User guide on Hopf bifurcation and time periodic orbits with `pde2path`

Hannes Uecker

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

May 11, 2020

Abstract

We explain the setup for using the `pde2path` libraries for Hopf bifurcation and continuation of branches of periodic orbits and give implementation details of the associated demo directories. See [Uec19] for a description of the basic algorithms and the mathematical background of the examples. Additionally we explain the treatment of Hopf bifurcations in systems with continuous symmetries, including the continuation of traveling waves and rotating waves in O(2) equivariant systems as relative equilibria, the continuation of Hopf bifurcation points via extended systems, and some simple setups for the bifurcation from periodic orbits associated to critical Floquet multipliers going through ± 1.

MSC: 35J47, 35B22, 37M20

Keywords: Hopf bifurcation, periodic orbit continuation, Floquet multipliers, partial differential equations, finite element method

Contents

1 Introduction

In [Uec19] we describe the basic algorithms in `pde2path` to study Hopf bifurcations[1] in PDEs of the form

$$M_d \partial_t u = -G(u, \lambda), \quad u = u(x,t),\ x \in \Omega,\ t \in \mathbb{R}, \tag{1.1}$$

where $M_d \in \mathbb{R}^{N\times N}$ is the dynamical mass matrix, and $G(u,\lambda) := -\nabla\cdot(c\otimes\nabla u) + au - b\otimes\nabla u - f$. Here $u = u(x,t) \in \mathbb{R}^N$, $x \in \Omega$ with $\Omega \subset \mathbb{R}^d$ some bounded domain, $d = 1,2,3$, $\lambda \in \mathbb{R}^p$ is a parameter (vector), and the diffusion, advection and linear tensors c, b, a, and the nonlinearity f, can depend on $x, u, \nabla u$, and parameters. The boundary conditions (BC) for (1.1) are of "generalized Neumann" form, i.e.,

$$\mathbf{n}\cdot(c\otimes\nabla u) + qu = g, \tag{1.2}$$

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, and over rectangular domains there additionally is the possibility of periodic BC in one or more directions. See [Uec19], and the steady state tutorials at [Uec20b], for more details on $c, b, \dots, g$.

`pde2path` spatially discretizes the PDE (1.1), (1.2) via piecewise linear finite elements, leading to the high–dimensional ODE problem (with a slight misuse of notation)

$$M\dot{u} = -G(u,\lambda), \quad u = u(t) \in \mathbb{R}^{n_u}, \quad G(u) = Ku - Mf(u), \tag{1.3}$$

where $n_u = n_p N$ is the number of unknowns, with n_p the number of grid points. In (1.3), M is the mass matrix, K is the stiffness matrix, which typically corresponds to the diffusion term $-\nabla\cdot(c\otimes\nabla u)$, and $Mf : \mathbb{R}^{n_u} \to \mathbb{R}^{n_u}$ contains the rest, which we often also call the 'nonlinearity'. However, (1.3) is really a sort of symbolic notation to express the spatially discretized version of (1.1), and K in (1.3) can also involve nonlinear terms and first order derivatives coming from $b\otimes\nabla u$ in (1.1). See, e.g., [RU19, RU17] for more details.

Here we first present implementation details for the four Hopf bifurcation test problems considered in [Uec19], thus giving a tutorial on how to treat Hopf bifurcations in `pde2path`. See [Uec20b] for download of the package, including the demo directories, and various documentation and tutorials. In

[1] i.e.: the bifurcation of (branches of) time periodic orbits (in short Hopf orbits) from steady states; accordingly, we shall call these branches Hopf branches;

particular, since the Hopf examples are somewhat more involved than the steady case we recommend to new users of `pde2path` to first look into [RU19], which starts with some simple steady state problems.

The first Hopf demo problem (demo `cgl`, subdir of `hopfdemos`) is a cubic–quintic complex Ginzburg–Landau (cGL) equation, which we consider over 1D, 2D, and 3D cuboids with homogeneous Neumann and Dirichlet BC. Next we consider a Brusselator system (demo `brussel`) from [YDZE02], which shows interesting interactions between Turing branches and Turing–Hopf branches. As a non–dissipative example we treat the canonical system for an optimal control problem (see also [dWU19]) of "optimal pollution" (demo `pollution`). This is still of the form (1.1), but is ill–posed as an initial value problem, since it features "backward diffusion". Nevertheless, we continue steady states and obtain Hopf bifurcations and branches of periodic orbits.

In §5 we give three tutorial examples for Hopf bifurcations in systems with continuous symmetries, namely a reaction-diffusion problem with mass conservation (demo `mass-cons`), a Kuramoto-Sivashinsky equation with translational and boost invariance (demos `kspbc2` and `kspbc4`), and a FHN type system with (breathing) pulses featuring an approximate translational symmetry (demo `symtut/breathe`). Such symmetries were not considered systematically in [Uec19]. They require phase conditions, first for the reliable continuation of steady states and detection of (Hopf or steady) bifurcations, which typically lead to the coupling of (1.1) with algebraic equations. To compute Hopf branches we then also need to set up suitable phase conditions for the Hopf orbits. See also [RU17], where preliminary results for the breathing pulses are discussed, and one more example of Hopf orbits for systems with symmetries is considered, namely modulated traveling fronts in a combustion model. Moreover, additional to [Uec19, RU17] we explain some further routines such as continuation of Hopf bifurcation points, and some simple setups for branch switching *from* Hopf orbits, i.e., for Hopf pitchforks (critical multiplier 1) and period doubling (critical multiplier -1). See also [Uec20a, §8] for further examples of period–doubling bifurcations in a classical two–component Brusselator system, following [YZE04].

In §6 we consider Hopf bifurcation in O(2) equivariant systems, which generically leads to coexistence of standing waves (SWs) and traveling waves (TWs). We start with the cGL equation over an interval with periodic BC (pBC) (demo `cglpbc`) and use an appropriate additional phase condition to continue TWs, periodic in the lab frame, but steady in the co–moving frame) as relative equilibria, from which we obtain modulated TWs via Hopf bifurcation in the co–moving frame. A similar approach for the cGL equation in a disk with Neumann BC (demo `cgldisk`) yields spiral waves as rotating waves (RWs), and meandering spirals as modulated RWs. Then we review and extend an example from [Uec19, §3.2], namely a reaction diffusion system in a disk and with Robin BC (demo `gksspirals`), following [GKS00]. In all these examples, the 'interesting' Hopf bifurcation points have multiplicity at least two, leading to SWs vs TWs (or RWs). Our default branch switching so far only deals with simple Hopf bifurcation points systematically, but we use and explain an ad hoc modification for Hopf points of higher multiplicity, allowing to select, e.g., TW vs SW branches. At the end of §6 we also explain a setup to compute periodic orbit branches with fixed period T (freeing a second parameter), and for non–autonomous systems, i.e., with explicit t–dependence of G.

The user interfaces for Hopf bifurcations reuse the standard `pde2path` setup and no new user functions are necessary, except for the case of symmetries, which requires one additional user function. For the basic ideas of continuation and bifurcation in steady problems we refer to [UWR14] (and the references therein), for `pde2path` installation hints and review of data structures to [dWDR$^+$20], for a general soft introduction to [RU19], and for the basic algorithms implemented in the `hopf` library of `pde2path` to [Uec19]. Thus, here we concentrate on how to use these routines, and on recent additions.

The basic setup of all demos 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, e.g., `cgl` (and later `brussel, ...`).
- Main script files, such as `cmds*d.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. Most examples are 2nd order semilinear, i.e., of the form $\partial_t u = -G(u) = D\Delta u + f(u)$ with diffusion matrix $D \in \mathbb{R}^{N\times N}$, and we put the 'nonlinearity' f (i.e., everything except diffusion) into a function `nodalf.m`, which is then called in `sG.m`, but also in mesh-adaption and in normal form computations. An exception is the KS equation, see §5.2.
- a function `oosetfemops.m` for setting up the system matrices.
- 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 plots.
- Some auxiliary scripts `auxcmds.m`, which contain commands, for instance for creating movies or for mesh–refinement, which are not genuinely related to the Hopf computations, 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` (an optimal control problem) 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 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. We give hints about the timing and indications of convergence, but we refrain from a genuine convergence analysis. In some cases (demos `cgl` in 3D and `brussel` in 2D, and `cgldisk, gksspirals`) we additionally switch off the on the fly computation of Floquet multipliers 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 (with about 120000 total degrees of freedom), may take several minutes. All computational times given in the following are from a 16GB i7 laptop with Linux Mint 18 and `Matlab` 2016b.

In §2 to §6 we explain the implementations for the four demos from [Uec19] and the extensions, and thus in passing also the main data-structures and functions from the `hopf` library. In particular, in §3 we extend some results from [Uec19] by Hopf point continuation, and in §5 explain the setup for the systems with symmetries and hence constraints. For the unconstrained theory, and background on the first three example problems, we refer to [Uec19], but for easier reference and to explain the setup with constraints, we also give the pertinent formulas in Appendix A. This also helps understanding a number of new functions, for instance for continuation of TWs and RWs, and for bifurcation from periodic orbits, and Appendix B contains an overview of the functions in the `hopf` library and of the relevant data structures for quick reference. 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`; 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 The cGL equation as an introductory example: Demo `cgl`

We consider the cubic-quintic complex Ginzburg–Landau equation

$$\partial_t u = \Delta u + (r + \mathrm{i}\nu)u - (c_3 + \mathrm{i}\mu)|u|^2 u - c_5|u|^4 u, \quad u = u(t,x) \in \mathbb{C}, \tag{2.1}$$

with real parameters r, ν, c_3, μ, c_5. In real variables u_1, u_2 with $u = u_1 + \mathrm{i}u_2$, (2.1) can be written as the 2–component system

$$\partial_t \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} \Delta + r & -\nu \\ \nu & \Delta + r \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} - (u_1^2 + u_2^2) \begin{pmatrix} c_3 u_1 - \mu u_2 \\ \mu u_1 + c_3 u_2 \end{pmatrix} - c_5 (u_1^2 + u_2^2)^2 \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}. \tag{2.2}$$

We set $c_3 = -1, c_5 = 1, \nu = 1, \mu = 0.1,$ and use r as the main bifurcation parameter. Considering (2.2) on, e.g., a (generalized) rectangle

$$\Omega = (-l_1\pi, l_1\pi) \times \cdots \times (-l_d\pi, l_d\pi) \tag{2.3}$$

with homogeneous Dirichlet BC or Neumann BC, or with periodic BC, we can explicitly calculate all Hopf bifurcation points from the trivial branch $u = 0$, located at $r = |k|^2 := k_1^2 + \ldots + k_d^2$, with wave vector $k \in \mathbb{Z}/(2l_1) \times \ldots \times \mathbb{Z}/(2l_d)$.

In particular from the bifurcation point of view, an important feature of the cGL equation (2.1) are its various symmetries, as also discussed in [RU17]. For homogeneous Neumann and Dirichlet BCs, (2.1) has the gauge (or rotational) symmetry $u \mapsto \mathrm{e}^{\mathrm{i}\phi}u$, i.e., (2.1) is equivariant wrt the action of the special orthogonal group SO(2). Periodic boundary conditions imply a translation invariance as an in general additional SO(2) equivariance, as on the real line. However, for wavetrains (or traveling waves) $u(t,x) = R\exp(\mathrm{i}(kx - \omega t))$, where $R > 0$ and $\omega, k \in \mathbb{R}$ are the amplitude, frequency and wave number, respectively, the rotation and translation have the same group orbits. Therefore, in contrast to the steady case [RU17], for our purposes here the gauge symmetry will not play a role. Additional, (2.1) has the reflection symmetry $x \mapsto -x$ (1D, and analogously for suitable BC over higher dimensional boxes with suitable BC). Thus, in summary, for pBC (and also for the cGL in a disk with e.g., Neumann BC where the role of spatial translation will be played by spatial rotation, the pertinent symmetry group is O(2). Consequently, many Hopf bifurcation points from the trivial solution will be at least double, and in §6 we discuss this case and the associated questions of the bifurcation of standing vs traveling waves, and their numerical treatment in `pde2path`.

Here we shall first focus on (2.1) over boxes with BC that break the translational invariance, such that only discrete symmetries remain, since, as noted above, for Hopf bifurcation the gauge invariance is equivalent to time shifts, and hence is automatically factored out by the phase condition for time shifts. Moreover, we shall choose boxes such that all HBPs are simple.

2.1 General setup

The cGL demo directory consists, as noted above, of some function files to set up and describe (2.2), some script files to run the computations, and a few auxiliary functions and scripts to explain additional features, or, e.g., to produce customized plots. An overview is given in Table 1, and for this 'first' demo we discuss the main files in some detail, while in later demos we will mainly focus on differences to this basic template.

Table 1: Scripts and functions in `hopfdemos/cgl`. Treating the 1D, 2D and 3D cases in one directory, the only dimension dependent files are the scripts, and the function cGLinit. The 2nd part of the table contains auxiliary functions and scripts which are not needed for the Hopf computations, but which illustrate additional `pde2path` features.

script/function	purpose,remarks
cmds*d	main scripts; for $* = 1,2,3$, respectively, which are all quite similar, i.e., mainly differ in settings for output file names. Thus, only cmds1d is discussed in some detail below.
p=cGLinit(p,lx,nx,par,ndim)	init function, setting up parameters and function handles, and, as the only space dimension d=ndim dependent points, the domain.
p=oosetfemops(p)	set FEM matrices (stiffness K and mass M)
r=sG(p,u)	encodes G from 2.2 (including the BC)
f=nodalf(p,u)	the 'nonlinearity' in (2.2), i.e., everything except $D\Delta u$.
Gu=sGjac(p,u)	Jacobian $\partial_u G(u)$ of G.
auxcmds1	script, auxiliary commands, illustrating stability checks by time-integration, and a posteriori computation of Floquet multipliers
auxcmds2	script, auxiliary commands, illustrating (adaptive) mesh-refinement by either switching to natural parametrization, or via `hopftref`
cmds1dconv	script showing some convergence of periods in 1D for finer t discretization
plotana1	plot analytical Hopf-branch for cGL for comparison with numerics, used in `cmds1d.m`, calls mvu=anafloq(rvek,s)
homov2d, homov3d	auxiliary functions to generate movies of Hopf orbits

As main functions we have

- `cGLinit.m`, which (depending on the spatial dimension) sets up the domain, mesh, boundary conditions, and sets the relevant function handles `p.fuha.sG=@sG` and `p.fuha.sGjac=@sGjac` to encode the rhs of (2.2);
- `sG.m` and `sGjac`, which encode (2.2) and the associated Jacobian of G;

Then we have three script files, `cmds*d.m`, where *=1,2,3 stands for the spatial dimension. These are all very similar, i.e., only differ in file names 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 $u \equiv 0$, $r \in \mathbb{R}$;
2. compute branches of periodic orbits (including Floquet multipliers) by calling `hoswibra` and `cont` again, then plotting.

Listings 5-4 discuss the dimension independent (function) files. We use the `OOPDE` setup, and thus we refer to [RU19] for a general introduction concerning the meaning of the stiffness matrix K, the mass matrix M and the BC matrices Q (and $G_{\rm BC}$, not used here), and the initialization methods `stanpdeo*D`, $* = 1,2,3$, setting up an `OOPDE` object which contains the geometry and FEM space, and the methods to assemble the system matrices.

```
function p=cGLinit(p,lx,nx,par,ndim) % (generic) init routine for cGL problem
p=stanparam(p); screenlayout(p); % set standard parameters and screenlayout
p.fuha.sG=@sG; p.fuha.sGjac=@sGjac; p.sw.jac=1; % rhs and Jac
p.nc.neq=2; p.nc.ilam=1; p.fuha.outfu=@hobra; % number of eq, cont-param, output
switch ndim % domain and BC, depending on spatial dim
  case 1; pde=stanpdeo1D(lx,2*lx/nx); p.vol=2*lx; p.x0i=10; % index for ts-plot
       bc=pde.grid.neumannBC('0'); % BC
  case 2; pde=stanpdeo2D(lx,lx/2,2*lx/nx); p.vol=2*lx^2; p.x0i=30;
       bc=pde.grid.robinBC('1','0');
  case 3; pde=stanpdeo3D(lx,lx/2,lx/4,2*lx/nx); p.vol=0.5*lx^3; p.x0i=200;
```

```
        bc=pde.grid.robinBC('1','0');
        p.plot.ng=20; p.plot.lev=[-0.1 0.1]; % 3D specific plot settings
        p.plot.levc={'blue','red'}; p.plot.alpha=0.5;
end
15 pde.grid.makeBoundaryMatrix(bc); p.nc.sf=1e3; p.pdeo=pde; % OOPDE setting of BC
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
20 p.plot.cm=hot; p.plot.bpcmp=9; % colormap for soln plot, comp.for branch-plot
[p.u,res]=nloop(p,p.u);fprintf('first res=%g\n',res); % start-point for cont
p.file.smod=10; p.sol.ds=0.1; p.nc.dsmax=0.5; % saving, stepsize, max stepsize
p.sw.bifcheck=2; p.nc.neig=20; % method for bifcheck, and # Evals used
p.nc.mu1=0.5; % be relaxed about possible bif-detection
```

Listing 1: `cgl/cGLinit.m`, which collects some typical initialization commands. `p=stanparam(p)` in line 2 sets the `pde2path` controls, switches and numerical constants to standard values; these can always be overwritten afterwards, and some typically are. In line 3 set the function handles to the rhs and its Jacobian, and similarly in line 4 we (re)set the output function handle to `hobra`, which can be used as a standard output when Hopf bifurcations are expected. In lines 5-14, depending on the spatial dimension, we create a 1D, 2D or 3D `OOPDE` objects, essentially consisting of the domain, the FEM setup and the boundary condition. In 1D, this is the interval (-lx,lx) with mesh width lx/nx, and homogeneous Neumann BC. In line 15 we finish this by preparing the associated BC matrices, and afterwards we put this `PDE`–object, the number of grid points, and the associated norm weight ξ into `p`. Calling `setfemops` in line 17 then immediately refers to `oosetfemops`, see Listing 5. In line 18 we initialize the solution vector (here with the explicitly known trivial solution $u = 0$ and append the parameters. In the remainder of cGLinit we set some additional controls, mostly explained by the comments. We only remark that p.sw.bifcheck=2 in line 23 tells `pde2path` to use algorithm HD2 [Uec19, §2.1] to detect bifurcations, by computing n_{eig}=p.nc.neig=20 eigenvalues near 0. This is a suitable choice since $\partial_u G(0)$ has no real eigenvalues. p.nc.mu1 in line 24 refers to μ_1 from [Uec19, Remark 2.2]. Finally, `p.vol` in lines 6,8 and 10 is used in the norm (2.4), and `p.x0i` is a point index for plotting the time-series $t \mapsto u(t, x_{\texttt{p.x0i}})$.

```
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;
```

Listing 2: `cgl/sG.m`. Given K and M from oosetfemops, we only need to compute the 'nonlinearity' f, which we outsource to `nodalf`, see Listing 3, and then compute the rhs G (or residual r) as $G(u) = Ku - Mf$. Note that the BC are already included in p.mat.K via line 7 of `oosetfemops`.

```
function f=nodalf(p,u) % nonlinearity for cGL
u1=u(1:p.np); u2=u(p.np+1:2*p.np); par=u(p.nu+1:end); % extract fields
r=par(1); nu=par(2); mu=par(3); c3=par(4); c5=par(5); % and parameters
ua=u1.^2+u2.^2; % aux variable |u|^2
5 f1=r*u1-nu*u2-ua.*(c3*u1-mu*u2)-c5*ua.^2.*u1;
f2=r*u2+nu*u1-ua.*(c3*u2+mu*u1)-c5*ua.^2.*u2;
f=[f1;f2];
```

Listing 3: `cgl/nodalf.m`. The 'nonlinearity' (which includes linear terms, i.e., everything except the diffusion terms) f from (2.2). We extract the two components u_1 and u_2, and the parameters from u, introduce an auxiliary variable ua= $|u|^2$, and write down the components of f in a standard `Matlab` way.

```
function Gu=sGjac(p,u) % Jacobian for cGL
n=p.np; [f1u,f1v,f2u,f2v]=njac(p,u); % the main work
Fu=[[spdiags(f1u,0,n,n),spdiags(f1v,0,n,n)];  % put partial derivatives
    [spdiags(f2u,0,n,n),spdiags(f2v,0,n,n)]]; % into (sparse) Jac
5 Gu=p.mat.K-p.mat.M*Fu; % multiply by -M and add Laplacian
end
```

```
   function [f1u,f1v,f2u,f2v]=njac(p,u) % local (no spat.derivatives) Jacobian
   u1=u(1:p.np); u2=u(p.np+1:2*p.np); par=u(p.nu+1:end);  % extract fields
10 r=par(1); nu=par(2); mu=par(3); c3=par(4); c5=par(5);  % and parameters
   ua=u1.^2+u2.^2;
   f1u=r-2*u1.*(c3*u1-mu*u2)-c3*ua-4*c5*ua.*u1.^2-c5*ua.^2;
   f1v=-nu-2*u2.*(c3*u1-mu*u2)+mu*ua-4*c5*ua.*u1.*u2;
   f2u=nu-2*u1.*(c3*u2+mu*u1)-mu*ua-4*c5*ua.*u1.*u2;
15 f2v=r-2*u2.*(c3*u2+mu*u1)-c3*ua-4*c5*ua.*u2.^2-c5*ua.^2;
   end
```

Listing 4: `cgl/sGjac.m`. Similar to `sG`, the main problem specific part is $\partial_u f$, put into `njac`, the implementation of which follows immediately from `nodalf`. Gu in line 5 is then rather generic.

```
  function p=oosetfemops(p) % FEM operators for cGL
  grid=p.pdeo.grid; % just a shorthand
  [K,M,~]=p.pdeo.fem.assema(grid,1,1,1); % assemble 'scalar' K and M
  [Q,~,~,~]=p.pdeo.fem.assemb(grid); % BC matrix
5 sf=p.nc.sf;  % stiff spring factor to implement DBC
  N=sparse(grid.nPoints, grid.nPoints); % 0-matrix, another shorthand
  p.mat.K=[[K+sf*Q N];[N K+sf*Q]]; % build 2-comp. system K
  p.mat.M=kron([[1,0];[0,1]],M);   % and M
```

Listing 5: `cgl/oosetfemops.m`. This sets the stiffness matrix K, the mass matrix M, and the BC matrix Q for (2.2); see [RU19, §1] for the meaning of these matrices. K$\in \mathbb{R}^{n_p\times n_p}$ in line 2 is the 'scalar' (i.e., one component) Neumann Laplacian, while M is the scalar mass matrix. Similarly, Q$\in \mathbb{R}^{n_p\times n_p}$ in line 3 is a BC matrix for one component, and its content depends on the BC set in lines 7, 9 and 11 of `cGLinit`. In line 6 of oosetfemops we create a zero matrix for convenience, and in lines 5,6 we then set up the FEM matrices for the *system* (2.2). Here both diagonal blocks of p.mat.K are equal, because so are the diffusion constants for both components in (2.2) and the BC we consider. However, this setup is quite flexible to implement also more complicated differential operators, including off-diagonal blocks ('cross diffusion'), first order differential operators, and different BC in different components.

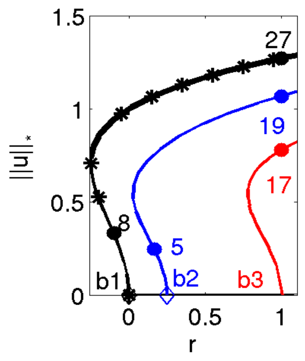
2.2 1D

In 1D we use Neumann BC, and $n_x = 31$ spatial, and (without mesh-refinement) $m = 21$ temporal discretization points. Listings 6 and 7 shows the main script file `cmds1d.m` for 1D computations (with some omissions wrt to plotting), while Fig. 1 shows some output generated by running `cmds1d`. The norm in (a) is

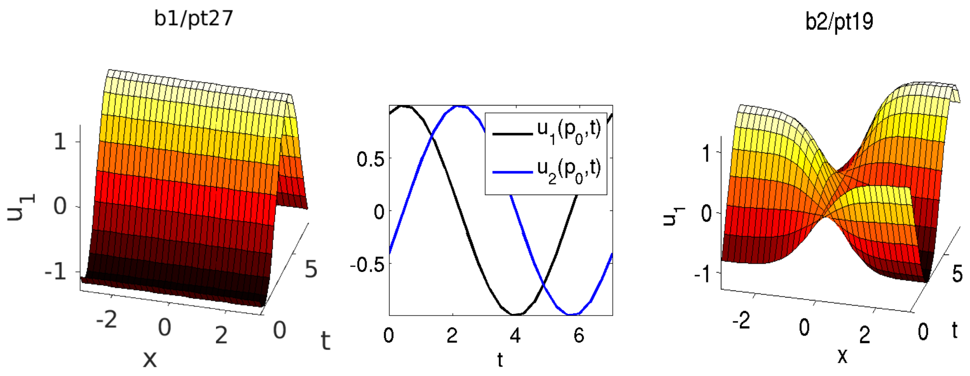
$$\|u\|_* := \|u\|_{L^2(\Omega\times(0,T),\mathbb{R}^N)}/\sqrt{T|\Omega|}, \tag{2.4}$$

which is our default for plotting of Hopf branches. During the continuation the default plotting routine `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. 1(b))). 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.

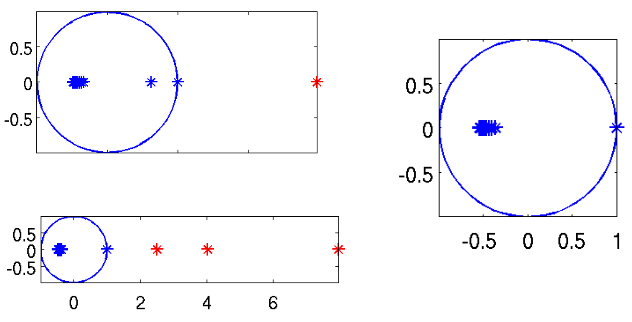
(a) BD, norm $\|u(\cdot,\cdot;r)\|_*$



(b) Example plots



(c) Multipliers at b1/pt8 (ind = 1) and b2/pt5 (ind = 3) (left), and at b1/pt27 (ind = 0) (right)



(d) left: BD, period $T(r)$. Right: numerical periods (for $m = 20, 40, 60$) and analytical period (black dots) on the 1st branch

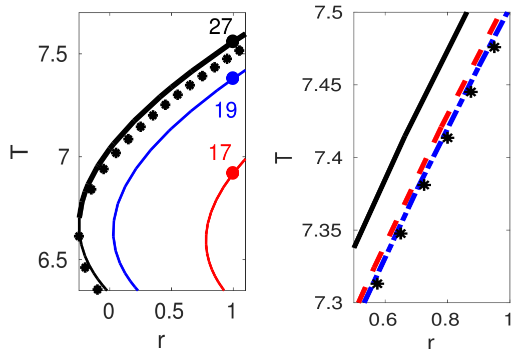


Figure 1: Selected outputs from `cmds1d.m`, i.e., numerical bifurcation diagrams, example plots and (leading 20) Floquet multipliers for (2.2) on the domain $\Omega = (-\pi, \pi)$ with Neumann BC, 30 grid–points in x. Parameters $(\nu, \mu, c_3, c_5) = (1, 0.1, -1, 1)$, hence bifurcations at (restricting 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). The thick part of the black line in (a) indicates the only stable periodic solutions. The black dots in (a) and (d) are from the (spatially homogeneous) analytical solution, see [Uec19]. For $m = 20$ there is a visible error in T. The right panel of (d) shows the numerical T for different m ($m = 20$ black, $m = 40$ red-dashed, $m = 60$ blue-dotted), which illustrates the convergence of the numerical solution towards the analytical solution (6.8). Similarly, the periods also converge on the other branches (see `cmds1dconv.m`). The second plot in (b) shows a time series at the point `p.hopf.x0i` from `b1/pt27`. See also Fig. 2(b) for a plot of `b3/pt17`.

As usual we recommend to run `cmds1d` cell-by-cell to see the effect(s) of the individual cells.

```
%% C1: init, and continuation of trivial branch
ndim=1; dir='hom1d'; p=[]; lx=pi/2; nx=30; % domain size and spat.resolution
par=[-0.05; 1; 0.1; -1; 1]; % r nu mu c3 c5
p=cGLinit(p,lx,nx,par,ndim); p=setfn(p,dir); % initialization and dirname
p.sw.verb=2; p=cont(p,20); % cont of (here) trivial branch, incl. bif-detec
%% C2: first 2 Hopf branches, run arclength from the start
para=4; ds=0.1; figure(2); clf; aux=[]; %aux.tl=60;
for bnr=1:2
    switch bnr
     case 1; p=hoswibra('hom1d','hpt1',ds,para,'1db1',aux); nsteps=30;
     case 2; p=hoswibra('hom1d','hpt2',ds,para,'1db2'); nsteps=20;
    end
    p.hopf.jac=1; p.nc.dsmax=0.5; p.hopf.xi=0.05; p.file.smod=5; p.sw.verb=2;
    p.hopf.flcheck=2; % switch for Floquet-comp: 0: off, 1:floq, 2: floqps
    bw=1; beltol=1e-6; belimax=5; % border-width, bel-parameters
    droptol=1e-3; AMGmaxit=200; % ilupack parameters (only needed if AMG=1)
    AMG=1; % set AMG=1 if ilupack available
    if ~AMG; p=setbel(p,bw,beltol,belimax,@lss); % use BEL without ilupack
    else p=setbel(p,bw,beltol,belimax,@lssAMG); p=setilup(p,droptol,AMGmaxit);
20  end
    t1=tic; p=cont(p,nsteps); toc(t1)
end
%% C3: on branch 3, use tomsol for initial steps, then switch to arclength
ds=0.2; para=3; p=hoswibra('hom1d','hpt3',ds,para,'1db3');
25 p.hopf.xi=0.05; p.hopf.jac=1; p.nc.dsmax=0.25;
p.hopf.tom.AbsTol=1e-4; p.hopf.tom.RelTol=1e-3; % tolerances for TOM
p=cont(p,5); % do 5 steps in natural parametrization
p.sw.para=4; % then switch to arclength
if ~AMG; p=setbel(p,bw,beltol,belimax,@lss); % use BEL without ilupack
30 else p=setbel(p,bw-1,beltol,belimax,@lssAMG); p=setilup(p,droptol,AMGmaxit); end
tic; p=cont(p,15); toc
```

Listing 6: `cgl/cmds1d.m` (first four cells). In cell 1 we initialize the problem and continue the trivial branch (with standard settings) to find the HBPs. In cell 2 we then compute the first 2 bifurcating Hopf branches in the arclength setting. See Appendix B for comments on `hoswibra`, which sets all the data structures for periodic orbit continuation and of course an initial guess, and thus is the main routine here. In line 15 we switch on the Floquet computation with `floq`, see §2.3. In line 16 we set parameters for the (optional) bordered elimination linear system solver `lssbel`, see [Uec19, Remark 2.3], and in line 17 for the preconditioned ilupack solver [Bol11] `lssAMG` as an inner solver for `lssbel`. This is optional, and controlled by the switch `AMG` in line 18. See lines 19,20 for the convenience functions to switch on these solvers and to set parameters. For the present 1D problem, both `lssAMG` or just `lss` are roughly equally fast, but for larger scale problems `lssAMG` is significantly faster. In any case, without `ilupack`, `lssbel` gives a significant speedup over `lss` for bordered systems, see also [UW17] for a tutorial on these solvers. In cell 3 we do the initial steps for the third Hopf branch in natural parametrization, which gives a refinement of the t-mesh by TOM from $m = 21$ to $m = 41$ (here uniform due to the harmonic nature of the time-dependence). We then switch to arclength and proceed as before.

```
cmp=9; wnr=3; figure(wnr);clf;plotbra('hom1d',3,cmp,'lsw',4); % label only HBPs
plotbra('1db1',3,cmp,'lab',[8,27]);  % ... some omissions
cmp=6; figure(wnr); clf; plotbra('1db1',3,cmp, 'lab',27, 'fp',1); % plot BD, T
hoplotf('1db1','pt27',1,1); figure(1); title('1db1/pt27'); % plot solns
```

Listing 7: `cgl/cmds1d.m` continued, to illustrate (with some omissions) the plot of bifurcation diagrams and solutions. Since in `cGLinit` we set p.fuha.outfu=@hobra, i.e., to the standard Hopf branch output, and since we have 5 parameters in the problem, the period T is at (user-)component 6 of the branch, then follow min and max, and component 9 contains the L^2 norm; see also [dW17] for details on the organization of the branch data and on `plotbra`.

Switching to continuation in another parameter works just as for stationary problems by calling `p=hoswiparf(...)`. See Cells 1 and 2 of `cgl/auxcmds1.m` for an example, and Fig. 2(a) for illustration. Cells 3 and 4 of `auxcmds1.m` then contain examples for mesh-refinement in t, for which there are essentially two options. The first is to use `p.sw.para=3` and the mesh-adaption of TOM, the second is `hopftref`, see Listing 8.

```
%% C1: change continuation param
p=hoswiparf('1db1','pt28','c5b',5,0.1); clf(2); p.usrlam=0.25; p=cont(p,20);
%% C2: plot BD and solns
bpcmp=6; pstyle=3; wnr=3; figure(wnr); clf;
plotbraf('c5b','pt18',3,bpcmp); xlabel('c_5'); ylabel('T');

%% C3: mesh-refinement in t using TOM:
p=loadp('1db2','pt10','1db1ref'); hoplot(p,4,1,1);
```

```
% switch to nat.-parametr., and reset tolerances, then cont
p=arc2tom(p); p.hopf.tau=[]; p.sol.ds=0.01;
p.hopf.tom.AbsTol=5e-5; p.hopf.tom.RelTol=5e-4; p=cont(p,5);
p.sw.para=4; p=tom2arc(p); p=cont(p,5); % switch back to arclength and cont
%% C4: mesh-refinement using hopftref
p=loadp('1db3','pt17','1db1ref');
% hogradinf(p); % info about max_t |udot| (here useless, since u is harmonic)
p=hopftref(p,4); p=cont(p,5); % bisect 5 intervals after t=4 and cont
%% C5: uniform mesh-refinement
p=loadp('1db3','pt17','1db1ref'); fac=2.3;
p=uhopftref(p,fac); p=cont(p,5); % increase # time-slices by fac, then cont
```

Listing 8: `cgl/auxcmds1.m`. Cells 1 and 2 illustrate switching to another continuation parameter, while cells 3 and 4 give simple examples of mesh-adaption in t. In cell 3 we use the error estimator build into TOM. In cell 4 we use `hopftref`, which is a purely ad hoc refinement, and which requires a time t^* where to refine from the user. In some cases, the convenience function `hogradinf(p)`, which inter alia returns the time t^* where $\|\partial_t u(\cdot,t)\|_\infty$ is maximal, is useful, though not in this problem since the solutions considered here are rather time harmonic. Note that in contrast to cell 3, or to the routine `meshada` for spatial mesh refinement, neither `hogradinf(p)` nor `hopftref` deal with error estimates in any sense.

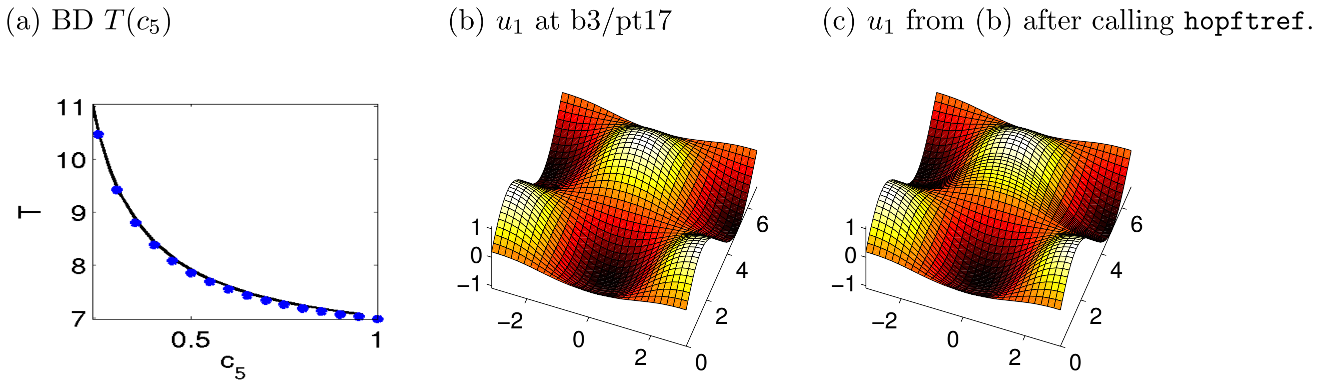


Figure 2: Example outputs from `auxcmds1.m`. (a) Continuing the solution b1/pt28 from Fig. 1(a,b) in c_5, with comparison to the analytical formula [Uec19, §3.1]. (b), (c) Solution at `b3/pt17` before and after mesh-refinement in t via `hopftref`, here near $t^* = 4$.

2.3 Remarks on Floquet multipliers and time integration

For the Floquet multipliers γ_j, $j = 1, \dots, n_u$ ($n_u = Nn_p$ with n_p the number of spatial discretization points, see (1.3)) we recall from [Uec19, §2.4] that we have two algorithms for their computation:[2]

- **FA1** (encoded in the function `floq`) computes $0 \le$ `p.hopf.nfloq` $\le n_u$ multipliers as eigenvalues of the monodromy matrix $\mathcal{M}$.
- **FA2**(encoded in `floqps`) uses a periodic Schur decomposition of the matrices building $\mathcal{M}$ to compute all n_u multipliers.

FA2 is generally much more accurate and robust, but may be slow.[3] See also line 15 in Listing 6. There always is the trivial Floquet multiplier $\gamma_1 = 1$ associated to translational in t, and we use $\mathrm{err}_{\gamma_1} := |\gamma_1 - 1|$ with the numerical γ_1 as a measure for the accuracy of the multiplier computation.

[2]in the software we typically call the Floquet multipliers μ instead of γ

[3]For `floqps` one needs to `mex` percomplex.f(F) in the directory `pqzschur`, see the README file there.

Furthermore we define the index of a periodic orbit u_H as

$$\mathrm{ind}(u_H) = \text{number of multipliers } \gamma \text{ with } |\gamma| > 1 \text{ (numerically: } |\gamma| > 1+\text{p.hopf.fltol}), \tag{2.5}$$

such that $\mathrm{ind}(u_H) > 0$ indicates instability.

On `b1` in Fig. 1, initially there is one unstable multiplier γ_2, i.e., $\mathrm{ind}(u_H) = 1$, which passes through 1 to enter the unit circle at the fold. 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 bifurcation 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`.

In `cgl/auxcmds2.m` we revisit these multiplier computations, and complement them with time-integration. For the latter, the idea 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 $e(t)$.[4] In Fig. 3(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

```
%% C1: plotting of precomputed multipliers
h=plotfloq('1db1','pt8');
%% C2: a posteriori compute and plot multipliers
aux=[];aux.wnr=6; aux.nfloq=10;
5 [muv1,muv2]=floqap('1db2','pt10',aux); axis tight;
%% C3: this cell only if percomplex has been mexed
aux.wnr=8; [muv1,muv2]=floqpsap('1db2','pt10',aux);
%% C4: time-integration, preparations
p=loadp('1db1','pt20'); hoplot(p,1,1); dir='stab1d1';
10 p.u(1:p.nu)=p.hopf.y(1:p.nu,1); u0=p.u(1:p.nu); p=setfn(p,dir);
ts=[]; t0=0; npp=50; nt=200; pmod=50; smod=5; tsmod=1; nc=0;
%% C5: time-integration (repeat if necessary)
[p,t0,ts,nc]=hotintxs(p,u0,t0,ts,npp,nt,nc,tsmod,pmod,smod,@nodalf,1);
figure(4); clf; plot(ts(1,:), ts(2,:)); % plot values at selected point
15 figure(5); clf; plot(ts(1,:), ts(3,:)); % plot difference in norm
%% C6: x-t plot; see in ts if there's something interesting after np
% periods, then plot around there
si=3*npp; incr=25; nt=5*npp/smod; wnr=2; cmp=1; vv=[30,70]; nt=15;
tintplot1d(dir,si,incr,nt,wnr,cmp,vv);
```

Listing 9: `cgl/auxcmds2.m`. Cells 1-3 deal with Floquet computations as indicated in the comments. Cells 4-6 deal with time integration. In line 10 we set up $u(\cdot, t_0)$ from the Hopf-orbit in `1db1/pt8` as an initial condition, and set some parameters. This is used in Cell 5 for time integration via `hotintxs` (see source for documentation), and Cell 6 plots the results, see Fig. 3.

[4] 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.

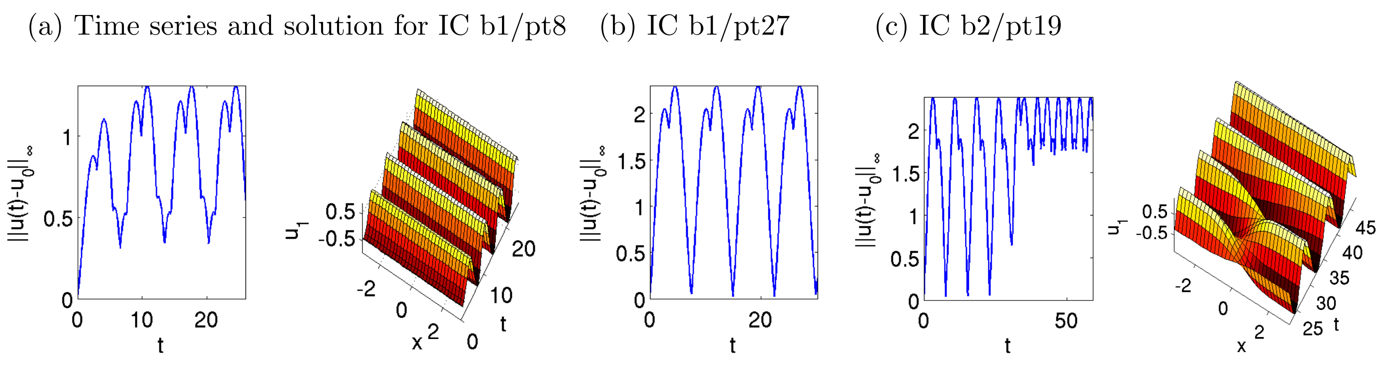


Figure 3: Selected output from `auxcmds2.m`, i.e., stability experiments for (2.2) in 1D. (a) IC h1/pt8, time series of $\|u(\cdot,t)-u_0\|_\infty$ and $u_1(x,t)$, showing the convergence to the larger amplitude solution at the same r. (b) IC h1/pt27 from Fig. 1, where we plot $\|u(\cdot,t)-u_0\|_\infty$ 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. 1, 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 4.

2.4 2D

In 2D we choose homogeneous Dirichlet BC for u_1, u_2, see lines 8,9 in `cGLinit`, and `oosetfemops.m`. Then the first two HBPs are at $r_1 = 5/4$ ($k = (1/2, 1)$, and $r_2 = 2$ ($k = (1, 1)$). The script file `cmds2d.m` follows the same principles as `cmds1d.m`, and includes some time integration as well, and in the last cell an example for creating a movie of a periodic orbits.

Figure 4 shows some results from `cmds2d.m`, obtained on a coarse mesh of 41×21 points, hence $n_u = 1722$ spatial unknowns, yielding the numerical values $r_1 = 1.2526$ and $r_2 = 2.01$. With $m = 20$ temporal discretization points, the computation of each Hopf branch then takes about 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 $\text{ind}(u_H) = 1$ up to the fold, and $\text{ind}(u_H) = 0$ afterwards, and on b2 $\text{ind}(u_H)$ decreases from 3 to 2 at the fold and to 1 near $r = 7.2$. Panel (c) illustrates the 2D analogue of Fig. 3(c), i.e., the instability of the second Hopf branch and stability of the first.

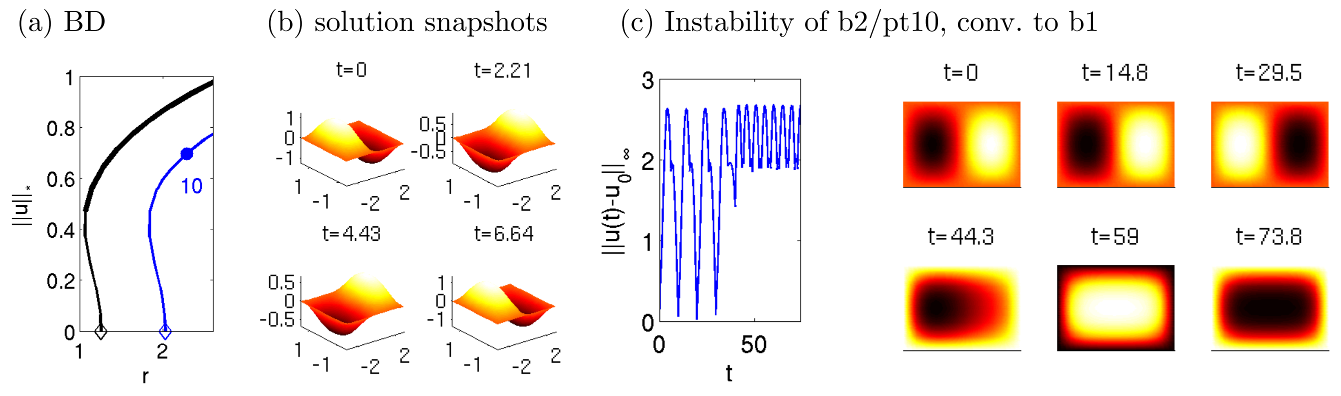


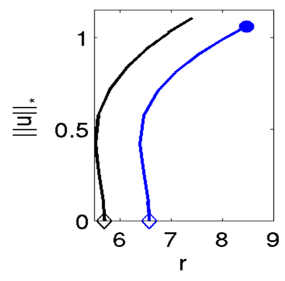
Figure 4: Example plots from `cmds2d.m`. (a) Bifurcation diagrams of the first 2 Hopf branches for (2.2) in 2D. (b) Solution snapshot from b2/pt10, at $t = 0, \frac{3}{10}T, \frac{6}{10}T, \frac{9}{10}T$. (c) Time integration starting from (b) ($t = 0$), with convergence to the first Hopf branch.

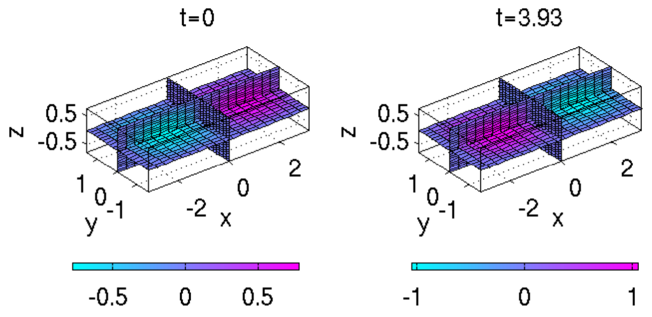
2.5 3D

To illustrate that exactly the same setup also works in 3D, in `cmds3d.m` and Fig. 5 we consider (2.2) 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=2912 points, thus 5824 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=5.47 and r_1=6.29. Again, this can be greatly improved by, e.g., halving the spatial mesh width, but then the Hopf branches become very expensive. Using $m = 20$, the computation of the branches (with 15 continuation steps each) in Fig. 5 takes about 10 minutes, and a call of `floqap` to a posteriori compute the Floquet multipliers about 50 seconds. Again, on b1, ind(u_H)=1 up to fold and ind(u_H)=0 afterwards, while on b2 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.

The script `cmds2d.m` follows the same principles as the 1D and 2D scripts. In 3D, the "slice plot" in Fig. 5(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.

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





(c) Example isoplot

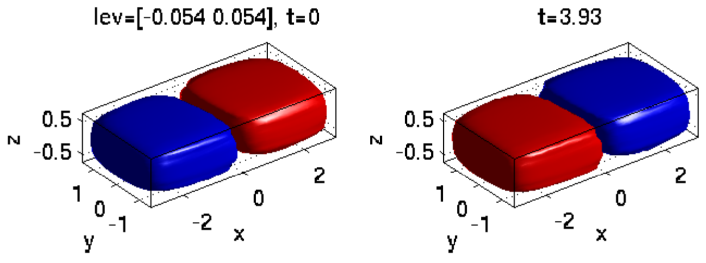


Figure 5: Example plots from `cmds3d.m`. (a) Bifurcation diagram of first 2 Hopf branches for (2.2) in 3D. (b,c) Solution snapshots at $t = 0$ and $t = T/2$ for the blue dot in (a); slice-plot in (b), and isolevel plot in (c) with levels $0.525m_1 + 0.475m_2$ and $0.475m_1 + 0.525m_2$, where $m_1 = \min_{x,t} u_1(x,t)$ and $m_2 = \max_{x,t} u_1(x,t)$.

3 An extended Brusselator: Demo `brussel`

In [Uec19, §3.2] we consider an example with an interesting interplay between stationary patterns and Hopf bifurcations, and where there are typically many eigenvalues with small real parts, such detecting

HBPs with `bifcheck=2` without first using `initeig` for setting a guess for a shift ω_1 is problematic. The model, following [YDZE02] is an 'extended Brusselator', namely the three component reaction–diffusion system

$$\partial_t u = D_u \Delta u + f(u,v) - cu + dw, \quad \partial_t v = D_v \Delta v + g(u,v), \quad \partial_t w = D_w \Delta w + cu - dw, \tag{3.1}$$

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 (3.1) on rectangular domains in 1D and 2D, with homogeneous Neumann BC for all three components. The system has the trivial spatially homogeneous steady state

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

and in suitable parameter regimes it shows co-dimension 2 points between Hopf, Turing–Hopf (aka wave), and (stationary) Turing bifurcations from U_s. A discussion of these instabilities of U_s in the $a - b$ plane is given in [YDZE02] for fixed parameters

$$(c, d, D_u, D_v, D_w) = (1, 1, 0.01, 0.1, 1). \tag{3.2}$$

In our simulations we additionally fix $a = 0.95$, and take b as the primary bifurcation parameter.

For the quite rich bifurcation results, which include primary spatially homogeneous and patterned Hopf bifurcations from $U \equiv U_s$, and Turing bifurcations from U_s followed by secondary Hopf bifurcations, we refer to [Uec19, §3.2]. Regarding the implementation, Table 2 lists the scripts and functions in `brussel`. Except for the additional component ($N = 3$ instead of $N = 2$) this is quite similar to `cgl`, with one crucial difference, in particular in 2D, on which we focus in §3.2. First, however, we shall focus on 1D and additional to [Uec19, §3.2] compute bifurcation lines in the a–b parameter plane by branch point continuation and Hopf point continuation, and compute secondary bifurcations *from* Hopf orbits.

Table 2: Scripts and functions in `hopfdemos/brussel`.

script/function	purpose,remarks
bru1dcmds	basic BDs for 1D, including some time integration.
bru1dcmds_b	extension of bru1dcmds, dealing with bifurcations from the primary Hopf orbit
bru2dcmds	script for 2D, including preparatory step `initeig` for guessing iω for Hopf bifurcations, and some time integration
cmdsHPc	script for Hopf and branch point continuation to compute Fig. 6(a)
auxcmds1	1D auxiliaries, illustrating spatial mesh refinement on Turing branches
auxcmds2	2D auxiliaries: illustration of problems with many small real eigenvalues
e2rsbru	elements to refine selector, interface to `OOPDE`'s equivalent of `pdejmps`
evalplot	script for plotting eigenvalues for linearization around spat. homogeneous solution, see [Uec19, Fig.7(b)].
bruinit	initialization as usual
oosetfemops	the FEM operator for (3.1), `OOPDE` setting
sG, sGjac, nodalf	rhs, Jacobian, and nonlinearity, as usual
bpjac, hpjac	computing (directional) second derivatives for BP and HP continuation

3.1 1D

Listing 10 shows the startup in 1D. We fix $a = 0.95$ and choose b as the continuation parameter, starting at $b = 2.75$, over the domain $\Omega = (-l_x, l_x)$, $l_x = \pi/k_{TH}$, where $k_{TH} = 1.4$ is chosen to have the first bifurcation from U_s to a Turing-Hopf (or wave) branch. This follows from, e.g., [YDZE02], see also [Uec19], but also from the bifurcation lines and spectral plots in Fig. 6(a,b), which we explain below. The 'standard' files such as `bruinit.m`, `oosetfemops.m`, `sG.m`, and `sGjac.m` are really standard and thus we refer to their sources. Moreover, regarding `initeig` in line 5 of `bru1dcmds.m` we refer to §3.2, where this becomes crucial. The continuation in line 6 of `bru1dcmds.m` then yields the first three bifurcations as predicted from Fig. 6(a,b), and subsequently further steady and Hopf bifurcations. See Fig. 6(c) for the BD of these branches, and [Uec19, §3.4] for further discussion. Here we first want to explain how Fig. 6(a) can be computed by Hopf point continuation (HPC) and branch point continuation BPC, see Listing 11.

```
  %% init, and cont of hom.steady branch
  ndim=1; dir='hom1d'; p=[]; lx=pi/1.4; nx=100;
  par=[0.95; 2.75; 1; 1; 0.01; 0.1; 1]; % a, b, c, d, Du, Dv, Dw
  p=bruinit(p,lx,nx,par,ndim); p=setfn(p,dir); p.sw.bifcheck=2;
5 p=initeig(p,4); p.nc.neig=[5, 5]; % init omv (compute guesses for eval shifts)
  p.sw.verb=2; p.nc.mu2=5e-3; p=cont(p,20);
```

Listing 10: `brussel/bru1dcmds.m` (first 6 lines). The `initeig` in line 5 is not strictly necessary in 1D, but useful for speed. The remainder of `bru1dcmds` computes a number of steady and Hopf bifurcations from `hom1d`, and some secondary Hopf bifurcations from Turing branches, and we refer to [Uec19] for the associated BDs and solution plots.

```
%% HP and BP continuations, first the wave (Turing-Hopf) branch
p=hpcontini('hom1d','hpt1',1,'hpc1'); huclean(p); p.sw.bprint=2;p.plot.bpcmp=2;
%[Ja,Jn]=hpjaccheck(p); pause % to check the correct impl. of hpjac
p.sol.ds=-0.01; p.nc.lammax=1.25; p.nc.lammin=0.8; p0=p; p=cont(p,20);
p=p0; p=setfn(p,'hpc1b'); p.sol.ds=-p.sol.ds; p=cont(p,20); % other direction
%% check HP continuation
p=hpcontexit('hpc1','pt5','t1'); % puts the HBP into dir 't1'
p=hoswibra('t1','hpt1',ds,para,'t1h'); p.nc.lammax=3.5; p=cont(p,5); % continue
%% BP-cont for Turing:
p=bpcontini('hom1d','bpt1',1,'bpc1'); p.sw.bprint=2; p.plot.bpcmp=2;
p.sol.ds=-0.01; p.nc.lammin=0.8; p.nc.lammax=1.15; p0=p; p=cont(p,20);
p=p0; p=setfn(p,'bpc1b'); p.sol.ds=-p.sol.ds; p=cont(p,20); % other direction
%% check BP continuation
p=bpcontexit('bpc1','pt5','t2'); % puts the BP into dir 't2'
p=swibra('t2','bpt1','t2b'); p.nc.lammax=3.5; p=cont(p,5); % swibra and cont
```

Listing 11: `brussel/bruHPCcmds.m` (first 15 lines). The ideas of HP continuation and of BP continuation are explained in §3.1.1 and [Uec20a], respectively.

3.1.1 Hopf point continuation

Similar to fold continuation and branch point continuation ([DRUW14] and [Uec20a, §3.4]), Hopf point continuation (HPC) can be done via suitable extended systems. Here we use [Gov00, §4.3.2]

$$H(U) := \begin{pmatrix} G \\ G_u\phi_r + \omega M\phi_i \\ G_u\phi_i - \omega M\phi_r \\ c^T\phi_r - 1 \\ c^T\phi_i \end{pmatrix} = 0 \in \mathbb{R}^{3n_u+2}, \quad U = (u, \phi_r, \phi_i, \omega, \lambda), \tag{3.3}$$

where $\mathrm{i}\omega \in \mathbb{R}$ is the desired eigenvalue of G_u, $\phi = \phi_r + \mathrm{i}\phi_i \in \mathbb{C}^{n_u}$ an associated eigenvector, and $c_r \in \mathbb{R}^{n_u}$ is a normalization vector. We thus have $3n_u + 2$ equations for the $3n_u + 2$ real unknowns U, and in [Gov00, Proposition 4.3.3] it is shown that (3.3) is regular at a simple Hopf bifurcation point.

Thus, (3.3) can be used for localization of (simple) Hopf points (implemented in the `pde2path` function `hploc`) if a sufficiently good initial guess U is given, and, moreover, freeing a second parameter w we can use the extended system

$$\mathcal{H}(U, w) = \begin{pmatrix} H(U, w) \\ p(U, w, \mathrm{ds}) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \mathbb{R}^{(3n_u+2)+1} \tag{3.4}$$

for HPC, where $p(U, w, \mathrm{ds})$ is the standard arclength condition, with suitable weights for the parameters λ, w. This requires the Jacobian

$$\partial_U H(U) = \begin{pmatrix} G_u & 0 & 0 & 0 & G_\lambda \\ \partial_u(G_u\phi_r) & G_u & \omega M & M\phi_i & \partial_\lambda(G_u\phi_r) \\ \partial_u(G_u\phi_i) & -\omega M & G_u & -M\phi_r & \partial_\lambda(G_u\phi_i) \\ 0 & c^T & 0 & 0 & 0 \\ 0 & 0 & c^T & 0 & 0 \end{pmatrix} \in \mathbb{R}^{(3n_u+2)\times(3n_u+2)}. \tag{3.5}$$

Here, G_u is already available, the $\partial_\lambda\cdot$ expressions are cheap from finite differences, as well as the w derivatives needed in the arclength continuation, and expressions such as ωM are easy. Thus the main task is to compute the directional 2nd derivatives

$$\begin{pmatrix} \partial_u(G_u\phi_r) \\ \partial_u(G_u\phi_r) \end{pmatrix} \in \mathbb{R}^{2n_u\times n_u}. \tag{3.6}$$

This can be done numerically, but this may be expensive, and for semilinear problems $G(u) = Ku - Mf(u)$ it is typically easy to write a function `hpjac` (or with some other problem specific name `name`) which returns (3.6), and which should be registered as `p.fuha.spjac=@hpjac` (or `p.fuha.spjac=@name`). For instance, for $N = 2$ components and $\phi_r = (\phi_1, \phi_2)$ we have

$$\partial_u(G_u\phi_r) = -M\partial_u \begin{pmatrix} f_{1,u_1}\phi_1 + f_{1,u_2}\phi_2 \\ f_{2,u_1}\phi_1 + f_{2,u_2}\phi_2 \end{pmatrix} = -M \begin{pmatrix} f_{1,u_1u_1}\phi_1 + f_{1,u_2u_1}\phi_2 & f_{1,u_1u_2}\phi_1 + f_{1,u_2u_2}\phi_2 \\ f_{2,u_1u_1}\phi_1 + f_{2,u_2u_1}\phi_2 & f_{2,u_1u_2}\phi_1 + f_{2,u_2u_2}\phi_2 \end{pmatrix}, \tag{3.7}$$

and we obtain the same expression for $\partial_u(G_u\phi_i)$ with $\phi_i = (\phi_1, \phi_2)$. Accordingly, `brussel/hpjac.m` returns $\begin{pmatrix}\partial_u(G_u\phi_r)\\ \partial_u(G_u\phi_r)\end{pmatrix}$ for the three component semilinear system (3.1). For general testing we also provide the function `hpjaccheck`, which checks `p.fuha.hpjac` against finite differences.

To initialize HPC, the user can call `p=hpcontini('hom1d','hpt1',1,'hpc1')`, see line 2 of Listing 11, where the third argument gives the new free parameter w. Here $w = a$, which is at position 1 in the parameter vector. This triples `p.nu` and sets a number of further switches, for instance for automatically taking care of the structure of $\partial_U H(U)$ in (3.5). For convenience `hpcontini` also directly sets `p.fuha.spjac=@hpjac`, which of course the user can reset afterwards. Then calling `cont` will continue (3.4) in w, and thus we produce the 'wave' and 'Hopf' lines in Fig. 6(a). Use `hpcontexit` to return to 'normal' continuation (in the original primary parameter).

Similarly, BPC is based on the extended system [Mei00, §3.3.2]

$$H(U) = \begin{pmatrix} G(u,\lambda)+\mu M\psi \\ G_u^T(u,w)\psi \\ \|\psi\|_2^2-1 \\ \langle\psi, G_\lambda(u,w)\rangle \end{pmatrix} = 0 \in \mathbb{R}^{2n_u+2}, \quad U=(u,\psi,w), \tag{3.8}$$

where (u,λ) is a (simple) BP (for the continuation in λ), ψ is an adjoint kernel vector, $w=(\lambda,\mu)$ with $w_1=\lambda$ the primary active parameter and $w_2=\mu$ as additional active parameter. The BPC requires the Jacobian $\partial_U H$ of which $\partial_u(G_u^T\psi)$ is potentially difficult to implement. However, again for semilinear problems $\partial_u(G_u^T\psi)$ has a similar structure as (3.7), see [Uec20a, §3.4] for further details. In particular, for (3.1) it can be implemented rather easily, see `bpjac.m`. The actual BPC is then initialized by calling `bpcontini`, see line 11 of Listing 11, and the BPC produces the 'Turing line' in Fig. 6(a). Using `bpcontexit` returns to 'normal' continuation.

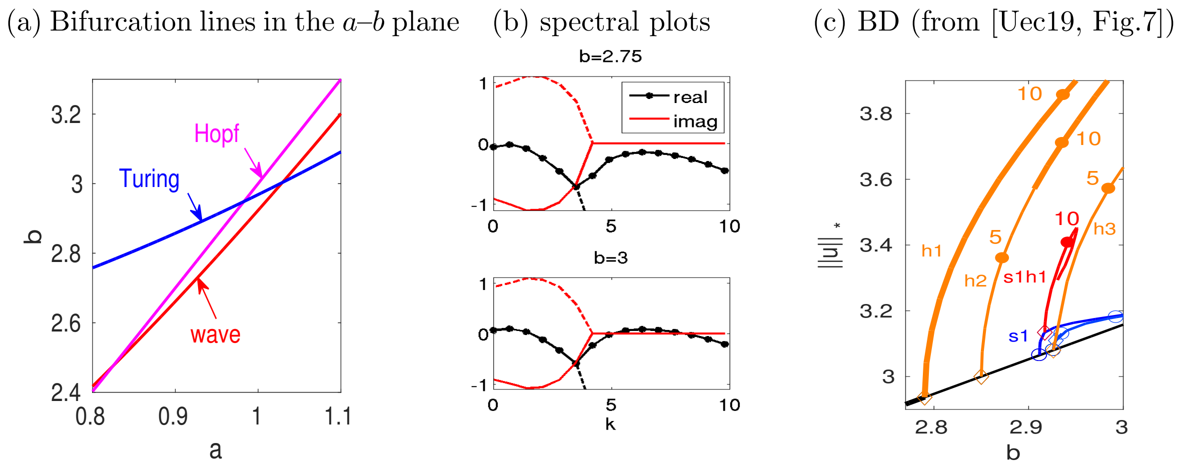


Figure 6: Results for (3.1), from `cmdsHPc.m` (a), `evalplot.m` (b), and `bru1dcmds.m` (c). (a) Bifurcation lines as obtained from branch point (Turing–line) and Hopf point (Hopf and wave line) continuation. Compare [Uec19, Fig.7a] or [YDZE02]. (b) Spectrum of the linearization of (3.1) around U_s, $a = 0.95$ fixed. The dots on the real part show the admissible wave numbers on the subsequently used 1D domain. (c) Bifurcation diagram of Hopf and Turing branches; see also [Uec19, Fig.7] for the associated solution plots.

3.1.2 Bifurcations from the first Hopf branch

As a second extension of what is presented for (3.1) in [Uec19, §3.3] we give some results on bifurcations from the first Hopf branch `h1` in Fig. 6(c), associated to Floquet multipliers going through 1.(Of course, a critical multiplier also goes through 1 for a periodic orbit fold as, e.g., for the cGL in §2, but here we are interested in genuine bifurcations.) The used method is described in Appendix A.2, together with the case of period doubling bifurcations associated to Floquet multipliers going through -1.

Remark 3.1. Our methods are somewhat preliminary in the following sense:

(a) The localization of the branch points uses a simple bisection based on the change of ind(u_H), cf. (2.5), as a multiplier crosses the unit circle. Such bisections work well and robustly for bifurcations from steady branches (and can always be improved to high accuracy using the above extended systems for BPs and HPs), but the bisection for critical multipliers is often more difficult (i.e., less accurate) due to many multipliers γ_j close to the unit circle, and also often due to sensitive dependence of the γ_j on the continuation parameter λ, and, moreover, on the numerical time discretization fineness. Typically, some trial and error is needed here.

(b) The computation of predictors for branch switching is currently based on the classical monodromy matrix $\mathcal{M}$, see (A.17) and (A.18). As explained in, e.g., [Lus01], see also [Uec19] and §4, this may be unstable numerically, in particular for non–dissipative problems. The multiplier computations should then be based on the periodic QZ-Schur algorithm (**FA2** algorithm in `pde2path`, see also §4), but for the branch switching predictor computations this has not been implemented yet. $\rfloor$

Despite Remark 3.1, for 'nice' problems the branch–switching seems to be robust enough. Figure 6 shows some results from `bru1dcmds_b`, and Listing 12 shows the basic commands. In line 2 we reload a point from the first Hopf branch and set `p.hopf.bisec` which determines how many bisections are done to localize a BP after the detection of an index change. Here (and in other problems) `p.hopf.bisec`=5 seems a reasonable value. We then continue further and find the two BPs on `1dh1` indicated in Fig. 6 where the red and magenta branches bifurcate. The branch switching is done in lines 4-8. Typically this requires a rather large `ds` (all this of course depends on scaling), and often one step with a large residual tolerance is needed to get on the bifurcating branch (which indicates that the predictor is not very accurate). However, once on the bifurcating branch, `p.nc.tol` can and should be decreased again. The same strategy is used for the second (magenta) bifurcating branch.

```
%% reload point from 1dh1 and run with more bisectionss
p=loadp('1dh1','pt10','1dh1b'); p.hopf.bisec=8; huclean(p); p=cont(p,20);
%% bifurcations FROM 1dh1
ds=0.5; aux.sw=1; p=poswibra('1dh1b','bpt1','t1',ds,aux);
p.sw.bifcheck=0; p.hopf.fltol=1e-2; % increase fl-tol due to large amplitude
p.nc.tol=1e-3; p=cont(p,1); % do 1 step with large tol to get on bif.branch
p.nc.tol=1e-8; p=cont(p,19); % decrease tol and continue further
```

Listing 12: `brussel/bru1dcmds_b.m` (first 7 lines). The remainder computes the 2nd bifurcating branch and deals with plotting.

3.2 2D

We close the discussion of (3.1) with some comments on the continuation of solution branches in 2D. See [Uec19, Fig.10] for example results, where we consider (3.1) on $\Omega=(-\pi/2,\pi/2)\times(-\pi/8,\pi/8)$. Already on this rather small domain the linearization of (3.1) around U_s has many small real eigenvalues. Therefore, the Hopf eigenvalues (with imaginary parts near $\omega_1 = 1$) are impossible to detect by computing just a few eigenvalues close to 0, as illustrated in Fig. 8(a,b). Thus, the preparatory

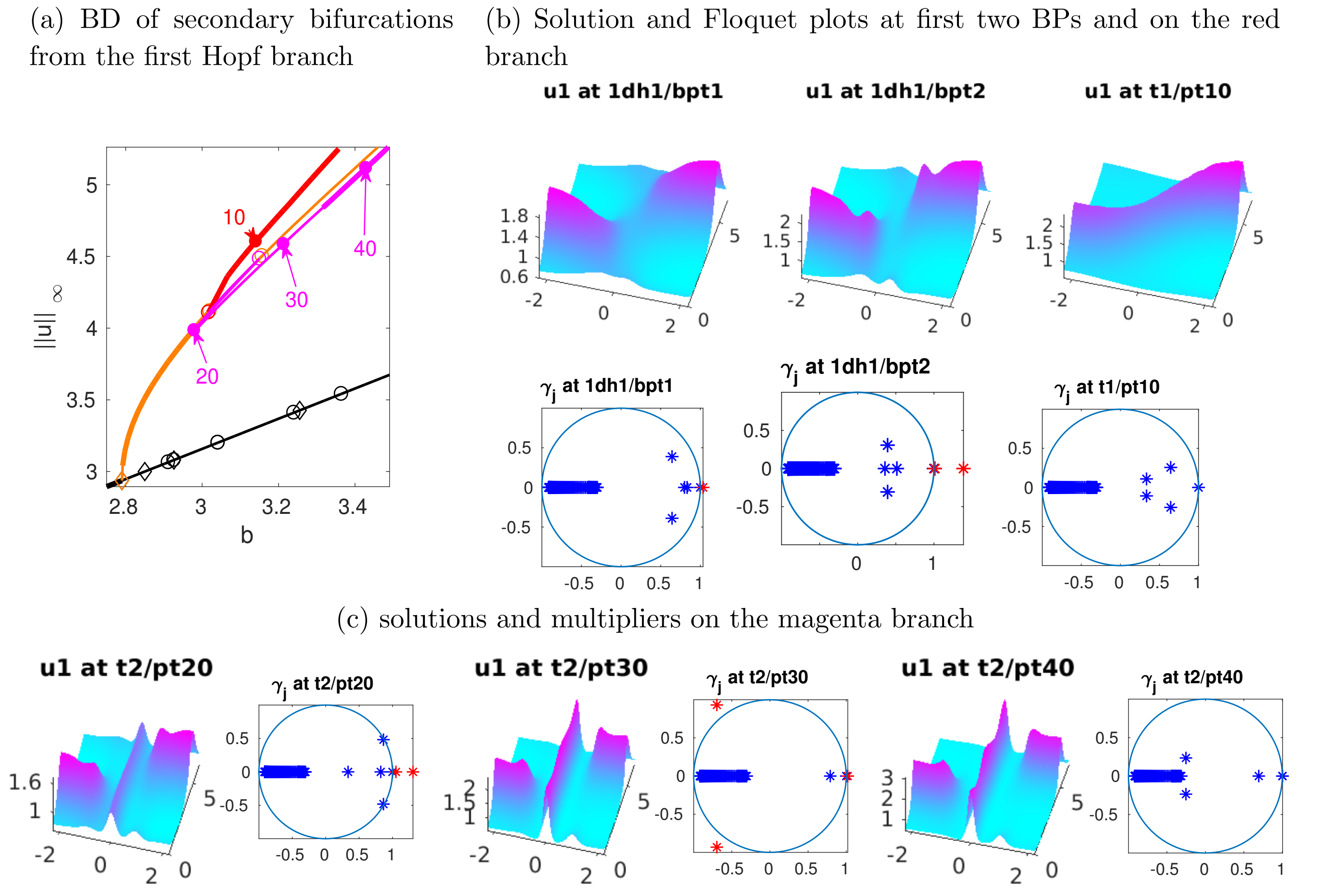


Figure 7: Results for (3.1) from `bru1dcmds_b.m`. (a) Bifurcation diagram, extenting Fig. 6(c) by the secondary pitchfork bifurcations from the primary Hopf branch (orange); first PD branch `t1` in red, second PD branch `t2` in magenta. (b,c) solution and Floquet plots. The magenta branch intermediately has index 3, before for increasing b first two unstable multipliers come back inside the unit circle via a Neimark-Sacker scenario, and then the last unstable γ goes through 1 and the orbit gains stability near $b = 3.3$.

step `initeig` already used in 1D in line 5 of Listing 10 becomes vital. This uses a Schur complement algorithm to compute a guess for the spectral shift ω_1 near which we expect Hopf eigenvalues during the continuation of a steady branch, see [Uec19, §2.1 and Fig.8] for illustration.

```
%% C1: init hom branch, with INITEIG, then use cont to find bifurcations
Du=0.01; Dv=0.1; Dw=1; c=1; d=1; a=0.95; b=2.75; lx=pi/2;
ndim=2; dir='hom2d'; p=[]; nx=60; par=[a b c d Du Dv Dw];
p=bruinit(p,lx,nx,par,ndim); p=setfn(p,dir); p.sw.spcalc=0; p.nc.mu2=0.5e-2;
5 p=initeig(p,4); p.nc.neig=[3, 3]; % init omv (compute guesses for eval shifts)
p.sw.bifcheck=2;  p=cont(p,30); % cont with just 3 evals near 0 and near om1
```

Listing 13: `brussel/bru2dcmds.m` (first Cell, i.e., initialization). The main issue is the preparatory step in line 6. This produces a (here quite accurate) guess 0.9375 for the candidate ω for imaginary parts at Hopf bifurcations, which, together with $\omega_0 = 0$, is put into `p.nc.eigref`. The remainder of `brucmds2.m` then continues the homogeneous branch and some bifurcating Hopf and Turing branches, including secondary bifurcations from the Turing branch to 'spotted' Hopf branches. Here, an adaptive spatial mesh refinement is helpful to increase accuracy.

```
function [p,idx]=e2rsbru(p,u)  % elements2refine selector as in pdejmps
E=zeros(1,p.pdeo.grid.nElements); par=u(p.nu+1:end); f=nodalf(p,u); a=0;
for i=1:1 % loop over the three components
 ci=par(4+i); fi=f((i-1)*p.np+1:i*p.np); ui=u((i-1)*p.np+1:i*p.np);
```

(a) $n_{\rm eig} = 200$ (b) $n_{\rm eig} = 300$ (c) $|g|$ from [Uec19, (2.11)] (d) $n_{\rm eig} = (3, 3)$ with $\omega_1 = 0.9375$

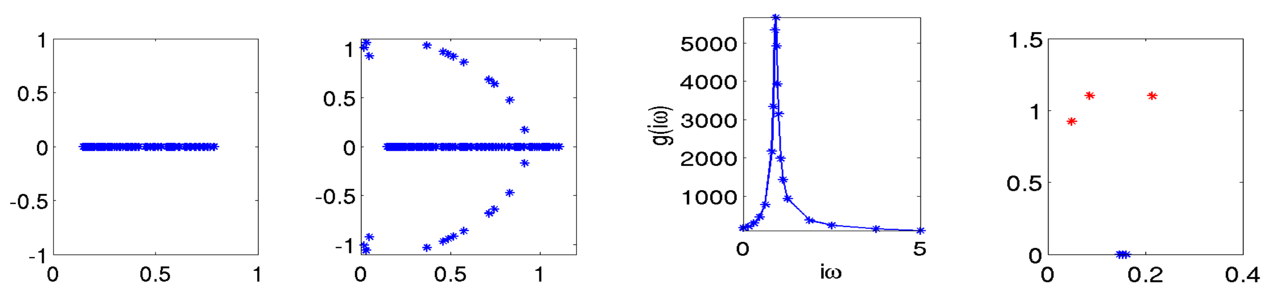


Figure 8: (a,b) $n_{\rm eig}$ smallest eigenvalues of the linearization of (3.1) around U_s at $b = 2.75$, remaining parameters from (3.2); **HD1** with $n_{\rm eig} = 200$ will not detect any Hopf points. (c) (**??**) yields a guess $\omega_1 = 0.9375$ for the ω value at Hopf bifurcation, and then **HD2** with $n_{\rm eig} = (3, 3)$ 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.

```
 E=E+p.pdeo.errorInd(ui,ci,a,fi); % sum up componentwise error-est.
end
p.sol.err=max(max(E));
idx=p.pdeo.selectElements2Refine(E,p.nc.sig); % select triangles to refine
```

Listing 14: `brussel/e2rsbru.m`. For a general discussion of error estimators in the `OOPDE` setting we refer to [RU19]. The only difference is that here we have a 3 component system, and thus we sum up the element wise errors over the components.

As indicated in the caption of Listing 13, at the start of the (1D and 2D) Turing branches we do some adaptive mesh–refinement. The used error estimator is given in Listing 14. The 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 A canonical system from optimal control: Demo `pollution`

In [Uec16, GU17], `pde2path` has been used to study infinite time horizon distributed optimal control (OC) problems, see also [dWU19] for a tutorial on OC computations with `pde2path`. As an example for such problems with Hopf bifurcations[5] we consider, following [Wir00], a model in which the states $v_1 = v_1(t, x)$ and $v_2 = v_2(t, x)$ are the emissions of some firms and the pollution stock, and the control $k = k(t, x)$ models the abatement policy of the firms. The objective is to maximize

$$J(v_0(\cdot), k(\cdot, \cdot)) := \int_0^\infty \mathrm{e}^{-\rho t} J_{ca}(v(t), k(t))\,\mathrm{d}t, \tag{4.1a}$$

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 local current value $J_c(v, k) = pv_1 - \beta v_2 - C(k)$, $C(k) = k + \frac{1}{2\gamma}k^2$, where $\rho > 0$ is the discount rate. Using Pontryagin's Maximum Principle, the so called canonical system for the states v and co-states (or Lagrange multipliers or shadow prices) λ can be formally derived as a first order necessary

[5]which so far could not be found in the systems studied in [Uec16, GU17]

optimality condition, using the intertemporal transversality condition

$$\lim_{t\to\infty} \mathrm{e}^{-\rho t} \int_\Omega \langle v, \lambda \rangle \, \mathrm{d}x = 0. \tag{4.2}$$

The canonical system reads

$$\partial_t v = D\Delta v + f_1(v,k), \quad v|_{t=0} = v_0, \tag{4.3a}$$
$$\partial_t \lambda = -D\Delta\lambda + f_2(v,k), \tag{4.3b}$$

where $f_1(v,k) = (-k, v_1 - \alpha(v_2))^T$, $f_2(v,k) = (\rho\lambda_1 - p - \lambda_2, (\rho + \alpha'(v_2))\lambda_2 + \beta)^T$, $\partial_{\mathbf{n}}\lambda = 0$ on $\partial\Omega$, and where the control k is given by

$$k = k(\lambda_1) = -(1+\lambda_1)/\gamma. \tag{4.3c}$$

For convenience we set $u(t,\cdot) := (v(t,\cdot), \lambda(t,\cdot)) : \Omega \to \mathbb{R}^4$, and write (4.3) as

$$\partial_t u = -G(u) := \mathcal{D}\Delta u + f(u), \tag{4.4}$$

where $\mathcal{D}$ =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 $\partial\Omega$ we have the initial condition $v|_{t=0} = v_0$ (only) for the states. A solution u of the canonical system (4.4) is called a *canonical path*, and a steady state of (4.4) (which automatically fulfills (4.2)) is called a *canonical steady state (CSS).* Due to the backward diffusion in λ, and since we only have initial data for half the variables, (4.4) is *not* well posed as an initial value problem. Thus, one method for OC problems of type (4.1) is to first study CSS, and then canonical paths connecting some initial states to some CSS u^*. This requires the so-called saddle-point property for u^*, and if this holds, then canonical paths to u^* can often be obtained from a continuation process in the initial states, see [dWU19].

A natural next step is to search for time–periodic solutions u_H of canonical systems, which obviously also fulfill (4.2). The natural generalization of the saddle point property is that

$$d(u_H) := \mathrm{ind}(u_H) - \frac{n_u}{2} = 0, \tag{4.5}$$

i.e., that exactly half of the Floquet multipliers are in the unit circle. In the (low–dimensional) ODE case, there then exist methods to compute connecting orbits to (saddle type) 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. A future aim is to extend these methods to periodic orbits of PDE OC systems.

However, in [Uec19, §3.4] we only illustrate that Hopf orbits can appear as candidates for optimal solutions in OC problems of the form (4.1), and that the computation of Floquet multipliers via the periodic Schur decomposition `floqps` can yield reasonable results, even when computation via `floq` completely fails.

For all parameter values, (4.4) has an explicit spatially homogeneous CSS, see [Uec19], and by a suitable choice of parameters we obtain Hopf bifurcations to spatially homogeneous and spatially patterned time periodic orbits. Concerning the implementation, Table 3 gives an overview of the involved scripts and functions. Since we again use the `OOPDE` setting, and since we restrict to 1D, although (4.4) is a four component system, much of this is very similar to the `cgl` demo in 1D, with the exceptions that: (a) we also need to implement the objective value and other OC related features; (b) similar to `brussel` it is useful to prepare the detection of HBPs via `initeig`; (c) we need to use `flcheck=2` throughout. Thus, in Listings 15-17 we comment on these points, and for plots illustrating the results of running `pollcmds.m` refer to [Uec19, §3.4].

Table 3: Main scripts and functions in `hopfdemos/pollution`.

script/function	purpose,remarks
pollcmds	main script
p=pollinit(p,lx,nx,par)	init function
p=oosetfemops(p)	set FEM matrices (stiffness K and mass M)
r=pollsG(p,u)	encodes G from (4.4); we avoid implementing the Jacobian here and instead use `p.sw.jac=1`
f=nodalf(p,u)	nonlinearity, called in `sG`.
jc=polljcf(p,u)	the (current value) objective function

```
function p=pollinit(p,lx,nx,par) % init-routine for pollution demo
p=stanparam(p); p.nc.neq=4; p.sw.jac=0; % numerical Jac
p.fuha.sG=@pollsG; p.fuha.jcf=@polljcf; % rhs, objective value,
p.fuha.outfu=@pollbra; % customized output (including objective function(s))
```

Listing 15: `pollution/pollinit.m` (first 4 lines). Additional to the rhs, in line 3 we set a function handle to the objective value, as usual for OC problems (see [dWU19]). Similarly, in line 4 we set `p.fuha.outfu` to a customized branch output, which combines features from the standard Hopf output `hobra` and the standard OC output `ocbra`. We do not set `p.fuha.sGjac` since for convenience here we use numerical Jacobians (`p.sw.jac=0` in line 1). The remainder of `pollinit.m` is as usual.

```
function jc=polljcf(p,u) % current value for pollution
par=u(p.nu+1:end); pr=par(2); vp=par(3); ga=par(5);
y=u(1:p.np); z=u(p.np+1:2*p.np);l1=u(2*p.np+1:3*p.np); % extract soln-components
k=-(1+l1)/ga; c=k+ga*k.^2/2; jc=pr*y-vp*z-c; % compute k, then J
```

Listing 16: `pollution/polljcf.m`, function to compute the current objective value. Called in `pollbra` to put the value on the branch (for plotting and other post-processing).

```
%% script for Hopf bif in pollution model Wirl2000, here with diffusion
close all; keep pphome
%% C1: init and continue trivial branch
p=[]; lx=pi/2; nx=40; par=[0.5 1 0.2 0 300]; % [del, pr, beta, a, ga];
p=pollinit(p,lx,nx,par); p=setfn(p,'FSS'); screenlayout(p); p.file.smod=2;
p=initwn(p,2,1); p=initeig(p); p.nc.neig=[5 5]; % find guess for omega_1
p.sw.bifcheck=2; p.sw.verb=2; p.nc.mu2=1e-3; % accuracy of Hopf detection
p.nc.ilam=1; p.sol.ds=0.01; p.nc.dsmax=0.01; p=cont(p,20); % cont of FSS
%% C2: cont of Hopf branches
para=4; ds=0.5; dsmax=1; xi=1e-2; figure(2); clf; aux=[]; aux.tl=25;
for j=1:2
 switch j
     case 1; p=hoswibra('FSS','hpt1',ds,para,'h1',aux); nsteps=15;
     case 2; p=hoswibra('FSS','hpt2',ds,para,'h2',aux); nsteps=25;
 end
p.hopf.xi=xi; p.hopf.jac=1; p.nc.dsmax=dsmax; p.hopf.y0dsw=0;
p.file.smod=1; p.hopf.flcheck=2; % use floqps for multipliers
p.usrlam=[0.5 0.6 0.7]; tic; p=cont(p,nsteps);  toc
```

```
end
```

Listing 17: `pollution/pollcmds.m` (first 19 lines). In cell 1 we use `initeig` to generate a guess for $\mathrm{i}\omega_1$ (the Hopf wave number), and set `neig` to compute 5 eigenvalues near 0 and near ω_1. For the computation of multipliers here we need to use `floqps`, see line 17. The remainder of `pollcmds` deals with plotting.

5 Hopf bifurcation with symmetries

If the PDE (1.1) has (continuous) symmetries, then already for the reliable continuation of steady states it is often necessary to augment (1.1) by n_Q suitable phase conditions, in the form

$$Q(u,\lambda,w)=0\in\mathbb{R}^{n_Q} \tag{5.1}$$

where $w\in\mathbb{R}^{n_Q}$ stands for the required n_Q additional active parameters, see [RU17] for a review. For instance, if (1.1) is spatially homogeneous and we consider periodic BC, then we have a translational invariance, and (in 1D) typically augment (1.1) by the phase condition

$$\langle\partial_x u^*,u\rangle=0\in\mathbb{R}, \tag{5.2}$$

where (for scalar u,v) $\langle u,v\rangle=\int_\Omega uv\,\mathrm{d}x$, and where u^* is either a fixed reference profile or the solution from the previous continuation step. We thus have $n_Q=1$ additional equations, and consequently must free 1 additional parameter.

Similarly, we must add phase conditions to the computation of Hopf orbits (additional to the phase condition (A.6) fixing the translational invariance in t). This is in general not straightforward, since (5.1), with (5.2) as an example, is not of the form $\partial_t u=Q(u,\lambda)$ and thus cannot simply be appended to (1.1). Instead, the steady phase conditions (5.1) must be suitably modified and explicitly appended to the Hopf system, see (A.9). Examples for the case (5.2) have been discussed in [RU17, §4], namely the cases of modulated fronts, and of breathers.

Here we give two more examples, and extend the breather example to compute period doubling bifurcations. The first example deals with Hopf orbits in a reaction diffusion system with mass conservation, and the second with Hopf orbits in the Kuramoto-Sivashinsky (KS) equation, where we need two phase conditions, one for mass conservation and one to fix the translational invariance. For both problems we restrict to 1D; like, e.g., the cGL equation, they both can immediately be transferred to 2D (where for the KS equation we need a third phase condition $q_3(u)=\langle\partial_y u^*,u\rangle=0$, cf. (5.9c)), but the solution spaces and bifurcations then quickly become "too rich", such that – as often – 2D setups only make sense if there are specific questions to be asked.

5.1 Mass conservation: Demo `mass-cons`

As a toy problem for mass conservation in a reaction diffusion system we consider

$$\partial_t u_1=\Delta u_1+d_2\Delta u_2+f(u_1,u_2),\quad \partial_t u_2=\Delta u_2-f(u_1,u_2),\ \text{in }\Omega, \tag{5.3}$$

$f(u_1,u_2)=\alpha u_1-u_1^3+\beta u_1u_2$, with parameters $d_2,\alpha,\beta\in\mathbb{R}$, and homogeneous Neumann BC. Then $m:=\frac{1}{|\Omega|}\int u+v\,\mathrm{d}x$ is conserved since $\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega(u+v)\,\mathrm{d}x=\int_{\partial\Omega}\partial_n u_1+(1+d_2)\partial_n u_2\,\mathrm{d}S=0$. Given a steady state (u,v) for some fixed α,β, this always comes in a continuous family parameterized by the "hidden" parameter m. Thus, to study steady states and their bifurcations we use the mass constraint

$$Q(u,\lambda) := \frac{1}{|\Omega|}\int u + v\,\mathrm{d}x - m = 0, \tag{5.4}$$

where as usual λ stands for the vector of all parameters. Given this additional equation, we have the differential-algebraic system

$$M\dot{u} = -G(u,\lambda), \qquad Q(u,\lambda) = 0, \tag{5.5}$$

and to compute solution branches we need 2 parameters, which we choose as α, β. If we restrict to $m = 0$, then we have two explicit branches of homogeneous solutions, namely $u_2 = -u_1$ and $u_1 = -\frac{\beta}{2} \pm \sqrt{\frac{\beta^2}{4} + \alpha}$. We choose the initial point $(\alpha,\beta)=(1,1)$, $u_1 = -1/2 - \sqrt{5/4}$, $u_2 = -u_1$ and continue in α.

As a Hopf version of (5.4) we use

$$Q_H(u(\cdot,\cdot)) := \sum_{i=1}^{m}\left(\int_\Omega (u_1(t_i,x) + u_2(t_i,x))\,\mathrm{d}x - m\right) \stackrel{!}{=} 0, \tag{5.6}$$

see Listings 19 and 20. In (5.6) i.e., we require the average (in t) mass to be conserved. Theoretically it would be sufficient to require $\int_\Omega (u_1(t_0,x) + u_2(t_0,x))\,\mathrm{d}x - m = 0$, but it turns out that (5.6) is more robust numerically, and that also with (5.6) we have $\left|\int_\Omega (u_1(t_i,x) + u_2(t_i,x))\,\mathrm{d}x - m\right| \le \texttt{tol}$ for all i, i.e., pointwise in t.

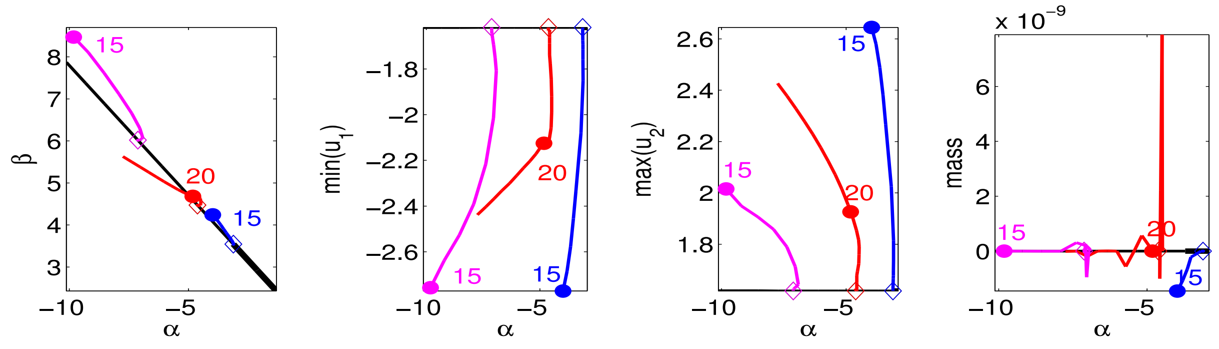
The implementation of (5.5) is rather straightforward, see Table 4 for an overview, and Listings 18–20. We fix $d_2 = 10$ and restrict to 1D, namely $\Omega = (-\pi,\pi)$. Figure 9(b) shows a basic bifurcation diagram, with various quantities as functions of α. The continuation of (5.5) in α with fixed $m = 0$ yields that the homogeneous solution u stays fixed, i.e., $u_1 = -1/2 - \sqrt{5/4}$, $u_2 = -u_1$ for all α, and that only β is adjusted, see the black lines in (b). (c) shows a number of Hopf orbits, where on each orbit we have $|Q(u(t,\cdot)| < 10^{-8}$ (see also the last plot in (a) for the average Q_H), where the tolerance for the Hopf orbits is 10^{-6}. These Hopf orbits are all unstable according to the associated Floquet multipliers, see also Fig. 10(a), and thus it is interesting to see the evolution of solutions starting on a Hopf orbit (with the numerical error acting as a perturbation of the true point on a Hopf orbit). In Fig. 10(b) we exemplarily show this for the case of $u(0)$ from `h3/pt15`; here, as in all other cases we considered, the time evolution converges to another stable spatially homogeneous steady state. From this we may again start continuation in, e.g., α and β, and find that this branch again typically shows some Hopf bifurcations.

Table 4: Scripts and functions in `hopfdemos/mass-cons`.

script/function	purpose,remarks
cmds1d	main script
mcinit, oosetfemops, sG, sGjac, nodalf	initialization, FEMops, rhs, Jac., and nonlinearity, as usual.
qf, qfjac	the phase condition (5.4), and its Jacobian.
qfh, qfhjac	the Hopf version (5.6) of (5.4), and its Jacobian.

```
function q=qf(p,u) % mass constraint int u1+u2 dx=0
M=p.mat.M(1:p.np,1:p.np); par=u(p.nu+1:end); m=par(4);
```

(a) BDs, parameter $\beta, \min(u_1), \max(u_2$ and mass as functions of α.



(c) Selected solution plots (both components for `h1/pt15`)

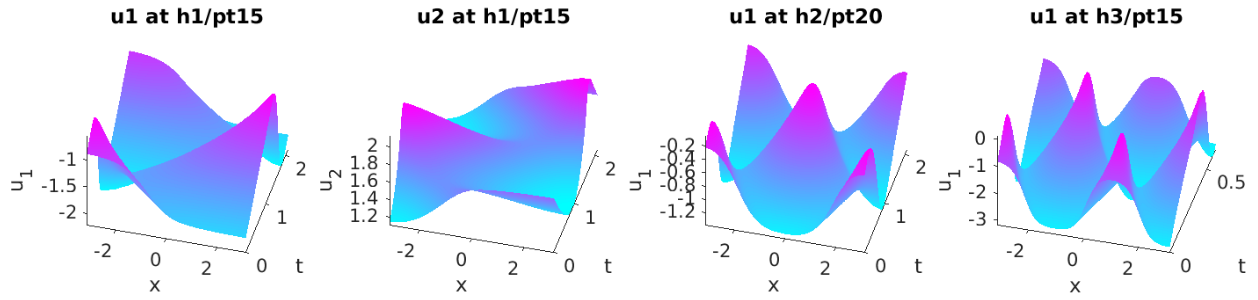


Figure 9: Continuation in α for (5.5) with $m = 0$. (a) Branch data on the homogeneous branch (black) and on three Hopf branches h1 (blue), h2 (red), and h3 (magenta). (b) Example solution plots.

```
u1=u(1:p.np); u2=u(p.np+1:2*p.np); q=sum(M*(u1+u2))/p.vol-m;
```

Listing 18: `mass-cons/qf.m`; mass constraint for steady state computations.

```
function q=qfh(p,y) % aux eqns in Hopf, here: mass constraint
M=p.mat.M(1:p.np,1:p.np); par=p.u(p.nu+1:end); m=par(4); q=0;
for i=1:p.hopf.tl; % sum up masses, i.e., conserve m on average
  u1=y(1:p.np,i); u2=y(p.np+1:2*p.np,i); q=q+sum(M*(u1+u2))/p.vol-m;
end
```

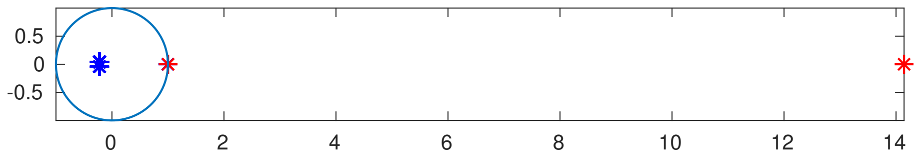
Listing 19: `mass-cons/qfh.m`; Hopf setting of mass constraint. The summing up (in t) of the masses turns out to be more robust, with the mass-constraint actually fulfilled pointwise (in t).

```
function qjac=qfhjac(p,y) % u-derivatives of qfh
qjac=zeros(1,p.hopf.tl*p.nu); j=p.mat.M(1:p.np,1:p.np)*ones(p.np,1)/p.vol;
for i=1:p.hopf.tl;  % same derivative at each time slice
    qjac((i-1)*p.nu+1:i*p.nu)=([j; j])';
end
```

Listing 20: `mass-cons/qfhder.m`, u–derivatives of Q_H, cf. last line of (A.10), where the parameter derivatives are done automatically via finite differences.

```
%% C1: init, and continuation of hom branch
ndim=1; dir='hom1d'; p=[]; lx=pi; nx=100; % domain size and spat.resolution
par=[1; 10; 0; 0; 1; 1];  % d1 d2 d3 m a1 b1
p=mcinit(p,lx,nx,par,ndim); p=setfn(p,dir); % initialization
p.nc.nq=1; p.fuha.qf=@qf;  % 1 steady constraint (mass), and its func.handle
p.sw.qjac=1; p.fuha.qfder=@qfjac; % use analytical jac for q, and func.handle
p.nc.xiq=0.1; p.nc.ilam=[5 6]; % weight of constr. in arclength, active vars
p=cont(p,20);  % run the continuation
%% C2: hopf with constraints, passed to hoswibra via aux vars in aux
para=4; ds=0.005; aux=[]; aux.dlam=0; aux.nqnew=0; aux.tl=50;
```

(a) Floquet multipliers for `h1/pt15`



(b) Time evolution of small perturbation of u from `h1/pt15` at $t = 0$, left u_1, right u_2

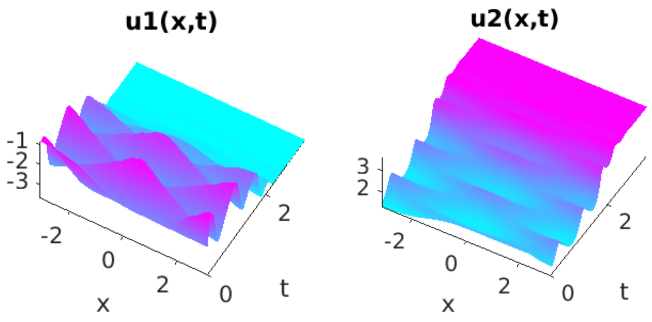


Figure 10: (a) Instability of `h1/pt15` as seen in its Floquet multipliers. (b) time integration, with convergence to another spatially homogeneous steady state.

```
aux.xif=50; aux.pcfac=10;  % weight factors, see hostanparam
aux.nqh=1; aux.qfh=@qfh; aux.qfhder=@qfhjac; % func.handles to hopf contraints
for i=1:3; % continue for 15 steps, first three with large tol
  p=hoswibra('hom1d',['hpt' mat2str(i)],ds,para,['h' mat2str(i)],aux);
  p.nc.ilam=5; p.hopf.ilam=6; p.sw.verb=0; p.hopf.sec=1; p.nc.dsmax=0.5;
  p.file.smod=5; p.nc.tol=1e-2; p=cont(p,3); p.nc.tol=1e-4; p=cont(p,12);
end
%% C3: time integrate from some point on Hopf orbit, preparations
p=loadp('h2','pt15'); p.u(1:p.nu)=p.hopf.y(1:p.nu,1); % load Hopf point, and
```

```
%% C6: continue from result of tint; again hopf for decreasing alpha
plotsol(p); p.sol.restart=1; p.sol.ds=-0.1; p.sw.para=2; % reset settings to
% steady case, in particular restore scalar stiffness matrix (-Laplacian)
p.mat.K=Ks; p=rmfield(p,'hopf'); p=setfn(p,'hom1d2'); p=resetc(p);
p.nc.nq=1; p.nc.ilam=[5 6]; p.fuha.headfu=@stanheadfu; p.fuha.ufu=@stanufu;
p=cont(p,10);
```

Listing 21: `mass-cons/cmds1d.m` (with some omissions) C1 continues the homogeneous branch, giving a number of Hopf bifurcations; here $u_1 = -(1+\sqrt{5})/2$ and $u_2 = -u_1$ stay fixed, and only β varies with α. In C2 we follow the first three Hopf branches, where we replace the stationary Hopf constraint in `qf` by the Hopf version `qfh`, see Fig. 9 for bifurcation diagrams and example Hopf solutions. All Hopf branches turn out to be unstable (from the Floquet multipliers), and thus in C3-5 we exemplarily look into the time evolution from the first point ($t = 0$) on the Hopf orbit `h1/pt15`. This converges to a (stable) homogeneous solution again, but at larger amplitude. Finally in C6 we use this as a starting point for continuation in α, and again find a number of Hopf bifurcations for decreasing α.

5.2 Mass and phase constraints: Demos `kspbc4` and `kspbc2`

The Kuramoto-Sivashinsky (KS) equation [KT76, Siv77] is a canonical and much studies model for long–wave instabilities in dissipative systems, for instance in laminar flame propagation, or for surface instabilities of thin liquid films. Here we consider the KS equation in the form

$$\partial_t u = -\alpha\partial_x^4 u - \partial_x^2 u - \frac{1}{2}\partial_x(u^2), \tag{5.7}$$

with parameter $\alpha > 0$, on the 1D domain $x \in (-2, 2)$ with periodic BC. (5.7) is thus translationally invariant, and has the boost invariance $u(x,t) \mapsto u(x-ct)+c$, and we need two phase conditions,

$$\frac{1}{|\Omega|}\int_\Omega u\,\mathrm{d}x = m, \text{ fixing the mass } m, \tag{5.8a}$$

$$\langle \partial_x u^*, u - u^* \rangle = 0, \text{ fixing the translational invariance.} \tag{5.8b}$$

Here fixing $m = 0$, (5.7) shows bifurcations from the trivial solution $u \equiv 0$ to stationary spatially periodic solutions at $\alpha_k = \left(\frac{2}{k\pi}\right)^2$, $k \in \mathbb{N}$. Next, for decreasing α we obtain secondary Hopf bifurcations from some branches of steady patterns, and for $\alpha \to 0$ the dynamics become more and more complicated, making (5.7) a model for turbulence. In [BvVF17], a fairly complete bifurcation diagram (with α in the range 0.025 to 0.4) has been obtained for (5.7) on $\Omega = (0,2)$ with *Dirichlet* BC, i.e., $u(0,t) = u(2,t) = \partial_x^2 u(0,t) = \partial_x^2 u(2,t) = 0$, where in particular many bifurcations have been explained analytically as hidden symmetries by extending solutions antisymmetrically to the domain $(-2,2)$ with periodic BC.

Here we directly study (5.7) in this setting, giving us the opportunity to also explain how to setup 4th order equations and periodic BC in `pde2path`. For the latter we only need to call `p=box2per(p,1)`, which generates matrices `fill` and `drop` which are used to transform the FEM matrices such as M and K to the periodic setting, see [DU17]. In order to implement 4th order equations there are basically two options:

(i) Since $-\partial_x^2 u = M^{-1}Ku$ in the FEM sense, (5.7) can be written in the `pde2path` FEM setting as $M\partial_t u = -\alpha K M^{-1} K u + Ku - \frac{1}{2}K_x(u^2)$. For pBC, K, M commute, and thus we can multiply by M to obtain $M^2\partial_t u = -\alpha K^2 u + MKu - \frac{1}{2}MK_x(u^2)$. Then letting $M_0{=}M$ and redefining $M{=}M^2$ we obtain $M\partial_t u{=} - \alpha K^2 u + M_0 K u - M_0 K_x(u^2)$. To incorporate the phase conditions (5.8) we introduce the parameters s for phase-conservation and ε for mass conservation, and thus ultimately consider the system

$$M\dot{u} = -\alpha K^2 u + M_0 K u - \frac{1}{2}M_0 K_x(u^2) + sK_x u + \varepsilon, \tag{5.9a}$$

$$0 = q_1(u) := \frac{1}{|\Omega|}\sum_{i=1}^{n_u}(M_0 u)_i - m, \tag{5.9b}$$

$$0 = q_2(u) := \langle \partial_x u^*, u - u^* \rangle, \tag{5.9c}$$

where $\frac{1}{|\Omega|}\sum_{i=1}^{n_u}(M_0 u)_i$ is the (Riemann sum) approximation of $\frac{1}{|\Omega|}\int_\Omega u\,\mathrm{d}x$, and u^* is a suitable reference profile. This set up is implemented in `kspbc4`, see below.

(ii) Alternatively we can rewrite the 4th order equation as a 2 component 2nd order system, for instance for (5.7) in the form

$$\partial_t u = -\alpha\partial_x^2 v - \partial_x^2 u - \frac{1}{2}\partial_x(u^2), \qquad 0 = -\partial_x^2 u + v. \tag{5.10}$$

By exploiting the mass matrix on the lhs of (1.3), (5.10) can be straightforwardly implemented in pde2path in the form

$$\mathcal{M}\dot{U} = -G(U), \quad U = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \ \mathcal{M} = \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix}, \ G(U) = -\begin{pmatrix} K & \alpha K \\ K & M \end{pmatrix} U + \begin{pmatrix} \frac{1}{2}K_x(u_1^2) \\ 0 \end{pmatrix}, \quad (5.11)$$

where M, K and K_x are the scalar mass, stiffness and advection matrices. Importantly, the spectral picture and time evolution for (5.11) are still fully equivalent to (5.7). Adding phase conditions like (5.9b,c), this is implemented in `kspbc2`, and yields the same results as the `kspbc4` set up, except for small differences wrt to Floquet multipliers, which in any case are somewhat delicate for constrained Hopf orbits, see Remark A.1.

The implementation of (5.9) is rather straightforward. See Table 5 for an overview, Listings 5.2–24 for pertinent sections from `oosetfemops, cmds1d, sG, qf` and `qfh`, while for `cmds2.m` and Jacobians/derivatives of `sG, qf` and `qfh` we refer to the m-files `sGjac, qjac` and `qfhjac`, respectively.

Table 5: Scripts and functions in `hopfdemos/kspbc4`.

script/function	purpose,remarks
cmds1	main script, steady state branches, and associated Hopf bifurcations of standing waves
cmds2	script for one traveling wave branch, and associated Hopf bifurcations of modulated traveling waves
ksinit, oosetfemops	initialization and FEMops; this is somewhat different from the other examples. `ksinit` also contains the call `p=box2per(p,1)` to set up the periodic BC; `oosetfemops` contains calls of `filltrafo`, and specifically the redefinition of M as M^2.
sG, sGjac	rhs, Jacobian; again somewhat different from before due to 4th order derivatives.
qf, qfjac	the phase conditions (5.9b,c), and the derivatives
qfh, qfhjac	the Hopf version of `qf` and its derivative

Figure 11(a) shows a basic bifurcation diagram of steady states, including one branch of traveling waves, obtained from `cmds2.m`. As predicted, at α_k we find supercritical pitchforks of steady branches. The first one starts out stable, and looses stability in another supercritical pitchfork around $\alpha = 0.13$ to a traveling wave branch (brown), which then looses stability in a Hopf bifurcation, see Fig. 12. However, here we first focus on Hopf bifurcations from the 2nd and 3rd primary branches, which first gain stability at some rather large amplitude, then loose it again in Hopf bifurcations, with the solution profiles at the HBP in (c). (b) zooms into the BD at low α, including the 4th steady branch, and three Hopf branches, while (d) shows selected Hopf orbits.

These results all fully agree with those in [BvVF17] (by extending the solutions from [BvVF17] antisymmetrically), who however proceed further by also computing some (standing) Hopf branches bifurcating in pitchforks and period doublings from the above (standing) Hopf branches. Naturally, these bifurcations are also detected in `pde2path`, but already their localization requires some fine tuning, e.g., small stepsizes. Moreover, with the given discretizations the branch switching then still often fails. This will be further studied elsewhere, and instead we illustrate period doubling with a model with better scaling properties in §5.3. On the other hand, for our periodic BC on the larger domain we also have traveling waves and Hopf bifurcations to modulated traveling waves. Some examples for these are considered in `cmds2.m`, see Fig. 12.

```
%% C1: init and zero-branch
al=0.42; m=0; par=[al; m; 0; 0];  % m=mass, par(3)=eps, par(4)=s (speed)
p=[]; lx=2; nx=100;p=ksinit(p,nx,lx,par); p=setfn(p,'0'); % domain and discr
p.nc.mu1=10; p.nc.mu2=1; % large spacing of evals, be loose about localization
```

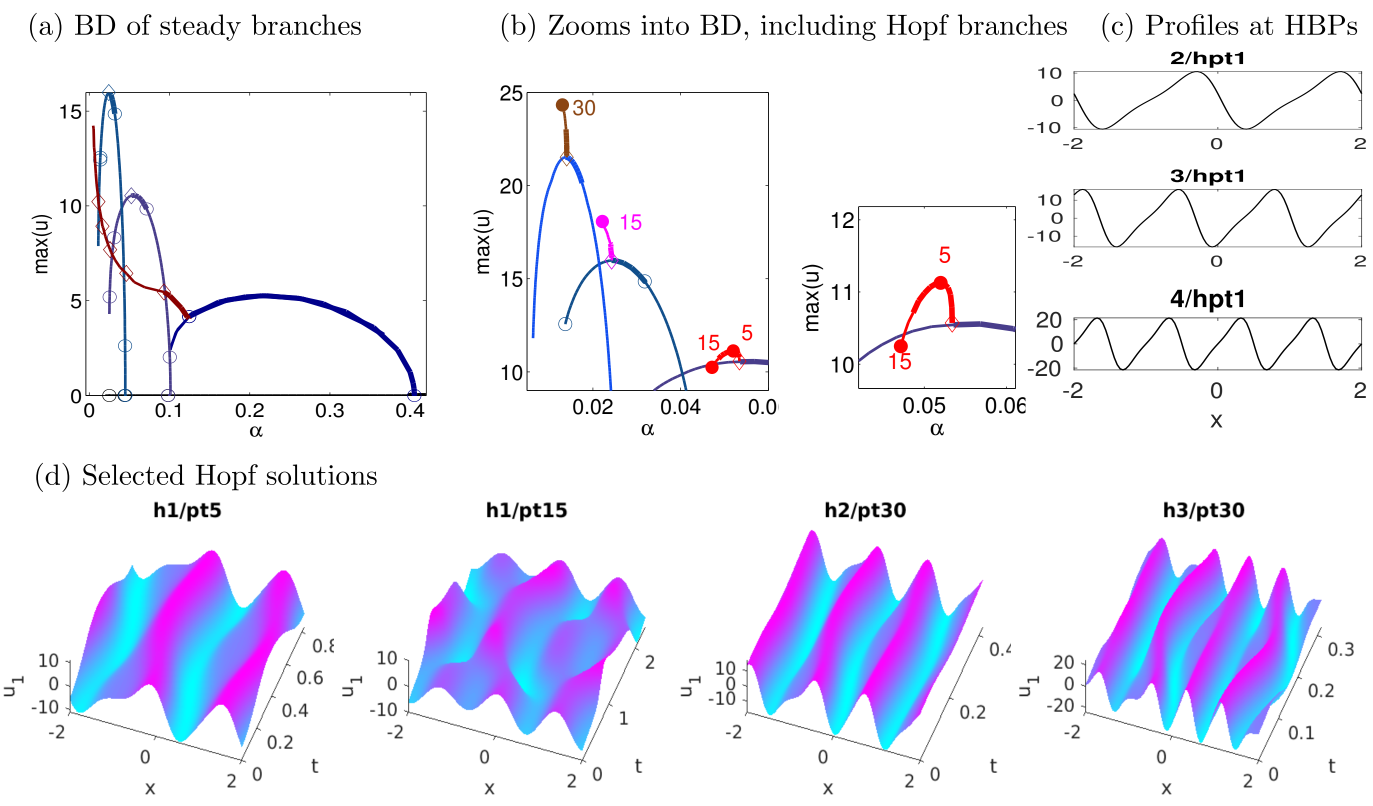


Figure 11: Results from `kspbc4/cmds1.m`. Bifurcation diagrams of steady solutions (except for the brown branch of traveling waves, see Fig. 12) (a), with zoom in (b), including the 4th steady branch and 3 Hopf branches h1 (red), h2 (magenta) and h3 (brown). For all these branches $m = \varepsilon = 0$ (numerically $\mathcal{O}(10^{-10})$, and except for the brown branch in (a) also $s = 0$. Profiles at the Hopf bifurcation points the steady branches in (c). In (d) we plot selected Hopf orbits and multiplier spectra. `h1` loses stability at $\alpha \approx 0.0486$ in a pitchfork (a multiplier becoming unstable at $\mu = 1$), and the largest multiplier of `h1/pt15` is $\mu_2 \approx 4000$. Also h2 is initially stable, but looses stability in a pitchfork at $\alpha \approx 0.024$, i.e., rather close to bifurcation, and a similar behaviour occurs on `h3`. See Fig. **??** for multiplier plots, and `cmds2.m` and Fig. 12 for further plots, for instance of solutions on the secondary brown branch in (a), and the Hopf bifurcations from this branch.

```
p.nc.ilam=[1 3]; p=cont(p,40); % initial steps
p.sol.ds=-0.001; p.nc.dsmax=0.001; % some more steps with smaller stepsize
p.nc.mu2=5; p=cont(p,20); p.file.smod=10; p=cont(p,10);
%% C2: compute branches of steady patterns
for i=1:4
    is=mat2str(i); p=swibra('0',['bpt' is],is,i*0.01);
    p.file.smod=20; p.sw.bifcheck=0; p=cont(p,5); % a few steps without PC
    p.u0x=p.mat.Kx*p.u(1:p.nu); % set profile for transl-invariance
    p.nc.nq=2; p.nc.ilam=[1 3 4]; p.tau=[p.tau; 0]; % now switch on PCs
    p.sw.bifcheck=2; p.nc.dsmax=0.2; p.nc.tol=1e-6; p=cont(p,30+i*60);
end
%% C3: 1st Hopf bifurcation
figure(2); clf; ds=0.1; clear aux; aux.dlam=0; aux.nqnew=0; aux.tl=30;
aux.xif=0.1; aux.y0dsw=2; % use PDE to set d/dt u_0 for phase-constr. (in t)
aux.nqh=2; aux.qfh=@qfh; aux.qfhder=@qfhjac; % func handles to hopf constraints
p=hoswibra('2','hpt1',ds,4,'h1',aux); p.nc.ilam=1; p.hopf.ilam=[3 4];
p.hopf.fltol=1e-2; p.hopf.nfloq=10; p.hopf.flcheck=2; p.sw.verb=0;
p.hopf.sec=1; p.nc.tol=1e-6; p.nc.dsmax=0.3; p.file.smod=5; p=cont(p,1);
p.hopf.flcheck=1; p=cont(p,14); % floqps fails for larger amplitudes,
% hence switch to floq: caution, only large multipliers seem correct
```

Listing 22: `kspbc4/cmds1.m`. Cell 1 deals with initialization and continuation of the trivial branch. Since the eigenvalues $\mu_k = -\alpha(k\pi/2)^4 + (k\pi/2)^2$ of the linearization around $u \equiv 0$ have a rather large spacing, in line 7 we set $\mu_{1,2}$ (see (A.2)) to rather large values. In Cell 2 we compute the first 4 branches of steady patterns. The phase condition $\langle \partial_x u^*, u\rangle = 0$ (2nd component of `qf`, see Listing 5.2) is only switched on after a few initial steps and then setting the reference profile $\partial_x u^* =$ `p.u0x`, because it only makes sense for u^* not spatially homogeneous. C3 computes the first Hopf branch `h1`, bifurcating from steady branch 2. We use a rather large Floquet tolerance `p.hopf.fltol`, see (2.5), because the Floquet computations do not remove the neutral directions, cf. Remark A.1. Moreover, for this problem `floqps` for the multiplier computations via periodic Schur decomposition sometimes fails (for unknown reasons), while `floq` (for unknown but maybe related reasons) seems somewhat unreliable for the small multipliers; the large multipliers (and hence the stability information) however always seem correct. The remainder of `kspbc4/cmds1.m` deals with the Hopf branches `h2` and `h3`, and with plotting.

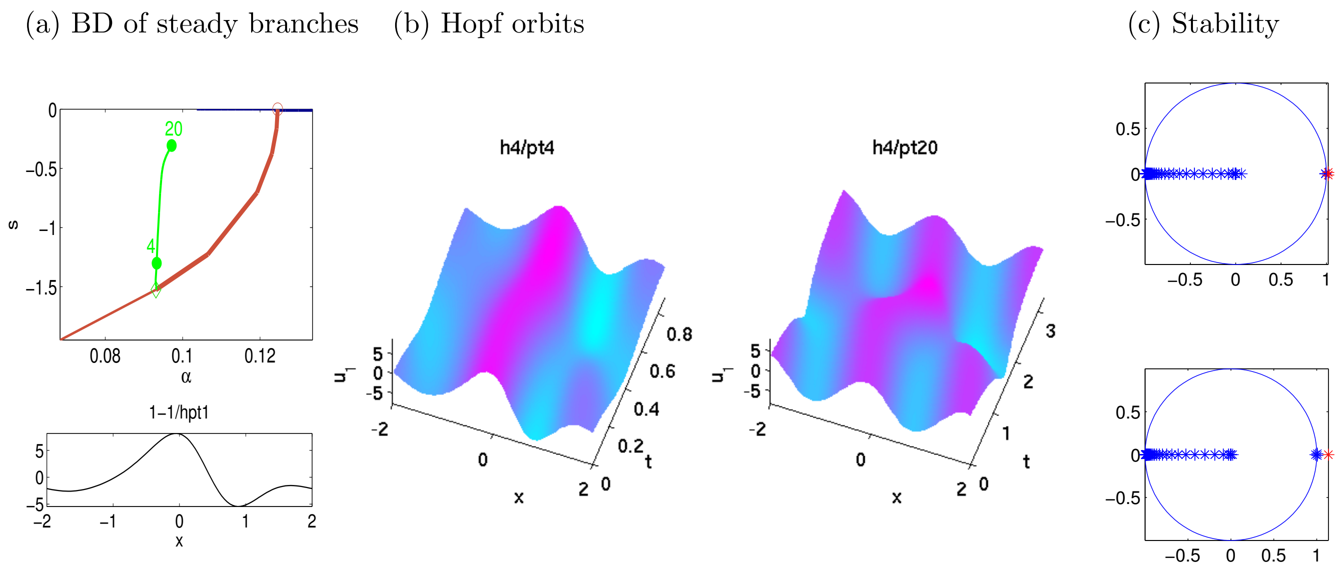


Figure 12: Results from `kspbc4/cmds2.m`. (a) Bifurcation diagram (s over α) of the traveling wave branch from Fig. 11(a), and of the first bifurcating modulated traveling wave branch (green), with the solution profile at bifurcation at the bottom. (b) shows two Hopf orbits on the green branch (in the frames moving with speeds s from (a), respectively), and (c) the associated Floquet multipliers. The bifurcation is subcritical, and the modulated traveling waves are (mildly) unstable.

```
function p=oosetfemops(p) % with filltrafo to transform to per.domain
gr=p.pdeo.grid;
[K,M,~]=p.pdeo.fem.assema(gr,1,1,1); Kx=convection(p.pdeo.fem,gr,1);
p.mat.K=filltrafo(p,K); M=filltrafo(p,M); p.mat.Kx=filltrafo(p,Kx);
p.mat.M0=M; p.mat.M=M^2; % save M as M0 and redefine M for the 4th order setup
```

```
function r=sG(p,u) % KS in 4th order formulation
K=p.mat.K; M0=p.mat.M0; Kx=p.mat.Kx; par=u(p.nu+1:end);
al=par(1); eps=par(3); s=par(4); u=u(1:p.nu); uxx=K*u;
r=al*K*uxx-M0*uxx+0.5*M0*(Kx*(u.^2))+s*M0*(Kx*u)+eps;
```

Listing 23: `kspbc4/oosetfemops.m` and `kspbc4/sG.m`. The mass matrix `p.mat.M` is redefined in `oosetfemops` to M^2, and the proper mass matrix is stored in `p.mat.M0`.

```
function q=qf(p,u) % mass (and phase) constraint for KS
par=u(p.nu+1:end); u=u(1:p.nu); % extract pars and u-vars
```

```
q=sum(p.mat.M0*u)/p.vol-par(2); % mass constraint
if p.nc.nq==2; % if active, then add phase constraint
  if isfield(p,'u0x'); u0x=p.u0x(1:p.nu); else u0x=p.mat.Kx*p.u(1:p.nu); end
  q=[q;u0x'*u];
end
```

```
function q=qfh(p,y) % aux eqns in Hopf, here: sum up shifts wrt u0
par=p.u(p.nu+1:end); m=par(2); n=p.nu;
q1=sum(p.mat.M0*y(1:n,1))/p.vol-m; % mass constraint (at initial slice)
tl=size(p.hopf.y,2); q2=0;  % phase constr, useful to define 'on average'
if isfield(p,'u0x'); u0x=p.u0x(1:p.nu); else u0x=p.mat.Kx*p.u(1:p.nu); end
for i=1:tl; u=y(1:n,i); q2=q2+u0x'*u; end
q=[q1;q2];
```

Listing 24: `kspbc4/qf.m` and `kspbc4/qfh.m`. The phase conditions for the steady and for the Hopf case.

5.3 Period doubling of a breather (demo `symtut/breathe`)

In [RU17, §4.2] we studied the RD system

$$\partial_t u = \partial_x^2 u + f(u,v), \quad \partial_t v = D\partial_x^2 v + g(u,v), \tag{5.12}$$

with homogeneous Neumann BC, $f(u,v) = u(u-\alpha)(\beta-u)-v$, $g(u,v) = \delta(u-\gamma v)$, with $\alpha,\beta,\gamma > 0$, and $0<\delta\ll 1$. For suitable parameters, this model has standing and traveling pulses (and traveling fronts), and for $\delta\to 0$ we find a Hopf bifurcation to breathers, see the bottom row of Fig. 13 for examples. In [RU17] this served as an example for the usefulness of constraints, here regarding the approximate translational invariance for the case of narrow breathers. It turns out that the 'primary breather branch' (red in (a)) looses stability in a period doubling bifurcation, yielding the magenta branch in (a), which starts out stable, and then looses stability in a torus bifurcation, see (b).

```
%% period doubling from breather
huclean(p); ds=0.5; p=poswibra('h1','bpt1','pd1',ds); p.nc.tol=0.5;
p.hopf.nqh=0; % switch off average speed constraints (allows better stepsizes)
p.sw.bifcheck=0; p.hopf.flcheck=1; p.nc.dsmin=1e-3; p.sw.verb=0;
p=cont(p,2); p.nc.tol=1e-2; p=cont(p,5); p.nc.tol=1e-6; p=cont(p,23);
```

Listing 25: `symtut/breathe/cmds1.m`, commands for the period doubling branch.

Figure 13 is computed in the script `symtut/breathe/cmds1.m`, see Listing 25 for the relevant code snippet. For more background on the demo `symtut/breathe` we refer to [RU17], and here only remark that:

- A good localization of the PD bifurcation point on the breather is crucial; here $\gamma_{\text{crit}} \approx -1.1$ obtained using `p.hopf.bisec=5` is good enough if we allow large residuals at startup of the magenta branch. After 5 steps we set `p.nc.tol=1e-6` again.
- For the magenta branch we switch off the translational constraints, i.e., set `p.hopf.nqh=0`. While the constraint is useful for narrow breathers (at the start of the red branch), the wider breathers interact strongly enough with the boundary and the constraint can be dropped. The magenta branch can also be computed with `p.hopf.nqh=1` but this becomes more expensive. Most importantly, due to a poorly localized BP (critical multiplier -1.12) we start with a very large tolerance `tol=0.5` to get onto the period-doubled branch, but we can subsequently decrease the tolerance to 1e-8 as usual.

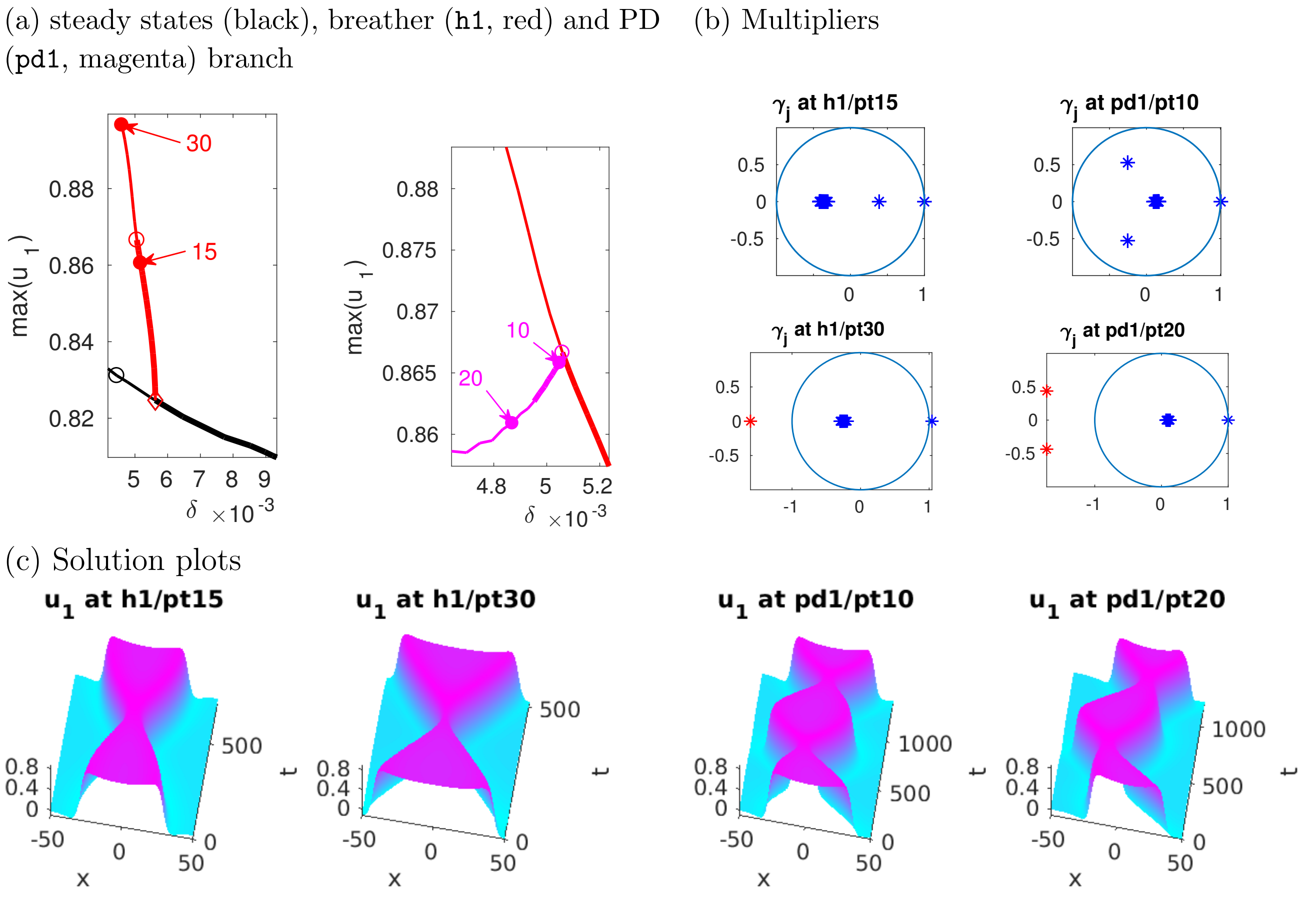


Figure 13: Period doubling bifurcation in (5.12), $(\alpha, \beta, \gamma) = (0.11, 1, 6)$, $D = 2$.

6 O(2) equivariance: traveling vs standing waves, and relative periodic orbits

In §2 we considered the cGL equation over domains which lead to simple HBPs, i.e., boxes with NBC or DBC, where moreover in 2D and 3D we chose suitable side-lengths l_x, l_y, l_z, in particular $l_x \neq l_y$. If for instance in 2D we instead chose a square domain, then naturally the 2nd HBP would be double, with oscillating 'horizontal' and 'vertical' stripes as two Hopf eigenfunctions.

For steady bifurcations, the higher multiplicities of BPs due to discrete symmetries and the associated multiple bifurcating branches can be dealt with systematically in `pde2path`, as described in [Uec20a]. For HBPs of higher multiplicity we do not yet provide similar routines, but rather treat them in an ad hoc way. Moreover, for Hopf problems multiple branches due to continuous symmetries are probably even more important than multiple branches due to discrete symmetries. In particular, O(2) equivariant Hopf bifurcations arise in a variety of settings, for instance for translational invariant problems (due to pBC) with reflection symmetry, and similarly for problems on circular domains, where the role of translational invariance is played by spatial rotations. Thus, here we first consider the cGL equation in boxes with pBC, and in a disk domain (demos `cglpbc` and `cgldisk`), and then review the demo `gksspirals` dealing with a RD model from [Uec19, §3.2] in the unit disk.

For background on O(2) equivariant Hopf bifurcation see, e.g., [GS02], and the references therein. Loosely said, the main result is that for double HBPs we generically obtain three bifurcating branches of Hopf orbits: left/right traveling waves (TWs), and standing waves (SWs), which correspond to equal amplitude superpositions of TWs. Importantly, the TWs are steady solutions in an appropriate co-moving frame, and are thus much cheaper to compute than general Hopf orbits.

6.1 The cGL equation in boxes with pBC: demo `cglpbc`

6.1.1 1D

. We consider a variant of (2.2), namely

$$\partial_t \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = -G(u,\lambda) \tag{6.1}$$
$$:= \begin{pmatrix} \Delta + r & -\nu \\ \delta^2\nu & \Delta + r \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} - (u_1^2+u_2^2)\begin{pmatrix} c_3u_1 - \mu u_2 \\ \mu u_1 + c_3 u_2 \end{pmatrix} - c_5(u_1^2+u_2^2)^2 \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + s\partial_x \begin{pmatrix} u_1 \\ u_2 \end{pmatrix},$$

first on the interval $\Omega = (-\pi,\pi)$ with pBC. The additional parameter δ can be used to break the phase invariance $u \mapsto \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} u$ (i.e. $u \mapsto \mathrm{e}^{\mathrm{i}\phi}u$ in complex notation) of (2.1), see below for further comments, and the parameter $s \in \mathbb{R}$ describes a frame moving with speed s, useful later for the continuation of TWs. As in §2 we fix $c_3 = -1, c_5 = 1$, $\nu = 1$, (and initially $\mu = 0.5$, $\delta = 1$ and $s = 0$), and use r as the primary bifurcation parameter.

The HBPs from the trivial solution are still

$$r_k = k^2, k \in \mathbb{N}, \text{ eigenvalues } \pm\mathrm{i}\omega \text{ with } \omega = \nu, \tag{6.2}$$

but now are double for $k > 0$. Two 'natural' two eigenfunctions are (in complex notation)

$$\phi_1(t,x) = \mathrm{e}^{\mathrm{i}(\omega t - kx)} \text{ and } \phi_2(t,x) = \mathrm{e}^{\mathrm{i}(\omega t + kx)}, \tag{6.3}$$

and the ansatz for bifurcating periodic orbits is

$$u = z_1\phi_1 + z_2\phi_2 + \text{h.o.t}, \quad (z_1, z_2) \in \mathbb{C}^2. \tag{6.4}$$

Thus, ϕ_1 corresponds to a right TW with speed ω/k, and ϕ_2 to a left TW with speed $-\omega/k$. For $s = 0$, (6.1) is O(2) equivariant, i.e., $G(\gamma u) = \gamma G(u)$ for all $\gamma \in \Gamma = \mathrm{O}(2)$. Here $\gamma = (m, \xi)$, $m \in \mathbb{Z}_2 = \{\pm 1\}$, $\xi \in \mathrm{SO}(2) = [0, 2\pi)$, and the action of γ on x and $u(x)$ is given by $(\gamma u)(x) = u(m(x+\xi))$ (reflection and translation). For each $k \in \mathbb{N}$, the subspace $X_k := \mathrm{span}\{\phi_1, \phi_2\}$ is Γ invariant, and the action of γ on (z_1, z_2) is

$$m : (z_1, z_2) \mapsto (z_2, z_1), \quad \xi : (z_1, z_2) \mapsto (\mathrm{e}^{\mathrm{i}\xi} z_1, \mathrm{e}^{-\mathrm{i}\xi} z_2). \tag{6.5}$$

The equivariant Hopf theorem ([GS02, Thm 4.9] or [Hoy06, Thm 4.6]) yields that generically for each k we have exactly three bifurcating branches, namely

$$\begin{cases} \text{rTW} & u(x,t) = z_1\phi_1 + \text{h.o.t} \\ \text{lTW} & u(x,t) = z_2\phi_2 + \text{h.o.t} \\ \text{SWs} & u(x,t) = z(\phi_1+\phi_2) + \text{h.o.t}, \end{cases} \tag{6.6}$$

where as usual h.o.t stands for higher order terms. Moreover, the TWs are solutions of the form

$$u(x,t) = v(x-st) \text{ with some speed } s \in \mathbb{R}, \tag{6.7}$$

i.e. *relative equilibria*, which means that we can find them as steady solutions of (6.1) with a suitable s. In fact, from the phase invariance for $\delta = 1$ we already have an explicit formula for TWs, namely

$$u(x,t) = R\mathrm{e}^{\mathrm{i}(\omega t - k\cdot x)}, \quad |R|^2 = -\frac{c_3}{2c_5} \pm \sqrt{\frac{c_3^2}{4c_5^2} + r - k^2}, \quad \omega = \omega(k,r) = \nu - \mu |R|^2, \tag{6.8}$$

but (6.7) is the more general result. Moreover, on the spaces X_k the additional SO(2) phase symmetry $\vartheta : u \mapsto \mathrm{e}^{\mathrm{i}\vartheta} u$ acts like time-shifts $u \mapsto u(t + \vartheta/\tau)$, and hence is not an additional symmetry for the Hopf bifurcation and does not need to be considered further.

Remark 6.1. In summary, at each $r_k = k^2$ we have the bifurcation of TWs and SWs, and our aim is to compute these numerically (even if for $\delta = 1$ we know the TWs analytically from (6.8)). Thus we face a similar problem like for steady bifurcations of higher multiplicity, discussed in detail in [Uec20a]: When computing the eigenfunctions associated to the eigenvalue $\mathrm{i}\omega$ of G_u we in general do not obtain the 'natural' ones $\phi_{1,2}|_{t=0} = \mathrm{e}^{\pm \mathrm{i}kx}$ from (6.3), but *some* linear independent $\tilde{\phi}_{1,2} \in \operatorname{span}\{\mathrm{e}^{\mathrm{i}kx}, \mathrm{e}^{-\mathrm{i}kx}\}$. Thus, even though we know the analytic form (6.6) of the bifurcating branches, this only applies to the natural basis $\phi_{1,2}$ of the center eigenspace. In principle we could compute the (3rd order) amplitude system on the center manifold associated to the basis $\tilde{\phi}_1 \mathrm{e}^{\mathrm{i}\omega t}, \tilde{\phi}_2 \mathrm{e}^{\mathrm{i}\omega t}$, and from this the coefficients $\tilde{z}_{1,2}$ for TWs and SWs. However, in contrast to the steady case, for which we provide routines to do so, we refrain from implementing this for the Hopf case in `pde2path`, because the general case of multiple Hopf bifurcations becomes significantly more complicated, see, e.g., [Kie79], or [Mei00] for the case where additional mode interactions with steady modes come into play. Instead, we proceed ad hoc, and require user input of coefficients z_1, z_2 (and $z_3, \dots, z_m$ in case of still higher multiplicity m, see §6.1.2). In practice this works quite well. ⌋

```
%% C1: init, and continuation of trivial branch
p=[]; lx=pi; nx=50; par=[-0.1; 1; -0.5; -1; 1; 0; 1];  % r,nu,mu,c3,c5,s,del
p=cGLinit(p,lx,nx,par); dir='01D'; p=setfn(p,dir); % initialize
p=box2per(p,1); p=cont(p,20); % switch on periodic BC and continue
%% C2: bif to SWs/TWs at HBP2;
figure(2); clf; aux=[]; aux.dlam=0; dir='01D'; hp='hpt2'; nsteps=40;
for sw=1:2
  switch sw
    case 1; aux.z=[1 -2i]; ndir='1dtw1'; pc=0; % TW
    case 2; aux.z=[1 0]; ndir='1dsw1'; pc=1;   % SW
  end
  if pc % SWs, need phase-condition, resp. can be enforced by PC
   aux.nqh=1; aux.qfh=@qfh; aux.qfhder=@qfhjac;
   p=hoswibra(dir,hp,0.1,4,ndir,aux);
   p.hopf.ilam=6; p.u0x=p.mat.Kx*p.hopf.tau(1:p.nu)'; % transl.phase cond
  else p=hoswibra(dir,hp,ds,4,ndir,aux);   end
  p.nc.dsmax=0.2;p.hopf.bisec=5;p.file.smod=1;p.sw.bifcheck=1;p=cont(p,nsteps);
end
```

Listing 26: `cglpbc/cmds1d.m` (first 2 cells). In C1, the only new command is `box2per`, which switches on the pBC. In C2 we compute a TW branch and a SW branch bifurcating from Hopf point 2 by 'guessing' and passing on to `hoswibra` coefficients `aux.z`. For the SW branch we additionally set the average translational PC `qfh` as in §5.2, with reference profile $u_0(x) = u_{\text{pred}}(0,x)$, where u_{pred} is the predictor for the Hopf orbit, and speed parameter s (given in `par(6)`).

In Listing 27 we give the start of cglpbc/cmds1d.m for (6.1) on $\Omega = (-\pi, \pi)$ with pBCs, which are switched on via box2per, see [DU17]. We ignore the first (spatially homogeneous $k = 0$) Hopf branch, and in C2 compute one TW branch and one SW branch bifurcating at the 2nd HBP, corresponding to wave number $k = 1$. Here we 'guess' by some trial and error the coefficients z_1, z_2 for each of these branches. Additionally, for the SW branch we set a translational PC (with $s = 0$). It turns out that this PC is usually enough to force SWs, even if the guess for the coefficients aux.z rather corresponds to a TW. See Fig. 14 for some results. The SW branch stays unstable up to $r = 2$ and beyond. The TW branch starts unstably (as expected, as the trivial branch is already unstable) with $\mathrm{ind}(u_H) = 5$, which turns into $\mathrm{ind}(u_H) = 4$ at the fold, into $\mathrm{ind}(u_H) = 2$ shortly after, and u_H becomes stable near $r \approx 1.25$. Both crossings of unstable multipliers into the unit circle are of torus type, and hence the bifurcating branches in this form currently can not be computed with pde2path. But as already said, the TWs can also be computed as relative equilibria, i.e., as steady states in a frame comoving with speed

$$s = \omega/k \text{ at bifurcation,} \tag{6.9}$$

where k is the spatial wave number. The pertinent branch switching is implemented in

p=twswibra(dir,fname,spar,kwnr,newdir,aux),

where spar is the index of s in the parameter vector, kwnr $= k$, and aux.z again can be used to pass the coefficients $z_{1,2}$ for the predictor guess. Complementing this with the PC qf, i.e., $\langle \partial_x u_0, u \rangle = 0$, we obtain the same TW as in C2 with a small error in the period T between the two methods, which vanishes if we increase the temporal resolution for the hoswibra solution. To obtain a space–time plot of TWs, use twplot.

```
%% C3: TWswibra, speed s=om/k, HBP2
aux.z=[1 -2i]; spar=6; kwnr=1; p=twswibra('01D','hpt2',spar,kwnr,'1dtw1b',aux);
p.u0(1:p.nu)=p.tau(1:p.nu); p.u0=p.u0'; p.nc.mu2=0.1;
p.u0x=p.mat.Kx*p.u0; p.u(1:p.nu)=p.u(1:p.nu)+0.01*p.tau(1:p.nu);
p.nc.nq=1; p.nc.ilam=[1;6];  % 1 phase-cond, speed as second parameter
p.fuha.qf=@qf; p.sw.qjac=1; p.fuha.qfder=@qjac;
p.sw.bprint=6; clf(2); p.nc.dsmax=0.05; p.sol.ds=0.03; p=cont(p,60);
%% C4: secondary bif via hoswibra from TW-cont
aux=[]; aux.dlam=0; aux.nqh=1; aux.nqnew=0; aux.tl=40; aux.qfh=@qfh;
aux.qfhder=@qfhjac; p=hoswibra('1dtw1b','hpt2',0.04,4,'1dtw1bs1',aux);
p.file.smod=2;  p.sw.bifcheck=0; p.hopf.ilam=6; p.nc.ilam=1; p=cont(p,80);
```

Listing 27: cglpbc/cmds1d.m (cells 3 and 4). In C3 we compute the TW branch as a relative equilibrium via twswibra. On this branch we find HBPs, and in C4 we compute secondary Hopf branches bifurcating from this relative equilibrium; see Fig. 14 for plots (as obtained from plotcmds.m, and text for further comments.

The continuation of the TW as a relative equilibrium yields HBPs on this branch, at the locations where the continuation as periodic orbits yields the (torus) BPs, see also Remark 6.2(a). Now we can use hoswibra to compute the bifurcating modulated TWs as *relative periodic orbits* (relPO). Figure 14(c) shows a zoom near the fold; the bifurcating branch seems to connect to the SW branch near $r \approx 0.89$, see also the solution plots in (e), where the bottom row shows the solutions in the lab frame, i.e., by shifting back $x \mapsto x + st$. Here we plot over $\Omega \times [0, 2T_2)$, where $T_2 \approx 2.28$ is the period in the moving frame. In general, Hopf orbits bifurcating from relative equilibria (i.e., in the moving frame) correspond to quasiperiodic solutions in the lab frame, and the quotient T_1/T_2 (with $T_1 = L/(ks)$, s the comoving speed, L the domain size, and k the wave number) varies continuously. At pt32

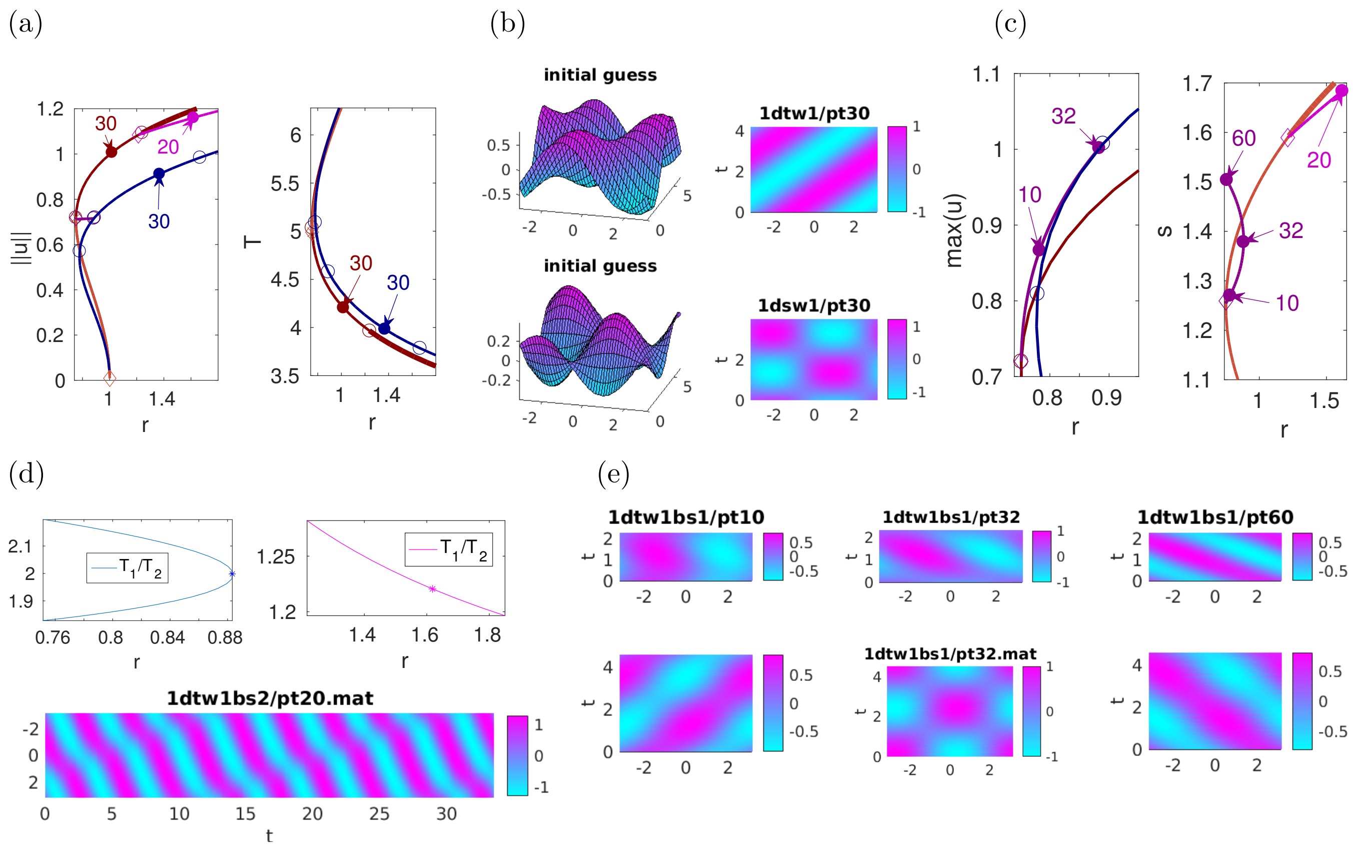


Figure 14: (6.1) on $\Omega = (-\pi, \pi)$ with pBC, $(\nu, \mu, c_3, c_5, \delta) = (1, 0.5, -1, 1, 1)$. (a) BD of TW (brown) and SW (blue) branches, and secondary bifurcation from TW branch (dark and light magenta). (b) intial guesses for TW and SW branches, and two solution plots. (c) Zoom into BDs near TW fold, including secondary branches. (d) Quotients (top) of periods $T_1 = 2\pi/s$ and T_2 where s is the frame speed and T_2 the period in the comoving frame. Left on 'connecting branch', right on magenta modulated TW. Bottom: solution plot in lab frame (bottom) at marked magenta point. (e) solution plots at marked points on dark magenta branch in comoving frame (top) and lab frame (bottom). At the fold (pt32) there is the resonance $T_1 = 2T_2$, and the combination of left traveling in the moving frame and the motion of the frame yields the SW with period $T = 2T_2$.

the solution is (approximately) $2T_1$ periodic in the lab frame and corresponds to the SW. Similarly, solution 20 on the magenta branch is approximately $11T_2 = 33.36$–periodic in the lab frame, see the bottom panel of (d).

Remark 6.2. (a) The multipliers of the periodic orbit $u(x,t)$ in the lab frame are given by $\gamma_j = \mathrm{e}^{-\mu_j T}$, where the μ_j are the eigenvalues of the linearization around the TW in the comoving frame. In detail, the ansatz $u(x,t) = v(x - st, t)$ yields $\partial_t v = -G(v) + s\partial_\xi v$, with linearization $\partial_t v = -G_u(v_0(\xi))v + s\partial\xi v =: -Lv$. As L is independent of t, the linear flow yields $v(T) = \sum_j c_j \mathrm{e}^{-\mu_j T}\phi_j$, where $v(0) = \sum_j c_j\phi_j$, and where for simplicity we assumed semisimple eigenvalues μ_j of L with associated eigenvectors (eigenfunctions) ϕ_j, $j = 1, \dots, n_u$.

(b) To plot the modulated TWs in the lab frame we use the function `lframeplot(dir,pt, wnr,cmp,aux)`. This first aims to determine the minimal $m \in \mathbb{N}$ such that $msT = qL$ for some (minimal) $q \in \mathbb{N}$, where T is the period in the frame moving with speed s and L is the domain size. Equivalently, $m = qL/sT$ for some (minimal) integer q, and the minimal is period $T^* = mT = \frac{q}{s}L$. Of course, $m = qL/sT \in \mathbb{N}$ numerically means $|m - \lfloor m \rfloor| <$tol, where tol can be passed as `aux.pertol`. This should not be taken too small, i.e., on the order of the expected error in the speed s and the period T. Naturally, this also ignores the fact that generically $T_1/T_2 \notin \mathbb{Q}$, and that the associated orbits are quasiperiodic rather than periodic (with a possibly large period). Alternatively, an integer m can be passed in `aux.m` to force the plot over $[0, mT]$. `lframeplot` at the end also reports the final (integer) grid-point shift of the transformation to the moving frame is given, which should be 0. ⌋

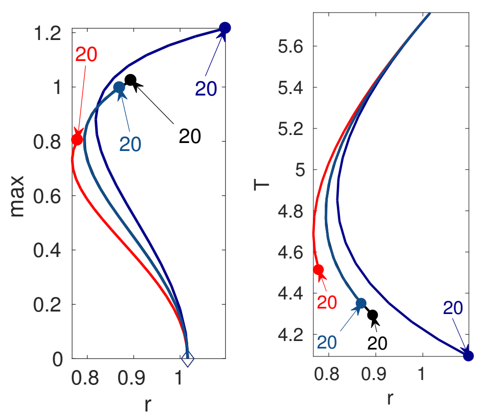
6.1.2 2D box with pBC in x

In Fig. 15 we give some very introductory results for (6.1) over the 2D square box $\Omega = (-\pi, \pi)^2$, with pBC in x and homogeneous Neumann BC in y. The 2nd HBP at (analytically) $r = 1$ is then triple, with Hopf eigenfunctions (in complex notation)

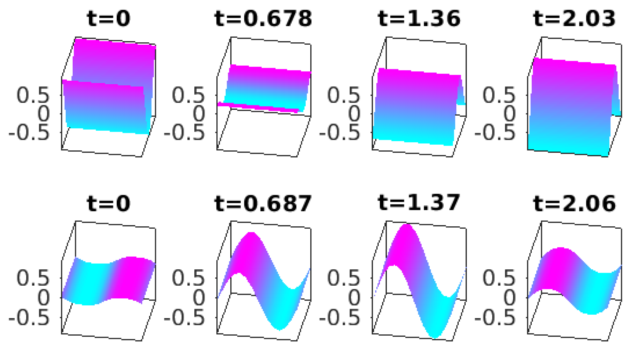
$$e^{i(\omega t - x)}, e^{i(\omega t - x)}, \cos(y)e^{i\omega t} \tag{6.10}$$

and modulo spatial translation we may expect at least five primary bifurcating branches: SW and TW (twice) in x, SW in y, and a mixed SW mode of the form $b(t)\sin(x)\cos(y)$. Four such branches are computed in `cmds2d.m` via 'educated' guesses of the three coefficients for the three numerical eigenfunctions replacing (6.10). Naturally, the TW branches can again also be computed as relative equilibria, and we find several secondary bifurcations. See `cmds2d.m`, but here we refrain from giving the further details.

(a)



(b)



(c)

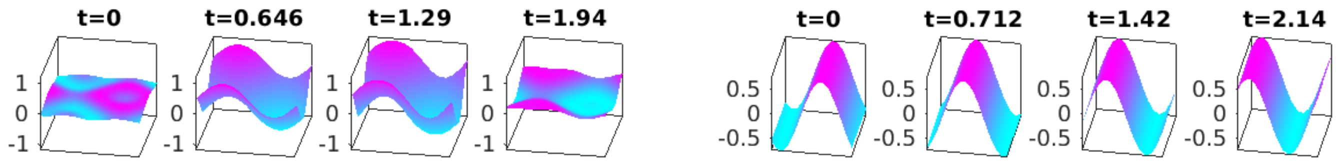


Figure 15: (6.1) on $\Omega = (-\pi, \pi)^2$ with pBC in x and Neumann BC in y, $(\nu, \mu, c_3, c_5, \delta) = (1, 0.5, -1, 1, 1)$. (a) BD of 4 branches bifurcating from the 2nd HBP $r = 1$: swy (black), tw (red), swx-y (dark blue), swx (light blue). (b,c) example solution plots (roughly half a period). swy (top) and (swx) in (b), swx-y (left) and tw (right) in (c).

6.2 The cGL equation in a disk: demo `cgldisk`

A situation very similar to the 1D-pBC case arises for (6.1) in a disk with Neumann BC (or other rotationally invariant BCs). The symmetry group is again O(2) where the role of spatial translations is now played by spatial rotations

$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto R_\vartheta \begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}, \tag{6.11}$$

$\vartheta \in [0, 2\pi)$. The generator $\partial_\vartheta R_\vartheta|_{\vartheta=0}$ of the associated Lie algebra acts on $u(x, y)$ as

$$\partial_\vartheta u(R_\vartheta(x, y))|_{\vartheta=0} = \left[\partial_x u \partial_\vartheta x(\vartheta) + \partial_y u \partial_\vartheta y(\vartheta)\right]|_{\vartheta=0} = -y\partial_x u + x\partial_y u =: K_{\text{rot}} u.$$

Hence the rotational phase condition reads $\langle K_{\text{rot}} u_0, u\rangle = 0$, where u_0 is a suitable profile, typically set at bifurcation. Similarly, the rotating wave (RW) ansatz $u((x, y), t) = \tilde{u}(R_{-st}(x, y), t)$ yields

$$\partial_t u = -sK_{\text{rot}}\tilde{u} + \partial_t\tilde{u} = -G(u) = -G(\tilde{u}), \text{ hence } \partial_t\tilde{u} = -G(\tilde{u}) + sK_{\text{rot}}\tilde{u},$$

after which we drop the $\tilde{}$ again.

For the implementation, in `oosetfemops` we generate K_{rot} via

```
po=getpte(p); x=po(1,:); y=po(2,:); p.mat.Krot=convection(fem,grid,[-y;x]).     (6.12)
```

The PC $q = \langle K_{\text{rot}} u_0, u\rangle = 0$ is implemented as

```
function q=qf(p,u); q=(p.mat.Krot*p.u0)'*u(1:p.nu); end     (6.13)
```

and the pertinent modification of `sG` reads `r=K*u-p.mat.M*f+s*Krot*u`.

The eigenfunctions v of Δ with Neumann BC have the form $u_{n,j}(x, y) = B_{n,j}(lr)g_n(\phi)$, where $g_n(\phi) = \mathrm{e}^{\mathrm{i}n\phi}$, l is a scaling factor, and $B_{n,j}$ is a Bessel function. These can be used to explicitly compute the HBPs from $u \equiv 0$, and to see that the HBPs are simple for $n = 0$ and double for $n \neq 0$ (with $\sin(n\phi)$ and $\cos(n\phi)$ the two basis functions in the angular direction). For $n = 0$, there is no angular dependence, and hence switching on the PC on such branches (with $u_0 = u_0(r)$) leads to a singular Jacobian because $K_{\text{rot}} u_0 = 0$. For coarse meshes, the rotational invariance is sufficiently broken for the continuation to work also without PC, but for finer meshes the PC becomes vital for robust continuation.

Table 6: Short overview of scripts and functions in `hopfdemos/cgldisk`; see sources for details.

script/function	purpose,remarks
cmds2d, plotcmds, cGLinit	main script, plotting commands, init function as usual
sG, sGjac, nodalf,njac	rhs, Jacobian, and nonlinearity, as usual
qf,qjac,qfh,qfhjac	phase conditions, based on `Krot` generated in `oosetfemops`, and versions for Hopf orbits
hoplott, plottip, lfplottip, gettip	some additional customized plot commands, and helper functions

Table 6 gives a short overview of the files for the implementation. In `cmds2d` we consider (6.1) in a disk with radius π, with base parameters $(\nu, \mu, c_3, c_5, \delta) = (1, -5, -1, 1, 1)$, and bifurcation parameter r as before. We changed μ in order to have pronounced 'spirals' [KH81, BKT90, Bar95, Sch98, SSW99] as RWs, see Fig. 16, and §6.3 for further more general comments. Relative periodic orbits then typically correspond to 'meandering spirals' which are most easily characterized by the motion of the tip. In order to compute the spiral tips with reasonable accuracy, at the start of `cmds2d` we locally refine the mesh near $(x, y) = 0$, see Fig. 16(a), leading to a mesh with about 1400 grid points. The temporal resolution for POs will be 30 gridpoints, and thus we will have about 84000 DoF, which we find a reasonable compromise between accuracy and speed; see also Remark 6.3.

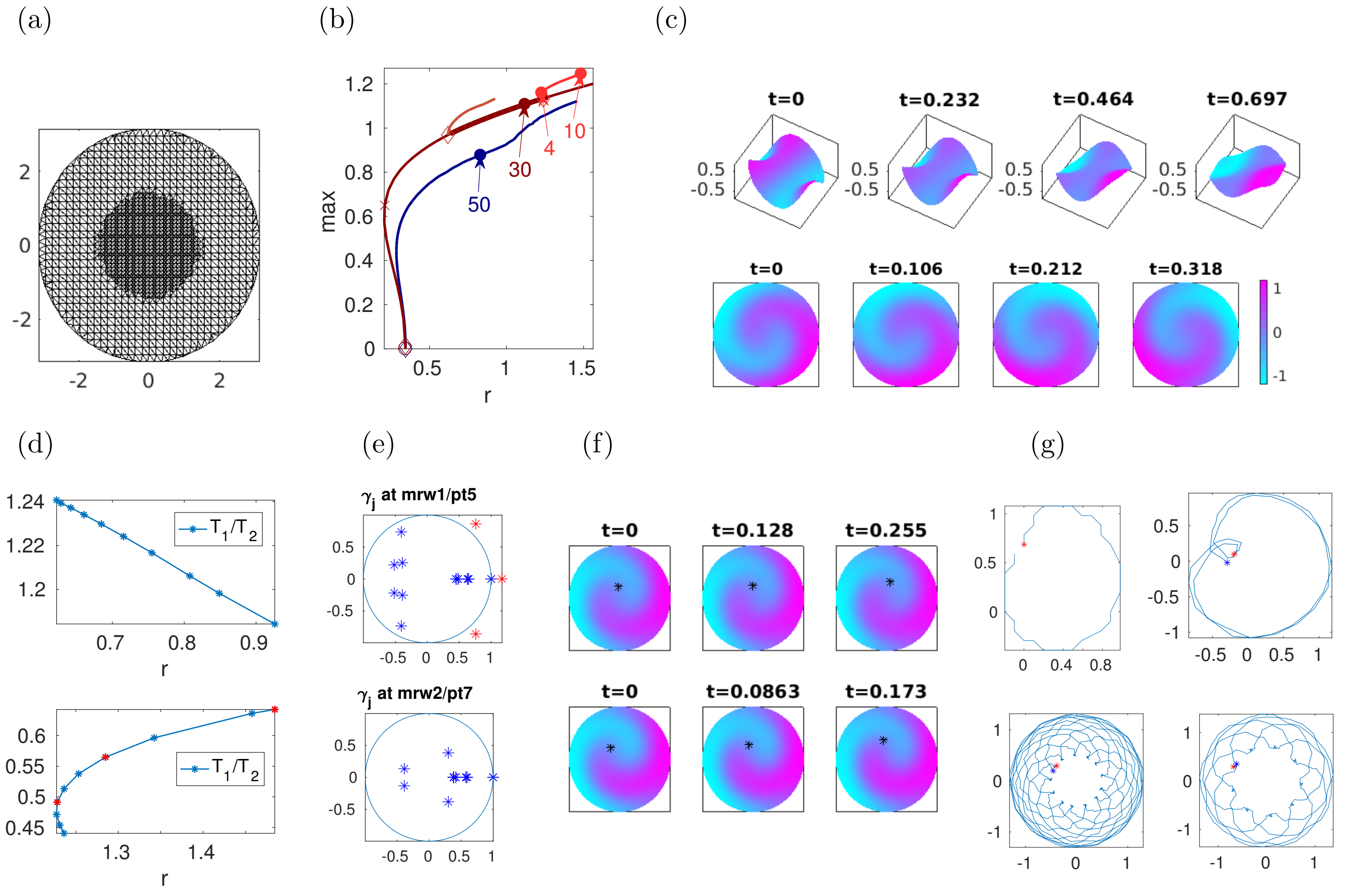


Figure 16: (6.1) on a disk with radius π and homogeneous NBC, $(\nu, \mu, c_3, c_5, \delta) = (1, -5, -1, 1, 1)$. (a) locally (near $\rho = 0$) refined mesh. (b) basic BD of SWs (blue), RWs (brown), and two branches of modulated RWs, mRW1 (light brown, unstable), and mRW2 (red, stable). (c) example plots (snapshots from roughly the first quarter period) of SW and RW. (d) Period quotients on mRW1 (top) and mRW2 (bottom) branches, $T_1 = 2\pi/s$ (lab-frame time period of RWs), T_2 =period on mRW2 in the rotating frame. The mRW1 branch is unstable and hence not discussed further, but the mRW2 branch is stable after the fold, and the red dots in the bottom panel correspond to the solutions used in (f,g). (e) Floquet spectra of selected mRW1 and mRW2. (g) Rotating frame plots of example solutions pt4 (top) and pt10 (bottom) from mRW2 branch (first $\frac{1}{5}$th of period), including computed tip positions. (g) Top: tip-path for mRW2/pt4 in comoving (left) and lab frame (right, $m = 2$). Bottom: lab frame paths of tips for mRW2/pt7 (left, $m = 13$) and mRW2/pt10 (right, $m = 9$).

Figure 16 (b) shows a basic BD of SWs, of RWs bifurcating at the second HBP, and two branches mRW1 and mRW2 of modulated RWs. As in Fig. 14 we omit the (stable) primary spatially uniform SW branch bifurcating at $r = 0$. The RW branch (dark brown) is stable between the first and second HBP, where mRW1 and mRW2 bifurcate. The SW branch is unstable. We now focus on mRW2, which bifurcates (slightly) subcritically, and is stable after the fold, which however for efficiency we check a posteriori. (d) shows the ratio of periods $T_1 = 2\pi/s$ (lab-frame period of RWs) and T_2 (period on mRW2 in the rotating frame).

To visualize the solutions in the lab frame one often uses the motion of the spiral tip $(\tilde{x}(t), \tilde{y}(t))$, where, if $(x(t), y(t))$ are the coordinates in the rotating frame, then

$$\begin{pmatrix} \tilde{x}(t) \\ \tilde{y}(t) \end{pmatrix} = R_{st} \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}. \tag{6.14}$$

There are different options how to define the tip coordinates $(x, y)(t)$. In the far field, in radial direction the spirals behave like plane waves, and thus the basic idea is to use the intersection of isolines of u_1 and u_2 to define $(x, y)(t)$. Alternatively, for 'good' spirals one can use the maximum of $|\nabla u_1 \times \nabla u_2|$, following [JSW89]. In [BKT90], a definition based on isolines of the nonlinearity $(f_1(u), f_2(u))$ has been used as numerically robust definition of the tip for the case of a FHN like model. All these definitions typically give approximately the same tip positions. Here we choose the intersection of the $f_1(u) = c_1$ and $f_2(u) = c_2$ level curves (with a small tolerances), with $c_1 = c_2 = 0.25$, which for all our spirals gives a unique tip which makes good sense visually, see Fig. 16(f) for example solutions, and `gettip` for the implementation. Moreover, with the locally refined spatial mesh and a sufficient time resolution, the tip paths $t \mapsto (x(t), y(t))$ in the comoving frame are 'reasonably smooth', see the first plot in (g) for an example. To plot the tip paths in the lab frame, we again choose a (minimal) $m \in \mathbb{N}$ such that $|mT_1 - qT_2| <$tol for some $q \in \mathbb{N}$. Using tol=0.025 and (6.14) for $0 \le t \le mT_1$ generates the flower–like patterns plotted in (g). In particular, at pt4 (top right of (g)) we are near $1 : 2$ resonance.

Similarly, the top panel of Fig. 16(d) shows the period quotient on the first mRW branch mRW1. On mRW1 we have a similar tip motion as on mRW2, but since these mRWs are unstable (see top panel of (e)), we skip further plots and discussion.

Remark 6.3. (a) The RWs can be plotted using `rwplot`. As in Remark 6.2(a), the multipliers of the RWs can be obtained from exponentiation of the linearization around the RW in the lab-frame. See below, in particular Remark 6.4, for comments on such spiral spectra.

(b) With the given settings, leading to about 84000 DoF for the Hopf orbits, the computation of the mRW branches (10 steps on each) takes about 400s. The tip paths in Fig. 16(g), computed on the given mesh (with no interpolation involved), suggest that the numerics are somewhat pushed to the limit here, i.e., the meshes (in x and t) may be somewhat under resolved. However, we did check that the results stay qualitatively the same when increasing the temporal resolution to 50. On the other hand, omitting the preparatory step of local mesh refinement near $(x, y) = 0$ the flower patterns in (g) become (even) more ragged, but other quantities such as the BPs and the periods only change slightly, which indicates that we have a correct general picture. To obtain smoother tip paths, alternative methods allowing finer meshes should be more appropriate, e.g., time-simulation [BKT90], or shooting methods for Hopf orbits [SN10, SN16]. ⌋

6.3 Reaction diffusion in a disk: Demo `gksspirals`

Spirals like in §6.2 occur in a variety of settings, i.e., in various excitable or oscillatory 2D RD systems. In particular, the cGL equation can be considered as a normal form near a Hopf bifurcation, and is an example of a so–called $\lambda - \omega$ system

$$\partial_t u = \Delta u + \lambda(u, v)u - \omega(u, v)v, \quad \partial_t v = \Delta v + \lambda(u, v)v + \omega(u, v)u, \tag{6.15}$$

where λ and ω are some functions of $u^2 + v^2$. See [PET94] for further discussion, where in particular $\lambda(u, v) = 1 - u^2 - v^2$ and $\omega(u, v) = 1 + q(u^2 + v^2)$, and where then the role of q (corresponding to μ in (6.1)) as a perturbation parameter for the existence theory of spiral waves and its role to determine their shape is discussed.

In [Uec19, §3.2], with associated demo `hopfdemos/gksspirals` we consider a two-component reaction diffusion system from [GKS00] on the unit disk with somewhat non–standard Robin–BC, namely

$$\begin{aligned} \partial_t u &= d_1\Delta u + (0.5+r)u + v - (u^2+v^2)(u-\alpha v), \\ \partial_t v &= d_2\Delta v + rv - u - (u^2+v^2)(v+\alpha u), \end{aligned} \tag{6.16}$$

$$\partial_{\mathbf{n}} u + 10u = 0, \quad \partial_{\mathbf{n}} v + 0.01v = 0, \tag{6.17}$$

where $\mathbf{n}$ is the outer normal. In [Uec19, §3.2], our focus was on the computation of the primary SW and RW branches, and we did not compute the RWs as relative equilibria, and thus also skipped the computation of modulated RWs as relative POs. Here we include this and thus extend the presentation in [Uec19, §3.2].

The eigenfunctions of the linearization around $(u,v)=(0,0)$ are again build from Fourier Bessel functions

$$\phi_m(\rho,\vartheta,t) = \Re(\mathrm{e}^{\mathrm{i}(\omega t + m\vartheta)} J_m(q\rho)), \quad m\in\mathbb{Z}, \tag{6.18}$$

where (ρ,ϑ) are polar-coordinates, and with, due to the BC (6.17), 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 trivial solution $(u,v)=(0,0)$ is stable up to $r_1\approx -0.21$ where a Hopf bifurcation with angular wave number $m=0$ in (6.18) occurs, and then further Hopf-bifurcations occur with $m=1,2,0,3,\ldots$.

Like (6.1), (6.17) has spatial symmetry group O(2), acting by rotations and reflections in x. The modes (6.18) with $m\neq 0$ have the symmetry of RWs, and the associated HBPs are double. Thus, as in (6.4) we use a modified branch switching

$$u(t) = u_0 + 2\varepsilon\alpha\mathrm{Re}(z_1\mathrm{e}^{-\mathrm{i}\omega_H t}\psi_1 + z_2\mathrm{e}^{\mathrm{i}\omega_H t}\psi_2) \tag{6.19}$$

with user provided $z_1, z_2\in\mathbb{C}$. If we apply (6.19) at a double Hopf bifurcation with $(z_1,z_2)=(1,0)$, then it turns out that the initial guess is sufficiently close to a RW for the subsequent Newton loop to converge to this RW. On the other hand, $z_1=1, z_2=\mathrm{i}$ at the HBPs with $m\geq 1$ yields bifurcation to SW. Alternatively, setting the PC (6.13) (and initializing $s=0$) forces SWs, and moreover makes the continuation of SWs more robust.[6]

First (script files `cmds1` and `cmds1sw`) we follow [GKS00] and set $\alpha=0$, $d_1=0.01$, $d_2=0.015$, and take r as the main bifurcation parameter. Then (script file `cmds2`) we set $\alpha=1$ (where α corresponds to q from [PET94]), see the comments following (6.15), let

$$(d_1,d_2)=\delta(0.01,0.015), \tag{6.20}$$

and also vary δ which corresponds to changing the domain size by $1/\sqrt{\delta}$. Note that the operator/matrix $\partial_\phi = K_{\mathrm{rot}}$ from (6.12), and hence the rotating wave speed s is independent of δ. In the end, this gives a model with distinguished bifurcations of spiral waves from the trivial solution $(u,v)\equiv 0$.

Figure 17(a) shows a basic bifurcation diagram for (6.16), (6.17), and (b) and (c) illustrate the

[6]The continuation of SW works also works without PC, because the FEM discretization destroys the (strict) rotational invariance, but it initially needs small stepsizes due to small eigenvalues, and in the initial continuation steps the angular phase of the SW pattern slightly shifts.

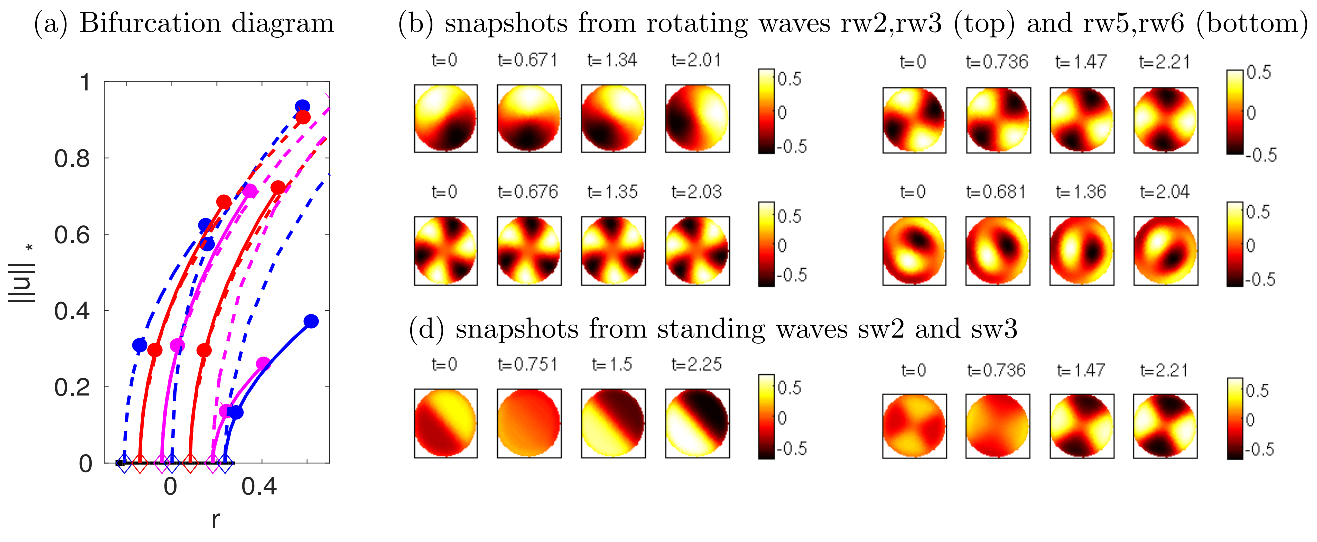


Figure 17: (a) Basic bifurcation diagram for rotating waves (full lines rw2, rw3, rw5, rw6, rw7), and standing waves (dashed lines sw1,…, sw7) for (6.16), (6.17), 10 continuation steps for each. On sw1 and the RW branches we mark the points 5 and 10. (c) Snapshots of u from the RW branches at the last points, $t = 0, T_j/9, 2T_j/9, T_j/3$, with T_j the actual period. (c) Snapshots of u from the RW branches sw2 and sw3. Run `cmds1` and `cmds1sw.m` to obtain these (and other) plots as in [Uec19, §3.2].

difference between rotating and standing waves. Otherwise we refer to [Uec19, §3.2] for further discussion of this and other plots generated in `cmds1, cmds1sw` and `cmds2`. Instead, here we focus on the implementation, and comment on results from `cmds3` about modulated RWs (not discussed in [Uec19, §3.2]).

Table 7: Scripts and functions in `hopfdemos/gksspirals`.

script/function	purpose,remarks
cmds1, cmds1sw	main script for $\alpha = 0$ in (6.16): continuation of RW (SW) branches
cmds2	similar to cmds1, but for $\alpha = 1$ in (6.16)
cmds3	continuation of RW (1–armed spiral) for $(\alpha, \delta) = (1, 0.25)$, branch–switching to mRW, and continuation in δ.
geo=circgeo(r,nx)	defining a circular domain
p=rotinit(p,nx,par)	initialization, as usual
sG, sGjac, nodalf	rhs, Jacobian, and nonlinearity, as usual
nbc	BC for (6.17), using the convenience function `gnbc`
auxcmds	making movies, using customized plotting from `homovplot(..)`
hoplotrot, proplot, levplot	some additional customized plot commands

Table 7 lists the pertinent files in `gksspirals`. The general setup is very similar to `cgldisk`, but a difference is that here we use the 'legacy' `pdetoolbox` setting. As a consequence, there is no file `oosetfemops.m`. Instead, in line 5 of `rotinit.m` we define a diffusion tensor, and the system matrices are then generated via the `pde2path` function `setfemops`. Additionally, we set up a function `nbc.m` defining the boundary conditions. See Listings 28-29. Afterwards, the files `sG.m` and `sGjac.m` encoding (6.16),(6.17), and the script files follow standard rules, where for the plotting we set up some customized functions derived from the default `hoplotsol`. Additional to the discussion in [Uec19, §3.2], in C4 of `cmds1` we compute the RWs via `twswibra` as relative equilibria. For this we need to compute $K_{\rm rot}$ in the `pdetoolbox`–setting, see `setfemops` for a pertinent local modification of the standard `setfemops` function. To compute level curves of spirals and from these the tips of spirals we take advantage of pdetoolbox plotting routines (see Fig. 18(b) for an example). The script cmds1sw differs from cmds1 only in setting the PC (6.13). cmds2 is similar to cmds1 but with $\alpha = 1$ and with continuation in the domain size to have more pronounced spirals.

```
   function p=rotinit(p,nx,par) % init for gksspirals-demo, legacy sfem=1 setting
   p=stanparam(p); screenlayout(p);
   p.file.dircheck=0; p.nc.ilam=1; p.nc.neq=2; p.fuha.outfu=@hobra;
   p.fuha.sG=@sG;p.fuha.sGjac=@sGjac; % rhs
 5 p.eqn.c=isoc([[0.01,0];[0,0.015]],2,1); % diffusion tensor (for setfemops)
   p.eqn.b=0; p.eqn.a=0; % no conv. or linear terms (put these into nodalf)
   p.fuha.bc=@nbc; p.fuha.bcjac=@nbc; % BCs
   p.mesh.geo=circgeo(1,nx); hmax=2/nx; p=stanmesh(p,hmax);
   p.sw.sfem=1; p.vol=2*pi; p.sw.bifcheck=2; p.nc.neig=20;
10 p=setfemops(p); % here using legacy setfemops, diff
```

Listing 28: gksspirals/rotinit.m (first 10 lines). The crucial difference to the other Hopf demos is that this is based on the old pdetoolbox setting. This leads to changes in lines 5-8, i.e., the setup of the tensors and BC needed by setfemops in line 10 (which does *not* call a function oosetfemops if p.sw.sfem$\neq -1$).

```
function bc=nbc(p,u) % BC for model rot
b1=10; b2=0.01; % q-vals in \pa_n u+q*u=0 formulation
c1=p.eqn.c(1); c2=p.eqn.c(end);  % diff. constants
b1=b1*c1; b2=b2*c2; % org q-vals need to be multipl. by diff-const.
enum=max(p.mesh.e(5,:)); % #ofboundary segments
g=[0;0];q=[[b1 0]; [0 b2]];
bc=gnbc(p.nc.neq,enum,q,g); % same BC on all bdry segments
```

Listing 29: gksspirals/nbc.m, using the 'generalized Neumann BC' convenience function gnbc.

In cmds3 we set $(\alpha, \delta) = (1, 0.25)$ and start by continuation of the one-armed spirals as relative equilibria. Here we again strain the numerics with a mesh of about 3000 points, and 20 points in time and hence about 120000 DoF for the continuation of the mRW.[7] Figure 18 shows some basic results from cmds3. The RW1 branch stabilizes shortly after bifurcation, where a branch of mRWs (mRW1) bifurcates. As expected, this is somewhat similar to the cGL case in Fig. 16, but there are also interesting differences: Again, the branch of RWs stabilizes at the HBP due to the subcritical bifurcation of mRW1, which itself are unstable. However, in contrast to the cGL case, here the RW1 branch then stays stable up to large $r > 10$, i.e., upon further continuation in r there does not seem to be a further bifurcation to mRWs. Moreover, there now seems to be a period locking between RW1 and mRW1, i.e., $T_1/T_2 = 1$ (within numerical accuracy[8]), where T_1 is the period on RW1 in the lab frame, and T_2 is the period on mRW1 in the frame rotating with speed s. Panels (c), (d) show tip–paths in the rotating and lab frames, and additionally, in (d) we also show the 20 largest multipliers at mRW1/pt25, of which 1 is unstable. (e) shows a continuation of rw1/pt15 in δ, and the associated spectrum at $\delta = 0.1$ (see Remark 6.4 for further comments). The period T only depends weakly on the domain size $1/\sqrt{\delta}$, as it should for reasonably well developed spirals for which the BC play a negligible role.

Remark 6.4. As in Remark 6.2(a), the multipliers γ_j of a RW can be obtained by exponentiation of the eigenvalues of the linearization in the co–rotating frame. Such spiral-wave spectra [SS06, WB06, SS07] show some interesting structure, e.g., near the imaginary axis they are periodic with period s, see Fig. 18(e) for an example. Heuristically, this can be explained as follows. In polar coordinates

[7]The finer mesh is again needed to somewhat accurately compute the tip paths, and, here also to compute the HBPs from the RWs with good accuracy.

[8]Increasing the temporal resolution mildly from 20 to 40, we obtain qualitatively the same behavior, and the quotient T_1/T_2 in the bottom panel of (a) moves closer to 1. See also Remark 6.3

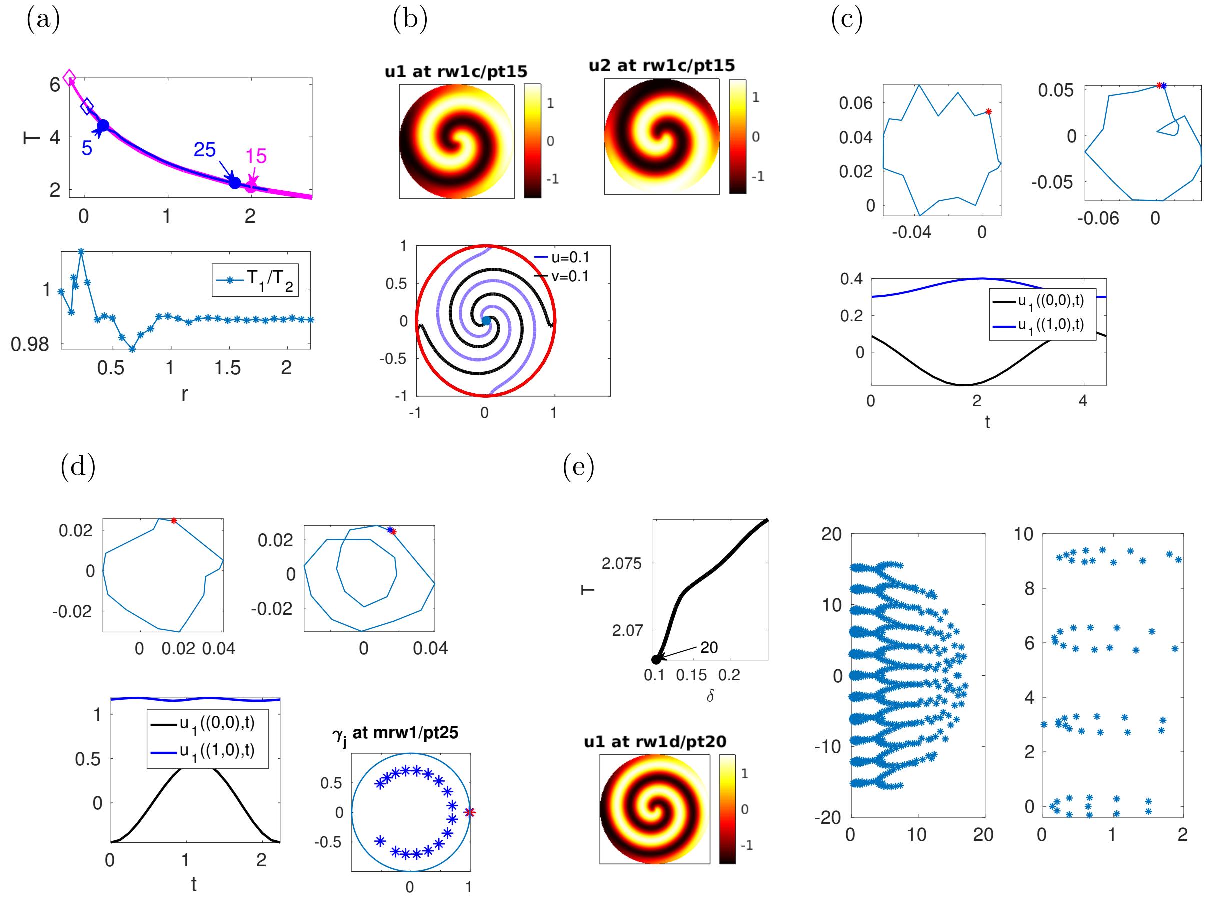


Figure 18: Results from `cmds3.m`. (a) top: BD ($\alpha = 1, \delta = 0.25$) of RW1 (magenta, rotating (spiral) wave) and mRW1 (blue, modulated (meandering) spiral wave bifurcating at HBP1 on RW1). The periods T_1 (of RW1 in the lab frame) and T_2 (of mRW1 in the rotating frame) are equal (within numerical accuracy), and also the speeds s_1 (of RW1) and s_2 (of mRW1, in the average sense) agree. Bottom: quotients of periods. (b) snapshots from RW1/pt15, and contours $u_1 = u_2 = 0.1$, used to define the spiral tip in (c,d). (c) top: tip paths for mRW1/pt5 in rotating frame (left) and lab frame (right); bottom: time series at $(x, y) = (0, 0)$ (black) and $(x, y) = (1, 0)$ (blue), illustrating that the modulation acts near the tip. (d) analogous data from mRW1/pt25, and additionally the 20 largest multipliers. (e) continuation of `rw1/pt15` to $\delta = 0.1$, and illustration of the typical spiral wave spectrum: 500 smallest eigenvalues (middle) and zoom near imaginary axis (right); $s \approx 3.04$.

(ρ, ϕ), the linearization around $u_{\rm RW}$ on the infinite plane reads

$$Lu = D(\partial_\rho^2 u + \frac{1}{\rho}\partial_\rho u + \frac{1}{\rho^2}\partial_\phi^2 u) + s\partial_\phi u + \partial_u f(u_{\rm RW}(\rho, \phi), \lambda)u, \tag{6.21}$$

where following for instance [SS07] we used the shorthand $D\Delta u$ for the diffusion terms with diffusion matrix D, and f for the remaining terms (without spatial derivatives). If we then let $\rho \to \infty$ in the eigenvalue problem $Lu = \mu u$ we formally obtain

$$D\partial_\rho^2 u + s\partial_\phi u + \partial_u f(u_{\rm RW}(\rho, \phi), \lambda)u = \mu u, \tag{6.22}$$

and if (u, μ) is an eigenpair, so is $(u{\rm e}^{{\rm i}\ell\phi}, \mu + {\rm i}s\ell)$, for each $s \in \mathbb{Z}$. This formal computation for the essential spectrum is explained in more detail in [SS06, WB06, SS07] and the references therein, where moreover it is explained that the eigenvalues on large but finite domains accumulate near the so called absolute spectrum, and how the (formal) order $\mathcal{O}(1/\rho)$ and $\mathcal{O}(1/\rho^2)$ terms in (6.21) may generate isolated point spectrum. ⌋

6.4 Extensions: fixed period T, and non–autonomous cases

All our examples so far deal with the autonomous case $M\partial_t u = -G(u)$ with no explicit time dependence of G, and with a free period T which is computed as a part of the solution. In the demo `cglext` we explain how to modify the setup to

- treat an explicit t–dependence of G, and/or
- free an additional parameter to deal with a fixed period T.

This also includes the option to compute and continue POs that are not generated in a Hopf bifurcation via the function `poiniguess`. For non–autonomous problems, we may also expect that we should drop the temporal phase condition (t–PC) (A.6). In any case, we need to have n equations in $n+1$ unknowns, where different combinations are possible:

- Fixing the period T (to the forcing period) and dropping the t–PC requires one free parameter λ, and `length(p.hopf.ilam)=p.hopf.nqh` (number of aux. parameters=number of aux. equations), with `nqh=0` the simplest case.
- If we keep the t–PC but fix T, then we need to free an additional parameter a, i.e., need `length(p.hopf.ilam)=p.hopf.nqh+1`.

To illustrate these setups we modify the cGL equation (2.1), in 1D, first to

$$\partial_t u = \partial_x^2 u + (r + \mathrm{i}\nu)u - (c_3 + \mathrm{i}\mu)|u|^2 u - c_5(t,x)|u|^4 u, \quad u = u(t,x) \in \mathbb{C}, \tag{6.23}$$

where, initially, $(\nu, \mu, c_3) = (1, 1, -1)$, r is the initial bifurcation parameter, and the 5th order coefficient c_5 may depend on t and x. Concretely, we use

$$c_5 = c_5^* + \alpha \tanh(10((t - \beta T)\mathrm{mod}T)) \sin x, \tag{6.24}$$

where α, β are additional constants which we put into `par(8)` and `par(9)`, respectively. This essentially corresponds to a step function at $t = \beta T$ (and $t = T$), allowing to deal with a free T.

For (6.23) with the multiplicative forcing (6.24) we still have the trivial branch $u \equiv 0$ for all α, and we can consider POs generated in Hopf–bifurcations from $u \equiv 0$. We mostly continue these POs including the PC generated at bifurcation, and then with either free T, or with fixed T ($= 2\pi$, generated at bifurcation) and then free ν. Indeed, for (6.23) with the forcing (6.24) we still have continua of (approximately) "time–shifted" POs: a different phase at bifurcation yields a different action of the forcing, but this effect is small at small amplitude, and altogether also at larger amplitude a small phase shift yields a slightly different solution (including a different ν or T). Thus we have continua of solutions parameterized by phase, and if desired it is possible to keep the phase condition (A.6). Relatedly, (6.23) has enough freedom to be quite insensitive wrt initial guesses for POs, i.e., almost any somewhat reasonable initial guess for a PO yields a PO via Newton iteration. Additionally, to check the setup with auxiliary equations and variables, we also treat (6.23) with pBCs, such that for SWs we need a PC in x.

As a second version, we consider the additive forcing

$$\partial_t u = \partial_x^2 u + (r + \mathrm{i}\nu)u - (c_3 + \mathrm{i}\mu)|u|^2 u + c_5(1 + \mathrm{i})\cos(t)\cos(\delta x), \tag{6.25}$$

such that for $c_5 \neq 0$ the trivial branch $u \equiv 0$ is lost. The factor $(1+\mathrm{i})$ in the forcing means that it acts the same way in both components of $u = u_1 + \mathrm{i}u_2$. For $\delta = 0$ and $c_5 \neq 0$ we can still get POs by simple guesses and Newton loops, where we *need* to drop the PC. For $\delta \neq 0$ we did not succeed to get POs for $c_5 \neq 0$ from guessing, and thus proceed by first computing Hopf–orbits for $c_5 = 0$ and then continuing these in c_5.

Table 8 shows the new/modified files in `cglext`, and Listings 30–33 show the some implementation details for (6.23), while the analogous functions `fofu2.m` and `nodalf2` for (6.25) are quite similar.

Table 8: Selected scripts and functions in `hopfdemos/cglext` (cGLinit and oosetfemops as before)

file	purpose,remarks
cmds1a	(6.23) with NBCs and fixed T, t–PC, and free ν.
cmds1b	(6.23) with NBCs, free T, hence no free aux variables.
cmds2	(6.23) with pBCs (hence an x–PC for SWs), fixed T, free ν
cmds3a	(6.25), no t–PC, x–independent forcing, initial POs can be 'guessed'.
cmds3b	(6.25), no t–PC, x–dependent forcing, initial POs via Hopf bifurcation for $c_5 = 0$.
sG, fofu	rhs and forcing function for (6.23), `fofu` called in `nodalf` and `sGjac`.
sG2, fofu2	rhs and forcing function for (6.25).

6.4.1 Multiplicative forcing

For (6.23) we compute POs with fixed $T = 2\pi$ and with keeping the t–PC. Thus we free ν by setting `p.hopf.ilam=2`, and fix T by setting `p.hopf.freeT=0`. This flags `pde2path` to remove T from the list of unknowns, and the column containing $\partial_T \mathcal{G}$ from the Jacobian $\mathcal{A}$, see (A.10). For the case of POs from `p=hoswibra(...,aux)`, `p.hopf.freeT=0` should be switched on by setting `aux.freeT=0`, see Listing 32. For the case of POs from `poiniguess`, this can similarly be done by setting `p.hopf.freeT=0` by hand, see Listing 33. Independent of whether T is free or not, to code a time–dependent G, we can access `p.t` (and `p.T`) set in the Hopf interface functions `hosrhs` and `hosjac`, see Listing 30 for an example.

Figure 19 shows sample results from `cmds1a` for (6.23) with $(c_3, \mu, c_5^*, \alpha, \beta) = (-1, 0.1, 1, 0.5, 0.5)$ on $\Omega = (-\pi, \pi)$ with NBCs, with as usual r as primary continuation parameter. We compute the first two bifurcating PO branches (black and red) via `hoswibra` from the trivial branch, and the third (blue) by `poiniguess` at finite amplitude. The results for free T (and fixed ν) in `cmds1b` are analogous, and therefore not plotted here.

```
function f=fofu(p,u) % forcing function, using:
% p.T=period (as set in hosrhs.m) and
% p.t=current time (normalized to (0,1))
% with error catching for steady problem (or if forcing pars are not set)
par=u(p.nu+1:end);
try; pa=par(8); T=p.T; t=T*p.t; tc=par(9); catch; pa=0; T=0; t=0; tc=0; end
x=getpte(p); n=p.nu/2; x=x(1:n)'; f=pa*tanh(10*(t-tc*T))*sin(x);
```

Listing 30: `cglext/fofu.m`, using the current period T and the current time t as set into `p.T` and `p.t` in the interface function `hosrhs` for calling `sG` for PO continuation.

```
function f=nodalf(p,u) % 'nonlinearity' for cGL (everything but diffusion)
n=p.nu/2; u1=u(1:n); u2=u(n+1:2*n); par=u(p.nu+1:end); % extract fields
r=par(1); nu=par(2); mu=par(3); c3=par(4); c5=par(5); % and parameters
ua=u1.^2+u2.^2; % aux variable |u|^2
```

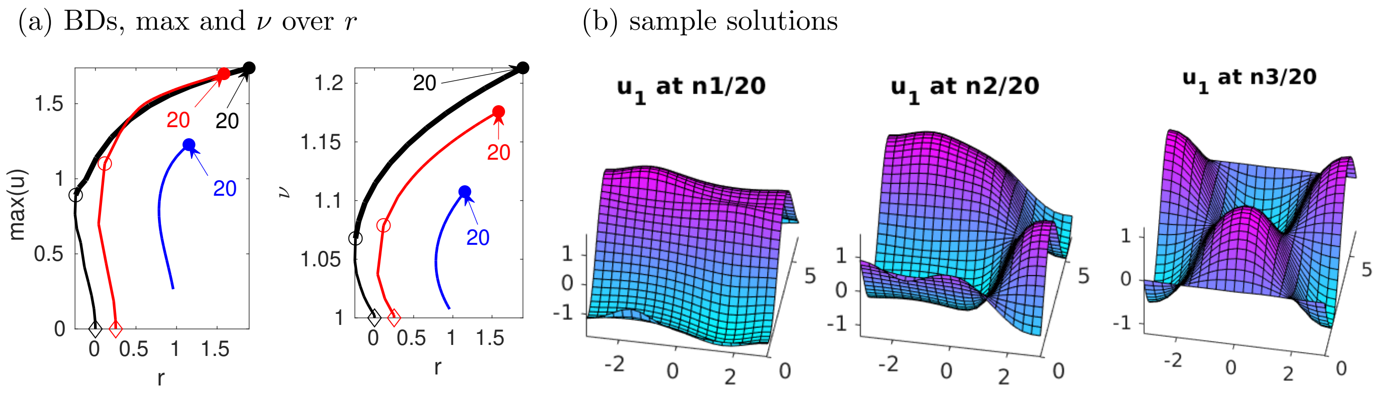


Figure 19: Sample outputs from `cmds1a.m`, (6.23) with NBCs, $\alpha = \beta = 0.5$, fixed period $T = 2\pi$ with ν as free parameter. First (black, n1) and second (n2, red) Hopf branches obtained via `hoswibra`, and third branch (n3, blue) obtained from `poiniguess`; sample solutions in (b).

```
c5=c5+fofu(p,u); % t-(and x)-dependent perturbation of c5;
f1=r*u1-nu*u2-ua.*(c3*u1-mu*u2)-c5.*ua.^2.*u1;
f2=r*u2+nu*u1-ua.*(c3*u2+mu*u1)-c5.*ua.^2.*u2;
f=[f1;f2];
```

Listing 31: `cglext/nodalf.m`, 'nonlinearity' in (6.23), calling `fofu` in l5.

```
%% bifs at HBP1 and 2
figure(2); clf; aux=[]; aux.dlam=0; aux.tl=40; ds=0.1; dir='0a'; aux.freeT=0;
for i=1:2
  hp=['hpt' mat2str(i)]; ndir=['sw' mat2str(i)];
  p=hoswibra(dir,hp,ds,4,ndir,aux); p.nc.dsmax=0.4;
  p=belon(p,2); p.hopf.ilam=2; p=cont(p,20); % free nu and go
end
```

Listing 32: Selection from `cglext/cmds1a.m`: Branch switching to the 1st Hopf orbit, fixing $T = 2\pi$ to the value computed at bifurcation, and instead freeing ν.

```
%% use poiniguess
p=loadp('0a','pt0','sw3'); % load some steady point (for discr.data)
nu=p.nu; p.u(nu+1)=0.5; % reset some parameter as desired
t=linspace(0,2*pi,40); ia1=0.0; ia2=0.1;% create guesses for IC
x=getpte(p); x=x'; uv=[cos(x);cos(x)];  p.u(p.nu+1)=1; % an x-dependent guess
tl=length(t); u=zeros(nu,tl);
for i=1:tl; u(:,i)=ia1+ia2*cos(t(i)+1)*uv; end
aux=[]; aux.ds=0.1; p=poiniguess(p,t,u,aux); p.sol.restart=0; p.sw.verb=0;
p.hopf.freeT=0; p.hopf.ilam=2; p=belon(p,2); p=cont(p,21); % fix T, free nu, go
```

Listing 33: 2nd selection from `cglext/cmds1a.m`, here using `poiniguess` to go to the third branch, again with fixed $T = 2\pi$ and free ν.

Figure 20 shows analogous sample results from `cmds2` where instead of NBCs we use pBCs. The main difference is that now the second HBP is double, and for $\alpha = 0$ there bifurcate TWs and SWs, cf. §6.1.1. Under the multiplicative forcing, the TWs (blue branch) become modulated TWs. Moreover, although the forcing in principle breaks the translational invariance in x for the SWs (red branch) the continuation is more robust by keeping the x–PC as in (5.8b). At the end of `cmds2.m` we again use `poiniguess` to compute PO branches from suitable initial guesses.

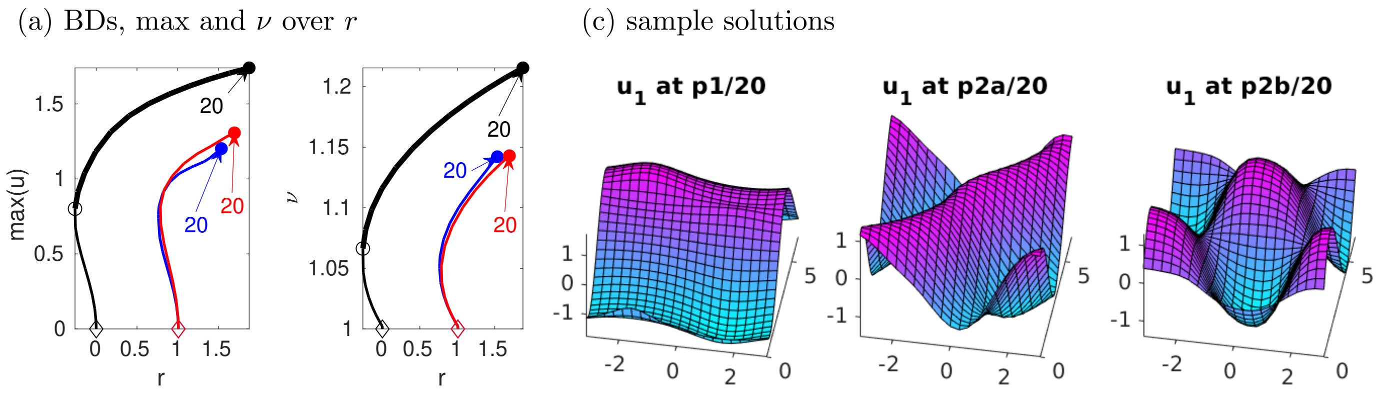


Figure 20: Sample outputs from `cmds2.m`, like Fig. 19 but with with pBCs. Consequently, the 2nd HP is double with standing waves and (modulated) TWs bifurcating.

6.4.2 Additive forcing

Figures 21 and 22 show sample results for the additive forcing (6.25). In Fig. 21, with x-independent forcing, we obtain a first PO at $(r, c_5) = (0.5, 0.5)$ via `poiniguess` with initial guess $u \equiv \sin(t+\phi)(1+\mathrm{i})$ where ϕ can be chosen rather arbitrarily, e.g., $\phi = 0$. Fixing $T = 2\pi$ and dropping the t–PC, the Newton loop then depending on ϕ converges to one of two spatially homogeneous POs, while with the t–PC switched on the Newton loop does not converge. Subsequently continuing in r to $r = 1$ (see (a)), and then switching to continuation in c_5 we get the bifurcation diagram in (b), with sample time-series of the spatially homogeneous solutions given in (c). Due to the $u \mapsto -u$ symmetry of the (unforced) cGL (6.25), we altogether have the symmetry $(u, c_5) \mapsto -(u, c_5)$ and hence restrict to positive c_5. The spatially homogeneous PO starts out unstably but becomes stable at larger amplitude, and there are several pitchfork bifurcations to (unstable) x–dependent POs (see (d) for a sample), where the branches form closed loops between BPs on the homogeneous branch, and show further 2ndary bifurcations.

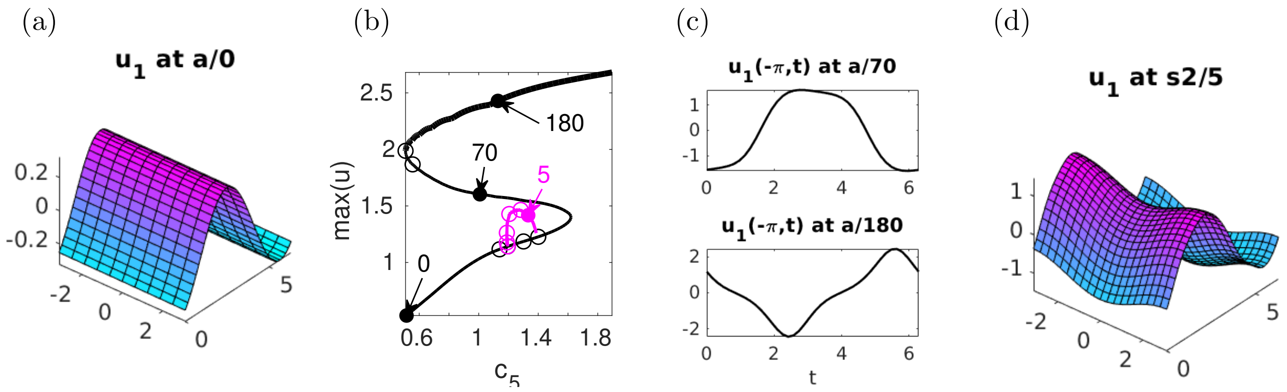


Figure 21: Sample outputs from `cmds3a.m`. (6.25) with NBCs, fixed period $T = 2\pi$, no t–PC, and $\delta = 0$ (x–homogeneous forcing). Initial PO via `poiniguess` with $(r, c_5) = (0.5, 0.)$, then continuation to $r = 1$ (see (a)), and subsequently continuation in c_5 (b). (c) sample plots.

For $\delta = 1$ and $c_5 \neq 0$ we were not able to find POs from simple initial guesses. Thus, in `cmds3b` we first set $c_5 = 0$ and compute POs via Hopf bifurcation from $u \equiv 0$ with fixed $T = 2\pi$ and free ν. Then switching to continuation in c_5 on the first (spatially homogeneous) and second ($\sin(x)$ type) Hopf branches at $r = 0.11$ and $r = 0.35$, respectively, we obtain the red (c1) and blue (c2) branches in

Fig. 22(a), with sample plots in (b). As expected, the $c_5(1+\mathrm{i})\cos(t)\cos(x)$ forcing becomes dominant with increasing c_5.

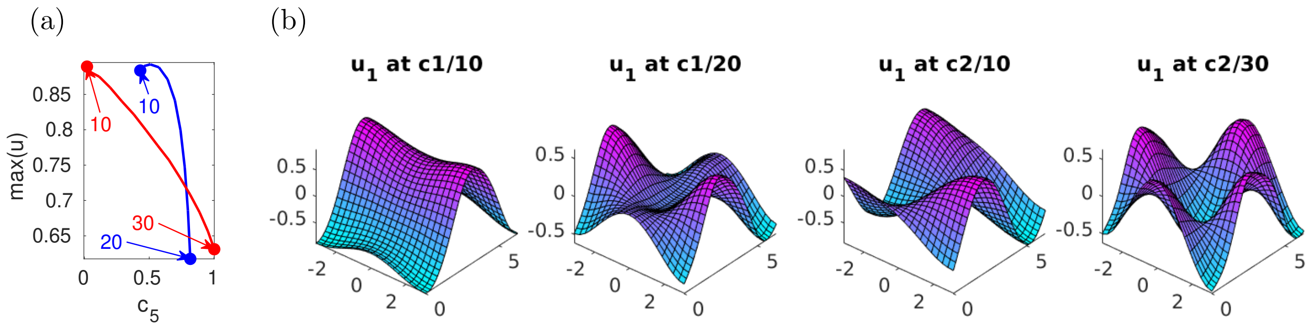


Figure 22: Sample outputs from `cmds3b.m`, (6.25) with forcing $c_5(1+\mathrm{i})\cos(t)\cos(x)$. We first let $c_5 = 0$ and continue the first two bifurcating Hopf branches in r, and then switch to continuation in c_5 at $r = 0.11$ (c1, red) and $r = 0.35$ (c2, blue), respectively.

A Some background and formulas

For details and background on the basic algorithms for Hopf branch point (HBP) detection and localization, branch–switching to and continuation of Hopf orbits, and Floquet multiplier computations we refer to [Uec19]. However, in §A.1 we first briefly repeat the pertinent formulas that are implemented in `pde2path`, including the augmented systems for Hopf computations with constraints. We focus on the arclength parametrization setting (`p.sw.para=4`), which is more convenient and robust than the 'natural parametrization' (`p.sw.para=3`). In Appendix A.2 we then give the formulas used for branch–switching as multipliers go through ± 1. We mix the presentation of formulas with their `pde2path` implementation, and in §B we give an overview of the used data structures and functions.

A.1 Basics

First of all, the detection of Hopf bifurcation points (HBPs) requires the `p.sw.bifcheck=2` setting [Uec19, §2.1], which is essentially controlled as follows:

`p.nc.eigref(1:ne)` contains shifts near which eigenvalues are computed; guesses for these shifts can be obtained via `initeig`. However, except for §(3), here we use `p.nc.eigref=0`.) (A.1)

A bisection for localization of a possible HBP is started if $|\mathrm{Re}\mu| <$ `p.nc.mu1` for the eigenvalue with the smallest abs. real part, and a HBP is accepted if $|\mathrm{Re}\mu| <$ `p.nc.mu2` at the end of the bisection. (A.2)

The default branch switching `hoswibra` to a Hopf branch generates

$$\lambda = \lambda_H + \delta_s\delta_\lambda \quad u(t) = u_0 + 2\alpha\delta_s\Re(\mathrm{e}^{-\mathrm{i}\omega_H t}\Psi), \tag{A.3}$$

as an initial guess for a periodic solution of (1.3) with period near $2\pi/\omega$. Here Ψ is the (complex) eigenvector associated to $\mathrm{i}\omega_H$, and δ_λ, α can be computed from the normal form

$$0 = r\left[\mu_r'(\lambda_H)(\lambda - \lambda_H) + c_1|r|^2\right]. \tag{A.4}$$

of the bifurcation equation on the center manifold, see [Uec19, §2.2]. After the coefficients δ_λ and α in (A.3) are computed (in `hogetnf`), δ_s is chosen in such a way that the initial step length is `ds` in the norm (A.8) below. However, the computation of δ_λ is currently only implemented for semilinear systems, i.e., in FEM form $M\dot{u} = Ku - Mf(u)$, and even then is often not very reliable. Thus, `hogetnf` can be skipped by calling `hoswibra(...,aux)` with `aux.dlam` set to some value, where usually `aux.dlam=0` is the best choice.

To compute Hopf orbits, after rescaling $t \mapsto Tt$ with unknown period T, the time evolution and periodicity condition for u read

$$M\dot{u} = -TG(u,\lambda), \quad u(\cdot,0) = u(\cdot,1), \tag{A.5}$$

and the time-translational phase condition and arclength equation read

$$\phi := \xi_\phi \int_0^1 \langle u(t), \dot{u}_0(t)\rangle \, \mathrm{d}t \overset{!}{=} 0, \tag{A.6}$$

$$\psi := \xi_{\mathrm{H}} \sum_{j=1}^m \langle u(t_j) - u_0(t_j), u_0'(t_j)\rangle_\Omega + (1-\xi_{\mathrm{H}})\big[w_T(T-T_0)T_0' + (1-w_T)(\lambda-\lambda_0)\lambda_0'\big] - \mathrm{ds} \overset{!}{=} 0, \tag{A.7}$$

where T_0, λ_0 and u_0 are from the previous step, ξ_{H}, w_T are weights, $'$ denotes differentiation wrt arclength, and $\langle u,v\rangle_\Omega$ stands for $\int_\Omega \langle u(x), v(x)\rangle \, \mathrm{d}x$, with $\langle a,b\rangle$ the standard $\mathbb{R}^N$ scalar product.[9] Numerically, we use $\langle u,v\rangle_\Omega = \langle Mu, v\rangle$, where M is the mass matrix belonging to the FEM mesh. The steplength is `ds` in the weighted norm

$$\|(u,T,\lambda)\|_\xi = \sqrt{\xi_{\mathrm{H}}\left(\sum_{j=1}^m \|u(t_j)\|_2^2\right) + (1-\xi_{\mathrm{H}})\big[w_T T^2 + (1-w_T)\lambda^2\big]}. \tag{A.8}$$

In (A.6), ξ_ϕ with standard setting $\xi_\phi = 10$ (`p.hopf.pcfac`, see Appendix B) is another weight, which can be helpful to balance the Jacobian $\mathcal{A}$, see (A.10). To improve convergence of Newton loops, it sometimes turns out to be useful to set ξ_ϕ to somewhat larger values. Also, while $\dot{u}_0$ in (A.6) is in principle available from $M\dot{u}_0 = -TG(u_0,\lambda_0)$ (which is used for `p.hopf.y0dsw=0`, it often appears more robust to explicitly approximate $\dot{u}_0$ via finite differences, for which we set `p.hopf.y0dsw=2`. Finally, for a system with n_H Hopf constraints $Q_H(u) = 0$, and hence n_H additional free parameters $a \in \mathbb{R}^{n_H}$, we also add $w_a \langle a - a_0, a'\rangle$ to ψ, where w_a is a weight for the auxiliary parameters a.

Letting $U = (u,T,\lambda,a)$, and writing $\mathcal{G}(U) = 0$ for (A.5) (see also (A.12)), in each continuation step we need to solve

$$H(U) := \begin{pmatrix} \mathcal{G}(U) \\ \phi(u) \\ \psi(U) \\ Q_H(U) \end{pmatrix} \overset{!}{=} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \in \mathbb{R}^{mn_u+2+n_H}, \tag{A.9}$$

where m is the number of time slices t_j, $j = 1,\ldots,m$, n_u the number of PDE unknowns at each time slice, and $n_H =:$ `p.hopf.nqh` is the number of Hopf constraints (encoded in `p.hopf.qfh`). To solve

[9]By default (`p.hopf.y0dsw=2`), $\dot{u}_0$ in (A.6) is evaluated (in `sety0dot`) by 2nd order finite differences; alternatively, `p.hopf.y0dsw=1` means first order FD, and `p.hopf.y0dsw=0` replaces $\dot{u}_0$ by $M\dot{u}_0 = -TG(u_0,\lambda_0)$.

(A.9) we use Newton's method, i.e.,

$$U^{j+1}=U^j-\mathcal{A}(U^j)^{-1}H(U^j),\quad \mathcal{A}=\begin{pmatrix}\partial_u\mathcal{G} & \partial_T\mathcal{G} & \partial_\lambda\mathcal{G} & \partial_a\mathcal{G}\\ \partial_u\phi & 0 & 0 & 0\\ \xi_{\rm H}\tau_u & (1-\xi_{\rm H})w_T\tau_T & (1-\xi_{\rm H})(1-w_T)\tau_\lambda & w_a\tau_a\\ \partial_u Q_H & \partial_T Q_H & \partial_\lambda Q & \partial_a Q_H\end{pmatrix}, \tag{A.10}$$

where of course we never form $\mathcal{A}^{-1}$ but instead use `p.fuha.blss` to solve linear systems of type $\mathcal{A}U = b$. These systems are of bordered type, and thus it is often advantageous to use bordered system solvers, see [UW17]. For the case of POs with fixed T (`p.hopf.freeT=0`), the second column of $\mathcal{A}$ in (A.10) is deleted, and consequently one additional parameter a must be freed, see §6.4 and Remark A.1b).

For the time discretization we have

$$u = (u_1, \dots, u_m) = (u(t_1), u(t_2), \dots, u(t_m)), \tag{A.11}$$

(m time slices, stored in `p.hopf.y(1:p.nu,1:m))`,

where $u_m = u1$ is redundant but convenient. To assemble $\mathcal{G}$ in (A.5) we use modifications of TOM, yielding, with $h_j = t_j - t_{j-1}$ and $u_0 := u_{m-1}$,

$$(\mathcal{G}(u))_j = -h_{j-1}^{-1}M(u_j - u_{j-1}) - \frac{1}{2}T(G(u_j) + G(u_{j-1})),\quad \mathcal{G}_m(u) = u_m - u_1. \tag{A.12}$$

The Jacobian is $\partial_u\mathcal{G} = A_1$, where we set, as it is also used for the Floquet multipliers with free γ (see [Uec19, §2.4] and §A.2),

$$A_\gamma = \begin{pmatrix} M_1 & 0 & 0 & 0 & \dots & -H_1 & 0\\ -H_2 & M_2 & 0 & 0 & \dots & 0 & 0\\ 0 & -H_3 & M_3 & 0 & \dots & 0 & 0\\ \vdots & \dots & \ddots & \ddots & \ddots & \vdots & \vdots\\ 0 & \dots & \dots & \ddots & \ddots & 0 & 0\\ 0 & \dots & \dots & 0 & -H_{m-1} & M_{m-1} & 0\\ -\gamma I & 0 & \dots & \dots & \dots & 0 & I\end{pmatrix}, \tag{A.13}$$

where $M_j = -h_{j-1}^{-1}M - \frac{1}{2}T\partial_u G(u_j)$, $H_j = -h_{j-1}^{-1}M + \frac{1}{2}T\partial_u G(u_{j-1})$, and I is the $n_u \times n_u$ identity matrix.[10] The Jacobians $\partial_u G \in \mathbb{R}^{n_u\times n_u}$ in M_j, H_j are computed as for steady state problems, e.g., via `p.fuha.sGjac`, or by `numjac` if `p.sw.jac`=0 (but still locally in time). The main interfaces between the TOM functions and the standard `pde2path` setup for steady problems are the functions

`hosrhs` and `hosjac`, (A.14)

[10]In (A.12) and (A.13) we assume that M is regular, i.e., (A.5) does not contain algebraic constraints as for instance for the 2nd order system formulation of the KS in `kspbc2`; see also Remark A.1 for this case.

which essentially call the `pde2path` functions `resi` and `getGu` at each time slice. To deal with non-autonomous problems, they also pass the time $t \in [0,1]$ and the period T via `p.t` and `p.T`, respectively, such that these can be used in, e.g., `sG` and `sGjac`, see, e.g., `fofu` in §6.4.

For $Q_H(u(\cdot,\cdot),\lambda,w,a)=0$, the user must provide a function handle `p.hopf.qfh`, similar to $0 = Q(u,\lambda,w) =$ `p.fuha.qf` for the steady case. Moreover, when switching to a Hopf branch (and if `p.nc.nq` was greater 0 for the steady continuation), the user has to drop the stationary constraints, i.e., reset `p.nc.nq=0` and `p.nc.ilam=p.nc.ilam(1)` to just the primary active parameter, while the other active parameters $a \in \mathbb{R}^{n_H}$ for Q_H should be set in `p.hopf.ilam` $\in \mathbb{N}^{n_H}$, which now acts as a pointer to these 'secondary' active parameters. Finally, the user must provide a function handle in `p.hopf.qfhjac` to a function that returns $\partial_u Q_H$ from the last line in (A.10). On the other hand, $\partial_T\mathcal{G}, \partial_\lambda\mathcal{G}, \partial_a\mathcal{G}$, and $\partial_T Q_H, \partial_\lambda Q_H$ and $\partial_a Q_H$ in (A.10) are cheap from numerical differentiation and hence taken care of automatically.

Remark A.1. a) If (A.5) contains algebraic constraint components as in `kspbc2`, then we modify (A.12). For instance, if the second component is algebraic, then we (automatically, in `tomassemF` and `tomassempbc`) set $(\mathcal{G}(u)_2)_j = -\frac{1}{2}TG(u_j)$, and accordingly also modify (A.13).

b) For non–autonomous problems it may be useful or necessary to drop the t–PC (A.6), which is flagged by `p.hopf.pc=0` (default=1). Then we must also decrease the number of unknowns by 1, for instance by fixing T via `p.hopf.freeT=0`, or, in case that `nqh`>0, by choosing `nqh-1` free auxiliary parameters a instead of `nqh` many. On the other hand, as noted above, for `p.hopf.freeT=0` and `p.hopf.pc=1`, one additional auxiliary parameter must be free, i.e., `length(p.hopf.ilam)=nqh+1`. See §6.4 for examples. ⌋

A.2 Floquet multipliers, and bifurcation from periodic orbits

The Floquet multipliers γ of a periodic orbit u_H are obtained from finding nontrivial solutions (v,γ) of the variational boundary value problem

$$M\dot{v}(t) = -T\partial_u G(u(t))v(t), \tag{A.15}$$

$$v(1) = \gamma v(0). \tag{A.16}$$

By translational invariance of (A.5), there always is the trivial multiplier $\gamma_1 = 1$. 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 (A.5) with $u(0)=u_0$ from u_H. Thus, $\mathcal{M}(u_0)$ depends on u_0, but the multipliers γ do not. $\mathcal{M}(u_0)$ has the eigenvalues $1,\gamma_2,\dots,\gamma_{n_u}$, where $\gamma_2,\dots,\gamma_{n_u}$ are the multipliers of the linearized Poincaré map $\Pi(\cdot;u_0)$, 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, see, e.g., [Kuz04, Theorem 1.6]. Thus, a necessary condition 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 $\gamma_{\text{crit}} = \gamma_2$ (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 $T(\lambda)/\tilde{T}(\lambda) \notin \mathbb{Q}$, then $\mathbb{R} \ni t \mapsto \tilde{u}(t)$

is dense in certain tori.

Numerics for (iii) are difficult even for low dimensional ODEs, but there are various algorithms for (i),(ii), and below we explain the simple ones so far used in `pde2path`. First we are interested in the computation of the multipliers. Using the same discretization for v as for u, it follows that γ and $v = (v_1, \dots, v_m)$ have to satisfy

$$v_1 = M_1^{-1} H_1 v_{m-1}, \quad v_2 = M_2^{-1} H_2 v_1, \quad \dots, \quad v_{m-1} = M_{m-1}^{-1} H_{m-1} v_{m-2}, \quad v_m = \gamma v_1, \tag{A.17}$$

for some $\gamma \in \mathbb{C}$. Thus, $\mathcal{M}(u_{j_0})$ can be obtained from certain products involving the M_j and the H_j, for instance[11]

$$\mathcal{M}(u_1) = M_1^{-1} H_1 M_{m-1}^{-1} H_{m-1} \cdots M_2^{-1} H_2. \tag{A.18}$$

Thus, a simple way to compute the γ_j is to compute the product (A.18) and subsequently (a number of) the eigenvalues of $\mathcal{M}(u_1)$. We call this **FA1** (Floquet Algorithm 1, implemented in `floq`), and using

$$\text{err}_{\gamma_1} := |\gamma_1 - 1| \tag{A.19}$$

as a measure of accuracy we find that this works fast and accurately for our dissipative examples. Typically $\text{err}_{\gamma_1} < 10^{-10}$, although at larger amplitudes of u_H, and if there are large multipliers, this may go up to $\text{err}_{\gamma_1} \sim 10^{-8}$, which is the (default) tolerance we require for the computation of u_H itself. Thus, in the software we give a warning if err_{γ_1} exceeds a certain tolerance tol_{fl}. However, for the optimal control example in §4, where we naturally have multipliers γ_j with $|\gamma_j| > 10^{30}$ and larger, **FA1** completely fails to compute any meaningful multipliers.

More generally, in for instance [FJ91, Lus01] it is explained that methods based directly on (A.18)

- 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, according to (A.19). See also [Kre01, Kre06] for similar ideas and results. We thus provide an algorithm **FA2** (Floquet Algorithm 2, implemented in `floqps`), which, based on `pqzschur` from [Kre01], computes a periodic Schur decomposition of the matrices involved in (A.18), from which we immediately obtain the multipliers, see [Uec19] for details.

Remark A.2. Also for $n_H > 0$, the computation of Floquet multipliers is based on (A.13), i.e., ignores the Hopf-constraints Q_H. Since these constraints are typically used to eliminate neutral directions, ignoring these typically leads to Floquet multipliers close to 1, additional to the trivial multiplier 1 from (time–) translational invariance. This may lead to wrong stability assessments of periodic orbits (see [RU17, §4] for an example), which however usually can identified by suitable inspection of the multipliers. ⌋

Here, additional to [Uec19], we give the (somewhat preliminary, see Remark 3.1) algorithms for branch switching for the case of $\gamma_{\text{crit}} = \pm 1$. First, the (simple) localization of a BP is done in

[11]In [Uec19, (2.40)] we used $\mathcal{M}(u_{m-1})$, but $\mathcal{M}(u_1)$ seems more convenient for branch–switching

hobifdetec.m via bisection. For the case $\gamma_{\text{crit}} = 1$ with associated eigenvector v_1 (i.e., $\gamma_{\text{crit}} \approx 1$ but not equal to the trivial multiplier, and hence $v_1 \neq \partial_t u_H(0)$) we then use (A.18), i.e.,

$$v_2 = M_2^{-1} H_2 v_1, \quad \dots, \quad v_{m-1} = M_{m-1}^{-1} H_{m-1} v_{m-2}, \tag{A.20}$$

and additionally $v_m = v_1$, to obtain a tangent predictor $V = (v_1, v_2, \dots, v_{m-1} | v_m)$ for the bifurcating branch.

For the case $\gamma_{\text{crit}} = -1$ we double the period T, i.e., set (recall that $t_1 = 0$)

$$t_{\text{new}} = \frac{1}{2}(t_1, t_2, t_3, \dots, t_{m-1}, 1 + t_1, 1 + t_2, \dots, 1 + t_{m-1}), \tag{A.21}$$

redefine $m = 2m$, and use $(V, -V|v_1)$ with V from A.20 as predictor. The pertinent function is poswibra.m, which, besides the orbit and branch point, and the new directory and the (initial) step length ds, can take some additional arguments aux. For instance, aux.sw=1 *forces* poswibra to take γ_{crit} from near 1, while aux.sw=-1 takes γ_{crit} from near -1. This is sometimes necessary because there may be γ_j close to both, ± 1, and the bisection for the localization does not distinguish these (or even multipliers elsewhere near the unit circle).

B Data structure and function overview

The Hopf setting naturally reuses and extends the stationary pde2path setting explained in, e.g., [dWDR+20]. As usual, here we assume that the problem is described by the struct p, and for convenience list the main subfields of p in Table 9.

Table 9: Main fields in the structure p for steady problems, see [dWDR+20] for more details.

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
pdeo	OOPDE data if OOPDE is used	mesh	mesh data (if the pdetoolbox is used)
plot	switches and controls for plotting	file	switches etc for file output
bel	controls for lssbel (bordered elimination)	ilup	controls for lssAMG (ilupack parameters)
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		

For the continuation of time-periodic orbits, the field p.hopf contains the pertinent data; it is typically created and filled by calling p=hoswibra(..). This inter alia calls p=hostanparam(p,aux), which can be used as a reference for the default values of the Hopf parameters. The unconstrained Hopf setting does not need any user setup additional to the functions such as p.fuha.sG, p.fuha.sGjac already needed for stationary problems. In case of Hopf constraints, the user has to provide two function handles in p.hopf.qfh and p.hopf.qfhder to functions which compute Q_H from (A.9) and the last row of $\mathcal{A}$ from (A.10), respectively. 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. Table 10 gives an overview of p.hopf, and Table 11 lists the main Hopf orbit related functions.

Table 10: Standard (and additional, at bottom) entries in `p.hopf`.

field	purpose
y	for `p.sw.para=4`: unknowns in the form $(u = (u_1, \dots, u_m) = (u(t_1), u(t_2), \dots, u(t_m))$, ($m$ time slices, y=$n_u \times m$ matrix); for `p.sw.para=3`: y augmented by $\tilde{y}$ and T, λ ($(2n_u{+}2) \times m$ matrix), see [Uec19].
y0d	for `p.sw.para=4`: $M\dot{u}_0$ for the phase condition (A.6), ($n_u \times m$ matrix); for `p.sw.para=3`: $M\dot{u}_0(0)$ for the phase condition [Uec19, (36)], ($2n_u$+2 vector).
y0dsw	(for `p.sw.para=4`) controls how $\dot{u}_0$ in (A.6) is computed: 0 for using the PDE (A.5), 2 for using FD (default).
pcfac	weight for the phase condition (A.6), default=10
tau	tangent, for `p.sw.para=4`, $(mn_u + 2 + n_H) \times 1$ vector, see third line in (A.10)
ysec	for `p.sw.para=3`, secant between two solutions (y_0, T_0, λ_0), (y_1, T_1, λ_1), $(2n_u{+}2) \times m$ matrix
sec	if sec=1, then use secant tau (instead of tangent) predictor for `p.sw.para=4`
t, T, lam	time discretization vector, current period and param.value
xi,tw,qw	weights for the arclength (A.7), xi=$\xi_{\rm H}$, tw=w_T, qw=w_a;
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-comp. during cont., 1 to use `floq`, 2 to use `floqps`
nfloq	# of multipliers (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
pcheck	if 1, then compute residual in hoswibra (predictor check)
bisec	# of bisection used for BP localizations
	Additional entries in case of Hopf constraints
ilam	pointer to the n_Q additional active parameters in `p.u(p.nu+1:end)`; the pointer to the primary active parameter is still in `p.nc.ilam(1)`.
qfh, qfhder	(handles to) functions returning the Hopf constraints $Q_H(U)$ and the derivatives (last lines of H in (A.9) and $\mathcal{A}$ in (A.10), respectively).
spar,kwnr	index of speed parameter, and spatial wave-nr for TW continuation, set in twswibra

Table 11: Overview of main functions related to Hopf bifurcations and periodic orbits; see `p2phelp` for argument lists and more comments.

name	purpose, remarks
hoswibra	branch switching at Hopf bifurcation point, see comments below
twswibra	branch switching at Hopf bifurcation point to Traveling Wave branch (which is continued as a rel.equilibrium)
hoswipar	change the active continuation parameter, see also swiparf
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 function called after each continuation step
plotfloq	plot previously computed multipliers
hpcontini	init Hopf point continuation
hpcontexit	exit Hopf point continuation
hpjaccheck	check user implementation of (3.6) against finite differences
hploc	use extended system (3.3) for Hopf point localization
hobifdetec	detect bifurcations *from* Hopf orbits, and use bisection for localization, based on multipliers
poswibra	branch switching *from* Hopf orbits
hobifpred	compute predictor for branch switching *from* Hopf orbits
hotintxs	time integrate (1.3) 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
hopftref	refine the t-mesh in the arclength setting at user specified time t^*
hogradinf	convenience function returning the time t^* where $\|\partial_t u(\cdot,t)\|_\infty$ is maximal; may be useful for hopftref.
initwn	init vectors for computation of initial guess for spectral shifts ω_j
hogetnf	compute initial guesses for dlam, al from the normal form coefficients of bifurcating Hopf branches, see [Uec19, (16)]
hocont	main continuation routine; called by cont if p.sol.ptype>2
hostanparam	set standard parameters
hostanopt	set standard options for hopf computations
hoinistep	perform 2 initial steps and compute secant, used if `p.sw.para=3`
honloopext,honloop	the arclength Newton loop, and the Newton loop with fixed λ
sety0dot	compute $\dot u_0$ for the phase condition (A.6)
tomsol	use TOM to compute periodic orbit in p.sw.para=3 setting.
tomassemG	use TOM to assemble $\mathcal{G}$, see [Uec19, (26)]; see also `tomassem, tomassempbc`
gethoA	put together the extended Jacobian $\mathcal{A}$ from [Uec19, (27)]
hopc	the phase condition ϕ from[Uec19, (19)], and $\partial_u\phi$.
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
poiniguess	generate initial guess for periodic orbit continuation based on `hopf` data structures, but without need of a HBP. Alternative to `hoswibra`, see §6.4.
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 p.sw.para=3, see also `hobc` and `hobcjac`

Besides `cont`, the functions `initeig`, `hoswibra`, `poswiba`, `hoplot`, `twswibra`, `floqap`, `floqplot`, `hotintxs`, `tintplot*d`, and `hopftref` are most likely to be called directly by the user, and `hobra` (the branch data) and `hostanufu` (called after each continuation step) are likely to be adapted by the user. As usual, all functions in Table 11 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`=30: number of (equally spaced) initial mesh-points in $t\in[0,1]$ (might be adaptively refined by TOM for `p.sw.para=3`, or via `hopftref` or `uhopftref` for `p.sw.para=4`).
- `aux.hodel`=1e-4: used for the finite differences in `hogetnf`.
- `aux.al, aux.dlam` (no preset): these can be used to pass a guess for α and δ_λ in (A.3) and thus circumvent `hogetnf`; useful for quasilinear problems and for problems with constraints (for

which `hogetnf` will not work), or more generally when the computation of α, δ_λ via `hogetnf` seems to give unreliable results.

- `aux.z`: The coefficients $z_1, \dots, z_m$ in the ad hoc modification (6.19) of (A.3) used for Hopf points of higher multiplicity m.

For the other functions listed above we refer to the m-files for description of their arguments, and to the demo directories for examples of usage and customization.

References

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

[BKT90] D. Barkley, M. Kness, and L. S. Tuckerman. Spiral-wave dynamics in a simple model of excitable media: the transition from simple to compound rotation. *Phys. Rev. A (3)*, 42(4):2489–2491, 1990.

[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.

[BvVF17] P.-L. Buono, L. van Veen, and E. Frawley. Hidden symmetry in a Kuramoto-Sivashinsky initial-boundary value problem. *Internat. J. Bifur. Chaos Appl. Sci. Engrg.*, 27(9), 2017.

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

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

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

[dWDR$^+$20] H. de Witt, T. Dohnal, J.D.M. Rademacher, H. Uecker, and D. Wetzel. pde2path - Quickstart guide and reference card, 2020.

[dWU19] H. de Witt and H. Uecker. Infinite time–horizon spatially distributed optimal control problems with pde2path – algorithms and tutorial examples, arxiv:1912.11135, 2019.

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

[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, 2008.

[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.

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

[GU17] D. Grass and H. Uecker. Optimal management and spatial patterns in a distributed shallow lake model. *Electr. J. Differential Equations*, 2017(1):1–21, 2017.

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

[JSW89] W. Jahnke, W. E. Skaggs, and A. T. Winfree. Chemical vortex dynamics in the Belousov-Zhabotinskii reaction and in the two-variable oregonator model. *J. Phys. Chem*, 93(2):740–749, 1989.

[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.

[Kie79] H. Kielhöfer. Hopf bifurcation at multiple eigenvalues. *Arch. Rational Mech. Anal.*, 69(1):53–83, 1979.

[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.

[KT76] Y. Kuramoto and T. Tsuzuki. Persistent propagation of concentration waves in dissipative media far from thermal equilibrium. *Prog. Theoret. Phys.*, 55(2):356–369, 1976.

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

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

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

[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.

[PET94] J. Paullet, B. Ermentrout, and W. Troy. The existence of spiral waves in an oscillatory reaction-diffusion system. *SIAM J. Appl. Math.*, 54(5), 1994.

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

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

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

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

[Siv77] G. Sivashinsky. Nonlinear analysis of hydrodynamic instability in laminar flames. I - Derivation of basic equations. *Acta Astronautica*, 4:1177–1206, 1977.

[SN10] J. Sánchez and M. Net. On the multiple shooting continuation of periodic orbits by Newton-Krylov methods. *Internat. J. Bifur. Chaos Appl. Sci. Engrg.*, 20(1):43–61, 2010.

[SN16] J. Sánchez and M. Net. Numerical continuation methods for large-scale dissipative dynamical systems. *Eur. Phys. J. Special Topics*, 225:2465–2486, 2016.

[SS06] B. Sandstede and A. Scheel. Curvature effects on spiral spectra: Generation of point eigenvalues near branch points. *PRE*, 016217:1–8, 2006.

[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.

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

[Uec19] H. Uecker. Hopf bifurcation and time periodic orbits with pde2path – algorithms and applications. *Comm. in Comp. Phys*, 25(3):812–852, 2019.

[Uec20a] H. Uecker. Pattern formation with pde2path – a tutorial, 2020.

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

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

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

[WB06] P. Wheeler and D. Barkley. Computation of spiral spectra. *SIADS*, 5:157–177, 2006.

[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.

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