Complex Swift–Hohenberg over intervals and disks, reference sheet, February 25, 2026

Hannes Uecker, Institut für Mathematik, Universität Oldenburg, hannes.uecker@uol.de

Introduction. In [UVK26] we consider dynamic patterns in the complex Swift-Hohenberg (CSH) equation

$$\partial_t u = (\lambda + \mathrm{i}\nu)u - (c_3 + \mathrm{i}\gamma)|u|^2u - |u|^4u - (1 + \nabla^2)^2u + \mathrm{i}\delta\nabla^2 u, \tag{1}$$

with $u = u(x,t) \in \mathbb{C}$, and coefficients $\lambda, \nu, \delta, c_3, \gamma \in \mathbb{R}$, over intervals $x \in (-l, l)$ with periodic boundary conditions (BCs), and over disks $D_R = \{x = r\mathrm{e}^{\mathrm{i}\vartheta} \in \mathbb{R}^2 : 0 \le r < R\}$ with Neumann BCs $\partial_n u|_{\partial D_R} = \partial_n \Delta u|_{\partial D_R} = 0$. Throughout we fix $c_3 = -2.5$ to have strongly subcritical bifurcations of, in 1D, the primary traveling and standing wave branches (TWs and SWs, respectively), which then yields secondary and tertiary bifurcations of modulated TWs (MTWs), localized SWs (LSWs), localized drifting waves (LDWs), and 2-frequency states. The 1D case is thus very rich in itself, but also serves as a "road map" to (dynamic) wall patterns in 2D, i.e., patterns attached to the perimeter of the disk ("breathing and rotating daisies"). This approach is similar to the 1D steady states of the real 1D SH equation used in [VKU21] as a road map for steady wall patterns over disks.

Symmetries and solution classes. Over intervals $(-l, l)$ with pBCs, wlog $l = m\pi$, the linearization at the trivial solution $u \equiv 0$ has eigenvalues $\mu_k = \lambda + (1 - k^2)^2 + \mathrm{i}(\nu - \delta k^2)$, $k \in \mathbb{Z}/m$, and hence the trivial branch $\lambda \mapsto u \equiv 0$ loses stability at $\lambda_c = 0$ in a wave bifurcation with wave number $k = 1$. The CSH equation has the usual $O(2)$ symmetry of spatial translation and reflection, and hence at $\lambda = \lambda_c = 0$ traveling waves and standing waves bifurcate simultaneously. Another important symmetry of (1) is the phase invariance, henceforth called gauge invariance: If $u(t, x)$ is a solution, so is $u(t, x)\mathrm{e}^{\mathrm{i}\vartheta}$ for any $\vartheta \in \mathbb{R}$. As a consequence, the TWs have the explicit single mode formula

$$u_{\mathrm{TW}}(x,t)=\alpha \mathrm{e}^{\mathrm{i}(kx-\omega_{\mathrm{TW}}t)},\quad \alpha^2 = -\frac{c_3}{2} \pm \sqrt{\frac{c_3^2}{4} + \lambda - (1-k^2)^2},\quad \omega_{\mathrm{TW}} = -\nu + \gamma\alpha^2 + \delta k^2, \tag{2}$$

which is useful to assess the standard numerical computation of TWs as relative equilibria in a frame comoving with (constant) speed. For this let $\tilde{u}(x,t) = u(x - \sigma(t), t)$ and after dropping the $\tilde{}$ seek TWs as steady states of

$$\partial_t u = (\lambda + \mathrm{i}\nu)u - (c_3 + \mathrm{i}\gamma)|u|^2u - |u|^4u - (1 + \partial_x^2)^2u + \mathrm{i}\delta\partial_x^2 u + s\partial_x u, \tag{3a}$$

$s = \sigma'$, together with the standard translational phase condition (PC)

$$q_{\mathrm{trans}}(u) := \langle \partial_x u_{\mathrm{old}}, u \rangle = 0. \tag{3b}$$

Thus, we have a nonlinear eigenvalue problem for the field u and the scalar s, and typically with primary (continuation) parameter λ.

Even more importantly, the gauge invariance implies that SWs can be factored as

$$u_{\mathrm{SW}}(x,t) = v(x)\mathrm{e}^{-\mathrm{i}\omega t}, \tag{4}$$

and hence we seek SWs as relative equilibria in a gauge rotating (and for later drift bifurcations possibly moving, $s \ne 0$) frame, i.e., as steady states of

$$\partial_t v = (\lambda + \mathrm{i}\nu)v - (c_3 + \mathrm{i}\gamma)|v|^2v - |v|^4v - (1 + \partial_x^2)^2v + \mathrm{i}\delta\partial_x^2 v + s\partial_x v + \mathrm{i}\omega v, \tag{5a}$$

which must be augmented by the gauge PC $q_{\text{gauge}}(v) := \langle v, \mathrm{i}v_{\text{old}}\rangle = 0$. Thus, the full PCs (translational and gauge) become

$$q_2(v) := (q_{\text{trans}}(v), q_{\text{gauge}}(v)) = (0,0), \tag{5b}$$

yielding a nonlinear eigenvalue problem for (u, s, ω), again with free λ. This avoids the expensive computation of SWs as time–periodic orbits (POs), and moreover allows to find LSWs as standard steady state bifurcations from SWs, and then LDWs as drift bifurcations from LSWs. For SWs and LSWs, $s = 0$ in (5a); a drift bifurcation then is a steady state bifurcation for (5a) with $s = 0$ at onset but then changing to $s \neq 0$. Finally, the computation of SWs and LSWs as equilibria for (5) also allows to find 2–frequency states via Hopf–bifurcation in (5).

On the TW branches there are Hopf points yielding bifurcations of MTWs as POs in a comoving frame, and for these we use a genuinely time-dependent frame speed $s(t)$ in (3a), and a pointwise in t translational phase conditions, namely

$$q_H(u, t_i) := \langle \partial_x u_*, u(t_i)\rangle = 0 \text{ at each time–discretization point } t_1 = 0, \dots, t_m = T, \tag{6}$$

where $u_* = u_*(x)$ is a reference profile, usually chosen as $u_*(x) = u_H(x)$, the profile at the HP.

As already said, we use the 1D case as a road map for 2D wall states, where TWs become rotating waves (RWs), MTWs become modulated RWs (MRWs), and LSWs refers to standing waves at the wall and additionally localized in angle ϑ. Tertiary bifurcations from these yield LDWs, drifting in angle. The implementations then are in all cases essentially the same as in 1D, i.e., instead of $s\partial_x u$ on the rhs of (3a) and (5a) we obtain $s\partial_\vartheta u$, and accordingly the systems and PCs become:

$$\left.\begin{array}{l} 0 = \partial_t u = (\lambda + \mathrm{i}\nu)u - (c_3 + \mathrm{i}\gamma)|u|^2 u - |u|^4 u - (1+\Delta)^2 u + \mathrm{i}\delta\Delta u + s\partial_\vartheta u, \\ 0 = q_{\text{rot}}(u) = \langle \partial_\vartheta u_{\text{old}}, u\rangle \end{array}\right\}, \tag{7}$$

for RWs as equilibria with free $s \in \mathbb{R}$;

$$\left.\begin{array}{l} 0 = \partial_t u = (\lambda + \mathrm{i}\nu)u - (c_3 + \mathrm{i}\gamma)|u|^2 u - |u|^4 u - (1+\Delta)^2 u + \mathrm{i}\delta\Delta u + s(t)\partial_\vartheta u, \\ 0 = q_H(u) = \langle \partial\vartheta u_*, u(t_i), \quad i = 1, \dots, m, \rangle \end{array}\right\}, \tag{8}$$

for MRWs with free function $t \mapsto s(t)$ (respectively vector $(s_1, \dots, s_m)$);

$$\left.\begin{array}{l} 0 = \partial_t v = (\lambda + \mathrm{i}\nu)v - (c_3 + \mathrm{i}\gamma)|v|^2 v - |v|^4 v - (1+\Delta)^2 v + \mathrm{i}\delta\Delta u + s\partial_\vartheta v + \mathrm{i}\omega v, \\ (0,0) = (q_{\text{rot}}(v), q_{\text{gauge}}(v)) = (\langle \partial_\vartheta v_{\text{old}}, v\rangle, \mathrm{i}v_{\text{old}}, v) \end{array}\right\}, \tag{9}$$

for SWs and LSWs with $s = 0$ and free ω, and for LDWs with free (s, ω).

Implementation. Technically, we write the complex 4th order system (1) as a real second order system for $U = (\mathrm{Re}u, \mathrm{Im}u, \mathrm{Re}(\Delta u), \mathrm{Im}(\Delta u)$. The parameter vector is sorted as `par` $= [\lambda, \nu, \gamma, c_3, c_5, s, \delta, \omega]$. For, e.g., continuation of RWs in (λ, s), see (7), the active parameters are thus chosen via `p.nc.ilam=[1,6]`, and `p.nc.nq=1`, `p.fuha.qf=@qf`, and `p.fuha.qfder =@qfjac`, while continuation of SWs, LSWs and LDWs in (λ, s, ω) (see (9)) is done via `p.nc.ilam=[1,6,8]`, and `p.nc.nq=2`, `p.fuha.qf=@qfR`, and `p.fuha.qfder=@qfRjac`.

In 2D, to obtain good (symmetry respecting) approximations with acceptable numbers of degrees of freedom, we proceed similar to [VKU21]: we use the piecewise quadratic (P2) FEM on meshes generated by subdivision and projection. Additionally, it turned out that the P2 FEM is also very helpful in 1D to get accurate comparison of the numerical speed s with the exact formula $s = \omega_{\text{TW}}/k$ with ω_{TW} given in (2).

The demo consists of three directories:

`cshlib/` contains common (dimension independent) files, e.g., the rhs `sGr` and Jacobian `sGrjac`, and the different PCs such as `qR` and derivatives of these such as `qRjac`, and some general files which in principle are independent of `csh` but are not in the main `pde2path` libraries yet. This in particular includes `hoswibramtw`, which turns the scalar $s\in\mathbb{R}$ from TW continuation into a function (vector) $(s_1,\dots,s_m)$ and sets up the appropriate data structures. For plotting in the labframe, $\sigma(t)=\int_0^t s(\tau)\,\mathrm{d}\tau$ is then reconstructed via the trapezoidal rule. See Table 1.

`csh1D/` contains `cshinit, oosetfemops` for 1D initialization, and then (mostly) scripts for computing and plotting 1D solutions (solution branches), and some minor convenience mods of `pde2path` library functions, e.g. `tintfreezex` and `xtplot`; see Table 2.

`csh2D/` is similar, see Table 3.

Thus, to run the scripts in `csh1D/` and `csh2D/`, add `cshlib/` to your `Matlab` library path (on top of the `pde2path` libraries *after* calling `setpde2path` in `pde2path/`), change into `csh1D/` (or `csh2D/`), and there start with `edit cmds1.m` and execute, as usually recommended cell–by–cell. Each of the directories also contains a file `aaa.m` with a short summary of the directory contents, roughly similar to the contents of Tables 1–3, respectively.

Table 1: Functions in `cshlib`. First block: functions for (1), 2nd block: more general files which are scheduled to become part of `pde2path/libs`.

file	purpose, remarks
`sGr sGrjac`	rhs (5a), and Jacobian; implements (1) if $s=\omega=0$ resp. (3a) if $\omega=0$; dimension independent.[1]
`nodalf`	"nonlinearity" of (1), i.e., terms without derivatives, called in `sGrjac`. Jacobian of `nodalf` in `njac`, called in `sGrjac`.
`qf,qjac`	$q_{\rm trans}$ (resp $q_{\rm rot}$ in 2D), and derivative.
`qfR,qfRjac`	both PCs $(q_{\rm trans},q_{\rm gauge})$ (resp $(q_{\rm rot},q_{\rm gauge})$ in 2D), and derivatives of these.
`eswibra`	initializes data-structures for SWs in the form $u(t,x)=v(x)\mathrm{e}^{-\mathrm{i}\omega t}$.
`hoswibramw`	initializes data-structures for the "$s=s(t)$" setting.
`sGmw`	internal for $s=s(t)$ setting; evaluation of $G(u,s(t_i))$; Jacobian in `sGmwjac.m`.
`assem1d2`	P2FEM matrix assembly 1D, incl.advection operators.
`assem6A`	P2FEM matrix assembly, incl.advection operators.
`hoassempbc`	simplified assembly of systems for POs; calls `huassem, huassem4`.

Table 2: Scripts and functions in `csh1D`; all figure numbers refer to [UVK26]. First block: Main scripts; 2nd block: Problem describing functions.

file	purpose, remarks
`cmds1`	$(\nu,\gamma,\delta)=(-1,0.5,1)$, Figs.2-4, output to `1/`, plotting in `cmds1plot`, including some DNS.
`cmds1c`	Fig.5, like `cmds1` but $l=8\pi$, output to `1c/`, plotting and DNS included.
`cmds2`	$(\nu,\gamma,\delta)=(0,-0.5,1)$, Figs.6,7, output to `2/`, plotting in `cmds2plot`
`cmds3`	$(\nu,\gamma,\delta)=(0,-1,0.5)$, Figs.8,9, output to `3/`, plotting in `cmds3plot`
`cmds3c`	$(\nu,\gamma,\delta)=(0,-1,0.5)$, $l=10\pi$, Fig.10, output to `3c/`, DNS and plotting.
`cshinit`	initialization of struct `p` with standard parameter values, set up of 1D dom.
`oosetfemops`	FEM operators (mass, diffusion, and advection matrices).

[1] The spatial dimension enters through the FEM operators, and these are set via `oosetfemops` in `csh1D/` and `csh2D/`, respectively.

Table 3: Scripts and functions in `csh2D`; all figure numbers refer to [UVK26].

file	purpose, remarks
`cmds1`	$(\nu,\gamma,\delta)=(-1,0.5,1)$, $R=5$, coarse mesh for MRWs, output to `1/`.
`cmds1b`	$(\nu,\gamma,\delta)=(-1,0.5,1)$, $R=5$, finer mesh, SWs, LSWs and rungs, out to `1b/`.
`cmds1plot`	Plotting of results from `cmds1` and `cmds1b`, Figs.11–14.
`cmds2(b)`	(ν,γ,δ)=$(0,-0.5,1)$, R=5, out to `2/`, `2b/`, plotting in `cmds2plot`, Figs.14,15.
`cmds3`	(ν,γ,δ)=$(0,-1,0.5)$, R=5, coarse resolution for MTWs, output to `3/`.
`cmds3b`	$(\nu,\gamma,\delta)=(0,-1,0.5)$, $R=5$, finer mesh for RWs and SWs, output to `3/`.
`cmds3plot`	Plotting, Fig.17, and also subsequent Figs.at $R=8$.
`cmds3c/d`	$(\nu,\gamma,\delta)=(0,-1,0.5)$, $R=8$, medium and fine meshes
`cshinit`	initialization of struct `p` with standard parameter values, set up of domain.
`oosetfemops`	FEM operators.
`plotsolr`	plot of $D_R\ni x\to \mathrm{Re}u(x)$, of $\partial D_R\ni\xi\to u(\xi)$ (real and imag along perimeter), and of $\tilde{u}(x,t)=u(x)\mathrm{e}^{-\mathrm{i}\omega t}$ (real part) along perimeter, u and ω taken from `p.u`.
`userplot`	calls `plotsolr` for perimeter plots also during continuation.
`tintxsmovpr`	DNS with output of perimeter/diameter data and movie, init and start.
`ctintmovpr`	continuation of DNS started in `tintxsmovpr`.

[UVK26] H. Uecker, N. Verschueren, and E. Knobloch. Dynamic patterns in the complex Swift-Hohenberg equation over intervals and disks, Preprint, 2026.

[VKU21] N. Verschueren, E. Knobloch, and H. Uecker. Localized and extended patterns in the cubic-quintic Swift-Hohenberg equation on a disk. *Phys. Rev. E*, (104):014208, 2021.