Implementation of the Thin-Film Equation on Prestructured, Switchable Substrates Using the `oomph-lib` Library

10th January 2022

Authors: Janik Suer
Moritz Stieneker
Svetlana Gurevich

Institute for Theoretical Physics, University of Münster

1 Tutorial: Prestructured Substrate

This tutorial shows how a switchable prestructured substrate for a one and two-dimensional thin-film model[1] can be implemented using the `oomph-lib` library[5]. In particular, the case of a moving step wettability profile [4] was considered. Within this tutorial, the code necessary to reproduce all the simulations and analysis is explained[2]. Table 1 gives an overview of the necessary files. In the accompanying source code, different wettability prestructures and switching patterns are already implemented, which can be used with minor adaptions to the supplied source code.

1.1 Theoretical Background

For the local height profile $h(x,t)$ of simple partially wetting liquids on a surface one can derive a partial differential equation called the thin-film equation, by using the Navier-Stokes equation with appropriate boundary conditions and the assumption of small gradients [8, 1, 11, 7]. The thin-film equation in the non-dimensionalized [6, 3, 10] form can be expressed as

$$\partial_t h(x,t) = \nabla Q(h) \nabla [P(h,x,t)]. \tag{1}$$

Here $Q(h) = h^3$ is the mobility resulting from the no-slip boundary conditions and $P(h,x,t)$ is the general pressure, which is equal to the variation of the free energy F_0 with respect to the film height. It is given by:

$$P(h,x,t) = -\Delta h - \Pi(h,x,t) = \frac{\delta F_0}{\delta h}, \tag{2}$$

where the first term $-\Delta h$ represents the so-called Laplace pressure. This pressure is a result of the surface tension at the air-liquid interface and is equal to the difference of pressure inside and outside the droplet (i.e. in the liquid and gas phase). The second term, $\Pi(h,x,t)$ is the so-called disjoining-pressure or Derjaguin pressure. It originates from the interaction between the liquid and solid surface and therefore also has to include the structure (i.e. wettability) of the substrate. The disjoining pressure is given by

$$\Pi(h,x,t) = \left(\frac{1}{h^6} - \frac{1}{h^3}\right)[1 + \rho g(x,t)], \tag{3}$$

where $g(x,t)$ is the structure-function and ρ is the wettability contrast. The product of $g(x,t)$ and ρ is the so-called wettability. In general, a higher wettability value results in a smaller contact angle. Hence a lower wettability results in larger contact angles, i.e. a increased droplet height if the available volume remains constant. Other forms of the disjoining pressure and

[1]For an in-depth introduction into the thin-film equation see [8]

[2]The necessary code is available online: `https://doi.org/10.5281/zenodo.5821537`.

the mobility are possible [1]. To avoid pinned contact lines this modeling approach uses a so-called precursor film. A thin liquid film which covers the entire substrate. Other possible solutions to this problem are depicted in [1].
In the following, we will look at different forms of $\rho g(x,t)$ and the effects on direct time simulations of these wettability profiles.

1.2 Stationary Wettability Pattern

One of the easiest (non-homogeneous) wettability profiles consists of two areas with different wettabilities. A possible way to obtain these two adjacent areas of different wettabilities with a smooth transition[3] between them is the logistic function (see [4] for more details):

$$\rho g(x) = \rho_{\mathrm{LW}} - \frac{\rho_{\mathrm{LW}} - \rho_{\mathrm{HW}}}{1 + e^{-\frac{x-c}{ls}}}. \tag{4}$$

Here, ρ_{LW} and ρ_{HW} correspond to a smaller and higher wettability value respectively. The constant c determines the x coordinate at which the transition between these two values occurs[4], while ls determines the smoothness of this transition. That is, a large value for ls corresponds to a smooth transition, whereas a small value would result in a steep transition. In the limes $ls \to 0$ the wettability profile goes over to a simple step function. The inhomogeneity profile is shown in fig. 1 for different values of ls relative to the domain size L. One can see how a small ls value results in a function resembling a simple step function, while for large values the transition area take up the entire domain. Here, ρ_0 is the mean value of ρ_{HW} and ρ_{LW}. Therefore it is equal to the wettability value at the steepest slope of $\rho g(x)$.
The implementation of this wettability pattern into `C++` code is shown in the following[5]:

[3] One could also use discontinuous jumps in theory but for this tutorial we stick to the continuous case.

[4] In particular, it determines the position of the steepest slope of the inhomogeneity profile.

[5] Note that the variables themselves are defined in the namespace `ThinFilm` within a different file (see section 1.4). The namespace is used here to access the parameters.

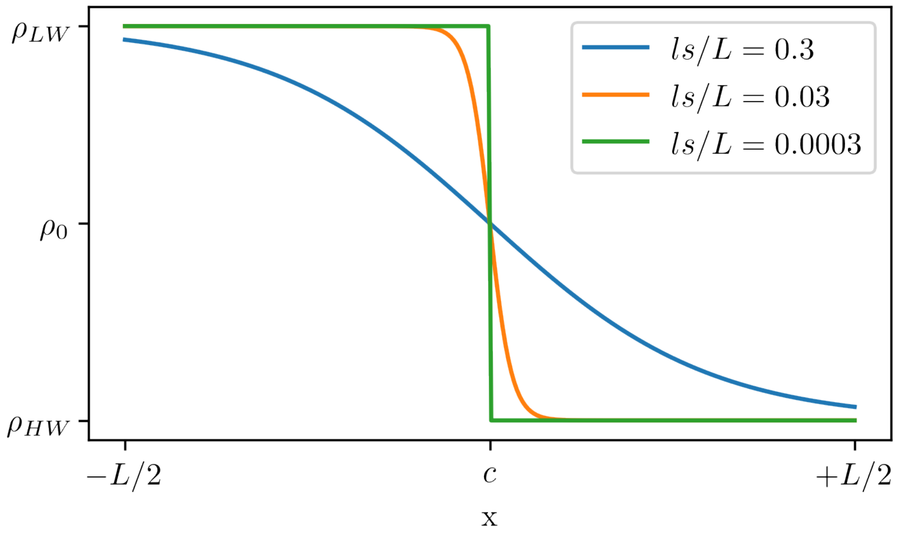


Figure 1: Logistic step function according to eq. (4) as a wettability profile with the high and low wettability values ρ_{HW} and ρ_{HW}, the mean wettability value ρ_0, the domain length L and the smoothness of the transition ls. The graph shows the wettability profile for three different values of ls.

```
using namespace ThinFilm;
double MyNDProblem<ELEMENT>::logistic(Vector<double> x, double
    t, double rho){
  return rhoLW - (rhoLW -rhoHW)/(1+exp(-(x[0] - cpos[0])/ls));
}
```

Listing 1: Implementation a stationary step wettability.

Here, `cpos[0]` $= c$ and the other variable equal to their counterparts in eq. (4) .
A droplet placed at the steepest slope of the wettability step experiences a wettability gradient and thus will move towards the area of high wettability [9]. It covers an increasingly large area of high wettability and will spread out since this is energetically favourable.

The initial height distribution of the droplet is given by eq. (5), where χ is the precursor film height. Equation (5) can be derived by approximating the droplet as a parabola of height h_0 and an initial contact angle θ_{init} which results in a slope equal to $\tan(\Theta_{\text{init}})$ at the contact line, i.e. the intersection of the droplet with the precursor film. The value of h_0 is equal to the maximum film height at $t = 0$. In order to include the precursor film in the equation for the initial film height the maximum function is being used, such that the minimum film height is equal to the precursor film height χ.

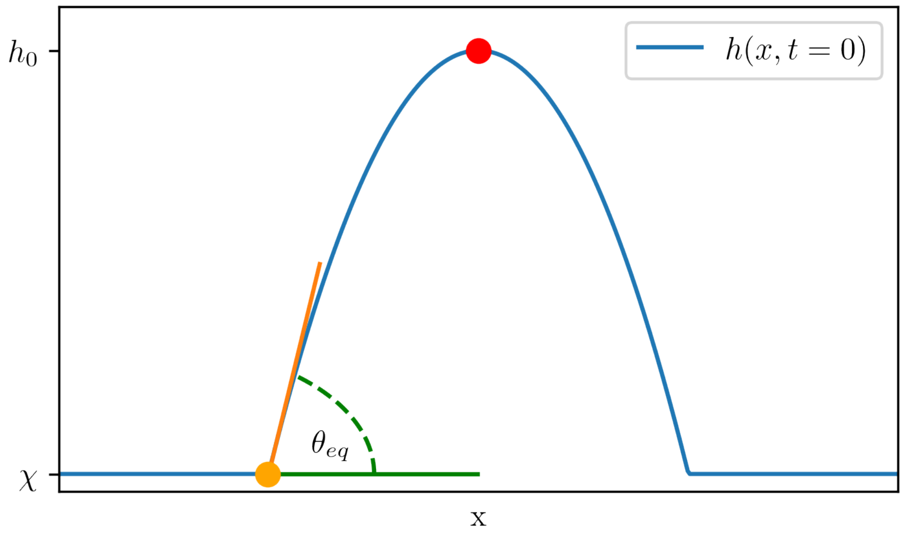


Figure 2: Height profile of the initial droplet. The slope at the contact line $\tan(\Theta_{\text{eq}})$ (orange) as well as the contact angle Θ_{eq} (green) and initial height (red) are shown.

A visualization of the initial droplet is shown in fig. 2. For the simulations presented here the initial conditions were $h_0 = 1$ and $\Theta_{\text{eq}} \approx 44.38°$.

$$h(x,t=0) = \max\left(-\frac{1}{4h_0}\left[\tan\left(\Theta_{\text{eq}}\right)\cdot x\right]^2 + h_0, \chi\right) \tag{5}$$

Snapshots of the one-dimensional direct time simulation using this wettability profile are shown in fig. 3. Figure 3a depicts the initial state of the droplet, which has not yet experienced the present inhomogeneity profile. Figure 3b shows the end state of the simulation. The droplet has moved away from the centre of the transition and into the more wettable area while also decreasing its height.

1.3 Moving Wettability Pattern

For each stationary pattern one can also construct a moving wettability pattern, preserving all used parameters whilst also introducing the speed of the inhomogeneity v_{inhom} as a new parameter. This velocity is inserted into the the inhomogeneity eq. (4) by applying a transformation to the x coordinate:

$$\tilde{x}(t) = \left[(x - v_{\text{inhom}}\cdot t) + L/2\right]\%L - L/2. \tag{6}$$

The reason for the minus sign in eq. (6) is that one wants the inhomogeneity profile to move towards higher x values for a positive velocity. This trans-

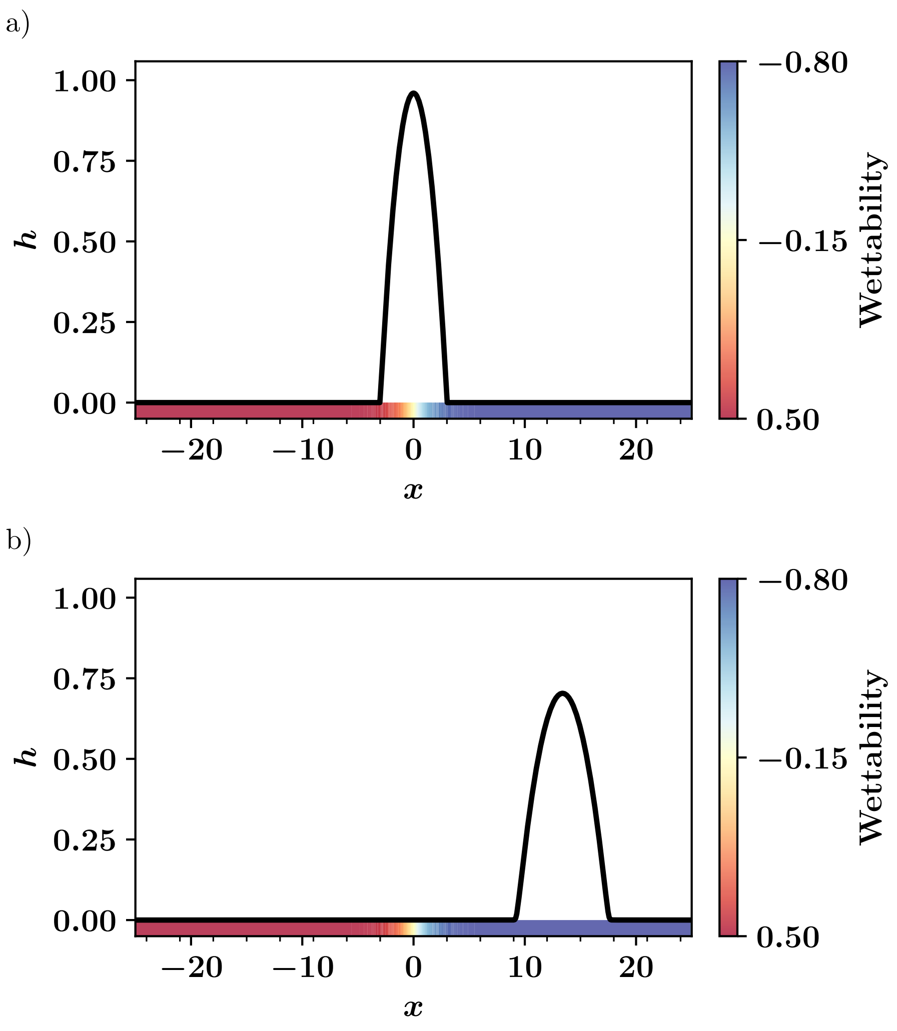


Figure 3: Snapshots of the direct time simulation of a one-dimensional drop on a substrate with a stationary logistic function as the wettability. The parameters for this simulation are $L = 50$, $ls = 1$, $c = 0$, $\rho_{\mathrm{LW}} = 0.5$, $\rho_{\mathrm{HW}} = -0.8$. a) initial state droplet placed at the center of the wettability transition, b) final state of the droplet, spread out over the area with high wettability.

formation takes periodic boundary conditions in x-direction into account for a domain centered at $x = 0$ with length L.
The inhomogeneity code is slightly modified and now reads[6]:

```
using namespace ThinFilm
template <class ELEMENT>
double MyNDProblem<ELEMENT>::moving_logistic(Vector<double> x,
                                              double t, double rho) {
  double L = L_nd[0];
  double x_tilde = x[0] - speed * t - cpos[0];
  double x_tilde_sign = (x_tilde > 0 ) - (x_tilde < 0 );
  double remapped_x_tilde = std::fmod(x_tilde + x_tilde_sign *
    L/2, L) - x_tilde_sign * L/2;
  return rhoLW - (rhoLW -rhoHW)/(1+exp(-remapped_x_tilde/ls));
  }
```

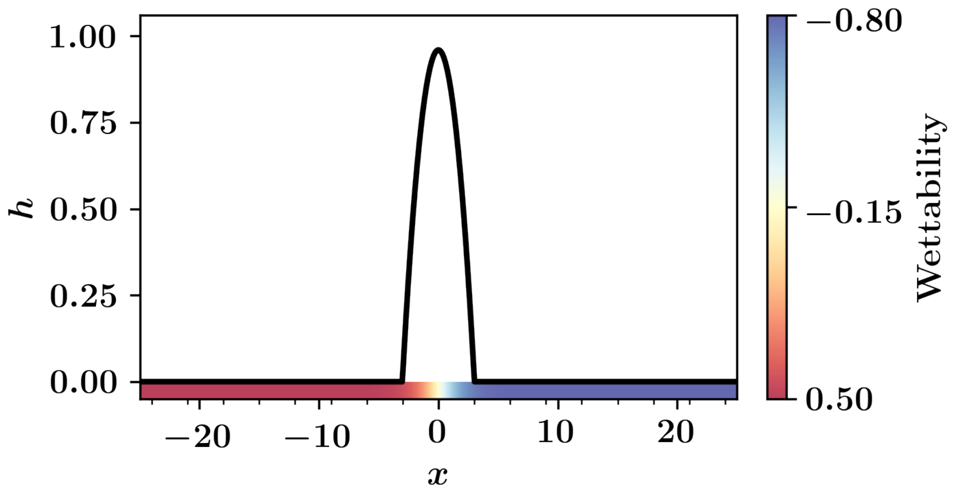
Listing 2: Implementation of the moving step wettability.

This sample is taken from `my_NDproblem.h`. The variable `speed` is equal to the variable previously denoted as v_{inhom}.
Direct time simulations of a time dependent wettability profile can show more involved dynamics then those of stationary ones. In case of the profile given in listing 2, the droplet still exhibits a initial behaviour analogous to the one shown in fig. 3. It starts to move away from the point of the steepest slope and towards the more wettable area, but since the step itself is also moving[7] the droplet will continue to experience a wettability gradient and therefore keep on moving away from the area of low wettability (see [2]). Snapshots of the direct one-dimensional time simulation are shown in fig. 4. In fig. 4a the droplet has not yet reached its equilibrium distance to the point of maximum steepness and is moving away from this point with a speed $v_{\mathrm{drop}} > v_{\mathrm{inhom}}$. In fig. 4b the droplet has reached the equilibrium distance and is moving at the constant speed $v_{\mathrm{drop}} = v_{\mathrm{inhom}}$.
One can imagine that this kind of behaviour will not sustain for any speed of the inhomogeneity. Once the step has reached a speed larger than some critical value the droplet will not be able to keep up with the step and fall behind. Since the inhomogeneity profile away from the point of the steepest slope is homogeneous (see fig. 1) the droplet will, once it has reached this area, stop moving.
This behaviour can be analyzed by observing the droplet speed while varying the inhomogeneity speed and keeping all other parameters constant. The droplet speed can be measured by looking at the temporal behaviour of the position of maximum height. This value is saved in the `trace.dat` file for each time step. The average velocity between timestep i and j, can be

[6]The `C++` code is not as compact as eq. (6) because we had to account for the sign sensitive definition of the modulo function.

[7]We only consider movement into the direction of the more wettable area here.

a)



b)

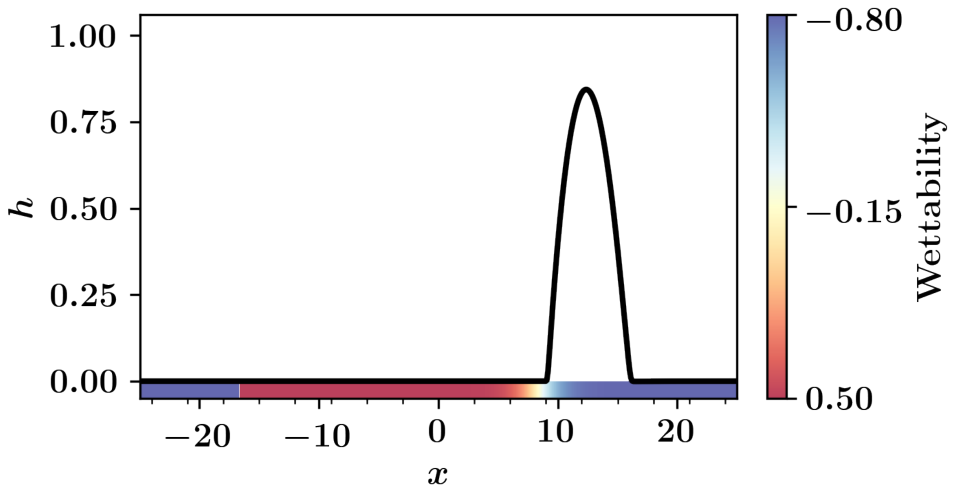


Figure 4: Snapshots of the direct time simulation of a one-dimensional drop on a substrate with a moving logistic function as the wettability. The parameters for this simulation are the same as in fig. 3, with the additional inhomogeneity speed $v_{\mathrm{inhom}} = 0.01$. The initial state is equal to fig. 3a. Figure 4a shows the droplet moving away from the inhomogeneity step with a velocity larger than v_{inhom}. Figure 4b shows the final state of the droplet, moving at a constant speed v_{inhom}.

obtained using finite differences:

$$\bar{v}_{\text{drop}} = \frac{h_{\max}[j] - h_{\max}[i]}{t[j] - t[i]}. \tag{7}$$

With $h_{\max}[i/j]$ and $t[i/j]$, equal to the position of maximum height and time at time step i/j respectively. In order to obtain the equilibrium drop velocity, i.e. the drop velocity for infinitely large times, one has to discard the first time steps, since at the beginning of the simulation the droplet will be moved by the inhomogeneity step regardless of the speed of the step. It is reasonable to start measuring once the droplet has reached its equilibrium distance from the step (see fig. 4b) or is already left behind. The closer the step speed is to the critical value, while still being larger than it, the longer the droplet will be moved by the step. Therefore the amount of steps which need to be discarded becomes increasingly large for step speeds close to the critical value This will result in deviations of the measured velocity from the equilibrium speed for finite time simulations as we will see later on.
The script `plot_speed_step_vs_speed_drop.py` is used to perform and visualize the analysis of the drop speed. It also contains a routine to plot the average drop velocity against the speed of the step (cf. fig. 5). The data used for this plot can be generated by using the `scan_parallel.py`[8] script. It uses multiple cores to perform direct one-dimensional time simulations of the problem (see section 1.4). Since we only want to consider one passing of the inhomogeneity profile, the integration time T has to be restricted such that the integration stops before the step reaches the domain border. A possible way, which was used for fig. 5 is:

$$T = 0.4 \cdot \frac{L - c}{v_{\text{inhom}}}. \tag{8}$$

In fig. 5 the average droplet velocity $\bar{v}_{\text{drop}}$ is plotted against the speed of the inhomogeneity profile v_{inhom}. It can be seen that the droplets velocity matches that of the step until a threshold value (which in this case is at around $v_{\text{inhom}} = 0.07$) is reached. The droplet then stops moving since it can no longer keep up with the step. As explained the before the finite time simulations lead to a non vanishing $\bar{v}_{\text{drop}}$ for step speeds slightly larger than the critical one. For increasingly long time simulations this effect will decrease.

1.4 Implementation details

The files relevant for the implementation of the inhomogeneity step are given in table 1 together with a short description. The backbone of the time

[8]One may use the adaptive mesh here, since the domain is large and only needs a high resolution near the droplet. See section 1.5 for more details on the adaptive mesh.

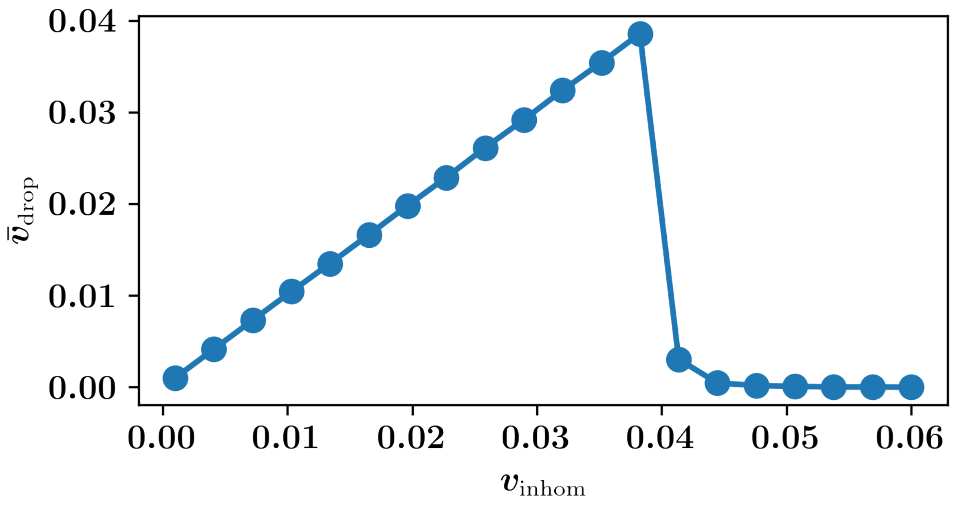


Figure 5: Average droplet velocity $\bar{v}_{\text{drop}}$ plotted against the speed of the inhomogeneity profile v_{inhom}. All parameters except the drop velocity are the same as in fig. 4.

simulations is file 5. Within this file the jacobian and residual are calculated. In both calculations the inhomogenity is included. For the jacobian this looks as follows:

```
double inhom = get_inhomogenity(x, time, rho);
double modulation = 1 + inhom;
```

Where the function `get_inhomogenity()` returns the wettability value and is equal to the term $\rho g(x,t)$ in eq. (3). The exact form of `get_inhomogenity()` is defined in file 1 through the function pointer `inhomfctpt`. The actual function it points to can either be set by using the constructor (either of the one or the two dimensional problem) or using the function `set_inhomogenity()`, which takes a string as an argument. The possible arguments for this function are listed in line 654−671 of file 1. The relevant argument for the previously discussed inhomogeneity is `"moving_logistic"`. The function pointer then points at the `moving_logistic()` function already shown in listing 2.
In the one- as well as in the two-dimensional case the boundary conditions in x-direction are periodic. For the y-direction we have chosen the natural boundary conditions of the Finite Element method, the Neumann boundary conditions.
For a simple time simulation, one can use the `run.py` file. It consists of three parts:

1. Initializing the problem using the `init_problem()` function provided by `scan_basics.py`. Here all parameters (ρ_{HW}, ρ_{LW}, ls, etc.) may be

Table 1: `C++` and `Python` scripts used for performing the simulations and analysis.

Number	Filename	Description
1	`my_NDproblem.h`	N dimensional problem with wettability pattern c of 4 uses 5, derivation of 4
2	`my_1Dproblem.h`	one dimensional derivation of 1
3	`my_2Dproblem.h`	two dimensional derivation of 1
4	`my_generic_problem.h`	general problem class
5	`prestructured_tfe _element.h`	element for prestructured surfaces derivation of `pde_elements.h`
6	`libprestructured _1Dtfe.cpp`	python bindings for simulations of 1D prestructured surfaces
7	`libprestructured _2Dtfe.cpp`	2D version of 6
8	`scan_basics.py`	convenient python functions for direct 1D time simulations uses 6
9	`scan_basics2D.py`	2D version of 8
9	`run.py`	Short script for performing single direct time simulations uses 2, 6, 8
10	`scan_parallel.py`	parallely performs multiple direct time simulations uses 2, 6, 8
11	`plot1D_fancy.py`	parallel plotting of each created `.dat` file
12	`plot_speed_step _vs_speed_drop.py`	Calculates the $\bar{v}_{\text{drop}}$ and plots it against v_{step}

handed over to the function as keyword arguments.

2. Performing the actual integration[9]. This can be done by:

   - `integrate_until_steady()`, which integrates the problem until the adaptive step size has reached `dt_max`[10],

   - `integrate(n)` or `integrate_for_time(T)` which take the number of integration steps n or the integration time T as arguments to specify how long the problem should be integrated.

   All functions are defined in file 4. For each time step a `.dat` file is created and saved within the `dat` folder of the temporary output folder (`"out"` by default). Also one line is written into the `trace.dat` file (in the output folder) which includes different measured quantities for example the current free energy and position of the maximum.

3. Cleaning up and plotting. The temporary output folder is moved into the `finalfolder`, together with the scipts used for the simulations and the parameters saved in the `params.json` file. Also the `.dat` files in `finalfolder` are plotted using `plot1D_fancy.py`.

Using the `run.py` file one can obtain the pictures shown in figs. 3 and 4. For the moving case the file should look like the following:

```python
import libprestructured_1Dtfe as TFElib
from parameter_scanning.scan_basics import *
from datetime import datetime

SCRIPTNAME = "run.py"
SCAN_BASICS_DIR = "parameter_scanning"

problem = init_problem(L = 50, N = 512, rhoHW = -0.8, rhoLW =
    0.5, ls = 1.0, speed = 0.01, inhomogenity =
    "moving_logistic")
problem.set_elements_to_equilibrated()
problem.integrate_for_time(1200)

timestamp = datetime.now().strftime("%Y%m%d-%H%M%S")
finalfolder = f"out_1D_run_{timestamp}"
tidy_up(problem,finalfolder,SCRIPTNAME,SCAN_BASICS_DIR)
# Plot output directly
os.system("./plotting_scripts/plot1D_fancy.py "+finalfolder)
```

[9]In order to use a moving wettability pattern the elements must first be set to equilibrated. This can be done using the function `set_elements_to_equilibrated()` defined in file 1.

[10]The variable is defined in file 4 and can also be set in the python script.

Changing the speed to zero will give the plots shown in fig. 3.
For plots like the one shown in fig. 5 one could perform one each simulation i.e. one for each value of v_{inhom} or use the `scan_parallel.py` file, which uses multiprocessing to scan the parameters given in the `parameter_sets` array. While all other parameters are the same as in section 1.5 the speed is chosen to be:

```
speed_array = np.linspace(0.001,0.06,20)
```

The script performs one `run.py` for each parameter set (without the plotting). The final output is a folder named according to the current date and time `out_1D_parameter_scan_YYYYMMDD_hhmmss`. Within this directory, one folder exists for each parameter scan including the same data as the ones created by `run.py`. In order to obtain a plot like the one in fig. 5 the script `plot_speed_step_vs_speed_drop` has to be executed with the final parameter scan directory as an argument. The resulting plot is then saved in a folder named `result` within the output directory.

1.5 Extension to Two Dimensions

For the extension of the one-dimensional case to two dimensions, one can use mostly the same code, i.e. `my_NDproblem.py`, since the contained base class `MyNDProblem` includes all the general code needed for arbitrary dimensions. The derived classes `My1DProblem` and `My2DProblem` inherit this code. The addition they provide is the code which is unique to the respective dimension. The change of the problem class now requires an adaptation of the used python bindings `libprestructured_2Dtfe.cpp`, the utility functions `scan_basics2D.py` as well as the run `run2D.py`, scan `scan_2D_parallel.py` and plotting `plot2D_fancy.py` scripts in python. All the code looks analogous to the one used before. Since we now handle a two-dimensional substrate, one has to provide two domain lengths `Lx`, `Ly` and number of nodes `Nx`, `Ny`.
Two snapshots of the two-dimensional problem are shown in fig. 6. The initially placed droplet is again given by a formula analogous to eq. (5) but since we are now considering a two-dimensional substrate the coordinate of the second dimension y is included as:

$$h(x,y) = -\frac{1}{4h_0}\tan\left(\Theta_{\text{eq}}\right)^2 \cdot \sqrt{x^2 + y^2} + h_0. \tag{9}$$

Figure 6a is the initially placed droplet (eq. (9)), whereas fig. 6b is the state of constant droplet speed $v_{\text{drop}} = v_{\text{inhom}}$. One can clearly see that the droplet is deformed and the curvature towards the low wettability area has decreased.
For the two dimensional case, the use of the adaptive mesh is very helpful. The number of nodes and distribution is adapted to the needed accuracy

in the given region. For this, the time step is performed and the error approximated if the error lies between max_error and min_error the node distribution is kept, otherwise it is adapted[11]. The values for the mesh errors are set in my_2Dproblem.cpp and can also be set when calling the constructor or the init_problem() function of the scan_basics_2D.py file. The number of refinements to be performed for each time step can be set with the variable nrefinements, which is also accessible outside the problem construction. The run2D.py file used for the simulation in fig. 6 looks like follows:

```python
import libprestructured_2Dtfe as TFElib
from parameter_scanning.scan_basics2D import *
from datetime import datetime

SCRIPTNAME = "run2D.py"
SCAN_BASICS_DIR = "parameter_scanning"

problem = init_problem(Nx=32, Ny = 32, Lx = 20, Ly = 20,
 speed = 0.02, rhoHW = -0.8, rhoLW = 0.5, nrefinements=1,
 max_error = 10**(-3), min_error=10**(-4),
 inhomogenity = "moving_logistic")
problem.set_elements_to_equilibrated()
problem.integrate_for_time(400)
timestamp = datetime.now().strftime("%Y%m%d-%H%M%S")
finalfolder = f"out_2D_testrun_{timestamp}"
tidy_up(problem,finalfolder,SCRIPTNAME,SCAN_BASICS_DIR)
# Plot output directly
os.system("./plotting_scripts/plot2D_fancy.py "+finalfolder)
```

[11]If the error is larger max_error the integration is repeated with higher accuracy. For an error smaller than min_error the integration is accepted and the mesh is adapted for the next step.

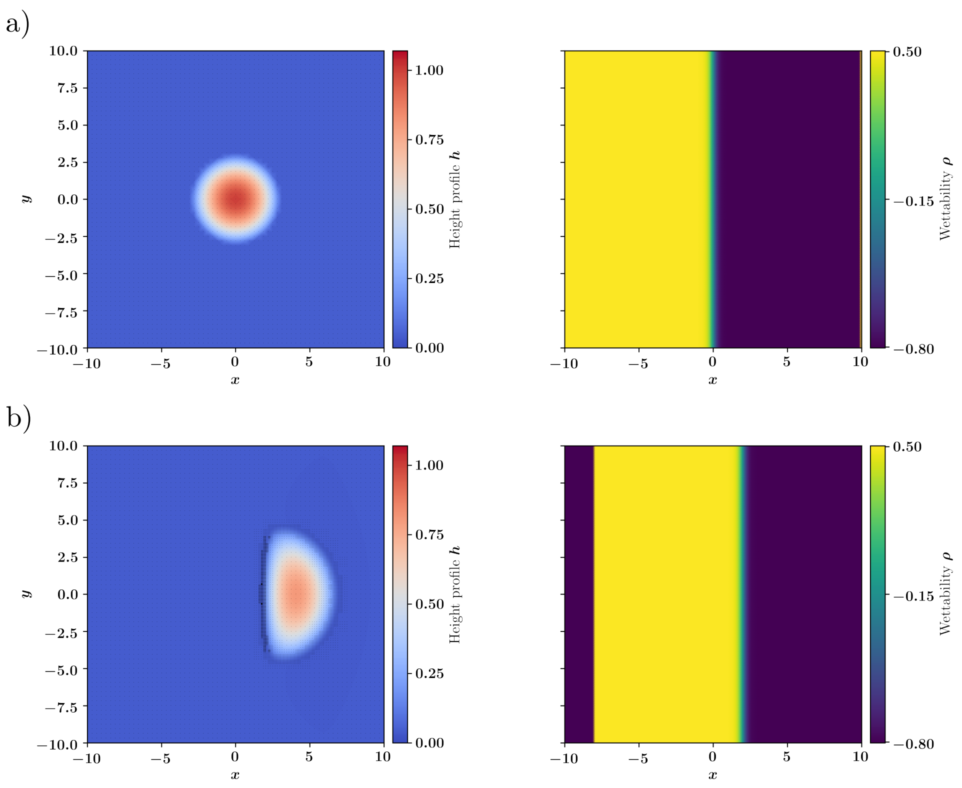


Figure 6: Snapshots of a direct time simulation of a two-dimensional drop on a substrate with a moving logistic function as the wettability. The parameters are $Nx = 32$, $Ny = 32$, $Lx = 20$, $Ly = 20$, $v_{\mathrm{inhom}} = 0.02$, $\rho_{\mathrm{HW}} = -0.8$, $\rho_{\mathrm{LW}} = 0.5$, `nrefinements`$= 1$, `max_error`$= 10^{-3}$, `min_error`$= 10^{-4}$. The nodes of the mesh are shown as black dots. The dark patch on the left side of the droplet in b) corresponds to a higher mesh resolution in this area.

Bibliography

[1] Daniel Bonn et al. “Wetting and spreading”. In: *Reviews of Modern Physics* 81 (2 2009), pp. 739–805. ISSN: 0034-6861. DOI: 10.1103/RevModPhys.81.739. URL: https://link.aps.org/doi/10.1103/RevModPhys.81.739.

[2] Manoj K. Chaudhury and George M. Whitesides. “How to Make Water Run Uphill”. In: *Science* 256.5063 (1992), pp. 1539–1541. ISSN: 0036-8075. DOI: 10.1126/science.256.5063.1539. URL: https://science.sciencemag.org/content/256/5063/1539.

[3] Sebastian Engelnkemper. “Nichtlineare Analyse physikochemisch getriebener Entnetzung - Statik und Dynamik”. PhD thesis. Westfälische Wilhelms-Universität Münster, 2017.

[4] Josua Grawitter and Holger Stark. “Steering droplets on substrates using moving steps in wettability”. In: *Soft Matter* 17.9 (2021), pp. 2454–2467. DOI: 10.1039/D0SM02082F.

[5] Matthias Heil and Andrew L. Hazel. “oomph-lib – An Object-Oriented Multi-Physics Finite-Element Library”. In: *Fluid-Structure Interaction.* Ed. by Hans-Joachim Bungartz and Michael Schäfer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 19–49. ISBN: 978-3-540-34596-1.

[6] Christoph Honisch et al. “Instabilities of Layers of Deposited Molecules on Chemically Stripe Patterned Substrates: Ridges versus Drops”. In: *Langmuir* 31.38 (Sept. 2015), pp. 10618–10631. ISSN: 0743-7463. DOI: 10.1021/acs.langmuir.5b02407. URL: https://ir.nctu.edu.tw/handle/11536/128271.

[7] Vladimir S. Mitlin. “Dewetting of Solid Surface: Analogy with Spinodal Decomposition”. In: *Journal of Colloid and Interface Science* 156.2 (1993), pp. 491–497. ISSN: 0021-9797. DOI: https://doi.org/10.1006/jcis.1993.1142. URL: https://www.sciencedirect.com/science/article/pii/S0021979783711422.

[8] Alexander Oron, Stephen H. Davis, and S. George Bankoff. “Long-scale evolution of thin liquid films”. In: *Reviews of Modern Physics* 69 (1997), pp. 931–980. ISSN: 0034-6861. DOI: 10.1103/RevModPhys.69.931.

[9] R. Shankar Subramanian, Nadjoua Moumen, and John B. Mclaughlin. “Motion of a Drop on a Solid Surface Due to a Wettability Gradient”. In: *Langmuir* 21.25 (2005), pp. 11844–11849. DOI: 10.1021/la051943i.

[10] Walter Tewes et al. “Comparing kinetic Monte Carlo and thin-film modeling of transversal instabilities of ridges on patterned substrates”. In: *The Journal of Chemical Physics* 146.9 (2017), p. 094704. ISSN: 0021-9606. DOI: `10.1063/1.4977739`. eprint: `https://doi.org/10.1063/1.4977739`.

[11] U Thiele. *Thin film evolution equations from (evaporating) dewetting liquid layers to epitaxial growth.* 2010. DOI: `10.1088/0953-8984/22/8/084019`.