Continuation of fold points, branch points, and Hopf points with constraints in `pde2path`

Hannes Uecker

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

September 6, 2023

Abstract

This somewhat technical note reviews the `pde2path` setup for continuation of fold points, branch points and Hopf points in case of constraints, and some demos for this, dealing with mass conservation and phase conditions.

Contents

1 Introduction

The `Matlab` package `pde2path` [Uec21, Uec23] computes solution branches for systems of the form

$$M_d \partial_t u = -G(u,\lambda), \tag{1a}$$
$$0 = q(u,\lambda). \tag{1b}$$

Here $u = u(x,t) \in \mathbb{R}^N$, x from some bounded domain $\Omega \subset \mathbb{R}^d$, $d = 1$, $d = 2$, or $d = 3$, (1D, 2D, 3D case), time $t \in I \subset \mathbb{R}$, and G stands for a PDE rhs including the boundary conditions (BCs); $M_d \in \mathbb{R}^{N\times N}$ is a (dynamical) mass matrix, which may be singular, and $\lambda \in \mathbb{R}^{n_p}$ denotes a parameter (vector); q denotes a (vector in $\mathbb{R}^{n_q}$, $n_q \ge 0$ of) constraint(s), the most common of which being phase conditions (PCs) in case of, e.g., periodic BCs, or mass constraints.

In `pde2path`, (1) is discretized by a finite element method (FEM), and with a slight abuse of notation we identify $u : \Omega \to \mathbb{R}^N$ (and $u(\cdot,t) : \Omega \to \mathbb{R}^N$) with the nodal values $u \in \mathbb{R}^{n_u}$ (resp. $u(t) \in \mathbb{R}^{n_u}$), $n_u = Nn_p$, where the number n_p of discretization points is usually large.

Important special points of solution branches of (1) are fold points (FPs), branch points (BPs), and Hopf points (HPs), and a useful feature of a numerical continuation and bifurcation package is the option of FP–, BP–, and HP–continuation (FPC, BPC and HPC, see also Table 1 for a list of acronyms), for which one has to free an additional parameter. Besides being important themselves due to topological changes occuring there (and new branches bifurcating at BPs and HPs), FPs, BPs or HPs for instance also often delimit the stability regions of solution branches, and hence FPC, BPC and HPC can be used to efficiently compute such stability regions in dependence of a second parameter.

In `pde2path`, BPC and HPC so far have only been used (and documented in demos) *without constraints, i.e., for $n_q = 0$ in (1).* In the present somewhat technical note we explain the setup with $n_q > 0$, and give some examples, namely FPC and BPC in Cahn–Hilliard type problems (with a mass constraint), and FPC and HPC in complex Ginzburg–Landau equations over a box with periodic BCs in one direction (q=translational PC), and over a disk (q=rotational PC). These extend the `pde2path` demos `ch` [Uec21, §6.9], `chtor` [Uec21, §10.9], and `cglpbc,cgldisk` [Uec21, §7.2]. Throughout we assume that G and q are sufficiently differentiable for all expressions to make sense, and that $0 \le n_q \ll n_u$, with in fact $n_q = 1$ in `ch,cglpbc` and `cgldisk`, and $n_q = 2$ in `chtor`.

Remark 1.1 a) Formally, we can set $\mathcal{G} = \begin{pmatrix} G \\ q \end{pmatrix}$ and $\mathcal{M}_d = \begin{pmatrix} M_d & 0 \\ 0 & 0 \end{pmatrix}$ and rewrite (1) as a system without (explicit) constraints, i.e., $\mathcal{M}_d \partial_t U = -\mathcal{G}(U, \lambda)$, where $U = (u, \beta)$, with parameter (vector) $\beta \in \mathbb{R}^{n_q}$. However, from a practical (implementation–wise) point of view we find it useful to maintain the distinction between G and q, and hence the form (1).

b) We here have the technical focus of how to set up FPC, BPC and HPC with constraints, and in the demos we only give brief remarks about the usefulness of FPC, BPC and HPC, i.e., about what we learn from these. See, e.g., [Kuz04, Chapter 8] for a general and detailed account. Other demos with FPC, BPC and/or HPC with some more remarks (but with $n_q = 0$) include, as subdirectories of `pde2path/demos/`: `acsuite/ac1D` and other subdirectories of `acsuite`, see [Uec21, Chapter 6], as simple introductory examples, but numerically expensive in 2D and 3D; `pftut/shEck`, [Uec21, §8.3.3], where BPC is used to compute the boundary of the Busse–balloon, `pftut/schnakpat`, [Uec21, §9.1], and `pftut/bruosc`, [Uec21, §9.3], where, e.g., BPC and HPC is used to find co–dimenssion-two points. The latter three give further examples how to set up functions `spjac.m`, `bpjac.m` and `hpjac.m` (see below) for systems of PDEs. ⌋

Table 1: Acronyms.

BP/FP/HP	branch/fold/Hopf point	BPC/FPC/HPC	branch–/fold–/Hopf–point continuation
BC	boundary condition	DBC/NBC/pBC	Dirichlet/Neumann/periodic BCs
FDs	finite diffences	PC	phase condition
CH	Cahn–Hilliard (problem)	cGL	complex Ginzburg Landau (equation)

2 The extended systems and the basic setup

2.1 Without constraints

For $n_q = 0$ (no constraints), the extended system used in `pde2path` for FPC, BPC and HPC are discussed in [Uec21, §3.6.1], and for convenience recalled here. For FPC we continue (in arclength)

$$H_{\mathrm{FP}}(U) = \begin{pmatrix} G(u,\Lambda) \\ G_u(u,\Lambda)\phi \\ \langle \phi, \phi \rangle - 1 \end{pmatrix} = 0 \in \mathbb{R}^{2n_u+1}, \quad U = (u, \phi, \Lambda), \tag{2}$$

where the *active* parameter vector $\Lambda = (\lambda, w)$ consists of the *new primary* parameter λ and the old primary parameter $w = \lambda_{\text{old}}$, and where here and in the following we write $G_u = \partial_u G$ where convenient. This gives $2n_u + 1$ equations in the $2n_u + 2$ unknowns $U = (u, \phi, \Lambda)$, and is combined with the usual arclength condition

$$p(U, s) := \xi \left\langle \begin{pmatrix} u_0' \\ \phi_0' \end{pmatrix}, \begin{pmatrix} u(s) - u_0 \\ \phi(s) - \phi_0 \end{pmatrix} \right\rangle_{\mathbb{R}^{2n_u}} + (1 - \xi) \langle \Lambda_0', \Lambda(s) - w_0 \rangle_{\mathbb{R}^2} - \mathrm{ds} \overset{!}{=} 0, \tag{3}$$

where $\xi > 0$ is a weight, typically $\xi = 1/n_u$, $U_0 = (u_0, \phi_0, \Lambda_0)$ is the solution in the previous step and $(u_0', \phi_0', \Lambda_0') = \frac{\mathrm{d}}{\mathrm{d}s}(u_0, \phi_0, \Lambda_0)$ is the tangent to the branch at U_0, and ds is the current continuation stepsize.

Similary, for BPC we use

$$H_{\text{BP}}(U) = \begin{pmatrix} G + \mu M_d \psi \\ G_u^T \psi \\ \|\psi\|_2^2 - 1 \\ G_{\lambda_{\text{old}}}^T \psi \end{pmatrix} = 0 \in \mathbb{R}^{2n_u+2}, \quad U = (u, \psi, \mu, \Lambda), \tag{4}$$

where ψ is an adjoint kernel vector of G_u, and for HPC we use

$$H_{\text{HP}}(U) := \begin{pmatrix} G \\ G_u \phi_r + \omega M_d \phi_i \\ G_u \phi_i - \omega M_d \phi_r \\ c^T \phi_r - 1 \\ c^T \phi_i \end{pmatrix} = 0 \in \mathbb{R}^{3n_u+2}, \quad U = (u, \phi_r, \phi_i, \omega, \Lambda), \tag{5}$$

where $\mathrm{i}\omega \in \mathrm{i}\mathbb{R}$ is the desired eigenvalue of G_u, $\phi = \phi_r + \mathrm{i}\phi_i \in \mathbb{C}^{n_u}$ is an associated eigenvector, and $c_r \in \mathbb{R}^{n_u}$ is used to fix the phase.

The Jacobian of, e.g., H_{FP} wrt to (u, ϕ, w) is

$$J_{\text{FP}} = \partial_{(u,\phi,w)} H_{\text{FP}} = \begin{pmatrix} G_u & 0 & \partial_w G \\ \partial_u (G_u \phi) & G_u & \partial_w (G_u \phi) \\ 0 & 2\phi^T & 0 \end{pmatrix} \in \mathbb{R}^{(2n_u+1)\times(2n_u+1)}. \tag{6}$$

In J_{FP}, the $n_u \times n_u$ blocks G_u and $\partial_u(G_u \phi)$ may be expensive for large n_u if done numerically by finite differences (FDs), but at least for semilinear PDEs these terms can easily (and quickly) be handled by user–provided functions, see the examples below. On the other hand, since $\dim(w) = 1 + n_q$ is small (here $\dim(w) = 1$), all w derivatives can be done quickly by FDs. The Jacobians of H_{BP} and H_{HP} are slightly more complicated, but with similar terms, discussed in §2.2 for $n_q > 0$.

In (6), and similarly in the following, we omit the derivative of H_{FP} wrt the new primary parameter λ, which together with the derivative of the arclength condition is automatically appended to $\partial_U H_{\text{FP}}$ by `pde2path`. Thus, for the arclength continuation of FPs, `pde2path` uses the Jacobian

$$\mathcal{A} = \begin{pmatrix} J_{\text{FP}} & \partial_\lambda H_{\text{FP}} \\ \partial_{(u,\phi,w)} p(U, s) & \partial_\lambda p(U, s) \end{pmatrix}, \tag{7}$$

but the user does not need to deal with this, and similar for BPC and HPC. The Jacobian (7) is then regular at a (quadratic) fold point, and the related Jacobians for BPC and HPC are regular at simple branch points and simple Hopf points, respectively.

In pde2path, FPC is initialized by `p=spcontini(dir,fpt,inewpar,newdir)` (as in `spectral continuation initialization`), where `dir/fpt` is the fold point to continue, `inewpar` is the (index of the) new primary parameter, and `newdir` is the directory for saving the results of the FPC.This automatically reorganizes variables and dimensions (also in case $n_q > 0$), and similarly for `bpcontini` and `hpcontini`, and the user only has to take into account the hints on building Jacobians given below. To switch back from FPC to regular continuation in the original parameter λ_{old}, use `p=spcontexit(dir,pt,newdir)`. For exiting BPC and HPC, similarly use `bpcontexit` and `hpcontexit`.[1]

2.2 With constraints

FP continuation. For $n_q > 0$, we write the extended system for FPC as

$$H_{\text{FP}}(U) = \begin{pmatrix} G \\ G_u\phi + G_{\tilde{w}}\tilde{\phi} \\ q \\ q_u\phi + q_{\tilde{w}}\tilde{\phi} \\ \langle \Phi, \Phi \rangle - 1 \end{pmatrix} \in \mathbb{R}^{2n_u+2n_q+1}, \quad U = (u, \phi, \Lambda, \tilde{\phi}), \quad \Lambda = (\lambda, w), \tag{8}$$

with now $w = (\lambda_{\text{old}}, \tilde{w}) \in \mathbb{R}^{1+n_q}$, where again λ_{old} is the old primary active parameter, $\tilde{w} \in \mathbb{R}^{n_q}$ are the old secondary active parameters, and where $\Phi = (\phi, \tilde{\phi}) \in \mathbb{R}^{n_u+n_q}$ is in the kernel of $\begin{pmatrix} \partial_u G & \partial_{\tilde{w}} G \\ \partial_u q & \partial_{\tilde{w}} G \end{pmatrix}$.
Accordingly, again omitting the derivative wrt the new primary λ, the Jacobian of H_{FP} reads[2]

$$J_{\text{FP}} = \begin{pmatrix} G_u & 0 & \partial_{\lambda_{\text{old}}} G & \partial_{\tilde{w}} G & 0 \\ \partial_u(G_u\phi + G_w\tilde{\phi}) & G_u & \partial_{\lambda_{\text{old}}}(G_u\phi + G_{\tilde{w}}\tilde{\phi}) & \partial_{\tilde{w}}(G_u\phi + G_{\tilde{w}}\tilde{\phi}) & G_{\tilde{w}} \\ q_u & 0 & \partial_{\lambda_{\text{old}}} & \partial_{\tilde{w}} q & 0 \\ \partial_u(q_u\phi + q_{\tilde{w}}\tilde{\phi}) & q_u & \partial_{\lambda_{\text{old}}}(q_u\phi + q_{\tilde{w}}\tilde{\phi}) & \partial_{\tilde{w}}(q_u\phi + q_{\tilde{w}}\tilde{\phi}) & q_{\tilde{w}} \\ 0 & 2\phi^T & 0 & 0 & 2\tilde{\phi}^T \end{pmatrix}. \tag{9}$$

Thus $J_{\text{FP}} \in \mathbb{R}^{(2n_u+2n_q+1)\times(2n_u+2n_q+1)}$ where the last three colums (of widths $1, n_q$ and n_q) are cheap via finite differences.

Typical examples of constraints are (for now assuming a scalar PDE)

$$(0 \overset{!}{=}) q(u) = \int_\Omega u \,\mathrm{d}x - m = \mathtt{a} * \mathtt{u} - \mathtt{m} \text{ (mass contraint, with mass parameter } m), \tag{10}$$

$$(0 \overset{!}{=}) q(u) = \int_\Omega (\partial_x u_0) u \,\mathrm{d}x = \mathtt{u0x}' * \mathtt{u} \text{ (1D phase constraint, where } u_0 \text{ is a reference profile)}, \tag{11}$$

using the continuous notation on the left, and the discrete `Matlab` notation on the right, with $\mathtt{a} = \mathtt{sum}(\mathtt{M}, 1)$ the column sum of the FEM mass-matrix `M`, and this motivates the following comments: Although numerical derivatives are also possible via `p.sw.qjac=0`, we generally recommend to treat

[1] There also is the option `p=spcontexit(dir,pt,newdir,outfu)`, where `outfu` can be set to a branch-output function. This may be needed if for FPC (or BPC or HPC) a different output function than for the regular continuation was used, see §4 for an example.

[2] we mainly display this obvious computation here for reference in the function `getGufoco.m` which assembles J_{FP}, and similarly for J_{BP} (`getGubpco.m`) and J_{HP} (`getGuhpco.m`) below

q_u via `p.sw.qjac=1` and `p.fuha.qjac=@myqjac` which implements $\partial_u q$. For the above examples,

$$q_u = \mathtt{a} \text{ (mass contraint)}, \tag{12}$$
$$q_u = \mathtt{u0x}' \text{ (1D phase constraint)}. \tag{13}$$

These are simple, and, perhaps more importantly, in case of integral constraints always dense, i.e., full row vectors, and hence cannot be treated efficiently by, e.g., `numjac`. Moreover, and we again believe typically, they are *linear* in u, and hence the (possibly expensive) second derivatives $\partial_u(q_u\phi)$ in (9) are *actually zero*, as are all other second derivatives of these constraints.

Remark 2.1 a) In summary, to generate (9), we may proceed as follows (assuming that the user knows the basic setup of the rhs G and q, and of $G_u = \partial_u G$ and $q_u = \partial_u q$; otherwise see [Uec21, Chapter 6], or the demos below):[3]

1. For semilinear PDEs, $\partial_u(G_u\phi)$ is often easy to implement in a function `Guphiu=myspjac(p,u,r)`, see the examples below. Then set `p.sw.spjac=1` and `p.fuha.spjac=@myspjac`.[4] For testing, call `spjaccheck(p)`, which compares the implementation in `myspjac` with the results of `numjac`.
2. If $\partial_u(G_u\phi)$ is not easy, and `p.sw.jac=1` (G_u is done by `p.fuha.sGjac`), then set `p.sw.spjac=0`; $\partial_u(G_u\phi)$ is then quickly (at least in 1D) and (usually) reliably generated by `numjac`.
3. If `p.sw.jac=0` (G_u is already done by `numjac`), then `p.sw.spjac=0` needs further care: In this case, `numjac` for $\partial_u(G_u\phi)$ must often be run with a rather larger `thresh`(old) vector, and our default setting (in `getGuphiu.m`) is `thresh`$\equiv$`1e3`. To see what increment δ (usually about $10^{-6}*$`thresh`) is then actually used for FDs in `numjac`, set `p.sw.verb` > 2, which triggers a call to `numjacinfo` in `getGuphiu`.[5] For flexibility, the user may also set `p.nc.njthreshsp` to a chosen (scalar) value, where however `p.nc.njthreshsp`$<10^2$ often gives bad Jacobians. Thus, if FPC with `p.sw.jac=0` and `p.sw.spjac=0` fails, we recommend trying a larger `p.nc.njthreshsp`, e.g. `p.nc.njthreshsp=1e4`.
4. Due to 3., instead of using `numjac`, the user can also set `p.sw.spjac=2`; $\partial_u(G_u\phi)$ is then generated by "brute force" FDs with stepsize $\delta =$ `p.nc.del`, which however can be quite slow.
5. If q is linear in u, and hence $\partial_u(q_u\phi) = 0$, then nothing needs to be done; $\partial_u(q_u\phi)$ is computed by `quuphi.m` which by default is set to the (dummy) library function `quuphi.m`, which returns $0 \in \mathbb{R}^{n_q\times n_u}$. If $\partial_u(q_u\phi) \neq 0$ but "easy to implement", then make a local copy of `quuphi.m` and modify accordingly.
6. If $\partial_u(q_u\phi) \neq 0$ and is difficult to implement, then set (the switch) `p.sw.spqjac=0` (overriding the default `p.sw.spqjac=1`), in which case $\partial_u(q_u\phi)$ is approximated by finite differences.

b) Additional to `p.nc.njthreshsp` in 3., we (newly!) introduced `p.nc.njthresh` as (possible) threshold for `numjac` for $\partial_u G$. This so far was set to $\delta :=$ `p.nc.del` (the FD stepsize), and this still applies if `p.nc.njthresh` is not set. However, in some cases it seems necessary to use a δ somewhat larger than the default $\delta = 10^{-4}$, and in this case it may be necessary to independently set `p.nc.njthresh`. See the comments in the demo–scripts below. $\rfloor$

[3]The demos provide the pertinent functions `sGjac` and `spjac` (and `bpjac` and `hpjac`) and then use the fastest option `p.sw.jac=1` and `p.sw.spjac=1` for Jacobians. However, to experiment with other settings, you can change these switches (e.g., by uncommenting the indicated lines in the scripts) to use numerical Jacobians, and get the same branches, but slower.

[4]In practice, we usually simply use `p.fuha.spjac=@spjac` and put the function `spjac.m` *in the current directory.*

[5]If `p.sw.jac=0` and `p.sw.verb` > 3, then `numjacinfo` is also called in `getGupde` for the computation of $\partial_u G$.

BP continuation. Using a similar sorting like in (8), we write the extended system for BPC with $n_q > 0$ as

$$H_{\rm BP}(U) = \begin{pmatrix} G+\mu M\psi \\ G_u^T\psi + q_u^T\tilde{\psi} \\ q \\ G_{\tilde{w}}^T\psi + q_{\tilde{w}}\tilde{\psi} \\ \langle \Psi, \Psi\rangle - 1 \\ \left(G_{\lambda_{\rm old}}^T, q_{\lambda_{\rm old}}^T\right)\Psi \end{pmatrix} \in \mathbb{R}^{2n_u+2n_q+2}, \quad U=(u,\psi,\Lambda,\mu,\tilde{\psi}), \quad \Lambda=(\lambda,w), \tag{14}$$

with again $w=(\lambda_{\rm old},\tilde{w})\in\mathbb{R}^{1+n_q}$, and where now $\Psi=(\psi,\tilde{\psi})\in\mathbb{R}^{n_u+n_q}$ is in the kernel of $\begin{pmatrix}\partial_u G & \partial_{\tilde{w}}G\\ \partial_u q & \partial_{\tilde{w}}q\end{pmatrix}^T = \begin{pmatrix} G_u^T & q_u^T \\ G_{\tilde{w}}^T & q_{\tilde{w}}^T\end{pmatrix}$. Hence, the Jacobian of $H_{\rm BP}$ reads

$$J_{\rm BP}=\begin{pmatrix} G_u & \mu M_d & \partial_{\lambda_{\rm old}}G & \partial_{\tilde{w}}G & M_d\psi & 0\\ \partial_u(G_u^T\psi+q_u^T\tilde{\psi}) & G_u^T & \partial_{\lambda_{\rm old}}(G_u^T\psi+G_{\tilde{w}}\tilde{\psi}) & \partial_{\tilde{w}}(G_u^T\psi+G_{\tilde{w}}\tilde{\psi}) & 0 & q_u^T\\ q_u & 0 & \partial_{\lambda_{\rm old}}q & \partial_{\tilde{w}}q & 0 & 0\\ \partial_u(G_{\tilde{w}}^T\psi+q_{\tilde{w}}^T\tilde{\psi}) & G_{\tilde{w}}^T & \partial_{\lambda_{\rm old}}(G_{\tilde{w}}^T\psi+q_{\tilde{w}}\tilde{\psi}) & \partial_{\tilde{w}}(G_{\tilde{w}}^T\psi+q_{\tilde{w}}\tilde{\psi}) & 0 & q_{\tilde{w}}^T\\ 0 & 2\psi^T & 0 & 0 & 0 & 2\tilde{\psi}^T\\ \partial_u(G_{\lambda_{\rm old}}^T\psi+q_{\lambda_{\rm old}}^T\tilde{\psi}) & G_{\lambda_{\rm old}}^T & \partial_{\lambda_{\rm old}}(G_{\lambda_{\rm old}}^T\psi+q_{\lambda_{\rm old}}^T\tilde{\psi}) & \partial_{\tilde{w}}(G_{\lambda_{\rm old}}^T\psi+q_{\lambda_{\rm old}}^T\tilde{\psi}) & 0 & q_{\lambda_{\rm old}}^T \end{pmatrix}. \tag{15}$$

Thus, $J_{\rm BP}\in\mathbb{R}^{2(n_u+n_q+1)\times 2(n_u+n_q+1)}$, and similar remarks as for (9) apply: The (possibly) expensive term is $\partial_u(G_u^T\psi+q_u\tilde{\psi})$, while all others can be easily and quickly done by FDs, for instance also using $\partial_u(G_{\lambda_{\rm old}}^T\psi+q_{\lambda_{\rm old}}^T\tilde{\psi})=\partial_{\lambda_{\rm old}}(\psi^T G_u+\tilde{\psi}^T q_u)$. If as in (12) and (13) we assume that q is linear in u, then $\partial_u(q_u^T\tilde{\psi})=0$, and for this we provide the (dummy) function `quupsi.m`; if $\partial_u(q_u^T\tilde{\psi})\neq 0$, then implement this in a local copy of `quupsi.m`. Thus, it mainly remains to compute $\partial_u(G_u^T\psi)$, for which the analog of Remark 2.1 applies: :

- Settings for `spjac`: If possible, supply a function `mybpjac.m` and set `p.fuha.spjac=@mybpjac` and `p.sw.spjac=1`. Otherwise, if `p.sw.jac=1`, then set `p.sw.spjac=0`.
  If `p.sw.jac=0`, try `p.sw.spjac=0`. If that fails, then try, e.g., `p.nc.njthreshsp = 1e4` or larger, or ultimately the fallback `p.sw.spjac=2` (slowest option).

HP continuation. The extended system for HPC with $n_q > 0$ reads

$$H_{\rm HP}(U) = \begin{pmatrix} G \\ G_u\phi_r+\omega M\phi_i+G_{\tilde{w}}\tilde{\phi}_r \\ G_u\phi_i-\omega M\phi_r+G_{\tilde{w}}\tilde{\phi}_i \\ q \\ q_u\phi_r+q_{\tilde{w}}\tilde{\phi}_r \\ q_u\phi_i+q_{\tilde{w}}\tilde{\phi}_i \\ C^T\Phi_r-1 \\ C^T\Phi_i \end{pmatrix} \in\mathbb{R}^{3(n_u+n_q)+2}, \quad U=(u,\phi_r,\phi_i,\Lambda,\omega,\tilde{\phi}_r,\tilde{\phi}_i), \quad \Lambda=(\lambda,w), \tag{16}$$

with again $w{=}(\lambda_{\rm old},\tilde w)\in\mathbb{R}^{1+n_q}$, and where now $\Phi{=}(\phi,\tilde\phi){\in}\mathbb{C}^{n_u+n_q}$, $\Phi=\Phi_r{+}{\rm i}\Phi_i$ with $\Phi_r,\Phi_i\in\mathbb{R}^{n_u+n_q}$, $\begin{pmatrix}G_u & G_{\tilde w}\\ q_u & q_{\tilde w}\end{pmatrix}\Phi={\rm i}\omega\Phi$, and where $C=(c,\tilde c)\in\mathbb{R}^{n_u+n_q}$ fixes the phase of Φ and can be chosen as Φ_r at initialization of the HPC. The Jacobian of $H_{\rm HP}$ reads

$$J_{\rm HP}{=}\begin{pmatrix}G_u&0&0&\partial_{\lambda_{\rm old}}G&\partial_{\tilde w}G&0&0&0\\ \partial_u(G_u\phi_r{+}G_{\tilde w}\tilde\phi_r)&G_u&\omega M&\partial_{\lambda_{\rm old}}(G_u\phi_r{+}G_{\tilde w}\tilde\phi_r)&\partial_{\tilde w}(G_u\phi_r{+}G_{\tilde w}\tilde\phi_r)&M\phi_i&G_{\tilde w}&0\\ \partial_u(G_u\phi_i{+}G_{\tilde w}\tilde\phi_i)&-\omega M&G_u&\partial_{\lambda_{\rm old}}(G_u\phi_i{+}G_{\tilde w}\tilde\phi_i)&\partial_{\tilde w}(G_u\phi_i{+}G_{\tilde w}\tilde\phi_i)&-M\phi_r&0&G_{\tilde w}\\ q_u&0&0&\partial_{\lambda_{\rm old}}q&\partial_{\tilde w}q&0&0&0\\ \partial_u(q_u\phi_r{+}q_{\tilde w}\tilde\phi_r)&q_u&0&\partial_{\lambda_{\rm old}}(q_u\phi_r{+}q_{\tilde w}\tilde\phi_r)&\partial_{\tilde w}(q_u\phi_r{+}q_{\tilde w}\tilde\phi_r)&0&q_{\tilde w}&0\\ \partial_u(q_u\phi_i{+}q_{\tilde w}\tilde\phi_i)&0&q_u&\partial_{\lambda_{\rm old}}(q_u\phi_i{+}q_{\tilde w}\tilde\phi_i)&\partial_{\tilde w}(q_u\phi_i{+}q_{\tilde w}\tilde\phi_i)&0&0&q_{\tilde w}\\ 0&c^T&0&0&0&0&\tilde c^T&0\\ 0&0&c^T&0&0&0&0&\tilde c^T\end{pmatrix}. \tag{17}$$

Thus, $J_{\rm HP}\in\mathbb{R}^{(3(n_u+n_q)+2)\times(3(n_u+n_q)+2)}$ and similar remarks as for $J_{\rm FP}$ and $J_{\rm BP}$ apply: The (possibly) expensive terms are the 2nd and 3rd $n_u\times n_u$ blocks in the first column, and the 4th and 5th $n_q\times n_u$ blocks, while all other can be done by FDs. However, if again we assume that q is linear in u, then $\partial_u(q_u\phi_r)=\partial_u(q_u\phi_i)=0$, and similar to above we provide the functions `quuphir.m` and `quuphii.m` to "compute" this; if $\partial_u(q_u\phi_r)\ne0$ or $\partial_u(q_u\phi_i)\ne0$, make suitable local copies of these functions. It essentially remains to generate $(\partial_u(G_u\phi_r),\partial_u(G_u\phi_i))$, and the analog of Remark 2.1 again applies:

- Settings for `spjac`: If possible, supply a function `myhpjac.m` and set `p.fuha.spjac=@myhpjac` and `p.sw.spjac=1`. This can be tested by `hpjaccheck`.
  If `myhpjac.m` is not available, and `p.sw.jac=1`, then set `p.sw.spjac=0`.
  If HPC with `p.sw.jac=0` and `p.sw.spjac=0` fails, then try, e.g. `p.nc.njthreshsp` = `1e4`, or ultimately the fallback `p.sw.spjac=2` (slowest option).

For HPC it may be useful to put the frequency ω on the branch for later plotting, and hence we provide a function `hpcbra.m`, see §4.

Remark 2.2 Table 2 lists the function needed for the "expensive terms"; additionally we remark again that providing, e.g., `spjac.m` or not is mainly a question of efficiency. In 1D, i.e., for "small" n_u, `p.sw.spjac=0` is usually fast enough. However, in 2D, `p.sw.spjac=0` and more so `p.sw.spjac=2` are often slow. For instance, for $n_u=7000$ in `chtor` (§3.2) `sGjac` and `spjac` are on the order of 0.001 seconds on the authors I7 laptop, while `numjac` for $\partial_u(G_u\phi)$ (`p.sw.spjac=0` with `p.sw.jac=1`) is on the order of 1 second, and the fallback `p.sw.spjac=2` (not needed here!) is about 10 seconds, which becomes minutes for `p.sw.jac=0`. ⌋

3 FPC and BPC in Cahn–Hilliard problems

We start with the classical example of the Cahn–Hilliard (CH) problem, concerned with stationary points of the energy

$$E_\varepsilon(u)=\frac{1}{2\sigma}\int_\Omega\frac{\varepsilon}{2}|\nabla u|^2+\frac{1}{\varepsilon}W(u)\,{\rm d}x, \tag{18}$$

where Ω is a bounded domain in $\mathbb{R}^d$, $\varepsilon>0$ is an interface energy parameter, W is a double–well potential with minima in $u_\pm$, and $\sigma=\int_{u_-}^{u_+}\sqrt{\frac12W(u)}\,{\rm d}u$ is a normalization. Wlog we choose

Table 2: Functions (function handles) for FPC, BPC and HPC. The first three are used if `p.sw.spjac`=1. To use, e.g., `myspjac.m` for FPC, set `p.fuha.spjac=@myspjac`, and similar `p.fuha.spjac=@mybpjac` or `p.fuha.spjac=@myhpjac` for BPC or HPC, respectively. The defaults set by `spcontini, bpcontini` and `hpcontini` are `spjac, bpjac` and `hpjac`, respectively; these must be provided by the user. If this is difficult, set `p.sw.spjac`=0 and see Remark 2.1. For the last four functions, the defaults are dummies returning 0.

handle `p.fuha.`	use	implements	default	`p.sw`	test
`spjac`	FPC	$\partial_u(G_u\phi)$	none	spjac	spjaccheck
`spjac`	BPC	$\partial_u(G_u^T\psi)$	none	spjac	bpjaccheck
`spjac`	HPC	$(\partial_u(G_u\phi_r), \partial_u(G_u\phi_r))^T$	none	spjac	hpjaccheck
`quuphi`	FPC	$\partial_u(q_u\phi)$	`quuphi`$(=0))$	spqjac	none
`quupsi`	BPC	$\partial_u(q_u^T\tilde{\phi})$	`quupsi`$(=0))$	none	none
`quuphir,quuuphii`	HPC	$\partial_u(q_u\phi_r)$, $\partial_u(q_u\phi_r)$	`quuphir(i)`$(=0))$	none	none

$W(u) = -\frac{1}{2}u^2 + \frac{1}{4}u^4 + \frac{1}{4}$ with $u_\pm = \pm 1$ and $\sigma = \sqrt{2}/3$. We impose the mass constraint

$$q(u) := \langle u\rangle - m \overset{!}{=} 0, \quad \langle u\rangle = \frac{1}{|\Omega|}\int_\Omega u\,\mathrm{d}x, \tag{19}$$

where $m = 0$ corresponds to equal volume fractions of "the phases ± 1", and we impose homogeneous Neumann BCs. Interestingly, for $m = 0$ and a sequence u_ε of minimizers of E_ε we have $E_\varepsilon(u_\varepsilon) \to |I|$ as $\varepsilon \to 0$, where the limit interface I is (a) minimal (surface). We refer to [Uec21, §6.9, §10.1] and the references therein for further background, and here only discuss the setup of FPC and BPC (as the problem is variational, Hopf points and time–periodic orbits cannot occur).

Introducing the Lagrange multiplier $\tilde{\lambda}$ and the Lagrangian $L = E_\varepsilon + \tilde{\lambda}q$, the pertinent system is

$$G(u) := -\varepsilon^2\Delta u - u + u^3 - \tilde{\lambda} = 0, \quad q(u) = \langle u\rangle - m = 0 \tag{20}$$

with $\partial_n u|_{\partial\Omega} = 0$. In 1D, the whole problem is symmetric under $x \mapsto -x$, and in general under $(u, m, \tilde{\lambda}) \mapsto -(u, m, \tilde{\lambda})$. Therefore we can restrict to $m \le 0$.

3.1 The problem over boxes

In the script `demos/pftut/ch/cmds1Db.m` and Fig.2 we choose $\Omega = (-1/2, 1/2)$ and first continue the homogeneous branch `h` (black) with $u \equiv m$, $\tilde{\lambda} = W'(m)$ in m, with free $\tilde{\lambda}$. Depending on the size of ε there are a number of (pitchfork, due to the equivariance $x \mapsto -x$) BPs on `h`. For $\varepsilon = 0.1$, the first branch `a1` bifurcates at $m = m_1(\varepsilon) \approx -0.6$, with one interface between $u = \pm 1$, and shows a fold around $m = -0.66$ after which it becomes stable; it then continues to $m = 0$, where the interface sits in the middle, i.e., at $x = 0$.

As already done in [Uec21, Fig.6.22] we can for instance continue the fold on `a1` in ε to see the existence (and stability) region (in m as a function of ε) of the one-interface pattern `a1`. The only nonlinear terms in G is u^3, hence $\partial_u G = -\varepsilon^2\Delta - 1 + 3u^2$ and $\partial_u(G_u\phi) = 6u\phi$ and `sG.m` and `sGjac.m`, and also `spjac.m` and `bpjac.m` are very simple, see Listing 1. Moreover, q is from (10), hence $\partial_u q =$ `a` is from (12), and in particular $\partial_u(q_u\phi) = 0$, which is already the default setting in `quuphi.m`. Thus, FPC of the (first) FP on branch `a1` in ε =parameter 2 is simply initialized and run by

$$\texttt{p} = \texttt{spcontini}('\texttt{a1}', '\texttt{fpt1}', 2, '\texttt{fpc1}'); \texttt{...set some switches...}, \texttt{p} = \texttt{cont}(\texttt{p}, 20);$$

see `cmds1Db.m`, 4th cell, and the green branch in Fig. 1(b). The physically interesting case is to decrease ε, and as we do so the FP moves to the left (in m, which together with $\tilde{\lambda}$ becomes a secondary active parameter). Here we also set `p.usrlam=0.05`, which forces output at $\varepsilon = 0.05$ if $\varepsilon = 0.05$ is passed during the continuation, which here happens at `pt9`. Calling `spcontexit('fpc1','pt9','b1')` we go back to continuation in m (at fixed $\varepsilon = 0.05$), and this yields the magenta branch `b1` in (c), see also the samples in (d).

Similarly we can do BPC in ε of the first BP on the homogeneous branch `h` to find the instability region of the homogeneous states, sometimes called spinodal region. This yields the red branch in (b), where $\varepsilon = 0.05$ is at `pt7`. Then calling, e.g., `p=bpcontexit('bpc1','pt7','pbpc1')` (with `pbpc` standing for PostBranchPointContinuation) and `p=swibra('pbpc1','bpt1','c1'); p=cont(p)`, we again obtain the magenta branch from (c).

Remark 3.1 For increasing ε, the BP `h/bpt1` moves to the right in m, and "vanishes" (in m=0) at ε=ε_0≈0.318. In arclength continuation this means that the branch of BPs folds back (in ε), and continues the symmetric BP (at positive m), see Fig. 1(b). Similarly, the FP on the one-interface branch moves to the right for increasing ε, and at $\varepsilon = \varepsilon_1 \approx 0.2$ "collides" with the BP. This means that at ε_1 the bifurcation changes from subcritical $\varepsilon < \varepsilon_1$ to supercritical $\varepsilon > \varepsilon_1$. The green branch actually consists of two branches (by the symmetry $x \mapsto -x$) which form a cusp at $\varepsilon = \varepsilon_1$, and depending on the numerical settings (tolerance, minimal stepsize, etc), near such cusps the continuation may fail or unreliably jump between the two branches.

Altogether we remark that we can have folds in FPC, BPC and HPC. We can also have BPs in FPCs, BPC and HPC, but we do not further pursue this here, and in our demos switch off BP (and HP) detection (and localization) by setting `p.sw.bifcheck=0`, and we switch off the computation of eigenvalues of $-\partial_U H$ by setting `p.sw.spcalc=0`. ⌋

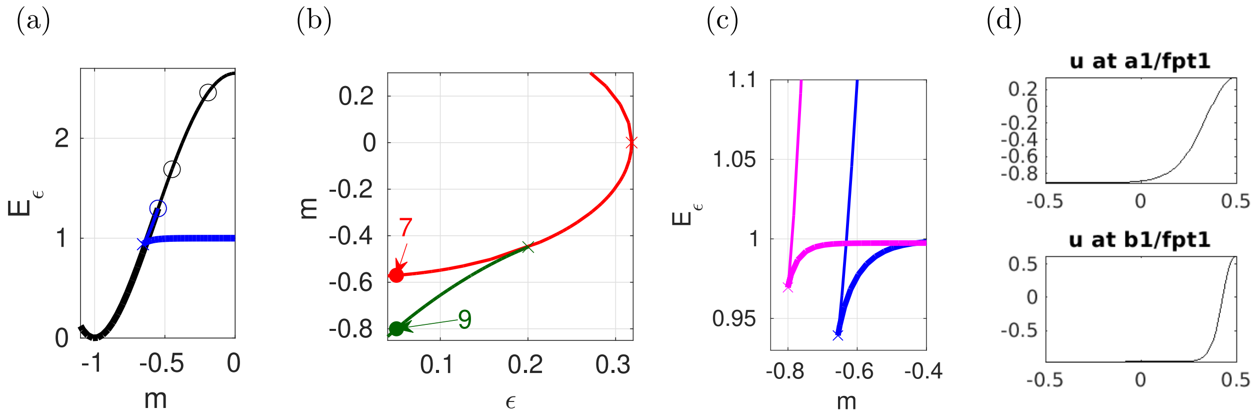

Figure 1: (20) over $\Omega = (-1/2, 1/2)$. (a) basic BD of the homogeneous branch `h` (black) and the one–interface branch `a1` (blue), $\varepsilon = 0.1$. (b) BPC (red) in ε of `h/bpt1` and FPC (green) in ε of `a1/fpt1`. (c) Zoom of `a1`, and the analogous branch `b1` after FPC exit at `pt9` in (b) $\varepsilon \approx 0.05$. (d) samples of `fpt1` on `a1` and on `b1`.

```
function r=sG(p,u) % PDE rhs for CH; first split u into pars and PDE-vars
par=u(p.nu+1:end); eps=par(2); lam=par(3); u=u(1:p.nu);
f=u+lam-u.^3; r=eps^2*p.mat.K*u-p.mat.M*f;  %  nonlinearity and rhs
```

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

```
function Gu=sGjac(p,u)  % PDE Jacobian
par=u(p.nu+1:end); eps=par(2); u=u(1:p.nu); % split u into pars and PDE vars
fu=1-3*u.^2; Fu=spdiags(fu,0,p.nu,p.nu); % f-deriv., put into sparse matrix
Gu=eps^2*p.mat.K-p.mat.M*Fu;  % the Jacobian matrix
```

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

```
function J=spjac(p,u) % \pa_u (G_u phi) for FPcont
phi=u(p.nu+1:2*p.nu); u=u(1:p.nu); % kernel vector, and PDE-u
fuu=-6*u; J=-p.mat.M*spdiags(fuu.*phi,0,p.nu,p.nu);
```

```
function J=bpjac(p,u) % \pa_u (G_u' psi) for BPcont
psi=u(p.nu+1:2*p.nu); u=u(1:p.nu); % adjoint kernel-vector and PDE-u
fuu=-6*u; J=-spdiags(fuu.*psi,0,p.nu,p.nu)*p.mat.M';
```

Listing 1: `ch/sG.m` and `ch/qf.m` implementing (20) (prepared by `chinit.m` and `oosetfemops.m` as usual) where `p.mat.vM` $= a$ from (10); `ch/sGjac.m` and `ch/qfder.m` as examples of simples derivatives, and `ch/spjac.m` and `ch/bpjac.m` as the only "nontrivial" functions needed for FPC and BPC.

Over 2D boxes, and more so over 3D boxes, the problem (20) becomes more complicated in the sense that for small ε there are many branches of solutions, e.g., different spots and stripes in 2D, and many secondary (and tertiary) bifurcations. Moreover, depending on the symmetry of the domain Ω, BPs from the homogeneous branch may have higher multiplicity, and for $\varepsilon \to 0$ the mesh–handling for the sharp interfaces becomes a more difficult problem than in 1D, see [Uec21, §6.9.2 and §6.9.3]. Nevertheless, the FPC and BPC (of simple BPs) works as before, and with the same files as in Listing 1, and in `cmds2Db.m` we illustrate this 2D setting.

3.2 The problem on a torus

In [Uec21, §10.1] and `pftut/chtor` we treat (20) on (the surface of) a torus, see Fig. 2, with Δ denoting the associated Laplace–Beltrami operator. For u non–constant in the horizontal angle ϑ, additional to the mass constraint we need a phase condition (PC) in the form $q_2(u) = \langle \partial_\vartheta u_0, u\rangle \overset{!}{=} 0$ where u_0 is a reference profile. In that case $n_q = 2$, and that gives us the opportunity to test/illustrate FPC and BPC with $n_q > 1$.

For the basic setup and patterns of (20) on a torus we refer to [Uec21, §10.1], and here and in `chtor/cmds2.m` and Fig. 2 only comment on issues of FPC and BPC. The black branch in (a) is again the spatially homogeneous solution $u \equiv m$, $\tilde\lambda = W'(m)$. If we continue the first bifurcating branch `b1` (blue) to $m = 0$, then at $m = 0$ this yields 2 vertical rings as interfaces between $u = \pm 1$, but here we are mostly interested in the fold around $m = -0.71$. Similarly, `b2` (red) at $m = 0$ yields 4 vertical rings at $m = 0$, but again we are only interested in the BP at $m \approx -0.46$, which yields a tertiary branch `b2-2` (magenta), and this again shows a fold, around $m \approx -0.51$. Importantly, these three branches have solutions which are nonhomogeneous in ϑ, and hence their continuation requires the PC $q_2(u) := \int_\Omega (\partial_\vartheta u_0) u \, d\Omega \overset{!}{=} 0$, where for u_0 we take the solution from the last continuation step, while $G(u)$ is augmented by $s\partial_\vartheta u$ with new (Lagrangian) parameter s. Thus, now

$$G(u) = -\varepsilon^2 \Delta u - u + u^3 - \tilde\lambda + s\partial_\vartheta u, \tag{21}$$

and $n_q = 2$ for such branches, with (here assuming a as a column vector, and using a discrete notation, where moreover $\partial_\vartheta u_0$ already contains multiplication by the mass matrix M)

$$q(u) = \begin{pmatrix} a^T u - m \\ (\partial_\vartheta u_0)^T u \end{pmatrix}, \quad \partial_u q = \begin{pmatrix} a^T \\ (\partial_\vartheta u_0)^T \end{pmatrix}, \quad \partial_{(m,s)} q = \begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix}. \tag{22}$$

For branches of solutions homogeneous in ϑ, only the first line of q is relevant. Relatedly, after branch switching from the homogeneous black branch we first do two continuation steps without the

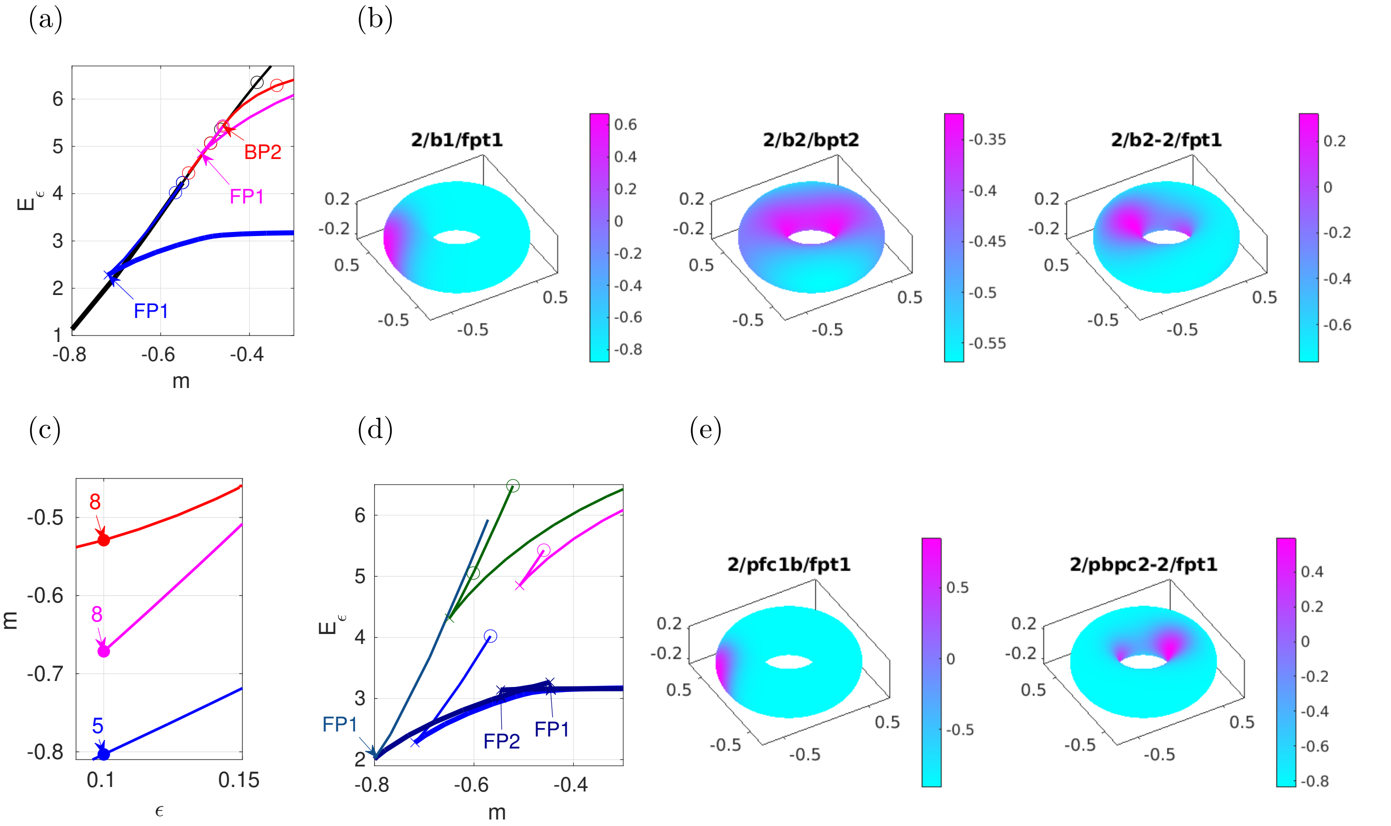

Figure 2: (20) over a torus with $(R, \rho) = (0.5, 0.25)$, and $\varepsilon = 0.15$ in (a,b). (a) homogeneous branch `h` (black) and the first two bifurcating branches `b1` (blue) and `b2` (red), and a tertiary branch `b2-2` (magenta). Samples in (b). (c) FPC and BPC of the marked points in (a). (d,e) switching back to continuation in m at $\varepsilon = 0.1$.

PC q_2 (as at bifurcation $\partial_\vartheta u_0 = 0$), and then switch on the PC.[6] Thus, we have two sets of q and q_u: First `qf.m,qfder` implementing only the mass constraint (the first line of (22)) and its derivative, to be run with `nq=1`, and second `qf2,qf2der`, implementing (22) and to be run with `nq=2`.

The parameter vector is $(m, \varepsilon\,\tilde\lambda, R, \rho, s)$ where R, ρ are the large and small radius of the torus. We fix $R = 0.5, \rho = 0.25$, and as before use m as the primary parameter for regular continuation with $\tilde\lambda$ as secondary active parameter, and hence `p.nc.ilam=[1,3]` ($n_q = 1$, no PC) or `p.nc.ilam=[1,3,6]` ($n_q = 2$, with PC). Altogether, due to the more complicated geometry and the PC, the files in `chtor/` are slightly more complicated than in `ch/`, but we refer to [Uec21, §10.1] for general discussion and put ample comments in the new script `cmds2.m` which deals with the FPC and BPC.

Fig.2(c) shows the continuation of points in (a) from ε=0.15 to ε=0.1.[7] (d) shows the new branches after `spcontexit` at `pt5` of the blue branch in (c), giving the lighter blue branch `pfc1` and `pfc1b`[8] , while the green branch `pbpc2-2` is obtained from `bpcontexit` at `pt8` of the red branch in (c), and then branch–switching at that BP. Naturally, `spcontexit` at the magenta `pt8` in (c) again yields the green branch, and the samples in (c) show that the effect of decreasing ε is to produce sharper interfaces between $u = \pm 1$.[9] Thus, these FPCs and BPCs (to still moderate ε) behave as expected,

[6] This sometimes introduces spurious bifurcation points, as without PC the 'almost zero' eigenvalue μ_0 may be negative; μ_0 is removed by switching on the PC, and this then changes the index of the solution. This can be fixed by after switching on the PC first doing one step without BP detection, but for simplicity we omit this here and in `cmds2.m`.

[7] Here we use a fixed mesh of 7000 points, and for $\varepsilon < 0.1$ we run into convergence problems as near the interfaces we would need a finer mesh. This can be done by adaptive mesh refinement, see `cmds1.m` and [Uec21, §10.1], including the convergence of $E_\varepsilon(u_\varepsilon)$ to the (limit) interface lengths, but for simplicity here we stick to the fixed mesh.

[8] with for instance `pfc1` and pfc1b standing for PostFoldContinuation1 and PostFoldContinuation1Backward

[9] This becomes more obvious at m closer to zero, see `cmds2.m` for such plots.

and the main remarkable effect is that the primary branch blue branch at ε=0.1 has two additional folds in m.

4 FPC and HPC in the complex Ginzburg–Landau equation

In `demos/hopf/cglpbc` we consider the complex Ginzburg–Landau (cGL) equation

$$\partial_t u = \varepsilon^2 \Delta u + (r + \mathrm{i}\nu)u - (c_3 + \mathrm{i}\mu)|u|^2 u - c_5|u|^4 u + s\partial_{x_1} u, \quad u = u(t, x) \in \mathbb{C}, \tag{23}$$

where we added the last term to deal with translational invariance (in x_1) for periodic BCs (in x_1). In (23), additionally to the choice of the domain, $\varepsilon > 0$ can be used to set the length scale $1/\varepsilon$, and we have the real parameters r, ν, c_3, μ, c_5, where typically the "driving force" r is used as the primary continuation parameter. Over 1D interval with periodic BCs the problem is $O(2)$ equivariant (spatial translations $x \mapsto x + \vartheta$, $\vartheta \in S^1$, and reflection symmetry $x \mapsto -x$), with HBPs on the trivial branch $u = 0$, and hence [Hoy06, §5.7] we obtain bifurcations of standing waves (SWs) and travelling waves (TWs) from the trivial solution. The TWs again need a PC $q(u) := \langle \partial_x u_0, u\rangle \overset{!}{=} 0$ and undergo Hopf bifurcations to modulated TWs, and thus we can illustrate HPC with $n_q > 0$.[10] Moreover, the cGL equation gives a 2-component real reaction diffusion *system* and hence gives an illustrative example how to implement `spjac` and `hpjac` for systems.

In the demo `demos/hopf/cgldisk` we transfer the setup to the cGL equation over a disk with Neumann-BCs, where the role of spatial translations is taken by spatial rotations $x = (x_1, x_2) \mapsto R_\vartheta x$, $\vartheta \in S^1$, again giving $O(2)$ equivariance. The TWs then become rotating waves (RWs), of spiral shape, and the Hopf-bifurcations from spirals yield modulated RWs, or 'meandering spirals'. The demos are already discussed in some detail in [Uec21, §7.2], without FPC and BPC, and thus here we focus on these aspects.

Written as a 2–component real system, (23) reads

$$\begin{aligned}\partial_t \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} &= -G(u, \lambda) \qquad (24)\\ &:= \begin{pmatrix} \varepsilon^2\Delta{+}r & -\nu \\ \nu & \varepsilon^2\Delta{+}r \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} - (u_1^2{+}u_2^2)\begin{pmatrix} c_3 u_1 - \mu u_2 \\ \mu u_1 + c_3 u_2 \end{pmatrix} - c_5(u_1^2{+}u_2^2)^2\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + s\partial_x \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.\end{aligned}$$

For all r, ν, c_3, μ, c_5, and pBCs, or homogeneous DBCs or NBCs, $u \equiv 0$ is a trivial solution, and we consider $r \mapsto u \equiv 0$ as the trivial branch. Depending on the domain (or equivalently ε) and the BCs, there are Hopf (for $\nu \neq 0$) bifurcations from this trivial branch and for $c_3 < 0$ (and $c_5 > 0$) the bifurcating (TW or SW) branches show folds and (for TWs) possibly further Hopf bifurcations.

4.1 Boxes

Choosing $\Omega = (-\pi, \pi)$ and pBCs, the Hopf points on the trivial branch are at $r_k = \varepsilon^2 k^2$, $k = 0, 1, 2, \ldots$. At each r_k, $k \geq 1$, we have a double Hopf point and three bifurcating branches, namely standing

[10]The cGL equation is also equivariant under phase rotations $u \mapsto \mathrm{e}^{\mathrm{i}\alpha} u$, $\alpha \in \mathbb{R}$, but for instance for SWs and TWs the spatial translations and phase rotations generate the same group orbits, and hence both are removed by the same phase conditions $q(u) = 0$, see, e.g., [Hoy06]. Thus $n_q = 1$.

waves (SWs), and left and right traveling waves (TWs) of the form

$$u(x,t) = a\mathrm{e}^{\mathrm{i}(\omega t \pm \varepsilon k x)}, \quad |a|^2 = -\frac{c_3}{2c_5} \pm \sqrt{\frac{c_3^2}{4c_5^2} + r - \varepsilon^2 k^2}, \quad \omega = \nu - \mu |a|^2, \tag{25}$$

where the phase of $a \in \mathbb{C}$ is free. Moreover, we compute that these branches have folds at

$$r = \varepsilon^2 k^2 - c_3^2/(4c_5)^2. \tag{26}$$

In Fig. 3 we focus on $\varepsilon = 1$ and the branch with $k = 1$ and FPC (and HPC) for this. First we note that for $k > 0$, to fix the (spatial) phase of the TW we need the PC

$$q(u) = \langle \partial_x u_0, u \rangle \overset{!}{=} 0, \tag{27}$$

and the addition of $s\partial_x u$ in (23) to treat TWs as *relative equilibria*, i.e., as fixed points in the comoving frame, i.e., in the frame moving with speed s. As we then continue the TW branch (black), we get the fold at $r = 3/4$ and additionally a HP at $r \approx 1.21$, where a branch of relative periodic orbits bifurcates, i.e., of time–periodic orbits in the comoving frame, aka modulated TWs, and where the TW branch gains stability. We can now do FPC and HPC with results as expected: in Fig. 3(b) we show the FPC (dashed line) and HPC (full line) in c_3. As c_3 decreases, the fold position behaves according to (26), and also the HP position moves to the left. Similarly, we can do FPC and HPC in ε, see the end of `cmds1Db.m` but we leave the analogous discussion to a 2D problem in the next demo.

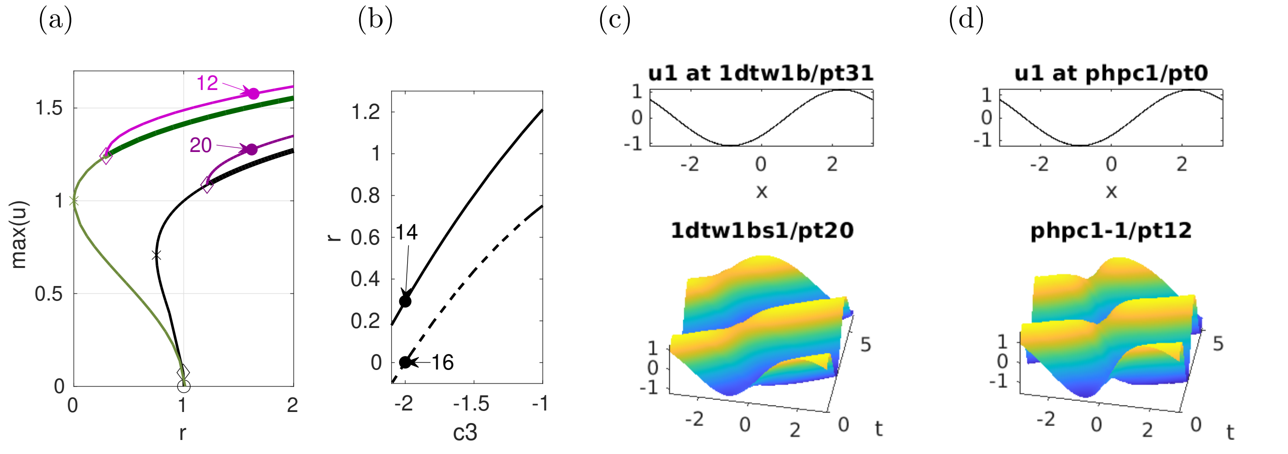

Figure 3: Results for (24) from `cmds1D.n`, $(\nu, \mu, c_5) = (1, 0.5, 1)$ and (initially) $c_3 = -1$. (a) TW branch `1dtw1b` (black) and mTW branch `1dtw1bs1` (violet, with pt20), and the same branches `phpc1` (green) and `phpc1-1` (light violet) after HPC to $c_3 = -2$. (b) HPC (full line) and FPC (dashed line) in c_3. (c,d) Samples.

It remains to discuss the setup of $\partial_u(G_u\phi)$ for FPC, and of $(\partial_u(G_u\phi_r), \partial_u(G_u\phi_i))$ for HPC (both are essentially the same). On the discrete level, (24) yields

$$G(u) = (\varepsilon^2 \mathcal{K} + s\mathcal{K}_x)u - \mathcal{M}f(u), \quad \mathcal{K} = \begin{pmatrix} K & 0 \\ 0 & K \end{pmatrix}, \quad \mathcal{K}_x = \begin{pmatrix} K_x & 0 \\ 0 & K_x \end{pmatrix}, \quad \mathcal{M} = \begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix}, \tag{28}$$

where K is the one–component stiffness matrix corresponding to $-\Delta$, K_x is the one–component advection matrix, M is the one–component mass matrix, and f is the "nonlinearity" (more precisely the terms without spatial derivatives)

$$f(u) = \begin{pmatrix} ru_1 - \nu u_2 \\ \nu u_1 + ru_2 \end{pmatrix} - (u_1^2+u_2^2) \begin{pmatrix} c_3u_1 - \mu u_2 \\ \mu u_1 + c_3u_2 \end{pmatrix} - c_5(u_1^2+u_2^2)^2 \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}. \tag{29}$$

Thus, $G_u = \varepsilon^2\mathcal{K} + s\mathcal{K}_x - \mathcal{M}\partial_u f(u)$, and since

$$\mathcal{M}(\partial_u f)\begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} = \mathcal{M}\begin{pmatrix} \partial_{u_1}f_1\phi_1 + \partial_{u_2}f_1\phi_2 \\ \partial_{u_1}f_2\phi_1 + \partial_{u_2}f_2\phi_2 \end{pmatrix}, \tag{30}$$

the general form of $\partial_u(G_u\phi)$ in this case is

$$\partial_u(G_u\phi) = -\mathcal{M}\begin{pmatrix} (\partial_{u_1}^2 f_1)\phi_1 + (\partial_{u_1}\partial_{u_2}f_1)\phi_2 & (\partial_{u_1}\partial_{u_2}f_1)\phi_1 + (\partial_{u_2}^2 f_1)\phi_2 \\ (\partial_{u_1}^2 f_2)\phi_1 + (\partial_{u_1}\partial_{u_2}f_2)\phi_2 & (\partial_{u_1}\partial_{u_2}f_2)\phi_1 + (\partial_{u_2}^2 f_2)\phi_2 \end{pmatrix}. \tag{31}$$

With the shorthand $|u|^2 = u_1^2 + u_2^2$ we have

$$\begin{aligned} \partial_u f(u) &= \begin{pmatrix} \partial_{u_1}f_1 & \partial_{u_2}f_1 \\ \partial_{u_1}f_2 & \partial_{u_2}f_2 \end{pmatrix} \qquad (32)\\ &= \begin{pmatrix} r-2u_1(c_3u_1-\mu u_2)-c_3|u|^2-4c_5|u|^2u_1^2-c_5|u|^4 & -\nu-2u_2(c_3u_1-\mu u_2)+\mu|u|^2-4c_5|u|^2u_1u_2 \\ \nu-2u_1(c_3u_2+\mu u_1)-\mu|u|^2-4c_5|u|^2u_1u_2 & r-2u_2(c_3u_2+\mu u_1)-c_3|u|^2-4c_5|u|^2u_2^2-c_5|u|^4 \end{pmatrix}, \end{aligned}$$

see also `sGjac.m` which calls `njac.m` implementing the "nodal Jacobian" (32). For (31) we need to compute the 6 expressions $\partial_{u_1}^2 f_1, \partial_{u_1}\partial_{u_2}f_1, \partial_{u_2}^2 f_1, \partial_{u_1}^2 f_2, \partial_{u_1}\partial_{u_2}f_2, \partial_{u_2}^2 f_2$, and then have

$$\partial_u(G_u\phi) = -\mathcal{M}\begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \tag{33}$$

with, e.g., `M11=spdiags(f1uu.*phi1+f1uv.*phi2,0,np,np)`, see `spjac.m`, where, e.g., `f1uv`$=\partial_{u_1}\partial_{u_2}f_1$, and where `phi1, phi2` are the two components of ϕ, extracted at the beginning of `spjac.m` from the extended data in `u`.

Similarly,

$$\begin{pmatrix} \partial_u(G_u\phi_r) \\ \partial_u(G_u\phi_r) \end{pmatrix} = -\begin{pmatrix} \mathcal{M}\begin{pmatrix} (\partial_{u_1}^2 f_1)\phi_{r1} + (\partial_{u_1}\partial_{u_2}f_1)\phi_{r2} & (\partial_{u_1}\partial_{u_2}f_1)\phi_{r1} + (\partial_{u_2}^2 f_1)\phi_{r2} \\ (\partial_{u_1}^2 f_2)\phi_{r1} + (\partial_{u_1}\partial_{u_2}f_2)\phi_{r2} & (\partial_{u_1}\partial_{u_2}f_2)\phi_{r1} + (\partial_{u_2}^2 f_2)\phi_{r2} \end{pmatrix} \\ \mathcal{M}\begin{pmatrix} (\partial_{u_1}^2 f_1)\phi_{i1} + (\partial_{u_1}\partial_{u_2}f_1)\phi_{i2} & (\partial_{u_1}\partial_{u_2}f_1)\phi_{i1} + (\partial_{u_2}^2 f_1)\phi_{i2} \\ (\partial_{u_1}^2 f_2)\phi_{i1} + (\partial_{u_1}\partial_{u_2}f_2)\phi_{i2} & (\partial_{u_1}\partial_{u_2}f_2)\phi_{i1} + (\partial_{u_2}^2 f_2)\phi_{i2} \end{pmatrix} \end{pmatrix}, \tag{34}$$

see `hpjac.m`. Both, (31) and (34), generalize in an obvious way to N–component (semilinear) reaction diffusion systems.

Finally, for $\tilde{w} = (r, s)$,

$$(\partial_u q)v = \partial_x u_{0,1}v_1 + \partial_x u_{0,2}v_2 \quad \text{and} \quad (\partial_{\tilde{w}}G)\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} v_1 & \partial_x v_1 \\ v_2 & \partial_x v_2 \end{pmatrix}, \tag{35}$$

and hence $\partial_u(\partial_{\tilde{w}}G) = 0$ and in particular $\partial_u(q_u\phi) = 0$ as before such that we can use the default functions (returning 0) `quuphi,quuphir` and `quuphii` in $J_{\rm FP}$ and $J_{\rm HP}$, cf. (9) and (17).

Remark 4.1 Again, the functions `sG, sGjac, spjac` and `hpjac` are essentially independent of the domain Ω, which in Fig.3 we took as a 1D interval. In [Uec21, §7.2.1] and `cglpbc/cmds2D.m` we generalize the computation of SW and TW branches to 2D boxes with pBCs in x_1 and (homogeneous) NBCs in x_2. The 3rd HBP on the trivial branch is then triple, giving five bifurcating branches, and in `cmds2Db.m` we now illustrate some FPC for this, but here we skip the details and instead next consider a disk. $\rfloor$

4.2 A disk

In `demos/hopf/cgldiskHPC` we consider

$$\partial_t u = \varepsilon^2 \Delta u + (r + \mathrm{i}\nu)u - (c_3 + \mathrm{i}\mu)|u|^2 u - c_5|u|^4 u + sK_{\mathrm{rot}}u, \quad u = u(t,x) \in \mathbb{C}, \tag{36}$$

over a disk of radius π (this precise value plays no role at all). This is (23) with $s\partial_x u$ replaced by $sK_{\mathrm{rot}}u$, $K_{\mathrm{rot}}u = -x_2\partial_{x_1}u + x_1\partial_{x_2}u$, and the PC (27) is replaced by

$$q(u) := \langle K_{\mathrm{rot}}u_0, u\rangle \overset{!}{=} 0. \tag{37}$$

Writing (36) as a two component real system proceeds as in (24), and in fact the files `sG.m,sGjac.m,` `spjac.m` and `hpjac.m` for (36) are exactly as in `demos/hopf/cglpbc` (with K_x replaced by K_{rot}). We again have the trivial branch $r \mapsto u \equiv 0$, with HBPs, double due to rotational symmetry (except the first), and the bifurcating branches are SWs and left/right rotating waves RWs. We refer to [Uec21, §7.2.2] for basic results on these, including more detailed plots of modulated RWs (mRWs), aka meandering spirals, often characterized by the motion of the spiral tip.[11]

Besides again using c_3 for FPC and HPC, here we also want to decrease ε (increase the disk size), and therefore we start with a reasonably fine mesh ($n_p \approx 2000$). This means that the computation of the mRWs is somewhat expensive (about 20Min for a branch of 10 continuation steps), and since we focus on HPC we only compute and plot a few mRWs. In Fig. 4, we give the BD of the first RW `rw1` (brown) for continuation in r, which gains stability at HP3[12] near $r = 0.68$, and loses it again at HP4 near $r = 1.23$. At these HPs, mRWs `mrw2` and `mrw3` bifurcate, and in particular `mrw3` contains stable solutions. See the top row of (c) for the profiles at these two HPs, which look rather similar, but the angular speeds $\omega/2\pi$ are different, namely $\omega = -11.3$ at `HP3` vs $\omega = -2.3$ at `HP3`. Similar to Fig. 3, in (b) we then do (FPC and) HPC in c_3. Here again all points move left in r as c_3 decreases, and the stability range of `rw1` between HP3 (red branch) and HP4 (blue branch) increases, while the solution profiles (c) do not change much.

In (d) we do a similar (FPC and) HPC in ε. Decreasing ε, HP3 and HP4 again move to the left, but also their distance (in r) decreases, and at $\varepsilon = \varepsilon_0 \approx 0.65$ they flip order. This could be problematic as it gives a double HP at ε_0 such that J_{HP} might not be regular. Here, however, the continuation works to $\varepsilon = 0.5$ (and smaller) without complaints. In (e) we go back to continuation in r at $\varepsilon = 0.75$ from `hpc2/pt36`, but naturally the FPC and the other HPC gives the same (blue) branch (brown branch taken from (a) for comparison). Here the stability of `rw1` corresponds to the (small) r–interval between `hpc1/pt62` and `hpc2/pt36` from (a). This illustrates the application of HPC to find stability ranges of solutions by continuing points that delimit these ranges.[13]

[11]`demos/hopf/cgldiskHPC` is a modification of `demos/hopf/cgldisk` since here we introduced the additional parameter ε which (for decreasing ε) also requires somewhat finer meshes, and we do this new in a new directory in order not to proliferate branches in `demos/hopf/cgldisk`; on the other hand, to keep the demo focussed we removed some additional functions present in `cgldisk/` (e.g., for plotting spiral tip paths).

[12]strictly speaking, this is the 2nd HP on the `rw1` branch; the first (at small amplitude of `rw1`, near $r = 0.43$) is spurious and wrongly detected due to the switching on of the PC after branch switching, cf. footnote 6.

[13]Of course, other HPs could move into the (supposed) "stability interval" delimited by `HP3` and `HP4`, but this does not occur here.

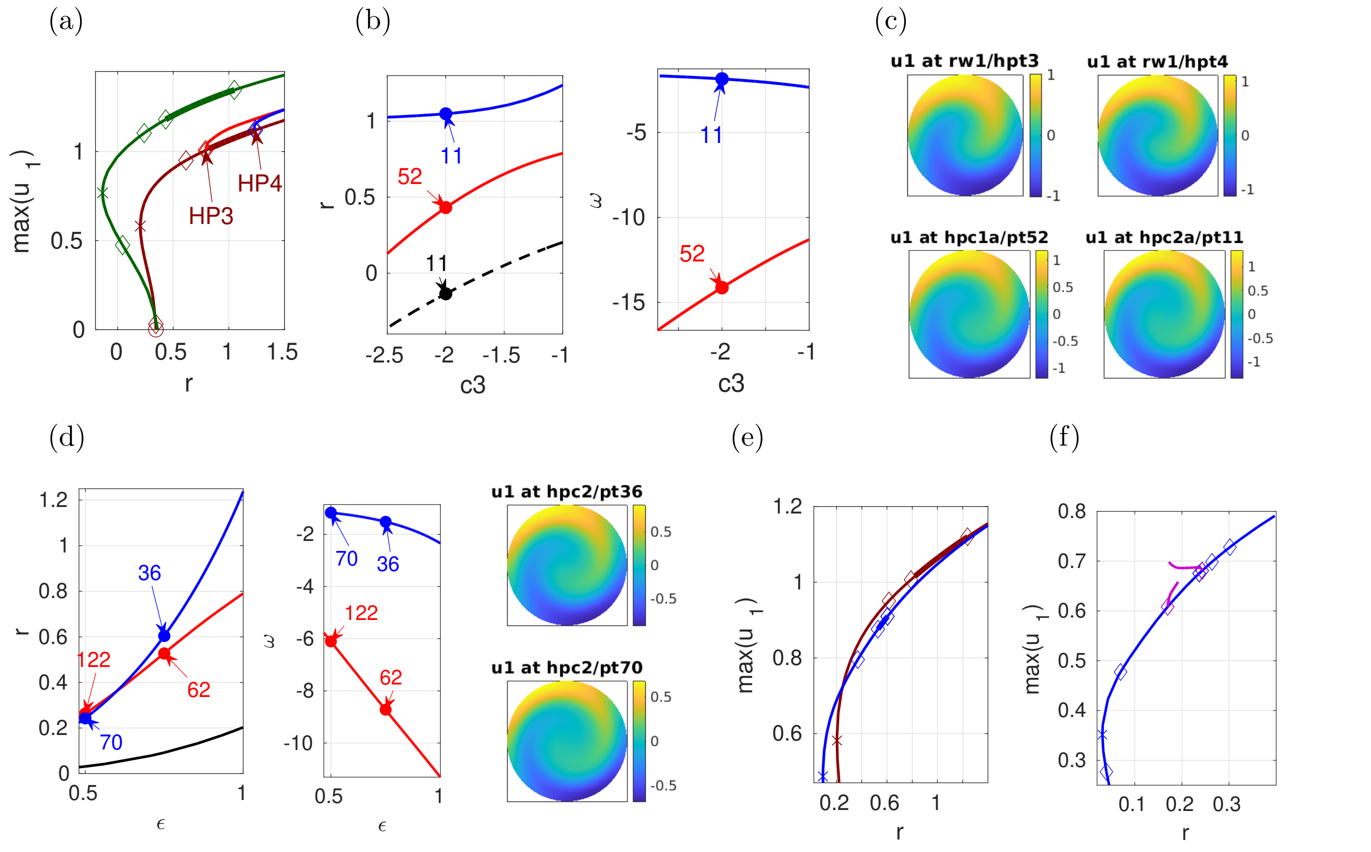

Figure 4: Results for (36) on a disk of radius π, $(\nu, \mu, c_5) = (1, 0.5, 1)$ and initially $c_3 = -1$, $\varepsilon = 1$. (a) BD of (original) RW1 (brown), with two mRWs bifurcating, giving a stability region for RW1 between HP3 and HP4. Green branch obtained after (either) FPC or HPC from RW1 to $c_3 = -2$. (b) FPC (black, dashed) and HPC `hpc1` (red, from HP3) and `hpc2` (blue, from HP4) in c_3, position in r (left), and ω (right), samples in (c). (d) FPC and HPC in ε, and two samples. (e) back to continuation in r at $\delta = 0.75$, where the (small) stability region of the blue RW1 is as determined in (d). (f) Continuation in r at $\delta = 0.5$, after mode crossing in (d).

For $\varepsilon < \varepsilon_0$ we do not expect stable parts on `rw1`, and in Fig. 4(f) we show `rw1` (blue) obtained from `hpcontexit` from `hpc1/pt70` (but again from `hpc2/pt122` or `spcontexit` at $\varepsilon = 0.5$ we obtain the same `rw1` branch), which is unstable for all r. However, to correctly detect this we (as always) need to make sure that we compute the relevant unstable eigenvalues, and for instance the 4 unstable eigenvalues $\mu_{1,2} \approx -0.06 \pm 5\mathrm{i}$ and $\mu_{3,4} \approx -0.09 \pm 8.5\mathrm{i}$ are "missed" with the intial setting of computing `p.nc.neig` $= 30$ closest to `p.nc.eigref` $= 0$, because too many stable eigenvalues with $\mathrm{Re}\mu > 0$ are closer to 0 on this larger domain. There are different options to refine the eigenvalue computation, see [Uec21, §3.3], but here we just choose the easy way and set `p.nc.neig=100`. For completeness, in (f) we also illustrate that Hopf branch-switching at the HPs still works, by plotting the obtained modulated RW branches as in (a).

References

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

[Kuz04] Y. A. Kuznetsov. *Elements of applied bifurcation theory. 3rd ed.* Springer, 2004.

[Uec21] H. Uecker. *Numerical continuation and bifurcation in Nonlinear PDEs.* SIAM, Philadelphia, PA, 2021.

[Uec23] H. Uecker. `www.staff.uni-oldenburg.de/hannes.uecker/pde2path`, 2023.