EDEW Version 1.0
A simulation tool for fluid handling by electrowetting effects

D 3.4: Report for EU project “Micrometer scale patterning of Protein and DNA-Chips”

Jan Lienemann, Andreas Greiner, and Jan G. Korvink

IMTEK – Institute for Microsystem Technology, Albert Ludwig University
Georges Köhler Allee 103, D-79110 Freiburg, Germany
lieneman@imtek.de Contract No. EU G5RD-CT-2002-00744

January 30, 2004

Contents

Abstract

We present EDEW (**E**volve **d**roplets by **e**lectro**w**etting), an application for the simulation of microscale fluid handling by electrowetting effects. The application program works as a driver for the command line surface evolver program which is specialized for the simulation of surface and interface tension effects. The goal is to provide a set of script templates and a user-friendly graphical user interface for the intuitive use of the program.

1 Introduction

Electrowetting devices are microfluidic components which allow for the motion of droplets on a substrate with a given grid of electrodes. Since on a 2D grid, the motion of a droplet is free in the plane as long as it sticks to the grid lines, these devices are particularly suitable for the design of reconfigurable fluidic circuits with programmable fluid paths [10, 7]. Reconfigurability means that the path of motion and thus the functionality of the device can be changed at runtime while the device is in use, and not only at design time as in other fluidic approaches.

The electrowetting grid performs three main functions (see Fig. 1):

Transport: To move the droplet along a path to or from other functional components like detectors, catalytic converters, supplies and waste outlets.

Splitting: To generate droplets from a reservoir, or to split a droplet into parts for parallel processing.

Merging/Mixing: To merge droplets and therefore mixing their contents. This can either be achieved by diffusion or with help of periodic actuation. However, since turbulent flow is hard to achieve at these dimensions, diffusion will still be the dominant mixing mechanism.

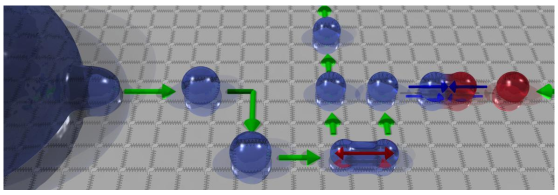


Figure 1: The three main functions of an electrowetting array: Splitting, moving, merging.

1.1 Electrowetting

The usual setup for electrowetting actuation is shown in Fig. 2. The system consists of a dielectric layer with electrodes attached to the bottom of the layer. It is essential that the dielectric layer is a good insulator with no pinholes, and that ions can not easily be trapped inside the layer; this would inhibit the correct function of the device.

The droplets are placed on top of the dielectric layer. They must be of conducting liquid (electrolyte); if non-conducting liquids are used, the

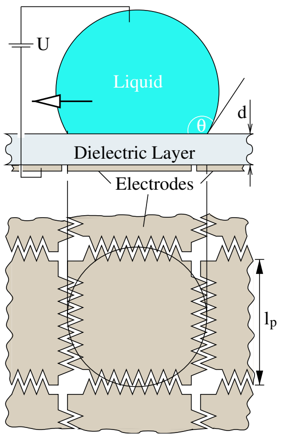


Figure 2: Setup for electrowetting. The top electrode is usually replaced by a glass lid with conducting ITO layer at the bottom.

droplet must be surrounded by a conducting liquid to achieve a reversed effect.

The droplet is contacted either with a wire as shown in Fig. 2, or is confined between the dielectric layer and a conducting top layer, e.g., a metal or ITO layer, forming a single large electrode.

By applying an electric voltage between the top and the bottom electrode, charge is accumulated at the droplet bottom as in a capacitor; since the capacity and thus the stored energy is proportional to the droplet's area of support, the droplet seeks to maximize its contact area. This results in a change of the contact angle (see Fig. 3). For a more rigid derivation see section 2.

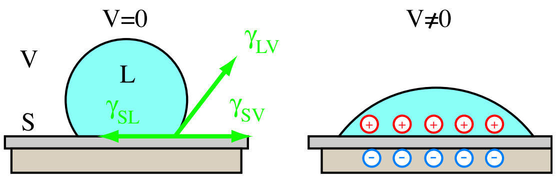


Figure 3: A droplet changing its contact angle by electrowetting. The arrows denote the surface/interfacial tensions.

Applying the voltage not to the electrode on which the droplet is sitting on, but to the adjacent electrode, the droplet seeks to increase its area of support on that electrode and therefore a motion to the next electrode takes place. Subsequent application of the algorithm to the next pair allows to transport the droplet over the complete area. By moving two droplets to the same spot, mixing can be achieved. Splitting requires more complicated actuation schemes, which can benefit from proper design tools to optimize the algorithms.

For the design of electrowetting devices, another key question is the shape of the droplet and its area of support on the substrate. These determine refracting properties for optical detections, the possibility of two droplets to touch and thus to merge, whether the transport works at all – i.e., whether the area of support is large enough such that the droplet can be actuated by the adjacent electrode – and how the splitting can be achieved.

1.2 Simulation of electrowetting

The goal of the tool presented in this article is to simulate the droplet deformation due to electrowetting, to judge if a certain shape can be achieved. It is not meant as a full computational fluid dynamics program; the simulation is quasistatic. However, since we can assume a damped system, the qualitative results match the real deformation, and after calibration of the time step due to the simple flow field in the droplet a good estimate of the behavior is possible.

For the simulation, the tool interfaces to the *Surface Evolver* program [3], which is a well known tool for the simulation of interface shapes.

The *Surface Evolver* is a command line interactive program for the study of surface shapes arising from surface tension effects and other energies. It is used in a wide range of research topics, including grain growth, foam modeling, packaging technologies, biology and geometry. The program "evolves" the surface, which is represented by a union of triangles, to an energy minimum by doing a gradient descent minimization. Different minimization schemes like steepest descent and conjugate gradients are available just as are second order schemes using the Hessian matrix. Its powerful scripting language makes it possible to use it as a library for custom applications.

The program is freely available and provided with extensive documentation [4, 5]. We recommend also reading the *Surface Evolver* manual to fully understand the templates provided with this program and be able to adapt them to future needs.

The surface is represented by vertices, edges, facets, and bodies. Bodies are defined by their bounding facets; the facets are flat triangles bounded by three edges (in contrast to faces, which may be bounded by more edges); edges are straight lines joining two vertices. Vertices represent points in the Euclidean $\mathbb{R}^3$ space.

The basic operation for the evolution of the surface is the iteration step, which moves the vertices along the energy gradient. The actual displacement is the product of the energy slope of the respective degree of freedom (DOF) and a global *scale factor*, which can be specified by the user or optimized by the *Surface Evolver*. An additional quantity correcting motion enforces global quantity constraints such as volume or attachment of the surface to walls.

For a facet with edges s_0 and s_1, the facet energy due to surface tension γ can be calculated by

$$E = \frac{\gamma}{2} \|s_0 \times s_1\| . \tag{1}$$

It is straightforward to show that the gradient $g_i = \partial E / \partial x_i$ of the first edge s_0 is then

$$g_{s_0} = \frac{\gamma}{2} \frac{s_1 \times (s_0 \times s_1)}{\|s_0 \times s_1\|} . \tag{2}$$

Summing up all gradient parts of the faces adjacent to a node yields the total free energy gradient for the vector motion.

1.3 Simulation tool

For a specific simulation problem, it is necessary to write a script file to specify the model along with constraints and surface energies. This requires some experience with the tool and time to get used to the programming language. We therefore provide a tool to simplify this process: A script template is provided, and the user only needs to specify model parameters in a user friendly graphical user interface (GUI). It is possible to execute the script directly from the program and to operate the simulation without having to learn the input language of the *Surface Evolver*. However, for experienced users direct interacting still remains possible.

The simulation is written in *Java* [1], a programming language created by *Sun Microsystems, Inc*. The great advantage of *Java* is its platform independence, therefore the program runs on a large variety of platforms without the need to recompile it or use platform dependent toolkits.

2 Background

The electrowetting effect has been known for a long time [9]. For a thin dielectric layer, an analytic model can be found by considering the change in free energy following a displacement of the contact line (see Fig.4) [11]. The free energy F in differential form can be expressed as

$$\mathrm{d}F = \gamma_{SL}\mathrm{d}A - \gamma_{SV}\mathrm{d}A + \gamma_{LV}\mathrm{d}A\cos\theta + \mathrm{d}U - \mathrm{d}W_B \tag{3}$$

where U is the energy stored in the electric field in the dielectric layer between droplet and electrode, γ_{SL}, γ_{SV} and γ_{LV} are the surface/interfacial tensions of the system at $V = 0$ (see Fig. 3), and W_B is the work the voltage must perform to built up the potential between droplet and electrode.

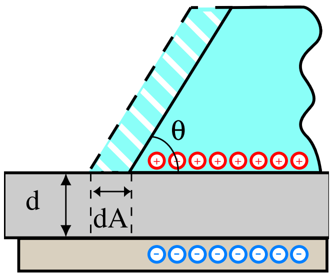


Figure 4: Schematic picture of the virtual displacement of the contact line.

Equilibrium and thus an energy minimum is reached when $\mathrm{d}F/\mathrm{d}A = 0$. For $V = 0$ (and thus $\mathrm{d}U = \mathrm{d}W_B = 0$), this leads to the Young equation

$$\cos\theta = \frac{\gamma_{SV} - \gamma_{SL}}{\gamma_{LV}}. \tag{4}$$

The energy stored in a capacitor with large area A, small plate distance d and relative dielectric constant ε_r of the material inbetween for a voltage V is given as

$$U = \frac{1}{2}CV^2 = \frac{1}{2}\frac{\varepsilon_r\varepsilon_0 A}{d}V^2, \tag{5}$$

where ε_0 is the dielectric constant of vacuum.

Now we assume that the droplet changes its area by $\mathrm{d}A$ because of moving the contact line. Then the energy of the electric field changes by

$$\frac{\mathrm{d}U}{\mathrm{d}A} = \frac{1}{2}\frac{\varepsilon_r\varepsilon_0}{d}V^2. \tag{6}$$

The additional energy is fed into the system by the voltage source, so that [11]

$$\frac{\mathrm{d}W_B}{\mathrm{d}A} = \frac{\varepsilon_r\varepsilon_0}{d}V^2. \tag{7}$$

$\mathrm{d}U/\mathrm{d}A$ and $\mathrm{d}W_B/\mathrm{d}A$ can be combined to an electrowetting term $\gamma_{EW} = \mathrm{d}W_B/\mathrm{d}A - \mathrm{d}U/\mathrm{d}A$, whereupon (3) reads

$$\frac{\mathrm{d}F}{\mathrm{d}A} = \gamma_{SL}\mathrm{d}A - \gamma_{SV}\mathrm{d}A + \gamma_{LV}\mathrm{d}A\cos\theta - \gamma_{EW} \tag{8}$$

with

$$\gamma_{EW} = \frac{1}{2}\frac{\varepsilon_r\varepsilon_0}{d}V^2. \tag{9}$$

The Young equation (4) then becomes

$$\cos\theta = \frac{\gamma_{SV} - \gamma_{SL} + \frac{1}{2}\frac{\varepsilon_r\varepsilon_0}{d}V^2}{\gamma_{LV}}. \tag{10}$$

We implement this model by modifying the interfacial tension of the droplet to the substrate, i.e., $\gamma_{SL} = \gamma_{SL,\,V=0} - \frac{1}{2}\frac{\varepsilon_r\varepsilon_0}{d}V^2$ on the parts of the contact area where it overlaps with the respective electrode.

Details of the implementation for electrowetting were presented in [8]. In principle, the surface energy on the substrate and on the top plate (for the confined droplet) is calculated by line integrals along the contact line. To convert the surface integral

$$E_{SL} = \int_S \gamma_{SL}\mathrm{d}A \tag{11}$$

into a line integral, we use the divergence theorem

$$\int_S \nabla\cdot\vec{v}\mathrm{d}A = \int_{\partial S} \vec{v}\cdot\mathrm{d}\vec{l} \tag{12}$$

with a $\vec{v}$ such that $\nabla\cdot\vec{v} = \gamma_{SL}$.

3 Installation

The simulation tool is designed to work on a variety of platforms. Main testing was done under Mac OS X, but it should work under every operating system where Java 1.4 or above is available. Please report to the authors if the application should present any problems in your environment.

The use of the concept UNIX in the following instructions refers to all UNIX-like operating systems such as Mac OS X, GNU Linux, Solaris, etc.

Since some of the functionality of this tool depends on the *Surface Evolver* program, we recommend to first install it (see [3]). The program also works without the *Surface Evolver*, e.g., if you want to run the scripts on a different computer, but then its main advantages such as a GUI for the *Surface Evolver* will not be available. We recommend installing it to `/usr/local/bin/evolver` under UNIX and to `C:\Program Files\Evolver\evolver.exe` under Microsoft Windows, since these are defaults. However, the paths can be adapted in the tool, so other locations are possible. For the proper operation of the *Surface Evolver*, you may need to install other programs, like an X-Server for the visualization of the simulation results. Please refer to the *Surface Evolver* documentation.

Further, you will need the *java runtime environment* with at least version 1.4 to run the application (JRE 1.4 or later). The JRE can be downloaded from [1] if it is not already installed on your computer.

To install the tool, please download the zip file from [2] and extract it to a temporary directory. Create a directory where you want the library files to reside (Recommendation: `/usr/local/lib/microprotein simulation tool/` under UNIX and `C:\Program Files\Microprotein Simulation Tool\` under MS Windows) and copy everything in the `library` directory of the zipfile to the respective location.

The program is started from the command line with

```
java -jar /path/to/jar/file/
Simtool.jar
```

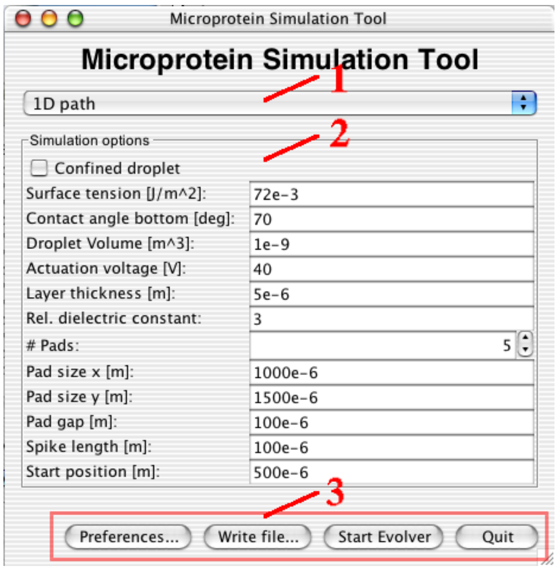
under UNIX and

```
java -jar C:\path\to\jar\file\
Simtool.jar
```

under Microsoft Windows (all characters on one line). The jar file is in the library directory and will also be copied to the library installation part.

4 User guide

On startup, the window shown in Fig. 5 is displayed. Please note that the look and feel of the GUI widgets depends on your platform; however, the functionality is the same on all platforms.



Microprotein Simulation Tool

1D path

Simulation options

☐ Confined droplet

Surface tension [J/m^2]:	72e-3
Contact angle bottom [deg]:	70
Droplet Volume [m^3]:	1e-9
Actuation voltage [V]:	40
Layer thickness [m]:	5e-6
Rel. dielectric constant:	3
# Pads:	5
Pad size x [m]:	1000e-6
Pad size y [m]:	1500e-6
Pad gap [m]:	100e-6
Spike length [m]:	100e-6
Start position [m]:	500e-6

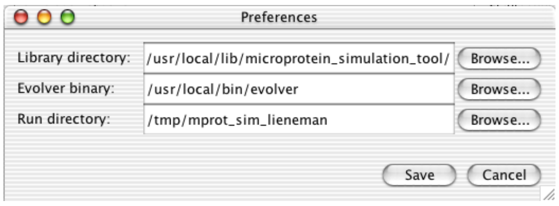
Figure 5: The main window of the Microprotein simulation tool.

The window features three main parts: The simulation chooser (1), the parameters for the chosen simulation (2) and the action buttons (3).

The simulation chooser lets you select the model template. As templates are added, more choices will become available. According to the chosen model, the input fields for the simulation parameters change. The description of these paramters is given in the model library (section 5).

The window presents four action buttons: The user preferences, where settings for paths and the surface evolver can be changed; a button to create a surface evolver file from the current configuration; a button to create the file and start the *Surface Evolver* with a temporary file; and a button to quit the application.

4.1 Preferences



Library directory:	/usr/local/lib/microprotein_simulation_tool/
Evolver binary:	/usr/local/bin/evolver
Run directory:	/tmp/mprot_sim_lieneman

Figure 6: The preferences window.

After clicking on the Preferences button, a window is displayed with fields for the most important program parameters (Fig. 6). If you have installed the files as recommended in section 3, no change to the default values is necessary; if not, you can browse your filesystem with the respective buttons.

The *library directory* is where you have the contents of the library directory installed to. The *evolver binary* is the program to execute to start the surface evolver; you can also add command line parameters or use own scripts. The *run path* is a temporary directory where the tool can place files to interface with the surface evolver. The tool makes sure that the filename of the temporary file it creates is new and unique, so this directory can host many subsequent simulations; however, the filenames need not necessarily be ordered. All directories are created if they do not exist; please make sure that you have write permissions on the last already existing directory in the path.

The temporary files are not removed so that they can be inspected and rerun later. The tool also creates a log file with all user input so that all actions can be repeated later.

The *Save* button closes the window, saves the values to the proper place on your computer (e.g., the registry on Microsoft Windows) and applies the new values to further actions. The *Cancel* button closes the window without saving and applying the changed values, which is also the case if you close the window by the respective icon.

4.2 Write file

The *Write file...* button opens up a file browser to specify the filename that the *Surface Evolver* script will be saved to. The file can be loaded with the surface evolver. Predefined actions, e.g., for initialization, are available and described in section 5.

4.3 Start evolver

The *Start evolver* button opens up three new windows and starts the *Surface Evolver* with a temporary script file. The most important window is the *Surface Evolver* control window, shown on the lower left of Fig. 7. In the top area, the input to and output from the *Surface Evolver* is presented. Below, the user can input commands to the running process. However, this is usually not necessary since the most important parameters can be controlled from the lower part of the control window. Please note that changing settings from the command line may lead to inconsistencies between the internal state of the *Surface Evolver* and the settings in the GUI (e.g., for the auto redraw feature, which can be toggled by the "D" command).

The commands are written to a log-file (temporary filename + ".log"), and all input and output from the "Evolver output" area is written to another file (temporary filename + ".out"). The name of the temporary file is shown in the first line.

An error message "Broken pipe" is usually a sign that the *Surface Evolver* process has died. This should not happen under normal circumstances; the reason can be a bug in the *Surface Evolver* or exhaustion of system resources due to other programs.

Below the command line, some buttons and checkboxes for controlling the output are available: "Show" to switch on and off the graphics window (right side of Fig. 7), "Quietgo" to suppress the energy and area output of each iteration, "Auto Redraw" to repaint the graphics after each iteration (very time consuming, recommended for debugging only), and "Redraw" to force a redraw. The button "Clear" clears the output window; its content is still preserved in the .log and .out-files.

All of these buttons do not modify the simulation itself; only the output behavior is changed.

The "Refine" button next to the "Redraw" button allows to refine the mesh; this modifies the simulation, but not the operating parameters.

Below, controls for changing the model are provided. These controls differ for each model and are therefore explained in section 5. With these buttons, the physics of the problem can be influenced. For example, voltage can be applied to pads, iteration steps can be performed, and the simulation can be initialized with the "Init" button.

If the "Show" checkbox is activated, a graphics window and a corresponding control window appears (right and upper left of Fig. 7). The meaning of the symbols in the control window and the corresponding *Surface Evolver* key are explained in Table 1.

Button	S.E.	Action
`CW`	c	Rotate clockwise by 6°
`(^)`	u	Rotate up by 6°
`CCW`	C	Rotate counterclockwise by 6°
`(<)`	l	Rotate left by 6°
`.`	s	Shrink by factor of 1.2
`O`	z	Zoom by factor of 1.2
`(>)`	r	Rotate right by 6°
`#`	R	Reset viewing parameters
`(v)`	d	Rotate down by 6°
`>*<`	m	Center image
`<|`	←	Translate image to left
`|>`	→	Translate image to right
`_^_`	↑	Translate image up
`-v-`	↓	Translate image down

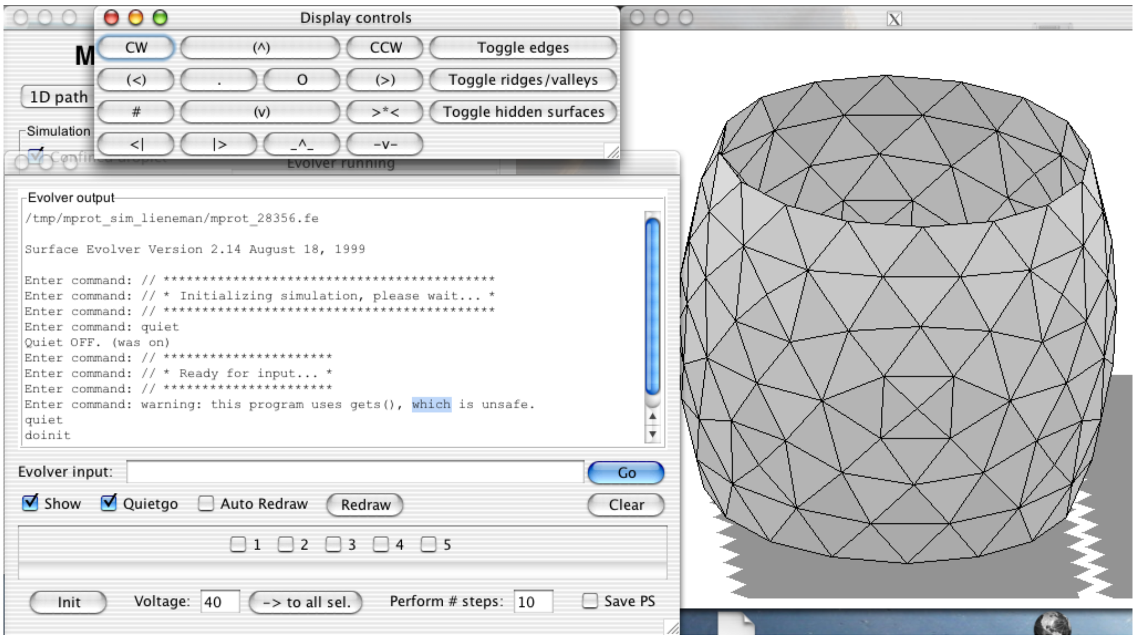
Table 1: Symbols in the Display controls window, corresponding *Surface Evolver* keys and action.

If you use the command line input, please note that for all buttons to work the *Surface Evolver* must show the main command prompt, i.e., it is not in the graphics mode ("s") or does not expect input for a specific command (e.g., "t").

The remaining buttons toggle the display of the element edges as black lines, specify a different coloring for convex and concave edges, and create a wireframe mode where the facet shading is switched off.

5 Model library

EDEW and its available models are under constant improvement; more models and features will be added to EDEW in the near future. We therefore recommend visiting the website for up-to-date information and the latest model library description.



```
/tmp/mprot_sim_lieneman/mprot_28356.fe

Surface Evolver Version 2.14 August 18, 1999

Enter command: // ******************************************
Enter command: // * Initializing simulation, please wait... *
Enter command: // ******************************************
Enter command: quiet
Quiet OFF. (was on)
Enter command: // *********************
Enter command: // * Ready for input... *
Enter command: // *********************
Enter command: warning: this program uses gets(), which is unsafe.
quiet
doinit
```

Evolver input:

☑ Show ☑ Quietgo ☐ Auto Redraw

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Voltage: 40 Perform # steps: 10 ☐ Save PS

Figure 7: The *Surface Evolver* control window, graphics controls and the graphics window.

5.1 Row of equidistant pads (1DPath)

This model simulates a row of electrodes with a single droplet on them. The configurable parameters that can be specified are listed in Table 2 (values with "*" only if "Confined" is checked) with dimensions as in Figure 8. Dimensions not shown are the distance of the bottom dielectric layer to the top layer for the confined droplet, the thickness of the dielectric layer and the droplet volume.

Parameter	Default	Confined?
Confined droplet	no	
Surface tension	$72\,\mathrm{J/m^2}$	
Contact angle top	$110\,^\circ$	*
Distance to top	1 mm	*
Contact angle bottom	$110\,^\circ$	
Droplet volume	1 nl	
Actuation voltage	40 V	
Layer thickness	1 µm	
Rel. dielectric constant	3	
Number (#) of pads	5	
Pad size x	1 mm	
Pad size y	1.5 mm	
Pad gap	0.1 mm	
Spike length	0.1 mm	
Start position x	0.5 mm	

Table 2: Parameters for 1DPath model. Parameters with * only if "Confined" is checked.

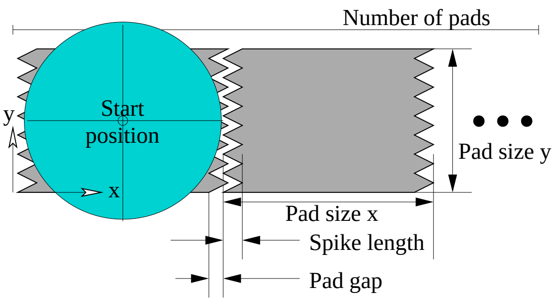


Figure 8: Dimensions for the 1DPath model.

The influence of the pad spikes is averaged along the x direction [8]. The interfacial energy has a linear decay along the spikes, until on the bulk of the pad it is constant until rising again when the spikes on the other side start.

The GUI for the *Surface Evolver* control window is shown in Fig. 7. The first row consists of checkboxes to apply voltages to the individual pads. The voltage applied depends on the current voltage settings, which can be specified at the next row in the "Voltage" field. When changing the voltage, it is important to press Enter on your keyboard once so that the change is propagated to the *Surface Evolver* process. When switching on a pad, the voltage is applied only to this pad; the others remain at their current voltage. To change all selected pads, press the button "-> to all sel.".

Before the simulation can be started, the simulation must first be initialized by pressing the “Init” button. In this step the mesh is refined and the droplet is evolved to an equilibrium state. Then, pads can be activated. To simulate, click into the “Perform # steps” field, enter a number of simulation steps to perform, and press Enter. One simulation step consists of several *Surface Evolver* iterations and mesh averaging and equiangulation steps which keep the dimensions of the mesh elements at about the same scale and prohibit large aspect ratios.

After every step, the step number, centroid x position of the droplet and the system energy are written to a datafile (temp. filename + “_t-centrx-E.dat”). The (empty) timestamp file (temp. filename + “_starttime.dat”) helps to determine the start of the simulation. The data files can be postprocessed to recover the energy gradient and therefore the force acting on the droplet. It is also possible to save an image after every step by checking the “Save PS” checkbox (temp. filename + “_time” + step number + “.ps”).

The centroid position is calculated by evaluating the integral

$$x_c = \int_V x \mathrm{d}v \bigg/ V \qquad (13)$$

over the complete volume V. This integral is implemented as surface integral by means of the divergence theorem. This poses another difficulty for the confined droplet because on the top and bottom contact areas no facets are present, and the automatic integration of the *Surface Evolver* will miss the contribution of this part. However, a good choice of the vector $\vec{v}$ in the divergence theorem (12) such that $\vec{v}$ is parallel to the contact area solves this problem.

5.2 Rectangular Channel (R4Channel)

This model simulates a liquid meniscus in a rectangular channel. For all four walls, different properties (thickness, dielectric constant, actuation voltage) can be given (See Table 3). The actuation voltage can be changed during the simulation. Further, the button “Equil.” evolves the surface until energy convergence is reached (change for 50 iterations smaller than 0.01% of the current energy value). The meaning of the remaining buttons is the same as explained above.

The simulation assumes that a counter electrode inside the liquid is placed somewhere else, so that all four walls have an isolating dielectric layer. A volume constraint takes care that the meniscus is always visible in the graphics window. One wall has a doubled line; this is the output, while the liquid enters from the opposite direction.

Parameter	Default
Surface tension	72 J/m^2
Channel width	100 µm
Channel height	100 µm
Contact angle (t, b, l, r)	110 °
Layer thickness (t, b, l, r)	1 µm
Actuation voltage (t, b, l, r)	40 V
Rel. dielectric constant (t, b, l, r)	3

Table 3: Parameters for the R4Channel model. (t, b, l, r): Parameters can be changed individually for each wall.

6 Extensions

It is not difficult to extend the program by adding new models. The base class for a model is “Simulation”. The class is abstract, and for a valid simulation the following methods must be overwritten:

public String toString():
This method simply returns a short string which is used in the simulation chooser of the main window to represent this model.

public void writefile(File outfile) throws IOException:
This method gets a File to write to. It is responsible for creating a Writer to send the data to. The library path for templates can be retrieved by the **static String Prefwin. getLibraryDir(Preferences prefs)** method. The method **protected void appendFileToWriter (String from, BufferedWriter to) throws IOException** can be used to copy template files. The method will usually take parameters directly from its own GUI objects. Handling of IOExceptions can be left to the simulation tool, however the outfile Writer needs to be closed by the method.

MyClass(Preferences myprefs, final JFrame outerframe):
The constructor for the class. MyClass must be replaced by the name of the new class. The objective of the contructor is to initialize the GUI part, which will appear as “Simulation options” in the main window. By calling **super(outerframe)**, the public field **GUI** which holds a JPanel to hold all parameter text fields and checkboxes is initialized with an empty

JPanel. The parameter **outerframe** is the main window to use e.g., for message boxes (the default constructor sets the field **outerfr** of the simulation class to this parameter, so that ActionListeners can use it; the field **prefs** is set to **myprefs**).

public JComponent getEvolverGUI(Evolverwin ev):
This method returns a JComponent (e.g., JPanel) to include in the lower area of the Evolver control window **ev**. The controls can use the method **public void Evolverwin. writeToEvolver(String s) throws IOException** to give input to the surface evolver, bust must handle possible IOExceptions (e.g., by showing a message).

The Sim1DPath class can be used as a template. After writing the class, it must be integrated into the main window by adding the line

```
simulations.add(new MyClass(pref,
outerframe));
```

below the line

```
/******* Simulation type *******/
```

in Mainwin.java.

7 Sample results

7.1 Moving droplet

This simulates moving a droplet by electrowetting. The simulation parameters are the default values as in Table 2. Figure 9 shows the results: a) After initialization, but before presseing the “Init” button; b) after pressing the “Init” button, the droplet shape is visible; c) After about 40 steps, the droplet has moved to the next position; d) Finally, the voltage is switched off and the droplet relaxes to its spherical shape.

7.2 Splitting a droplet

We repeat the experiment in [6] using the values in Table 4. The command `hessian` can be used near the end to speed up convergence.

The resulting droplet shape just prior to the splitting is shown in Fig. 10. The actual splitting of the droplet is not implemented; however, it can easily be determined from the graphical output.

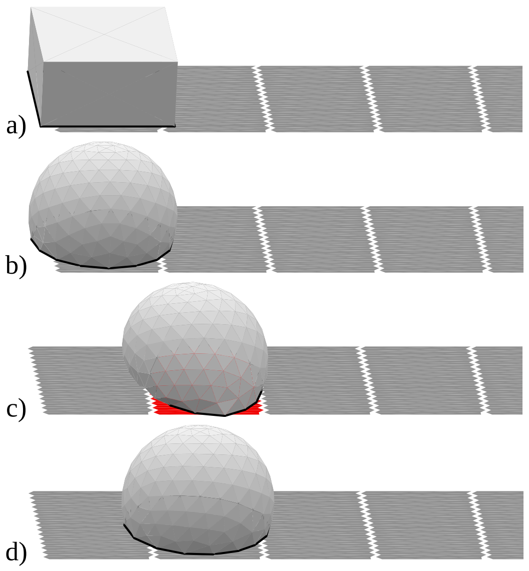


Figure 9: Simulation results for moving droplet: a) before initialization, b) after initialization, c) moved to second pad, d) relaxating after resetting voltage.

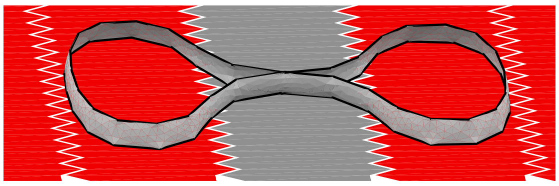


Figure 10: Splitting a droplet by electrowettting.

7.3 Energy curve for controlled motion

This example demonstrates the use of the command line. The simulation parameters are the same as the the default, except for the switch for the confined droplet which was turned on, and the number of pads, which was set to 2. After running the initialization, we activate the second pad. We then fix the centroid of the droplet (`fix centrx`) and redefine the simulation step to (all code entered on one line):

```
dostep:={
 printf "** \%5.0f **\n",tt;
 {V;u;V}4;
 centrx.target+=20e-6;
 equil;
 write_data; redraw;
 saveps;
 tt+=1;}
```

where `equil` is a predefined macro which iterates until convergence of the total energy.

The result is a energy curve similar to the one presented in [8]. By taking the derivative w.r.t. *x* of this

Parameter	Value
Confined droplet	Yes
Surface tension	72e-3
Contact angle top	120
Distance to top	80e-6
Contact angle bottom	120
Droplet volume	((5e-4)ˆ2*π*80e-6)
Actuation voltage	25
Layer thickness	1e-7
Rel. dielectr. const	2
# Pads	7
Pad size x	600e-6
Pad size y	1500e-6
Pad gap	100e-6
Spike length	160e-6
Start position	1890e-6

Table 4: Parameters for droplet splitting experiment.

curve, the force F on the droplet for each position can be estimated, $F = -\partial E/\partial x$.

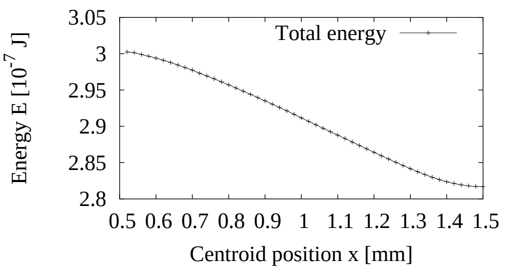
8 Summary and Conclusions

EDEW is a simulation tool for the Microprotein project that allows for a large variety of quasistatic fluidic simulations of microdrops moved by electrowetting. It interfaces to the well known and powerful code of the *Surface Evolver*, but without the need to understand the command language. By providing templates for programs in the *Surface Evolver*, specific problems can be simulated using a user friendly Java based graphical user interface without the need, but with the possibility, to use the command line. The tool can easily be extended with more model classes. It is platform independent and thus allows for simulation on large variety of computers.

The simulation results provide insight to the energy configuration of liquid droplets on electrodes and help judge whether and under which circumstances specific operations are possible, and how to optimize the geometrical and operating parameters of a given device. By postprocessing the output, it is also possible to recover capillary forces.

Acknowledgments

The authors acknowledge the work done by Kenneth A. Brakke and other contributors to the *Surface Evolver* program, and that he made the program available for free together with source code and extensive documentation.

a)


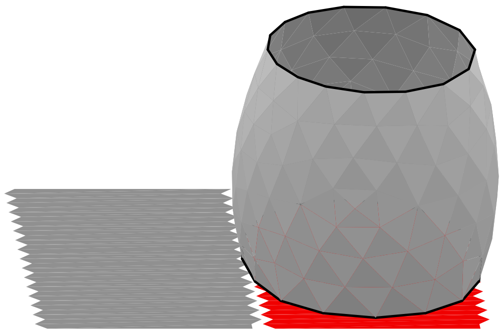
b)


Figure 11: a) Energy curve for moving droplet to the right; b) droplet after 50 iterations of controlled motion to the right side.

References

[1] http://java.sun.com.

[2] http://www.imtek.de/simulation/microprotein.

[3] http://www.susqu.edu/facstaff/b/brakke/evolver.

[4] Kenneth A. Brakke. The surface evolver. *Experimental Mathematics*, 1(2):141–165, 1992.

[5] Kenneth A. Brakke. *Surface Evolver Manual, Version 2.20.* Susquehanna University, Selinsgrove, PA 17870, August 2003.

[6] Sung Kwon Cho, Hyejin Moon, Jesse Fowler, and Chang-Jin Kim. Splitting a liquid droplet for electrowetting-based microfluidics. In *Proceedings of the ASME International Mechanical Engineering Congress and Exposition*, number IMECE2001/MEMS-23831, New York, NY, November 11–16 2001. ASME.

[7] Jie Ding, Krishnendu Chakrabarty, and Richard B. Fair. Reconfigurable microfluidic system architecture based on two-dimensional electrowetting arrays. In *Proc. MSM 2001*,

pages 181–185, Hilton Head Island, South Carolina, USA, March 19–21, 2001.

[8] Jan Lienemann, Andreas Greiner, and Jan G. Korvink. Electrode shapes for electrowetting arrays. In *Proc. Nanotech 2003*, volume 1, pages 94–97, Cambridge, USA, February 2003. NSTI.

[9] M. G. Lippmann. Relations entre les phenomenes electriques et capillaires. *Ann. Chim. Phys.*, 5(11):494–549, 1875.

[10] Michael G. Pollack, Richard B. Fair, and Alexander D. Shenderov. Electrowetting-based actuation of liquid droplets for microfluidic applications. *Applied Physics Letters*, 77(11):1725–1726, September 11, 2000.

[11] H. J. J. Verheijen and M. W. J. Prins. Reversible electrowetting and trapping of charge: model and experiments. *Langmuir*, 15(20):6616–6620, 1999.