Optimal Control Applications and Methods*, 22(5–6):251–280, 2001.

[BT07] W.J. Beyn and V. Thümmler. Phase conditions, symmetries, and pde continuation. In *Numerical continuation methods for dynamical systems*, pages 301–330. Springer, Dordrecht, 2007.

[BT10] Ph. Beltrame and U. Thiele. Time integration and steady-state continuation for 2d lubrication equations. *SIAM J. Appl. Dyn. Syst.*, 9(2):484–518, 2010.

[CG09] M. Cross and H. Greenside. *Pattern Formation and Dynamics in Nonequilibrium Systems*. Cambridge University Press, 2009.

[DF91] E. Dockner and G. Feichtinger. On the optimality of limit cycles in dynamic economic systems. *Journal of Economics*, 53:31–50, 1991.

[Doe07] E. J. Doedel. Lecture notes on numerical analysis of nonlinear equations. In *Numerical continuation methods for dynamical systems*, pages 1–49. Springer, Dordrecht, 2007.

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

[DU16] T. Dohnal and H. Uecker. Bifurcation of Nonlinear Bloch waves from the spectrum in the nonlinear Gross-Pitaevskii equation. *J. Nonlinear Sci.*, 26(3):581–618, 2016.

[DWC+14] H. A. Dijkstra, F. W. Wubs, A. K. Cliffe, E. Doedel, I. Dragomirescu, B. Eckhardt, A. Yu. Gelfgat, A. L. Hazel, V. Lucarini, A. G. Salinger, E. T. Phipps, J Sanchez-Umbria, H Schuttelaars, L. S. Tuckerman, and U. Thiele. Numerical bifurcation methods and their application to fluid dynamics: Analysis beyond simulation. *Communications in Computational Physics*, 15:1–45, 2014.

[FJ91] Th. F. Fairgrieve and A. D. Jepson. O. K. Floquet multipliers. *SIAM J. Numer. Anal.*, 28(5):1446–1462, 1991.

[GAP06] S. V. Gurevich, Sh. Amiranashvili, and H.-G. Purwins. Breathing dissipative solitons in three-component reaction-diffusion system. *Phys. Rev. E*, 74:066201, 2006.

[GCF+08] D. Grass, J.P. Caulkins, G. Feichtinger, G. Tragler, and D.A. Behrens. *Optimal Control of Nonlinear Processes: With Applications in Drugs, Corruption, and Terror*. Springer Verlag, 2008.

[GF13] S. V. Gurevich and R. Friedrich. Moving and breathing localized structures in reaction-diffusion system. *Math. Model. Nat. Phenom.*, 8(5):84–94, 2013.

[GKS00] M. Golubitsky, E. Knobloch, and I. Stewart. Target patterns and spirals in planar reaction-diffusion systems. *J. Nonlinear Sci.*, 10(3):333–354, 2000.

[Gov00] W. Govaerts. *Numerical methods for bifurcations of dynamical equilibria.* SIAM, 2000.

[GS96] W. Govaerts and A. Spence. Detection of Hopf points by counting sectors in the complex plane. *Numer. Math.*, 75(1):43–58, 1996.

[GU16] D. Grass and H. Uecker. Optimal management and spatial patterns in a distributed shallow lake model. Preprint, 2016.

[Hag82] P.S. Hagan. Spiral waves in reaction-diffusion equations. *SIAM Journal on Applied Mathematics*, 42:762–786, 1982.

[HM94] A. Hagberg and E. Meron. Pattern formation in non-gradient reaction-diffusion systems: the effects of front bifurcations. *Nonlinearity*, 7:805–835, 1994.

[KH81] N. Kopell and L.N. Howard. Target pattern and spiral solutions to reaction-diffusion equations with more than one space dimension. *Advances in Applied Mathematics*, 2(4):417–449, 1981.

[Kre01] D. Kressner. An efficient and reliable implementation of the periodic qz algorithm. In *IFAC Workshop on Periodic Control Systems*. 2001.

[Kre06] D. Kressner. A periodic Krylov-Schur algorithm for large matrix products. *Numer. Math.*, 103(3):461–483, 2006.

[Küh15a] Chr. Kühn. Efficient gluing of numerical continuation and a multiple solution method for elliptic PDEs. *Appl. Math. Comput.*, 266:656–674, 2015.

[Küh15b] Chr. Kühn. Numerical continuation and SPDE stability for the 2D cubic-quintic Allen-Cahn equation. *SIAM/ASA J. Uncertain. Quantif.*, 3(1):762–789, 2015.

[Kuz04] Yu. A. Kuznetsov. *Elements of applied bifurcation theory*, volume 112 of *Applied Mathematical Sciences*. Springer-Verlag, New York, third edition, 2004.

[LR00] K. Lust and D. Roose. Computation and bifurcation analysis of periodic solutions of large-scale systems. In *Numerical methods for bifurcation problems and large-scale dynamical systems (Minneapolis, MN, 1997)*, volume 119 of *IMA Vol. Math. Appl.*, pages 265–301. Springer, New York, 2000.

[LRSC98] K. Lust, D. Roose, A. Spence, and A. R. Champneys. An adaptive Newton-Picard algorithm with subspace iteration for computing periodic solutions. *SIAM J. Sci. Comput.*, 19(4):1188–1209, 1998.

[Lus01] K. Lust. Improved numerical Floquet multipliers. *Internat. J. Bifur. Chaos*, 11(9):2389–2410, 2001.

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

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

[MT04] F. Mazzia and D. Trigiante. A hybrid mesh selection strategy based on conditioning for boundary value ODE problems. *Numerical Algorithms*, 36(2):169–187, 2004.

[NS15] M. Net and J. Sánchez. Continuation of bifurcations of periodic orbits for large-scale systems. *SIAM J. Appl. Dyn. Syst.*, 14(2):674–698, 2015.

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

[Prü16] U. Prüfert. OOPDE: FEM for Matlab, www.mathe.tu-freiberg.de/nmo/mitarbeiter/uwe-pruefert/software, 2016.

[Sch98] A. Scheel. Bifurcation to spiral waves in reaction-diffusion systems. *SIAM journal on mathematical analysis*, 29(6):1399–1418, 1998.

[SDE+15] E. Siero, A. Doelman, M. B. Eppinga, J. D. M. Rademacher, M. Rietkerk, and K. Siteur. Striped pattern selection by advective reaction-diffusion systems: resilience of banded vegetation on slopes. *Chaos*, 25(3), 2015.

[Sey10] R. Seydel. *Practical bifurcation and stability analysis. 3rd ed.* Springer, 2010.

[SGN13] J. Sánchez, F. Garcia, and M. Net. Computation of azimuthal waves and their stability in thermal convection in rotating spherical shells with application to the study of a double-hopf bifurcation. *Phys. Rev. E*, page 033014, 2013.

[SS07] B. Sandstede and A. Scheel. Period-doubling of spiral waves and defects. *SIAM J. Appl. Dyn. Syst.*, 6(2):494–547, 2007.

[SSW99] B. Sandstede, A. Scheel, and C. Wulff. Bifurcations and dynamics of spiral waves. *J. Nonlinear Sci.*, 9(4):439–478, 1999.

[TB00] L. S. Tuckerman and D. Barkley. Bifurcation analysis for timesteppers. In *Numerical methods for bifurcation problems and large-scale dynamical systems (Minneapolis, MN, 1997)*, volume 119 of *IMA Vol. Math. Appl.*, pages 453–466. Springer, New York, 2000.

[TW96] O. Tahvonen and C. Withagen. Optimality of irreversible pollution accumulation. *Journal of Environmental Economics and Management*, 20:1775–1795, 1996.

[Uec15] H. Uecker. The pde2path add-on library p2poc for solving infinite time-horizon spatially distributed optimal control problems — Quickstart Guide. Preprint, 2015.

[Uec16a] H. Uecker. Optimal harvesting and spatial patterns in a semi arid vegetation system. *Natural Resource Modelling*, 29(2):229–258, 2016.

[Uec16b] H. Uecker. pde2path, www.staff.uni-oldenburg.de/hannes.uecker/pde2path, 2016.

[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.

[VE01] V. K. Vanag and I. R. Epstein. Inwardly rotating spiral waves in a reaction–diffusion system. *Science*, 294, 2001.

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

[WIJ13] I. Waugh, S. Illingworth, and M. Juniper. Matrix-free continuation of limit cycles for bifurcation analysis of large thermoacoustic systems. *J. Comput. Phys.*, 240:225–247, 2013.

[Wir96] Fr. Wirl. Pathways to Hopf bifurcation in dynamic, continuous time optimization problems. *Journal of Optimization Theory and Applications*, 91:299–320, 1996.

[Wir00] Fr. Wirl. Optimal accumulation of pollution: Existence of limit cycles for the social optimum and the competitive equilibrium. *Journal of Economic Dynamics and Control*, 24(2):297–306, 2000.

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

[YE03] L. Yang and I. R. Epstein. Oscillatory Turing patterns in reaction–diffusion systems with two coupled layers. *PRL*, 90(17):178303–1–4, 2003.

[ZHKR15] D. Zhelyazov, D. Han-Kwan, and J. D. M. Rademacher. Global stability and local bifurcations in a two-fluid model for tokamak plasma. *SIAM J. Appl. Dyn. Syst.*, 14(2):730–763, 2015.