

# Münsteranian Torturials on Nonlinear Science

edited by Uwe Thiele, Oliver Kamps, Svetlana Gurevich

## Continuation

### DRIST: linear stability of sliding drops on an inclined homogeneous substrate

Uwe Thiele<sup>1</sup>

Version 3, Jan 2022

For updates see [10.5281/zenodo.4546407](https://doi.org/10.5281/zenodo.4546407)

---

<sup>1</sup>with support of Frank Ehebrecht, Frank Lengers, Meshari Alesemi, Dimitri Tseluiko, Thomas Seidel, Simon Hartmann

### Abstract

The tutorial DRIST is one of a series of tutorials on the practical application of numerical path-continuation methods for problems in soft matter and pattern formation. It is part of the “Münsteranian Torturials on Nonlinear Science”. The tutorial employs continuation techniques to calculate selected stationary states of a driven thin-film equation together with their linear stability, i.e., steady profiles, their eigenvalues and eigenfunctions are continued in parallel. Retracing the various steps for this particular problem, you will be able to apply the technique to any equation of this form. Beside `auto07p` you will need `MatLab` to be able to use this tutorial. It is recommended to consider this tutorial after the tutorials CCH [1] and/or SLIDROP [2].

## 1 Model

The tutorial DRIST is part of the “Münsteranian Torturials on Nonlinear Science”, a series of hands-on tutorials that shall facilitate the practical application of numerical path-continuation methods [3, 4, 5] for problems in soft matter and pattern formation by lowering the entrance threshold for systems where side conditions as, e.g., conservation laws and translational invariance have to be taken into account. The present tutorial is based on the code package `auto07p` [6] and also employs `MatLab` [7]. An overview of all available tutorials in the series and a description of a recommended sequence of working through them is given in Ref. [8]. It shows how to determine the linear stability of one-dimensional (1d) stationary solutions of equations like the driven thin-film equation (as in tutorial SLIDROP [2], or Refs. [9, 10]),

$$\partial_t h = -\partial_x \left\{ Q(h) \partial_x \left[ \frac{\partial_{xx} h}{Bo} - \partial_h f(h) \right] + \chi(h) \right\} \quad (1)$$

together with their linear stability employing continuation. A special case is the driven Cahn Hilliard equation (see tutorial CCH [1], or Refs. [11, 12, 13, 14]). We have included the so called Bond number  $Bo$  in order to have a general form of the thin-film equation ( $Bo$  is used, e.g., in [10]. As it is always possible to include  $Bo$  in the  $x$ -scale in this tutorial we set  $Bo = 1$  in the following even though all provided codes use the general eq. (1).

As for such an equation with lateral driving, in general, eigenvalues and eigenfunctions are complex, one is not able to detect all eigenvalues as branching points. This is only possible for real eigenvalues (see tutorial LINDROP [15]). In consequence, the presented technique is based on a combination of `auto07p` [6, 3, 16] and `Matlab` [7] steps and may easily be adapted to related equations. `Auto07p` is employed for all continuation steps and `MatLab` for the solution of the eigenvalue problem. The general procedure was first employed in the context of transversal (front) instabilities in [17, 9] and for sliding drops on heated substrates in [10].

The example in the tutorial utilizes  $Q(h) = 1$ ,  $\chi(h) = \frac{D}{2} h^2$  and  $f''(h) = 3h^2 - 1$ , i.e., the driven Cahn Hilliard equation. Corresponding results are published in [14].

We now introduce the coordinate frame moving with  $v$  by  $\tilde{x} = x - vt$  (cf. tutorial SLIDROP [2]) and determine the steady state solutions  $h_0(\tilde{x})$  in the comoving frame. After dropping the tildes we obtain:

$$-v \partial_x h_0 = -\partial_x \{ Q(h_0) \partial_x [\partial_{xx} h_0 - \partial_h f(h_0)] + \chi(h_0) \}. \quad (2)$$

Integrating eq. (2) once gives the equation for the steady states in the comoving frame, i.e., stationary travelling states in the laboratory frame

$$0 = Q(h_0) \partial_x [\partial_{xx} h_0 - \partial_{h_0} f(h_0)] + \chi(h_0) - v h_0 - C_0. \quad (3)$$

To calculate the linear stability, one employs eq. (1) in the comoving frame, introduces the ansatz  $h(\tilde{x}, t) = h_0(\tilde{x}) + \epsilon h_1(\tilde{x}) e^{\beta t}$ , and linearizes in  $\epsilon \ll 1$ . After dropping the tildes the linearised problem can be written as a linear eigenvalue problem of the form

$$\beta h_1(x) = \mathcal{L}[h_0] h_1(x), \quad (4)$$

with the complex growth rate  $\beta$  as eigenvalue, the complex perturbation mode  $h_1(x)$  as eigenfunction and  $\mathcal{L}[h_0]$  as a linear operator acting on  $h_1(x)$ . In expanded form this equation is

$$\begin{aligned} \beta h_1(x) = & -Q_0 \partial_{xxxx} h_1 + Q_0 \partial_{xx} ((\partial_{hh} f_0) h_1) + v \partial_x h_1 - \partial_x ((\partial_h \chi_0) h_1) \\ & - (\partial_h Q_0) [\partial_x (h_1 \partial_{xxx} h_0 - 2 (\partial_{hh} f_0) (\partial_x h_0) h_1) + \partial_x h_0 \partial_{xxx} h_1 + (\partial_{hh} f_0) (\partial_x h_0) h_1] \\ & - (\partial_{hh} Q_0) [\partial_x (\partial_{xx} h_0 - \partial_h f_0) (\partial_x h_0) h_1], \end{aligned} \quad (5)$$

where  $Q_0 := Q(h_0)$ ,  $\partial_h Q_0 := \partial_h Q(h_0)$  etc. and

$$\begin{aligned} \partial_{xx} (h_1 \partial_{hh} f_0) &= h_{1xx} f_{hh}(h_0) + 2 h_{1x} h_{0x} f_{hhh}(h_0) + h_1 h_{0xx} f_{hhhh}(h_0) + h_1 h_{0x}^2 f_{hhhh}(h_0). \\ \partial_x ((\partial_h \chi_0) h_1) &= \partial_{hh} \chi_0 h_{0x} h_1 + (\partial_h \chi_0) h_{1x} \end{aligned} \quad (6)$$

The eigenvalues and eigenfunctions of eq. (5) (i.e., of the operator  $\mathcal{L}[h_0]$ ) are calculated by the MatLab code `get_eigen.m`. It determines the eigenvalues and eigenfunctions for a certain solution  $h_0$  obtained, e.g., with the tutorial `SLIDROP` [2]. The eigenvalues/eigenfunctions are then separated into real and imaginary cases which are then handled separately with continuation codes `linslidrop_real.f90` and `linslidrop.f90`, respectively.

To that end, we introduce in eq. (5) the real quantities  $\beta_r, \beta_i, h_1^{(r)}, h_1^{(i)}$  by writing  $\beta = \beta_r + i\beta_i$  and  $h_1(x) = h_1^{(r)}(x) + i h_1^{(i)}(x)$ . This results in the two equations

$$\begin{aligned} Q_0 \partial_{xxxx} h_1^{(r)} &= -\beta_r h_1^{(r)} + \beta_i h_1^{(i)} + [\mathcal{L}[h_0] h_1^{(r)} - Q_0 \partial_{xxxx} h_1^{(r)}] \\ Q_0 \partial_{xxxx} h_1^{(i)} &= -\beta_r h_1^{(i)} - \beta_i h_1^{(r)} + [\mathcal{L}[h_0] h_1^{(i)} - Q_0 \partial_{xxxx} h_1^{(i)}] \end{aligned} \quad (7)$$

To use `auto07p`, we write eq. (7) as a system of first-order autonomous ordinary differential equations on the interval  $[0, 1]$ . Therefore, we introduce for the steady state  $u_1 = h_0 - \bar{h}_0$ ,

$u_2 = \frac{dh_0}{dx}$  and  $u_3 = \frac{d^2 h_0}{dx^2}$ , for the real part of the eigenfunction  $u_{4+j} = \frac{d^j h_1^{(r)}}{dx^j}$  ( $j = 0, 1, 2, 3$ ) and,

analogously, for the imaginary part of the eigenfunction  $u_{8+j} = \frac{d^j h_1^{(i)}}{dx^j}$  ( $j = 0, 1, 2, 3$ ).

In total, this results in a system of  $3 + 4 + 4 = 11$  ordinary differential equations. For the sake of brevity, here we only give the 7 equations for the case  $\beta_i = h_1^{(i)} = 0$  and the case  $Q = 1$  of the example studied here (see Appendix for the full set of equations).

$$\begin{aligned} \dot{u}_1 &= L u_2 \\ \dot{u}_2 &= L u_3 \\ \dot{u}_3 &= L (\partial_{hh} f(u_1 + \bar{h}_0) u_2 - \chi(u_1 + \bar{h}_0) + v(u_1 + \bar{h}_0) + C_0) \\ \dot{u}_4 &= L u_5 \\ \dot{u}_5 &= L u_6 \\ \dot{u}_6 &= L u_7 \\ \dot{u}_7 &= L [(-\beta_r + f'''(u_1 + \bar{h}_0) u_3 + f''''(u_1 + \bar{h}_0) u_2^2 - \chi''(u_1 + \bar{h}_0) u_2) u_4 + \\ & \quad (2u_2 f'''(u_1 + \bar{h}_0) + v - \chi'(u_1 + \bar{h}_0)) u_5 + f''(u_1 + \bar{h}_0) u_6] \end{aligned} \quad (8)$$

where  $L$  is the physical domain size and dots and primes denote derivatives with respect to  $\xi = x/L$  and  $h$  respectively.

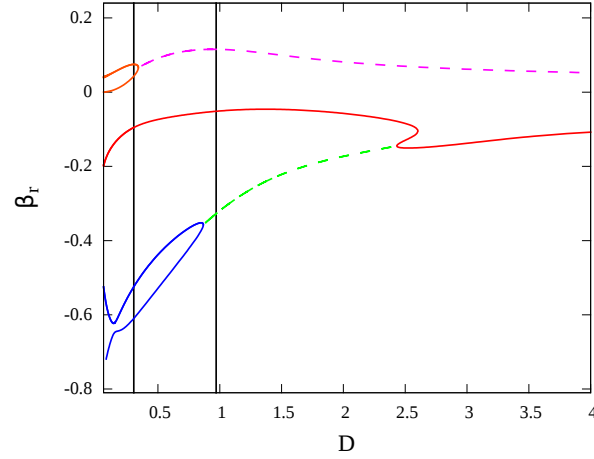
eq. (8) are the equations used in `linslidrop_real.f90` for the continuation of real eigenvalues in the case of  $Q = 1$ . The full set of equations for the continuation of complex eigenvalues with `linslidrop.f90` is listed in the Appendix.

The whole procedure is the following:

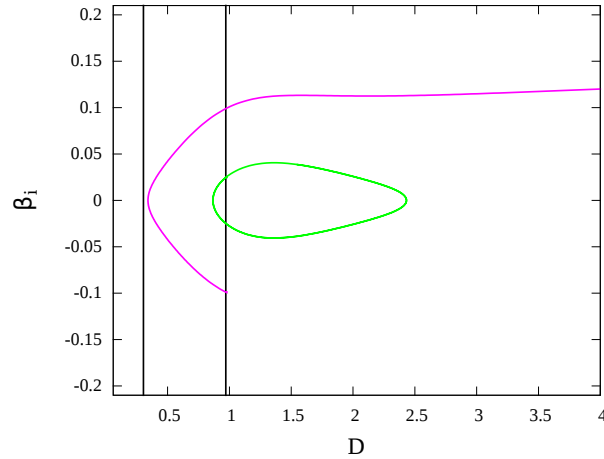
1. A continuation run of tutorial SLIDROP with `auto07p` determines stationary solutions of eq. (1) and writes solutions at particular parameter values into solution files.  
Run 1 follows sitting drop solutions, in particular, for two identical drops that sit beside each other (`ANZ=2` in `slidrop.f90`).  
Then, Run 11 increases the inclination of the substrate from zero, and continues the sliding drops that still have the  $L/2$  translational symmetry as the drops in run 1. Two branching points are detected that correspond to pitchfork bifurcations that break this symmetry.  
Run 111 then switches to one of these side branches with the broken symmetry.
2. The solution files are read into the MatLab code `get_eigen.m` that determines eigenvalues and eigenfunctions at those parameter values. The code then generates the respective `c.-` files for further continuation.
3. The obtained eigenvalues/eigenfunctions are then continued together with the stationary solutions via the `auto07p` codes `linslidrop_real.f90` (real eigenvalues) and `linslidrop.f90` (complex eigenvalues).

## 2 Runs:

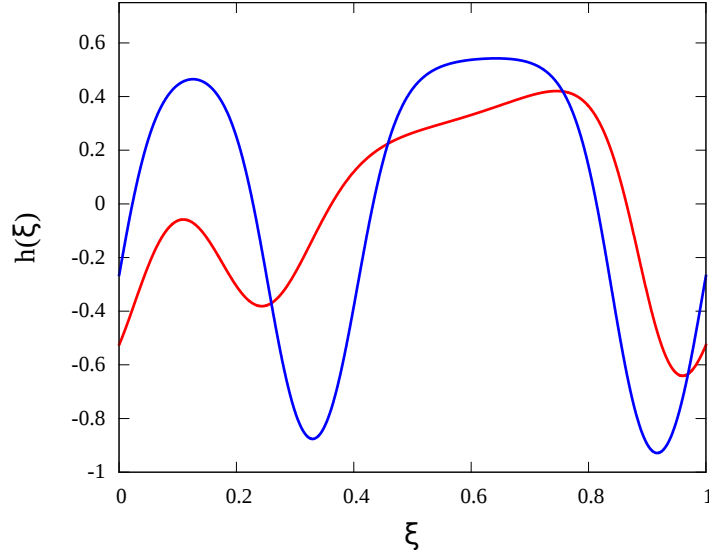
The diagrams in Figs. 1-4 are determined through the continuation runs presented in the following table. The white fields describe what the individual runs do and mention important parameter settings including necessary changes. The grey fields give the `auto07` commands on the left when using the (modern) `Python` interface and on the right when using the more classic command line approach.



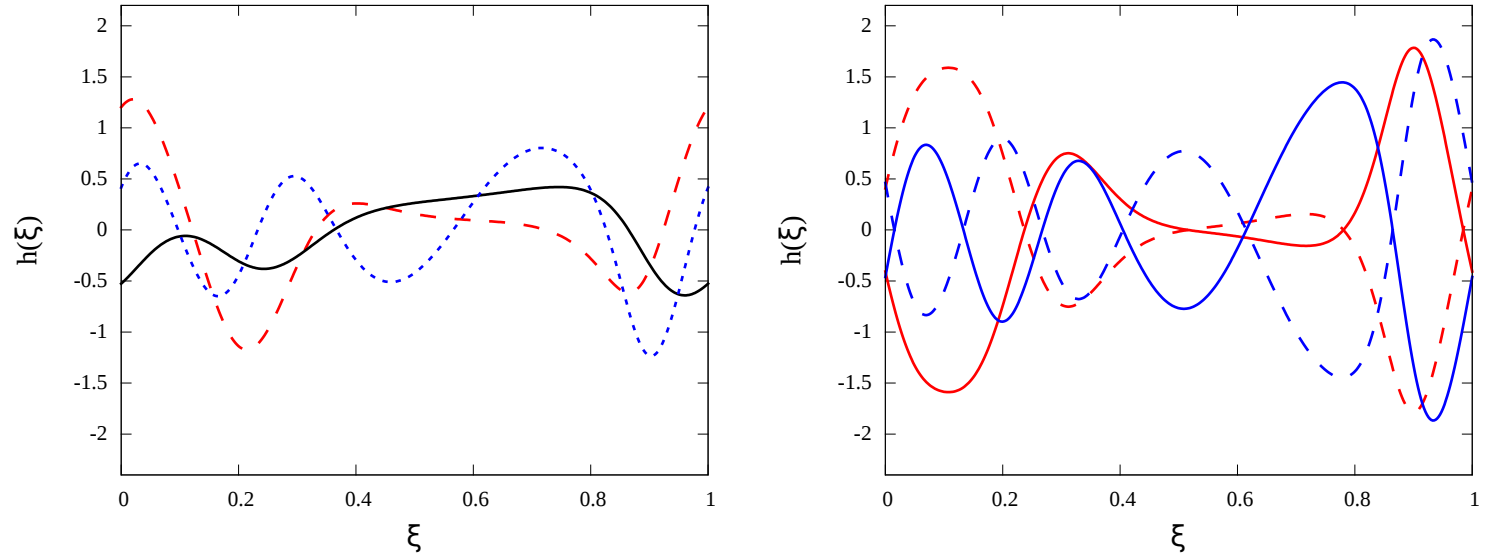
**Figure 1:** Shown is the real part of eigenvalues of eq. (4) as obtained by continuation runs 2 to 41. Full lines correspond to real eigenvalues, dashed lines to complex ones. Vertical black lines mark the positions on which the explicit calculation of eigenvalues by the `matlab` code is performed. The right one corresponds to the first calculation in `run ML1` and the left one serves to validate the continuation in `run ML2`.



**Figure 2:** Shown is the imaginary part of eigenvalues of eq. (4) as obtained by continuation runs 2 to 41. The colors correspond to the real parts in Fig.(1).



**Figure 3:** Stationary state  $h_0$  that is used in `get_eigen.m`. The solution in red corresponds to LAB28 of run 1 used in `runML1` and the blue solution to LAB35 used in `runML2`.



**Figure 4: Left:** Stationary state and real part of two complex eigenfunctions. The stationary state of run 1 is marked in full lines black. The dashed red line corresponds to the unstable complex eigenvalue used in run 4 and the dotted blue line to the stable one used in run3.

**Right:** Imaginary part of the calculated eigenfunctions in `runML1`. The full lines correspond to negative imaginary part of the eigenvalue, the dashed lines to the positive. The colors match to the corresponding real part in the left panel.

Python interface command line	Terminal command line
<i>auto</i>	
<p><b>run 1</b> Determine steady solutions without driving (<math>\text{PAR}(41) = D = 0</math>) as a function of domain size <math>L</math>, starting at the critical <math>L_c</math> with a small amplitude sinusoidal solution. Mean thickness is fixed. Compute the branch of periodic solutions for <math>\bar{h}_0 = 0.4</math>.  <b>Continuation parameters:</b> <math>L</math> (<math>\text{PAR}(5)</math>), <math>C_0</math> (<math>\text{PAR}(6)</math>) and <math>v</math> (<math>\text{PAR}(42)</math>)  <b>Parameters:</b> <math>\text{IPS} = 4</math>, <math>\text{ISP} = 0</math>, <math>\text{ISW} = 1</math>, <math>\text{ICP} = [5, 6, 42]</math>;  Starting from function <code>STPNT</code>;  Save output-files as <code>b.slidrop1</code>, <code>s.slidrop1</code>, <code>d.slidrop1</code></p>	
<code>r1=run(e='slidrop',c='slidrop.1',sv='slidrop1')</code>	<code>@@R slidrop 1</code> <code>@sv slidrop1</code>
<p><b>run 11</b> Restart at domain size <math>L = 22</math>, keep mean thickness <math>h_0</math> fixed and perform a continuation in driving strength <math>D</math>.  <b>Continuation parameters:</b> <math>D</math> (<math>\text{PAR}(41)</math>), <math>C_0</math> (<math>\text{PAR}(6)</math>), <math>v</math> (<math>\text{PAR}(42)</math>)  <b>Parameters:</b> <math>\text{IPS} = 4</math>, <math>\text{ISP} = 2</math>, <math>\text{ISW} = -1</math>, <math>\text{ICP} = [41, 6, 42]</math>;  Starting from LAB2 of run 1;  Save output-files as <code>b.slidrop11</code>, <code>s.slidrop11</code>, <code>d.slidrop11</code></p>	
<code>r11=run('slidrop1',e='slidrop',c='slidrop.11',sv</code>	<code>@@R slidrop 11 slidrop1</code> <code>@sv slidrop11</code>
<p><b>run 111</b> Restart at a branching point from run 11 (LAB4), switch branches (<math>\text{ISW} = -1</math>) and continue in <math>D</math> (<math>\text{PAR}(41)</math>). One writes down every 5th solution (<math>\text{NPR} = 5</math>) to be able to calculate the linear stability at those points;  <b>Continuation parameters:</b> <math>D</math> (<math>\text{PAR}(41)</math>), <math>C_0</math> (<math>\text{PAR}(6)</math>), <math>v</math> (<math>\text{PAR}(42)</math>)  <b>Parameters:</b> <math>\text{IPS} = 4</math>, <math>\text{ISP} = 2</math>, <math>\text{ISW} = -1</math>, <math>\text{ICP} = [41, 6, 42]</math>;  Starting from LAB4 of run 11;  Save output-files as <code>b.drist1</code>, <code>s.drist1</code>, <code>d.drist1</code></p>	
<code>r111=run('slidrop11',e='slidrop',c='slidrop.111</code> <code>sv='drist1')</code>	<code>@@R slidrop 111 slidrop11</code> <code>@sv drist1</code>
<p><b>run ML1:MatLab</b>  Compute eigenfunctions and eigenvalues at <math>\text{PAR}(41) = D \approx 0.97</math> (LAB28 of run111) by using MatLab (<code>sol_nbr=28</code> in line 27 of <code>get_eigen.m</code>). This code will calculate the 7 biggest eigenvalues and the corresponding eigenfunctions;  Starting from <code>*.drist1</code>;  Save output-files as <code>ml_real_outX.dat</code> or <code>ml_cplx_outX.dat</code> depending on the number of the eigenvalue <code>X</code> and whether the eigenvalue ist real or complex.</p>	
	<code>matlab -nodesktop -nosplash -nojvm -r</code> <code>get_eigen</code>
<p><b>run 2</b> Follow the real eigenvalue (<code>ml_real_out5.dat</code>) extracted by the <code>matlab</code> code in parameter <math>D</math> (<math>\text{PAR}(41)</math>), <math>v</math> (<math>\text{PAR}(42)</math>), <math>\beta_r</math> (<math>\text{PAR}(7)</math>) and <math>C_0</math> (<math>\text{PAR}(6)</math>);  <b>Continuation parameters:</b> <math>D</math> (<math>\text{PAR}(41)</math>), <math>v</math> (<math>\text{PAR}(42)</math>), <math>C_0</math> (<math>\text{PAR}(6)</math>), <math>\beta_r</math> (<math>\text{PAR}(7)</math>);  <b>Parameters:</b> <math>\text{IPS} = 4</math>, <math>\text{ISP} = 0</math>, <math>\text{ISW} = 1</math>, <math>\text{ICP} = [41, 42, 6, 7]</math>;  Starting from <code>ml_real_out5.dat</code>;  Save output-files as <code>b.linslidrop2</code>, <code>s.linslidrop2</code>, <code>d.linslidrop2</code></p>	

<code>r2=run(e='linslidrop_real',c='linslidrop_real.5 sv='linslidrop2')</code>	<code>@@R linslidrop_real 5 @sv linslidrop2</code>
<p><b>run 3</b> Follow the negative complex eigenvalue (<code>ml_cplx_out7.dat</code>) extracted by the <code>matlab</code> code in parameter <math>D</math> (PAR(41)), <math>v</math> (PAR(42)), <math>\beta_r</math> (PAR(7)), <math>\beta_i</math> (PAR(8)) and <math>C_0</math> (PAR(6)).;</p> <p><b>Continuation parameters:</b> <math>D</math> (PAR(41)), <math>v</math> (PAR(42)), <math>C_0</math> (PAR(6)), <math>\beta_r</math> (PAR(7)), <math>\beta_i</math> (PAR(8));</p> <p><b>Parameters:</b> IPS= 4, ISP= 0, ISW= 1, ICP= [41, 42, 6, 7, 8];</p> <p>Starting from <code>ml_cplx_out7.dat</code>;</p> <p>Save output-files as <code>b.linslidrop3</code>, <code>s.linslidrop3</code>, <code>d.linslidrop3</code></p>	
<code>r3=run(e='linslidrop',c='linslidrop.7',sv='linslidrop3')</code>	<code>@@R linslidrop 7 @sv linslidrop3</code>
<p><b>run 31</b> Restart at a fold (LAB4) (marked by L in <code>auto07p</code>). Treating the fold as a bifurcation, we try to switch branches (ISW= -1) and indeed go onto the branch of purely real eigenvalues. Note that branch switching is possible even though no BR point was detected.</p> <p><b>Continuation parameters:</b> <math>D</math> (PAR(41)), <math>v</math> (PAR(42)), <math>C_0</math> (PAR(6)), <math>\beta_r</math> (PAR(7)), <math>\beta_i</math> (PAR(8));</p> <p><b>Parameters:</b> IPS= 4, ISP= 0, ISW= -1, ICP= [41, 42, 6, 7, 8];</p> <p>Starting from LAB4 of run 3;</p> <p>Save output-files as <code>b.linslidrop31</code>, <code>s.linslidrop31</code>, <code>d.linslidrop31</code></p>	
<code>r31=run('linslidrop3',e='linslidrop',c='linslidrop3',sv='linslidrop31')</code>	<code>@@R linslidrop 31 linslidrop3 @sv linslidrop31</code>
<p><b>run 4</b> As in run 3, but now follow the unstable complex eigenvalue (<code>ml_cplx_out1.dat</code>). ;</p> <p><b>Continuation parameters:</b> <math>D</math> (PAR(41)), <math>v</math> (PAR(42)), <math>C_0</math> (PAR(6)), <math>\beta_r</math> (PAR(7)), <math>\beta_i</math> (PAR(8));</p> <p><b>Parameters:</b> IPS= 4, ISP= 0, ISW= 1, ICP= [41, 42, 6, 7, 8];</p> <p>Starting from <code>ml_cplx_out1.dat</code>;</p> <p>Save output-files as <code>b.linslidrop4</code>, <code>s.linslidrop4</code>, <code>d.linslidrop4</code></p>	
<code>r4=run(e='linslidrop',c='linslidrop.1',sv='linslidrop4')</code>	<code>@@R linslidrop 1 @sv linslidrop4</code>
<p><b>run 41</b> Restart at L Point (LAB2) of run 4 and proceed as in run 31.</p> <p><b>Continuation parameters:</b> <math>D</math> (PAR(41)), <math>v</math> (PAR(42)), <math>C_0</math> (PAR(6)), <math>\beta_r</math> (PAR(7)), <math>\beta_i</math> (PAR(8));</p> <p><b>Parameters:</b> IPS= 4, ISP= 0, ISW= -1, ICP= [41, 42, 6, 7, 8];</p> <p>Starting from LAB2 of run 4;</p> <p>Save output-files as <code>b.linslidrop41</code>, <code>s.linslidrop41</code>, <code>d.linslidrop41</code></p>	
<code>r41=run('linslidrop4',e='linslidrop',c='linslidrop4',sv='linslidrop41')</code>	<code>@@R linslidrop 41 linslidrop4 @sv linslidrop41</code>



<b>run ML2:MatLab</b>	
<p>Compute eigenfunctions and eigenvalues at <math>\text{PAR}(41) = D \approx 0.3</math> (LAB21 of run1) by using MatLab. LAB21 corresponds to the 14th solution in run1 as it starts with LAB8, so change line 6 of <code>get_eigen.m</code> to <code>sol_nbr=14</code>. This code will calculate the 7 biggest eigenvalues and the corresponding eigenfunctions. This run should serve as a validation of the previous procedure. Compare the calculated eigenvalues with those calculated by continuation (see Fig.1); Starting from <code>*.drist1</code>;  Save output-files as <code>ml_real_outX.dat</code> or <code>ml_cplx_outX.dat</code> depending on the number of the eigenvalue X</p>	
	<code>matlab -nodesktop -nosplash -nojvm -r get_eigen</code>
<code>clean()</code>	<code>@cl</code>

Table 1: Commands for running tutorial DRIST.

### 3 Remarks:

- We can switch to branches of real eigenvalues in run 31 and run 41 by using the equation file for complex eigenvalues, thereby we would be able to analyze the whole parameter space just by using the equation file `linslidrop.f90`.
- Adaptation of the codes to other  $\chi$ ,  $Q$  and  $f$  are, in general, possible. To do so one has to define  $\chi$ ,  $Q$  and  $f$  in the `.f90`-files (lines 43-48 in `slidrop.f90`, lines 50-64 in `linslidrop_real.f90` and lines 60-74 in `linslidrop.f90`) and in the Matlab-code in lines 382-442. One must also have in mind that the parameter indices in lines 34-53 may have to be changed.

### 4 Tasks:

1. Use the code `get_eigen.m` somewhere else in parameter space, e.g. for  $D > 3$  as one will get some eigenvalues not visible in Fig.1.
2. Increase the number of eigenvalues calculated by `get_eigen.m` by increasing `MaxEVs` (line 35).
3. Follow the same procedure for the main branch of periodic steady states in the comoving frame of run 11. One should observe eigenvalues crossing 0 at least at the detected branching points.
4. Adapt the procedure for another kind of thin-film equation, e.g. the for the case of sliding drops on a heated substrate in [10].

## Appendix

The full set of equations incorporated in `linslidrop.f90` are:

$$\begin{aligned}
\dot{u}_1 &= Lu_2 \\
\dot{u}_2 &= Lu_3 \\
\dot{u}_3 &= \frac{Bo \cdot L}{Q_0} (\partial_{hh} f(u_1 + \bar{h}_0) u_2 - \chi(u_1 + \bar{h}_0) + v(u_1 + \bar{h}_0) + C_0) \\
\dot{u}_4 &= Lu_5 \\
\dot{u}_5 &= Lu_6 \\
\dot{u}_6 &= Lu_7 \\
\dot{u}_7 &= \frac{Bo \cdot L}{Q_0} [(Q_0 f'''(u_1 + \bar{h}_0) u_3 + Q_0 f''''(u_1 + \bar{h}_0) u_2^2 - \chi''(u_1 + \bar{h}_0) u_2) u_4 + \\
&(2Q_0 u_2 f'''(u_1 + \bar{h}_0) + v - \chi'(u_1 + \bar{h}_0)) u_5 + Q_0 f''(u_1 + \bar{h}_0) u_6 - \beta_r u_4 + \beta_i u_8] \\
&- \frac{Bo \cdot L}{Q_0} (Q'_0) \left[ \frac{u_2 u_7}{Bo} - 2f'''(u_1 + \bar{h}_0) u_2^2 u_4 - 2f''(u_1 + \bar{h}_0) u_2 u_5 - \chi(u_1 + \bar{h}_0) u_5 \right. \\
&\quad \left. - \chi'(u_1 + \bar{h}_0) u_2 u_4 + v \cdot (u_1 + \bar{h}_0) u_5 + v u_2 u_4 + C_0 u_5 \right. \\
&\quad \left. + f''(u_1 + \bar{h}_0) (u_2 u_5 - u_3 u_4) + u_4 (f'''(u_1 + \bar{h}_0) u_2^2 + f''(u_1 + \bar{h}_0) u_3) \right] \\
&\quad - \frac{Bo \cdot L}{Q_0} (Q''_0) [(v(u_1 + \bar{h}_0) + C_0 - \chi(u_1 + \bar{h}_0)) u_2 u_4] \\
\dot{u}_8 &= Lu_9 \\
\dot{u}_9 &= Lu_{10} \\
\dot{u}_{10} &= Lu_{11} \\
\dot{u}_{11} &= \frac{Bo \cdot L}{Q_0} [(Q_0 f'''(u_1 + \bar{h}_0) u_3 + Q_0 f''''(u_1 + \bar{h}_0) u_2^2 - \chi''(u_1 + \bar{h}_0) u_2) u_8 + \\
&(2Q_0 u_2 f'''(u_1 + \bar{h}_0) + v - \chi'(u_1 + \bar{h}_0)) u_9 + Q_0 f''(u_1 + \bar{h}_0) u_{10} - \beta_r u_8 - \beta_i u_4] \\
&- \frac{Bo \cdot L}{Q_0} (Q'_0) \left[ \frac{u_2 u_{11}}{Bo} - 2f'''(u_1 + \bar{h}_0) u_2^2 u_8 - 2f''(u_1 + \bar{h}_0) u_2 u_9 - \chi(u_1 + \bar{h}_0) u_9 \right. \\
&\quad \left. - \chi'(u_1 + \bar{h}_0) u_2 u_8 + v \cdot (u_1 + \bar{h}_0) u_9 + v u_2 u_4 + C_0 u_9 \right. \\
&\quad \left. + f''(u_1 + \bar{h}_0) (u_2 u_9 - u_3 u_8) + u_8 (f'''(u_1 + \bar{h}_0) u_2^2 + f''(u_1 + \bar{h}_0) u_3) \right] \\
&\quad - \frac{Bo \cdot L}{Q_0} (Q''_0) [(v(u_1 + \bar{h}_0) + C_0 - \chi(u_1 + \bar{h}_0)) u_2 u_8]
\end{aligned}$$

## References

- [1] U. Thiele. *Münsteranian Torturials on Nonlinear Science: CCH - sliding clusters and travelling waves in the convective Cahn-Hilliard equation*. Ed. by U. Thiele, O. Kamps, and S. V. Gurevich. 2021. DOI: [10.5281/zenodo.4546352](https://doi.org/10.5281/zenodo.4546352).
- [2] U. Thiele. *Münsteranian Torturials on Nonlinear Science: SLIDROP - sliding drops on an inclined homogeneous substrate*. Ed. by U. Thiele, O. Kamps, and S. V. Gurevich. 2021. DOI: [10.5281/zenodo.4546381](https://doi.org/10.5281/zenodo.4546381).
- [3] E. Doedel, H. B. Keller, and J. P. Kernevez. “Numerical analysis and control of bifurcation problems (I) Bifurcation in finite dimensions”. In: *Int. J. Bifurcation Chaos* 1 (1991), pp. 493–520. DOI: [10.1142/S0218127491000397](https://doi.org/10.1142/S0218127491000397).
- [4] B. Krauskopf, H. M. Osinga, and J. Galan-Vioque, eds. *Numerical Continuation Methods for Dynamical Systems*. Dordrecht: Springer, 2007. DOI: [10.1007/978-1-4020-6356-5](https://doi.org/10.1007/978-1-4020-6356-5).

- [5] H. A. Dijkstra et al. “Numerical bifurcation methods and their application to fluid dynamics: Analysis beyond simulation”. In: *Commun. Comput. Phys.* 15 (2014), pp. 1–45. DOI: [10.4208/cicp.240912.180613a](https://doi.org/10.4208/cicp.240912.180613a).
- [6] E. J. Doedel and B. E. Oldeman. *AUTO07p: Continuation and bifurcation software for ordinary differential equations*. Montreal: Concordia University, 2009. URL: <http://indy.cs.concordia.ca/auto>.
- [7] MATLAB. *Version 9.9.0.1570001 (R2020b) Update 4*. 2020.
- [8] U. Thiele. *Münsteranian Torturials on Nonlinear Science: Overview of available tutorials on path continuation*. Ed. by U. Thiele, O. Kamps, and S. V. Gurevich. 2021. DOI: [10.5281/zenodo.4548111](https://doi.org/10.5281/zenodo.4548111).
- [9] U. Thiele and E. Knobloch. “Front and back instability of a liquid film on a slightly inclined plate”. In: *Phys. Fluids* 15 (2003), pp. 892–907. DOI: [10.1063/1.1545443](https://doi.org/10.1063/1.1545443).
- [10] U. Thiele and E. Knobloch. “Thin liquid films on a slightly inclined heated plate”. In: *Physica D* 190 (2004), pp. 213–248. DOI: [10.1016/j.physd.2003.09.048](https://doi.org/10.1016/j.physd.2003.09.048).
- [11] C. L. Emmott and A. J. Bray. “Coarsening dynamics of a one-dimensional driven Cahn-Hilliard system”. In: *Phys. Rev. E* 54 (1996), pp. 4568–4575. DOI: [10.1103/PhysRevE.54.4568](https://doi.org/10.1103/PhysRevE.54.4568).
- [12] A. A. Golovin et al. “Convective Cahn-Hilliard models: From coarsening to roughening”. In: *Phys. Rev. Lett.* 86 (2001), pp. 1550–1553. DOI: [10.1103/PhysRevLett.86.1550](https://doi.org/10.1103/PhysRevLett.86.1550).
- [13] M. D. Korzec et al. “Stationary solutions of driven fourth- and sixth-order Cahn-Hilliard-type equations”. In: *SIAM J. Appl. Math.* 69 (2008), pp. 348–374. DOI: [10.1137/070710949](https://doi.org/10.1137/070710949).
- [14] D. Tseluiko et al. “Effect of driving on coarsening dynamics in phase-separating systems”. In: *Nonlinearity* 33 (2020), pp. 4449–4483. DOI: [10.1088/1361-6544/ab8bb0](https://doi.org/10.1088/1361-6544/ab8bb0).
- [15] U. Thiele. *Münsteranian Torturials on Nonlinear Science: LINDROP - linear stability of steady states on a horizontal homogeneous substrate*. Ed. by U. Thiele, O. Kamps, and S. V. Gurevich. 2021. DOI: [10.5281/zenodo.4546375](https://doi.org/10.5281/zenodo.4546375).
- [16] E. Doedel, H. B. Keller, and J. P. Kernevez. “Numerical analysis and control of bifurcation problems (II) Bifurcation in infinite dimensions”. In: *Int. J. Bifurcation Chaos* 1 (1991), pp. 745–72. DOI: [10.1142/S0218127491000555](https://doi.org/10.1142/S0218127491000555).
- [17] J. M. Skotheim, U. Thiele, and B. Scheid. “On the instability of a falling film due to localized heating”. In: *J. Fluid Mech.* 475 (2003), pp. 1–19. DOI: [10.1017/s0022112002001957](https://doi.org/10.1017/s0022112002001957).