SYNTHESIS OF MICROFLUIDIC CHIPS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Vladimir Kibardin
December 2013

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Mark Horowitz, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Stephen Boyd

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

James Weaver

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

Abstract

Microfluidic chips are devices that allow one to manipulate nano- and pico-liters of fluids and microscopic objects, such as cells and complex molecules, and are widely employed in bioengineering, molecular biology, chemistry and adjacent fields. We focus our work on soft lithography technology used to create microfluidic chips in a number of bio labs. The growing complexity of these chips increases the probability and the cost of a design mistake, thus making them harder to design and debug.

In this thesis, we address these challenges by introducing appropriate design tools. First, we will present a robust compact modelling framework that enables rapid simulations of microfluidic chips via existing circuit simulators, such as SPICE, and validate these models on two simple devices, a long channel and a pump. Then, we frame our model as a geometric program, enabling us to perform fast and global optimizations of complete chip designs. Finally, we introduce a synthesis tool that conceives a design of a chip using parameterized building blocks and optimizes its performance.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Miniaturization is a common engineering trend holding for at least a century across many fields. Logic circuits, for example, have gone a long way from large vacuum tube to few atoms wide transistor structures. This drastic change in scale has enabled today's ubiquity of computers; made of vacuum tubes, contemporary processors could easily occupy several houses, consume megawatts of power and still be futilely slow.

Microfluidics is a result of a similar development in areas dealing with fluid. The ability to handle small amounts of liquid substances is a foundation for a lot of today's high-tech devices. Inkjet printers, for instance, employ microfluidics to spray small droplets of ink on the paper. In chemical or biological labs, different tools, like vials or mixers, can be integrated on a single microfluidic chip, increasing the efficiency and capabilities of the researchers.

This work focuses on one type of microfluidic devices, a valve-based soft-litho fluidics, invented by S Quake [1] that is used in a number of bio labs. By incorporating valves with routing channels, the technology has grown from simple single-channel devices to sophisticated high throughput lab-on-a-chip systems. As the complexity of these devices grows, the probability of making a mistake grows, so does the cost of mistakes. In integrated circuits industry, this growing complexity led to creation of circuit simulators which used compact models, and then logic design and optimization tools. Since soft-litho microfluidics are following a similar path, this thesis creates a similar set of tools for microfluidic chips, starting with compact models and moving on to design with higher level objects and optimization tools.

To better understand the modelling and synthesis work, the next chapter reviews soft-litho microfluidic manufacturing and the structures that are built in this process. The properties of these structures can be simulated using finite element methods, and these are reviewed next. Building compact models for these structures is not a new idea, so the chapter ends by reviewing some of the compact models that have been used to model microfluidic circuits, focusing on those that are applicable to soft-litho circuits.

With this background, Chapter 3 then presents our derivation of simple, non-linear compact models for channels and valves; the two basic building blocks in a soft-litho system. Interestingly, one can use the same potential based approach as is used to derive the I-V curve for a MOS device to get the pressure-flow relationship for a channel. These derived models are then validated against FEM simulations, and actual devices.

Our potential based formulation of the compact model has some additional advantages that are explored in Chapter 4. Here will look at incorporating optimization, so we can reduce the work a user must do to complete a design by computing many values in the resulting design to optimize its overall performance. This chapter shows that using our compact model many of the optimization problems can be stated as a geometric program, which is a form that can be solved efficiently. We use this technique to create a very simple synthesis system for soft-litho chips.

Chapter 2

Background

While there's no clear definition, typically, a device is referred to as *microfluidic* if it operates on small volumes of fluids (nano- or pico- liters), and has features of small size (e.g. channels that are 100μ wide). Such devices exploit physical effects enabled on such a scale (e.g. capillary effects) to manipulate (transport, isolate, sort, mix) small portions of reagents and samples. With such a diverse set of functions, microfluidics has been used in a variety of applications across many fields.

Adoption of microfluidics by researchers in microbiology and related fields allowed them to integrate many existing lab tools (such as vials, mixers, reactors) on a single device, as well as create new ones. Our work focuses on soft lithography[1], one of the microfluidic technologies, was used in a diverse set of biological experiments, including single cell analysis, culturing, DNA and protein molecules manipulations and so forth [2, 3, 4, 5, 6, 7].

2.1 Design and fabrication of soft-litho devices

The fabrication process begins with the creation of a set of masks that are printed on a transparent film. The feature size of the mask is about 10μ. The masks contain the desired geometry of the layer. The film is then used to mask the exposure of the photoresist layer ($10\mu - 100\mu$ thick) to the UV light (see Figure 2.1). The part

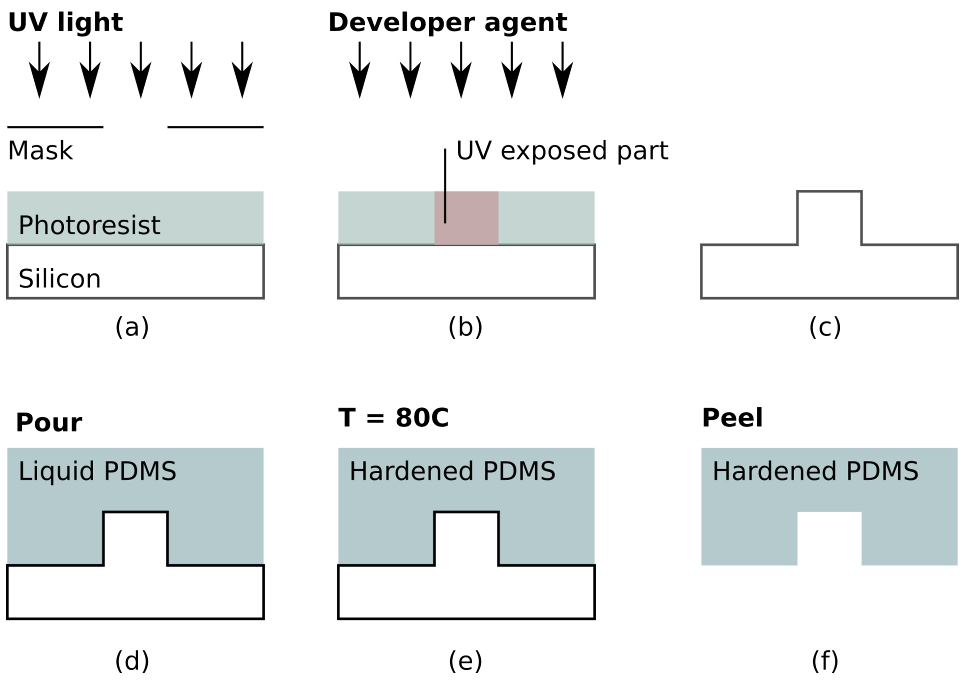


Figure 2.1: Fabrication of a single layer of a chip using photo-lithography. (a) A silicon wafer is covered with photoresist, exposed to UV through a mask. (b) The exposed part becomes insoluble; the rest is washed off by a developing agent. (c) The relief that remains after developing. (d) Liquid PDMS is poured on the mould. (e) It is baked at an elevated temperature to harden. (f) The PDMS film is peeled off the waver and becomes a layer for the chip.

exposed to the UV becomes insoluble to the developing agent (e.g. some acid), while the rest is washed off. The resulting mould contains an inverse relief of the layer (e.g. it has a concave part where a channel should be). Then, a soft polymer (e.g. PDMS) in a liquid form is poured onto the mould and hardened in an oven. The process is finalized by peeling the PDMS film off the mould. The layer's thickness varies from 100μ to $0.5cm$.

Figure 2.1(f) depicts the resulting film with a cavity that will become a channel when

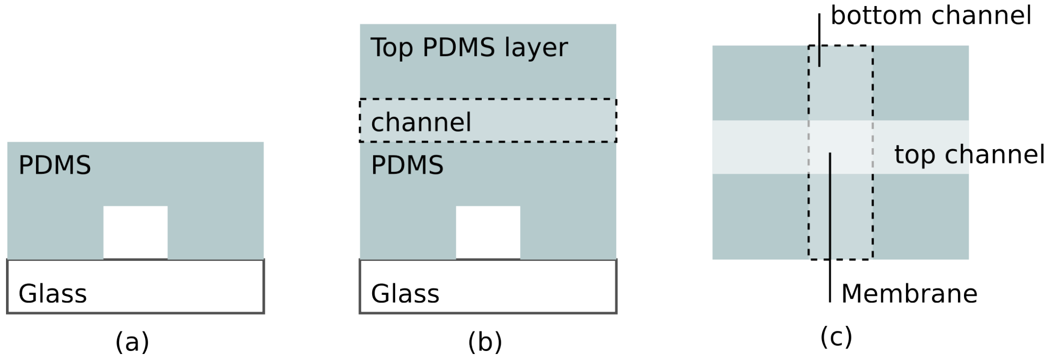


Figure 2.2: Fabrication of a valve. (a) The PDMS film is bonded to the glass substrate. (b) The second layer of PDMS that contains an orthogonal channel is bonded to the top of the first PDMS layer. (c) The top-down view of the construction; two orthogonal channels, one on top of the other.

bonded with a glass substrate (see Figure 2.2). The described layer fabrication steps can be repeated to build another PDMS layer that we can place on top of the PDMS-glass construction. In Figure 2.2(b,c), a device called a push-down valve is formed by merging 2 PDMS layers that contain intersecting orthogonal channels.

The push-down valve plays the major part in the 2-layer soft-litho microfluidics [1]. The valve can obstruct the bottom channel if the gas or fluid in the top channel is pressurized. The reason for that is the thin and flexible PDMS membrane between the channels (see 2.2(c)) deforms under pressure.

Figure 2.3 shows how the valve behaves at different pressures of the control channel. Lower pressures cause slight deformations of the membrane, whereas under higher pressures the membrane fully occupies the volume under the control channel, fully

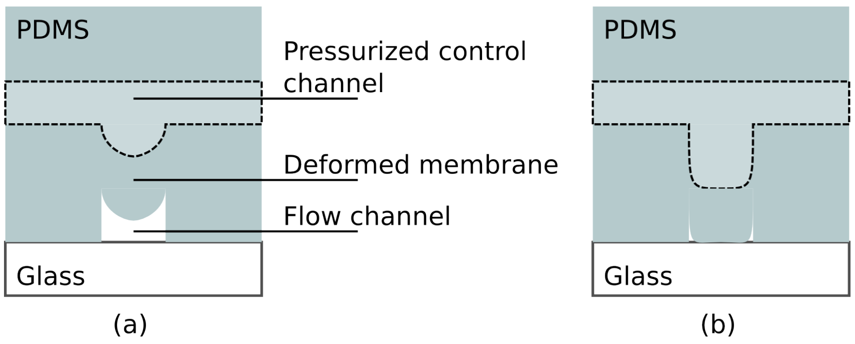
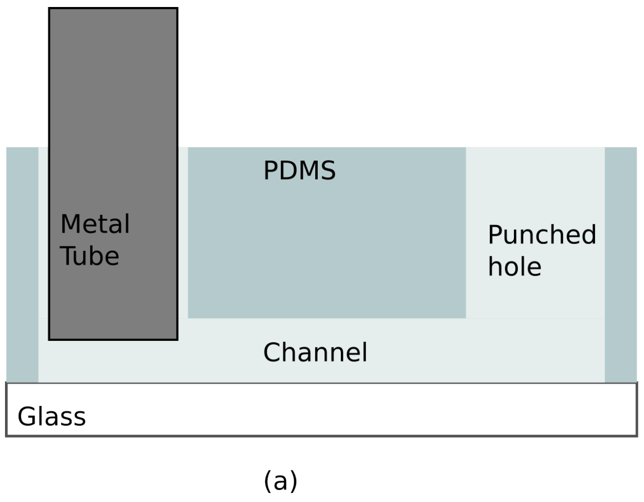


Figure 2.3: A microfluidic push-down valve in action. (a) The top (control) channel is pressurized and the membrane is pushed into the bottom (flow) channel, partially blocking the flow through it. (b) Higher pressure is applied to the control channel, making the membrane fully expand into the flow channel, therefore completely obstructing the flow.

blocking the flow channel (see Figure 2.3 (b)).

To enable the delivery of the pressure to the control channels and flow channels, vertical holes are punched in a chip. These holes are much wider ($> 1000\mu$) than the channels so that a metal tube can be inserted into the chip (see Figure 2.4(a)). The tubes are connected to a pressure source (e.g. a solenoid pump array) via tygon (or any other flexible material) tubing, while the source can be operated by a computer through a USB port.

A sufficiently complex microfluidic chip can have hundreds of valves and channels [8], however not all of them need to be connected to a pressure source. Certain structures can aid in the reduction of the number of control tubes, e.g. a multiplexer (mux) and a demultiplexer (demux). Figure 2.5 shows a simple 4-way demux that can route a fluid from the inlet (channel 0) to any of the 4 channels. While the same functionality can be achieved by having only 1 valve per flow line, it will require 4 control inlets. As we increase the number of flow lines, the demux can save us quite a few control terminals; in fact, if we have $2N$ control lines, a demux can be build that addresses 2^N flow lines. For example, to address 1024 flow lines with a demux, only 20 control inlets are required, while a naive 1024-inlet solution won't even fit on a chip, because of the size of the punch hole!



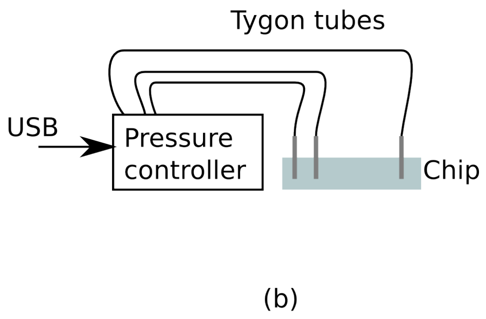


Figure 2.4: The way PDMS chips are operated. (a) A metal tube is inserted into a wide ($> 1000\mu$) vertical hole connected to a channel. (b) An array of metal tubes is connected to a pressure controller via tygon tubing. The pressure controller interfaces with a computer through a USB port.

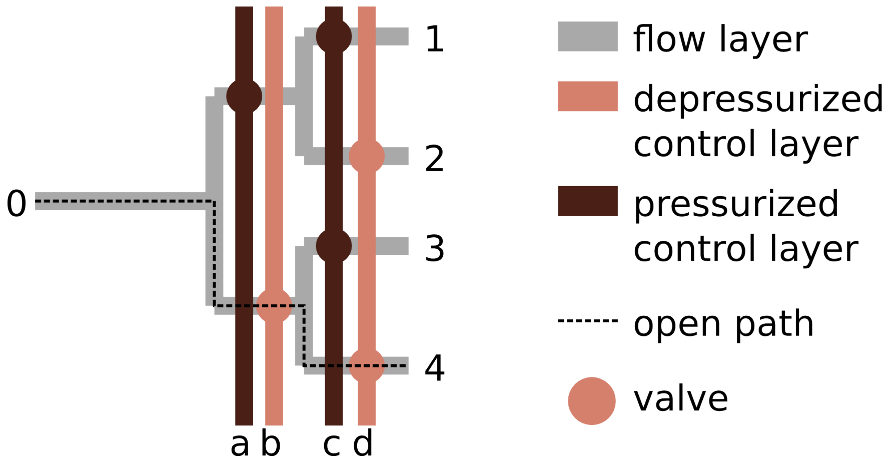


Figure 2.5: A 4-way multiplexer/demultiplexer. Control lines a and c are pressurized, therefore flow lines 1,2 and 3 are blocked and the only possible path is between 0 and 4. If, for example, we pressurize lines a and d, the open path will be 0 to 2. And by pressurizing a and b, we'll block all possible paths.

A microfluidic multiplexor is also a demultiplexer if the flow direction is reversed, i.e. it can route any of its inlets into a single outlet. For example, in Figure 2.5, if the flow lines are connected to 4 different sources of reagents, the multiplexer can connect each of the sources to the outlet 4. Again, being a copy of demux, the multiplexer offers a logarithmic reduction in the number of control lines needed to address a set of sources.

The mux/demux is a very powerful device, featured in many design solutions [7], [9],

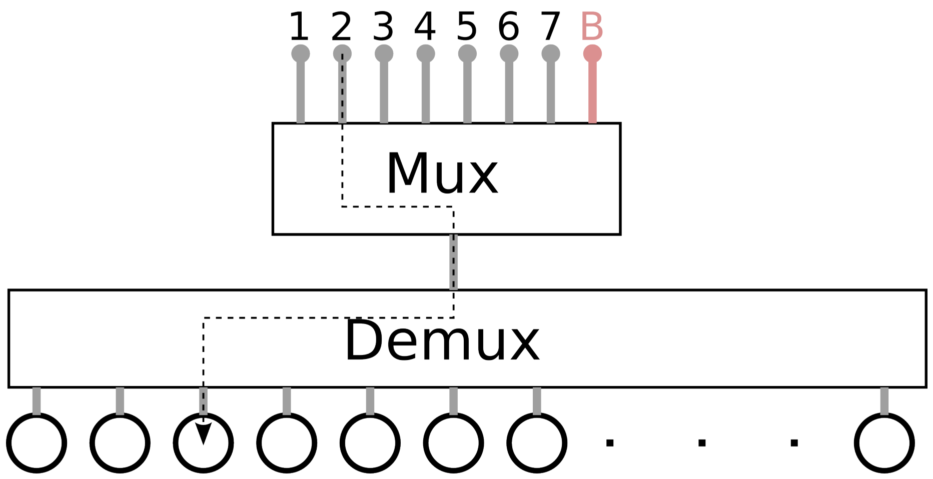


Figure 2.6: A block diagram of an antibiotic testing chip. A multiplexer (mux) is connected to 7 antibiotic sources and one source of bacteria cells. The mux is connected to the demux that addresses a set of 35 cell traps. First, all the cell traps are populated with bacteria, each of the chambers is filled a distinct combination of 3 antibiotics (mixed with feeding media). Finally, the chambers are observed to assess the sensitivity of the bacteria to the drug mixtures.

[10], [11]. For example, consider a simplified antibiotic test chip shown in Figure 2.6. The chip's purpose is to identify a combination (or combinations) that are most efficient in treating a certain bacterial infection. To do so, the chip isolates bacteria cells in special traps, then fills the traps with mixtures of say 3 drugs (it can also be different concentrations of the same antibiotic). The bacteria colony is then allowed to grow and the chambers are observed to see which of the drug mixtures affected the growth the most.

At the higher level, the chip is composed of a mux, a demux and a set of cell chambers.[1] But before a designer can produce a layout drawing of the chip, she needs to answer certain questions. First, she needs to decide on the number of chambers, and therefore, the number of control and flow inlets. More chambers mean more experiments can be done in parallel, fewer chambers result in a smaller chip that will be less prone to fabrication defects.

When the number of chambers is decided on, the designer needs to choose the widths of the channels. On one hand, thinner channels reduces the area of the design; for example, the multiplexer's area can be reduced by a factor of six, if the thinnest possible channels are chosen instead of the widest. On the other hand, wider channels give higher flow rates and lower delays. Again, the difference in flow rate between the widest and the narrowest channels can be an order of magnitude.

The choice of the widths depends on the constraints imposed by the experiment. For example, a chip in [3] deals with mammalian cells that are needed to be planted into the chambers in under than 15 minutes, otherwise they start dying. In this case, the designer might want the chip to be as fast as possible. If the chip is limited by the maximum allowed inlet pressure, as in [4], the designer may opt for wider channels.

However, if the experiment isn't time sensitive, a good solution may be to reduce the area as much as possible, while keeping an acceptable performance (i.e. delays and flow rate). This may be the case in our hypothetical antibiotic chip; doing all bacteria operations in hour is fine, but a day is probably too much.

While for simpler chips, design choices are easy to make, but as the complexity of the designs grows, chances of a mistake increases. Non-linear effects introduced by elastic PDMS and continuous fluid flow add to this probability. Modelling is a common way to address the challenge.

Simulation and modelling methods employed in microfluidics can be partitioned into three categories. The first is a physics-driven model aimed at simulating all relevant phenomena, typically, by numerically solving a set of partially differential equations

[1] A working chip will be slightly more complex, as it requires a place holder for the drug mixture as well as a number of flush lines to prevent combinations from mixing.

(PDE). These models are computationally heavy and typically not used to model an entire chip, rather to predict the performance of a small but critical part of the chip. Physical models also require specialized software and extensive domain expertise.

The second class is reduced order models, which are derived through analytical simplification of full blown physical models. While being much lighter in terms of required computational power, reduced order models are still based on pretty complex PDEs, not easily comprehensible for an unprepared user. Also, these models are not as universal as physics-based ones, and extension of their scope require significant effort.

Compact models are the third group of techniques. Typically, these models are much less detailed than the previous two, modelling a small set of aggregate characteristics, e.g. total flow rates and pressure instead of 3D velocity and force fields. Compact models are typically faster than reduced order ones, and significantly faster than physics-based models, and are much easier to use, understand and analyze. Moreover, a compact modelling approach can utilize existing mature and stable simulators, such as SPICE circuit simulator.

In the following sections, we'll discuss these three groups in greater detail, beginning with Finite Element Method as a common choice of physics-based model.

2.2 FEM

Finite element modelling or method (FEM) is a widely adopted family of simulation techniques applicable to a variety of domains of physics, such as electricity and magnetism, thermodynamics, fluid dynamics, solid body modelling, acoustics, chemistry and others. Essentially, FEM uses numerical methods to (approximately) solve partial differential equations describing a physical system.

Although FEM and its derivatives vary a lot in techniques and algorithms, they all go through two general steps. First, given a description of geometry of a system, an algorithm creates a *mesh* that corresponds to the geometry. Next, the mesh used as a guide to the discretization of the set of partial differential equations. Finally, an iterative algorithm is used to solve these discrete equations. As a result, the method

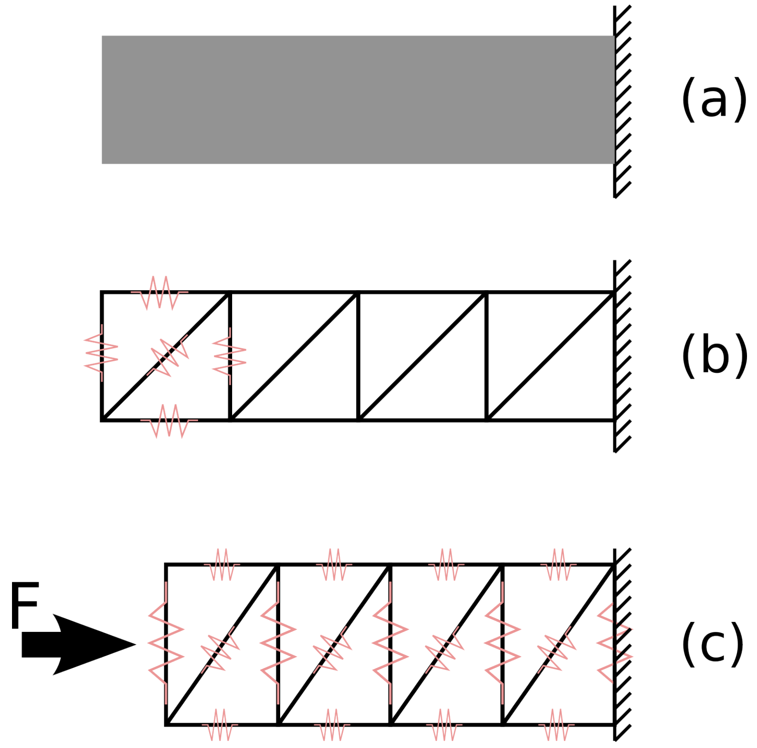


Figure 2.7: A simplified example of finite element method. (a) A block of an elastic material and a wall. (b) The corresponding mesh (black) and springs (red) representing the interaction of the mesh's nodes. (c) A force F is applied to the free side of the meshed block, causing some of the strings to expand and some to shrink, causing the deformation of the mesh that approximates the real deformation of the block.

provides a set of physical values in the nodes of the mesh.

In Figure 2.7 (a), a block made of an elastic material (e.g. some resin) lies on the floor near a wall. Suppose, we want to know how the block deforms if we push it against the wall. The continuous linear elastic model for the block is a set of ten first and second order PDEs. To approach the problem using FEM, the block is first meshed (see Figure 2.7 (b)), i.e. is divided into a set of primitives (triangles in this case) and the PDEs are turned into finite differential equations.

The last step is roughly the same as replacing nodes and the edges of the mesh with finite masses and springs respectively (see Figure 2.7 (b), red springs). So, pushing the block against the wall is approximated with application of a force F to the two leftmost nodes of the mesh (see Figure 2.7 (c)). As the result, horizontal springs shrink, but vertical ones expand, since they are coupled by the diagonal springs.

This example deals with a stress-strain problem, i.e. when a non-moving rigid body is deformed under certain applied forces. In our research, the investigation into the behaviour of PDMS was done primarily using stress-stain analysis. In its solid form, PDMS is typically modelled as a hyper-elastic material, in contrast to linearly elastic ones.

In a 1D case, linear elasticity means linear relationship between an applied force and the resulting displacement, e.g. a deformation of a spring with a stiffness coefficient k can be evaluated by the Hooke's law: $x = F/k$. For a hyperelastic material, this relationship isn't linear and depends greatly on the material. One of the most popular hyper-elastic models used for modelling of PDMS is a nearly incompressible neo-Hookean solid.

As it follows from the name, the model is an extension of the Hooke's law to non-linear stress-strain curves (see Figure 2.8). In a three-dimensional case, the material's properties are defined mainly by its density (ρ), (initial) bulk modulus (κ), and (initial) shear modulus (μ), coefficients that describe the reaction of the material to uniform compression and shearing (see Figure 2.9). PDMS is typically modelled as a weakly compressible hyper-elastic material, i.e. $\kappa << \mu$.

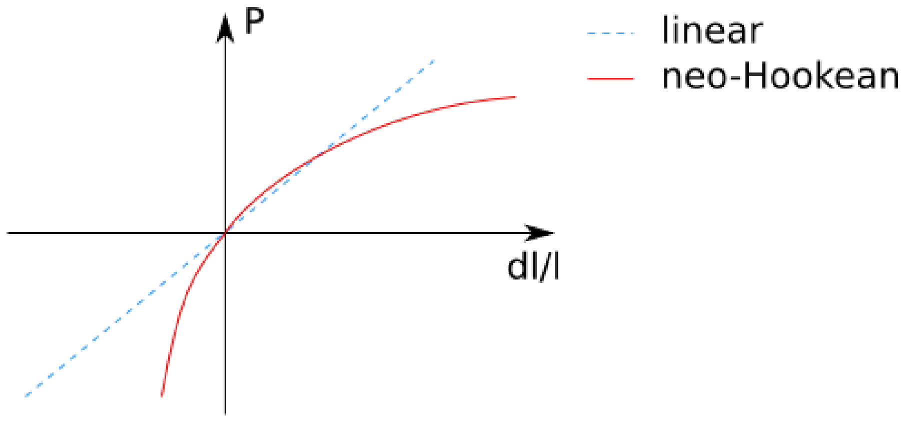


Figure 2.8: A stress-strain (e.g. relative force vs relative displacement) curve for a linearly elastic and a hyper-elastic material. Positive dl/Ls mean stretching and negative mean compressing. A hyper-elastic material is more prone to stretching than compressing in a single dimension case.

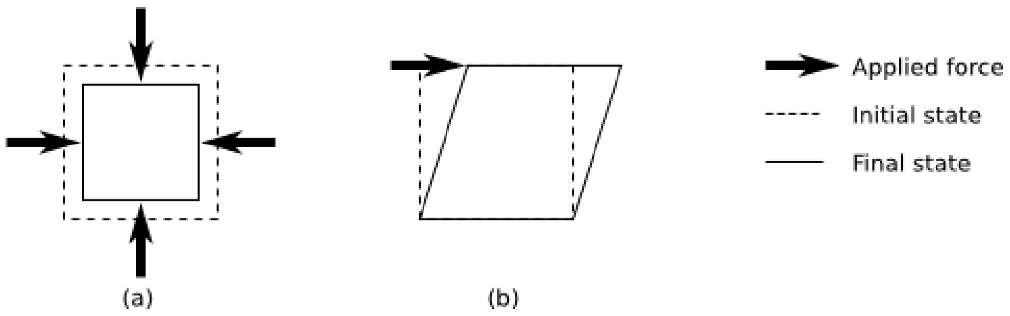


Figure 2.9: Explanation of the bulk and shear moduli. (a) Solid's reaction to a uniform compression is characterized by the bulk modulus. (b) Shear modulus describes the reaction to shearing.

The second part of the microfluidic system to be simulated is the fluid, the media that occupies the channels within the chip. Typically, to simulate the behaviour the fluid domain, the Navier-Stokes equation is used. Similar to the neo-Hookean model, this equation is based on first principles, i.e. conservation laws and general structure of the media.

Navier-Stokes equation is a non-linear partial differential equation that solves for the velocity field given the pressure boundary conditions. The fluid's properties are generally characterized by two parameters: density (ρ) and viscosity (μ), a parameter that describes the loss of energy due to particle friction (e.g. water has low viscosity, while oil has a higher one). The higher viscosity and density, the harder is to move the fluid.

Fluid flow inside a fully filled microfluidic channel has a low Reynolds number, meaning the flow is laminar. The velocity profile of the flow is typically parabolic-like[2], with zero velocity at the channel's boundaries and maximal velocity at the center.

Finally, the biggest benefit of the FEM approach is the ability to couple two (or

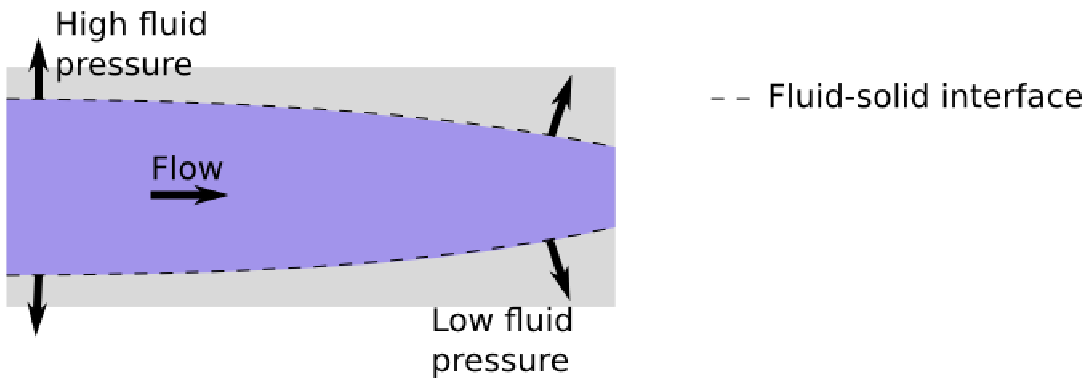


Figure 2.10: Solid-fluid interaction diagram. Higher pressure at the beginning of the channel causes higher deformation of the channel and vice-versa, low pressure at the outlet produces small displacements in the PDMS.

more) domains into a single system of equations and produce its approximated solution. When we deal with the PDMS-liquid system, the interface between the two domains is quite obvious – the inner surfaces of the channels. Figure 2.10 shows the

[2]Exactly parabolic for a channel with a circular cross-section

way two domains are joined in a single system. The interface surfaces in both fluid and solid share the same force field and the same displacement, i.e. higher pressures cause higher displacements of the solid and, therefore, expanding the flow cone.

The coupled (or multiphysics) problem is pretty complex and solving it via FEM requires significant computational resources, especially in a non-stationary regime, i.e. where time is a separate variable. For example, a microfluidic channel of a medium length can take several hours to produce seconds of simulated time. Moreover, a stationary valve's simulation can take days to converge.

In microfluidics, FEM and similar techniques are commonly used for modelling of a small (and possibly critical) part of a chip. In [12], a single cell trapping structure is modelled using computational fluid dynamics (a subset of FEM with no wall deformation). The simulation was used to increase the probability of a cell being captured, as well as to evaluate the total pressure drop across the structure. The latter was used to make sure, that the flow rate through a sequence of traps will be sufficient for the experiments. Similar traps for embryos were simulated in [13].

Simulations of deformable solids are also employed in microfluidics. In [14], scaling properties of a push-up PDMS valve were evaluated using finite element modelling. Dependence between valve's geometric dimensions (like height, width and thickness) and actuation pressure was established using the simulation. A passive valve is modelled in [15]. The simulation was used to optimize the pressure-flow relationship of the valve. A valve-like structure, microfluidic micro-lens, is simulated in [16] to evaluate the relationship between applied pressure and optical characteristics (like focal length).

2.3 Reduced order and compact models

Another approach sometimes used in modelling is called reduced order models. The main idea is to take PDE describing some physical system and use certain assumptions and system's characteristics to analytically reduce and/or approximate these equations. Then, the simplified PDEs can be integrated either numerically or even analytically. In [17], this approach was used to model an oscillatory flow in a slightly deformable tube. A microfluidic structure for reducing flow fluctuations is modelled in [18].

While often being accurate and allowing for non-numerical (e.g. symbolic) analysis of the system, reduced order models aren't easy to derive and work with. Moreover, due to constantly increasing computational power available to researchers, it's getting easier to perform numerical simulations of the full-blow physics-base models. A compact model, on the other hand, can be seen as a reduced order model of an elementary device, rather that the system as a whole.

In semiconductor integrated circuits, a transistor is the main building block. The device has highly complex underlying physics that described the diffusion of the electrons within a heterogeneous solid structure, including quantum mechanics and electromagnetism. By the time the number of the transistors per an integrated circuit reached hundreds, circuit simulation was an inevitable step in the design flow. While providing sufficient accuracy, transistor models based on actual physics were too computationally expensive to be used. On the other end of the spectrum, small-signal linearized models had an insufficiently narrow scope for some applications. To address the problem compact models were developed.

Though not having a precise definition, a compact model is an accurate, lightweight and robust model of an object (e.g. a device, a transistor, or a microfluidic channel). These benefits come at a price. First of all, a compact model is much less general, than a physics-based one, such as FEM. For example, in a transistor's compact model, geometrical dimensions (e.g. transistor's length or width) can be varied, while the overall shape (e.g. rectangular) is a fixed parameter. Second, compact models often hide actual physics from the user by approximating functions derived from a physical simulation or an analytically derived physical model.

To give a sense of the complexity reduction provided by compact models, consider a bipolar transistor. The physical model of the device consists of a dozen of partial differential equations in a three-dimensional space and a set of 2-d boundary conditions.

Developed in mid 50s, Ebers-Moll compact model reduces the static I-V relation to a set of three simple (non-differential) equations of the form:

$$i_B = \frac{I_s}{\beta_F}(e^{V_{BE}/V_T} - 1) + \frac{I_s}{\beta_R}(e^{V_{BC}/V_T} - 1),$$

where β's and V_T are parameters and the rest are variables, I's and i's are currents and V's are voltages. The set of equations contain the minimally sufficient number of non-linear functions (exponents in this case) necessary for production of accurate results.

Transistor compact models, however, revealed their true powers only after an efficient circuit simulator, SPICE, was created in mid 70s [19]. The tool was capable of taking a circuit description as an input, producing intermediate differential equations that couple elements of the circuit (e.g. transistors) and used numerical methods to solve these equations.

The tool aids circuit designers in various ways. First, the simulation can show if the integrated circuit in question satisfies the requirements and specifications. Second, if it doesn't, the tool can be used for debugging, since it reveals all the currents and the voltages inside within the circuit. Finally, the simulator is also used to optimized circuit's characteristics; for example by doing sensitivity analysis.

Existing microfluidic compact models address some, but not all, effects that take place in soft-litho microfluidics. In [20], linear hydraulic-electric analogy is used to model flow in a network of microfluidic channels. Non-linear flow-pressure relation in a rigid-walls device is modelled in [21] as a $Q(\Delta P)$ function. Similar approach is used for a single layer valve with deformable elements in [15]. Capacitive effects (i.e. the ability to accumulate fluid) of elastic microfluidic structures are modelled as linear relations between change in volume and applied pressure. In [22] and [23] compact (capacitive) models of elastic membranes are used to model pumps. A capacitive model of a membrane is also used in [18] to analyze the performance of a fluid stabilizing device. Similar models are used in [24] and [25] to address linear expansion of elastic channels. In [26], an elastic channel is modelled as a linear RC network.

To address both non-linear flow-pressure relation and non-linear capacitive effects typical for soft-litho microfluidics, we generalized existing compact modelling ideas and constructed models of a valve and a channel. Then, we used SPICE to simulate devices composed of these building blocks. We used the results of the simulation to validate the compact models by comparing the results with an experiment involving a real chip. Next, we produced parametrizable models for more complex devices, such as networks of channels and used the model for the global optimization of a design.

Chapter 3

Compact Modelling of Soft-litho Microfluidics

3.1 Intro

Soft-lithography allows one to manufacture a variety of microfluidic structures. They range from simple devices like inlets to complex multi-layer structures, like pressure traps or turbulent mixers. While one can make a large number of devices, most chips use only two: valves and channels. Most of the more complex structures are actually just a composition of these two building blocks. For example, a pump is a set of 3 valves connected by flow channels; a cell chamber is simply a (large or wide) segment of a channel. Since a significant share of microfluidic chips are solely composed of valves and channels, they are an obvious choice for our first compact modelling.

For each device, we'll show how the compact model was derived from the first principles. Then, we'll describe how we fit and validated the model using COMSOL FEM package[27]. Finally, we'll present our experimental verification of the models.

3.2 Channel

In a general sense, a microfluidic channel is a fluid-filled cavity inside a bulk of PDMS. For modelling purposes, we pose an additional requirement for an unpressurized channel to have a constant cross-section shape (e.g. a rectangular one). This doesn't restrict the generality much, as most variability can be addressed by modelling a sequence of constant-shaped channels. [1]

Soft-litho microfluidics uses channels in two ways: flow and control. Flow channels are meant to conduct fluid from one place to another. They are connected from an inlet with some positive pressure to an unpressurized outlet. Control channels have an inlet as one terminal and a dead end at the other end with one or more valves along the way. They are used to actuate the valves by applying pressure to the inlet.

Microfluidic channels also vary in length, cross-sectional shape and its location within a chip (i.e. layer). In this section, we will develop a single compact model that accounts for these parameters.

3.2.1 Model

First, consider a rigid channel, i.e. one which doesn't deform under pressure (for example, a cavity made in glass). Its cross-section shape is constant and doesn't depend on the pressure. In a rigid flow channel, the stationary flow rate Q is linear in the pressure drop ΔP and can be very accurately approximated by Poiseuille's law:

$$Q = G\Delta P,$$

where G is hydraulic resistance of the channel and is linear in channel's length L. When pressure is applied to a rigid control channel's inlet, it propagates to its end with a speed of sound (almost instantly, compared to speeds of other processes) in incompressible fluids. Given that, the equivalent circuit is just a resistor (see Figure 3.1).

[1] If there was enough need, we could generate a model for truly variable shaped channels using the same approach.

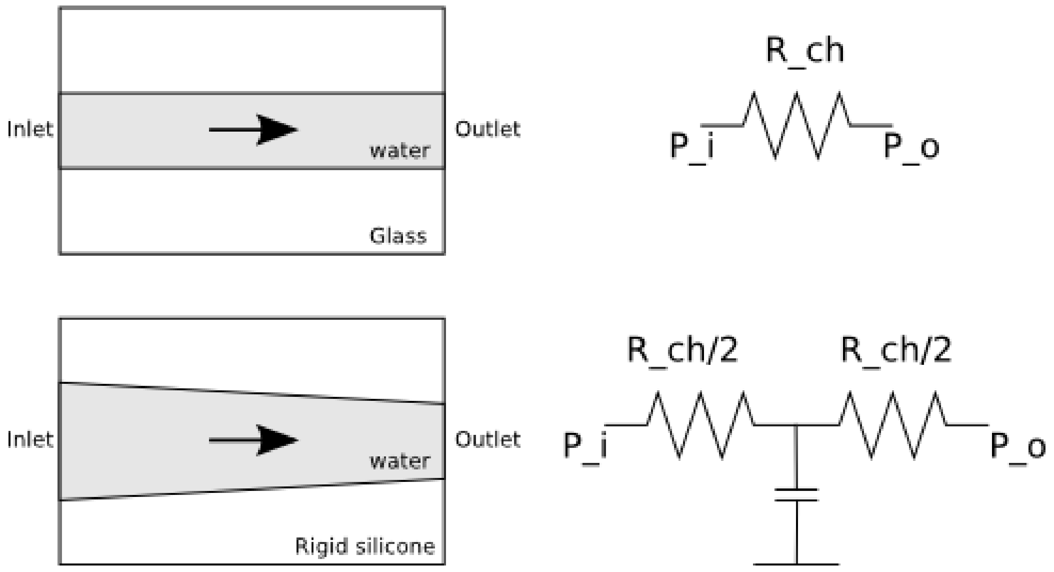


Figure 3.1: Simple microfluidic channels. An absolutely rigid channel (top left) in a glass bulk doesn't deform under working pressures. An equivalent circuit (top right) is a resistor with a value (R_{ch}) defined by Poiseuille's law. P_i and P_o are the inlet and outlet pressures respectively. Slightly deformable channel (bottom left) has a linear pressure drop and deformation across the channel. A T-section with two resistors and one capacitor is the equivalent circuit (bottom right)

Suppose now, the channel isn't completely rigid, but its pressure-induced deformations are slight (or linear), i.e. $\Delta S << S$, where S is the cross-sectional area. If the flow is constant, the deformation will linearly decrease from the inlet to the outlet (see Figure 3.1). Therefore, the hydraulic conductance is a linear function of ΔP:

$$G = k(P_i + P_o)/2 + g_0 = kP_{avg} + g_0$$

where $P_{avg} = (P_i + P_o)/2$ is the average between inlet and outlet pressures, g_0 is conductance of the non-deformed channel and k is a positive coefficient, i.e. the more pressure we apply, the higher is conductance. The linear increase of the cross-sectional area (and therefore the hydraulic conductance) is sometimes called compliance of the channel [25]. The Q vs ΔP law is:

$$Q = kP_{avg}\Delta P + g_0\Delta P,$$

and isn't linear any more. More importantly, the flow rate Q doesn't depend solely on $\Delta P = P_i - P_o$, but both on P_i and P_o. Unfortunately, the approach is not applicable to large channel deformations. However, we can address the non-linearity using potential functions.

Consider a channel's section with a very small length dl (see Figure 3.2). The pressure is constant inside the section and the flow rate Q is:

$$Qdl = g(p)dp,$$

where the $u(p)$ function is a linear conductance of the channel. Integration of both parts of the equality yields:

$$QL = \int_{p(0)}^{p(L)} g(p)dp = U(p_i) - U(p_o),$$

where $U(p)$ is the integral of $u(p)$ and L is the length of the channel. The $U(p)$ is the said potential function, since the flow rate Q is linear in its dependence on ΔU. This function has a number of interesting properties worth mentioning.

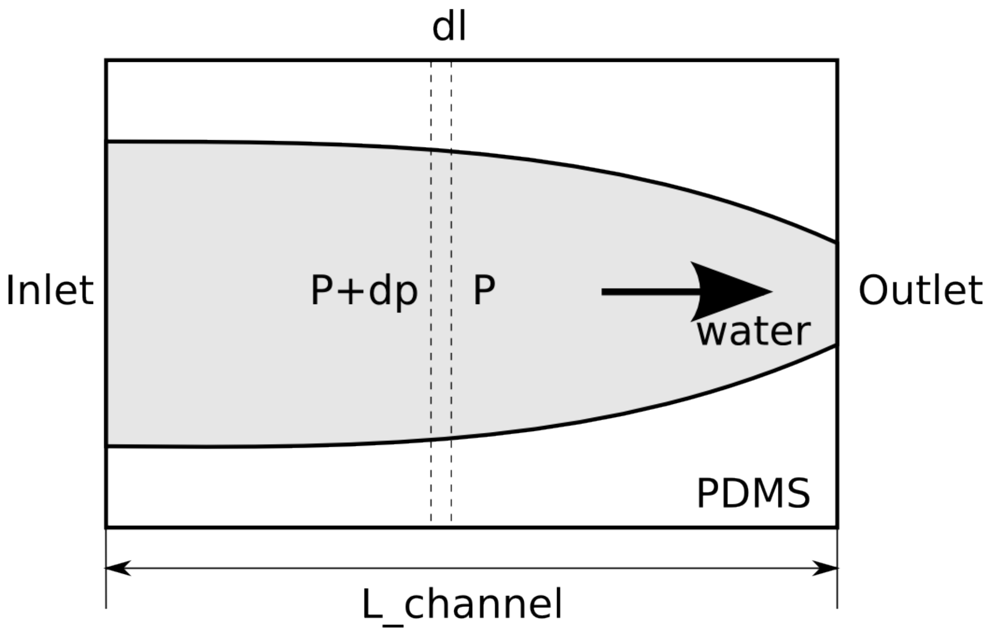


Figure 3.2: A general microfluidic channel of length $L_{channel}$ in a PDMS bulk and its small section of length dl. The pressure drop and the deformation are non-linear across the channel. The section is pressurized at P and has dp pressure difference.

First of all, the potential function is the only characteristic of a channel (and the fluid it is filled with) needed to calculate the flow rate given the in- and outlet pressures; it models all the fluid-solid interaction physics that takes place in the chip. In case of a rigid channel, for example, the potential reduces to an affine function.

Second, the microfluidic potential function is defined up to a constant, a property shared with all potential functions, i.e. $U(p) + K$ provides the same results as $U(p)$. For consistency reasons, however, we'll define the constant such that $U(0) = 0$.

Finally, the potential drops linearly across the channel: $U(P(x))$ is linear in x. This is easy to see, if we divide a channel into two parts of lengths x and $L - x$. Then, the potential drop equations are:

$$Q = \frac{U(P(x)) - U(P(0))}{x} = \frac{U(P(L)) - U(P(x))}{L - x},$$

where $P(x)$ is the pressure at a point x. A simple transformation leads to:

$$\begin{aligned}
\Big(U(P(L)) - U(P(x))\Big)x &= \Big(U(P(x)) - U(P(0))\Big)(L - x) \\
U(P(L))x - \cancel{U(P(x))x} &= U(P(x))L - \cancel{U(P(x))x} + U(P(0))L - U(P(0))x \\
U(P(x)) &= \Big(U(P(L)) - U(P(0))\Big)\frac{x}{L} + U(P(0)).
\end{aligned}$$

In transient regime (e.g. during pressurization), the behaviour of the deformable

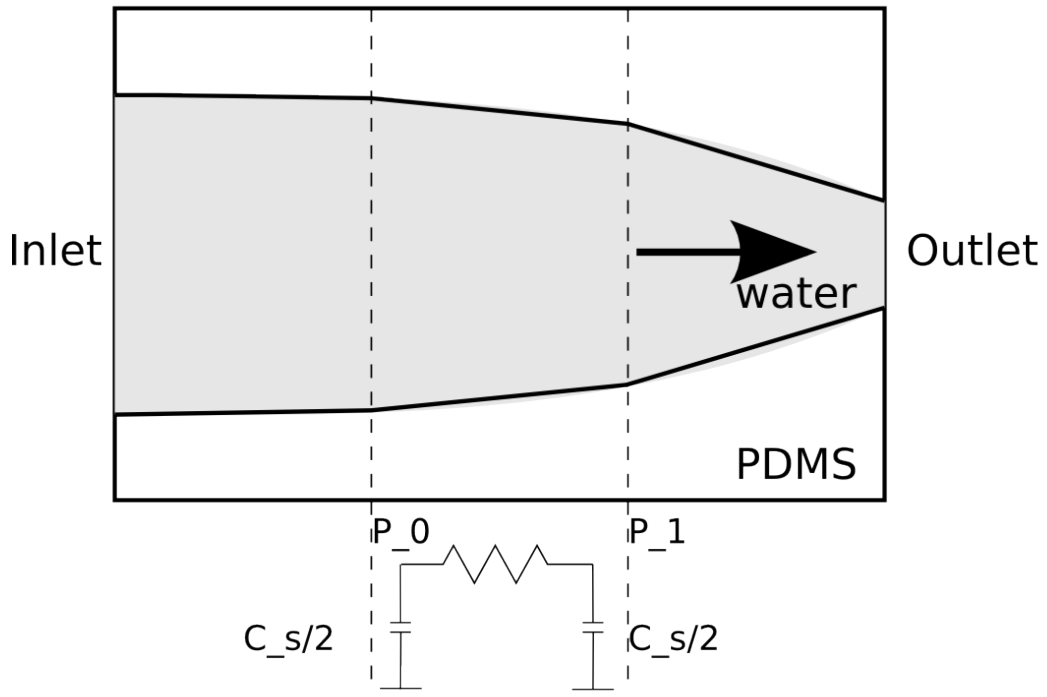


Figure 3.3: The channel is partitioned into 3 sections, the volume of each section is approximated by the volume of the cone section; the area drop linearly across the cone, therefore its volume is $P_0C_s/2 + P_1C_s/2$, where P_0 and P_1 are pressures at the beginning and the end of the section and C_s is its capacitance.

channel differs from that of a rigid channel. A deformable channel has more fluid inside under positive pressures. And due to non-zero hydraulic resistance, it'll take some time to fill in the volume difference. As a result, the pressure doesn't propagate to the end instantaneously. To model the process, we need a notion of hydraulic capacitance C:

$$C(P) = \frac{dV}{dP},$$

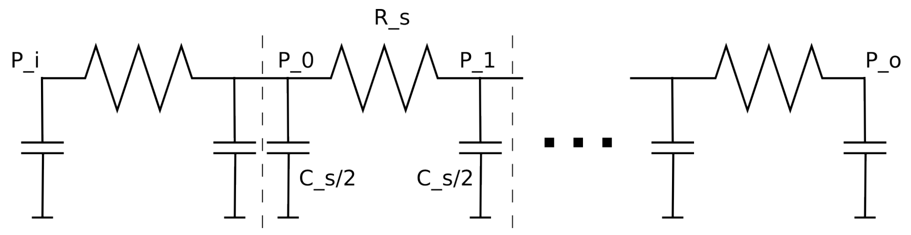


Figure 3.4: Equivalent circuit of a channel, a series of Π-sections. C_s is the capacitance of the section and is equal to C_{ch}/N, an N-th fraction of the channel's capacitance. The flow Q through a non-linear resistor R_s is $QL_s = U(P_1) - U(P_0)$, where L_s is the length of the section.

i.e. ratio between volume and pressure. In the case of small deformations, cross-section area S drops linearly across the channel for any ΔP, therefore C is constant and the resulting equivalent circuit is shown in Figure 3.1 [26].

In a general case, where the $S(x)$ curve is a non-linear one, we can divide the channel into a number of sections, the channel can be approximated by a cone section to high precision (see Figure 3.3). This in turn means the volume of the section can be modelled as two constant capacitors at both ends of the section:

$$\Delta V_{segment} = L_{segment} \frac{\Delta S(x_0) + \Delta S(x_1)}{2} = \frac{C_s}{2} P_0 + \frac{C_s}{2} P_1,$$

where $V_{segment}$ and $L_{segment}$ are the volume and the length of the segment, $S(x)$ is the section area at a point x, C_s is the capacitance of the section and P_x is the pressure at a point x. Putting it all together, the model of the section is a Π-section, i.e. a non-linear resistor with two capacitors (see Figure 3.3) and an equivalent circuit of a channel is a series or a chain of such Π sections (see Figure 3.4).

The circuit is a form of a distributed RC line and is very common for the world of integrated circuits; its properties are well understood and it is easy simulate using standard software packages like SPICE. Another advantage of the model is low number of parameters; we only need to know the potential function and the capacitance value. We will extract these parameters using Finite Element Modelling, which is discussed next.

3.2.2 Finite Element Modelling

To determine the channel's model parameters, we used COMSOL Multiphysics Finite Element Modelling software. It allowed us to simulate fluid-solid interactions of structures of arbitrary geometry in both stationary and transient regimes. We did both model fitting and verification of the model in COMSOL.

In COMSOL, modelling is a 4-step process. At first, you define the geometry of the structures to be simulated. Next, the structures are assigned to domains (e.g. fluid or solid) that define the physics to be modelled within the structure. Material properties, boundary conditions and domain interfaces are set up as well. Then, the stimuli and simulation parameters are defined and the simulation is executed. Finally, the user can extract simulation results using various probing methods. All four steps depend on the type of the channel and the purpose of the simulation.

Geometry

For both fitting and verification, we modelled a long water-filled channel in a bulk of PDMS [2](see Figure 3.5). The dimensions of the PDMS bulk were typically $1cm \times 1cm \times 1cm$. Experiments showed that at such sizes, the results were insensitive to the changes in the geometry of the bulk. The height of the bulk also corresponds to a typical height of a PDMS chip. Accordingly, the length of the channel was $10mm$ (at least 100x of the channel's width). This $L >> W$ condition allowed us to diminish the boundary effects at the ends of the channel.

We modelled both push up and push down channels. For a push-down configuration, flow channels were at the bottom of the bulk, while control channels were about 80μ above the floor. For push-up channels, the control and flow channels switch positions. The actual height of the control channel was calculated as a sum of the height of the flow channel and thickness of the membrane (see Figure 3.6). Both values are assessed

[2]While PDMS bulk is bonded to glass substrate and therefore PDMS isn't the only material in the chip, we bypassed the need to directly model the glass part by using appropriate boundary conditions.

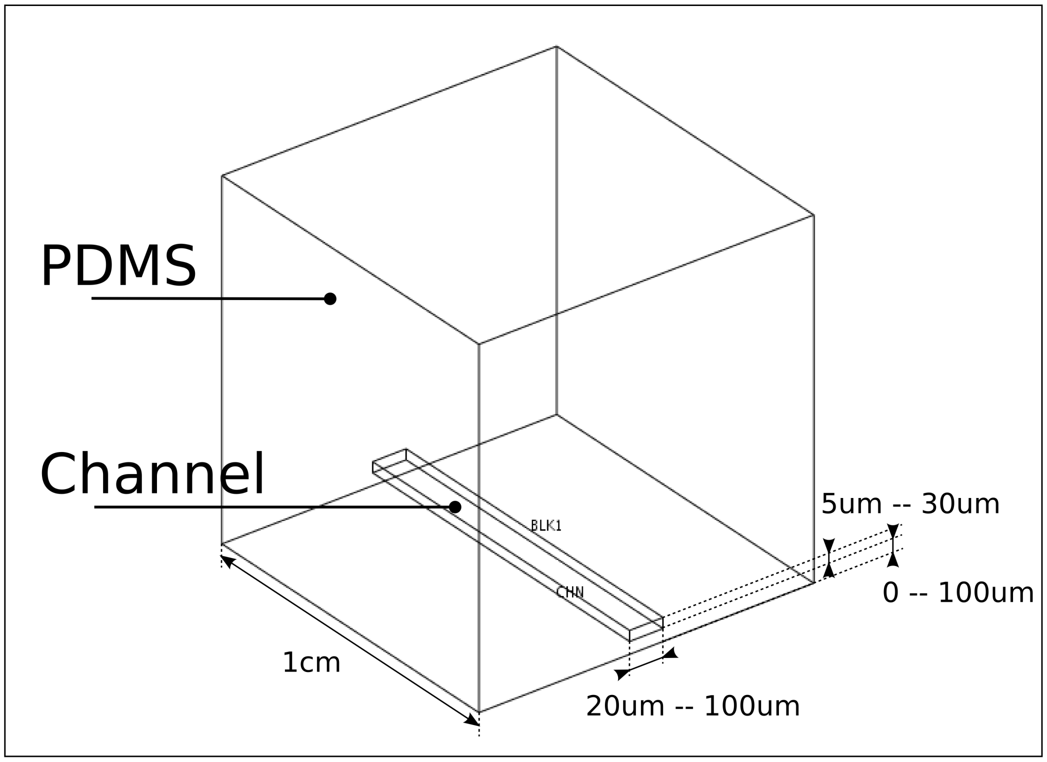


Figure 3.5: A typical setup of the channel's simulation in COMSOL (not to scale). The bulk has a cavity of the channel's shape, so that the channel and the bulk together form a cube with a side of $1cm$. The width of the channel ranges from 20μ to 100μ and its height varies from 5μ to 30μ. The elevation of the channel is typically between 0 and 100μ.

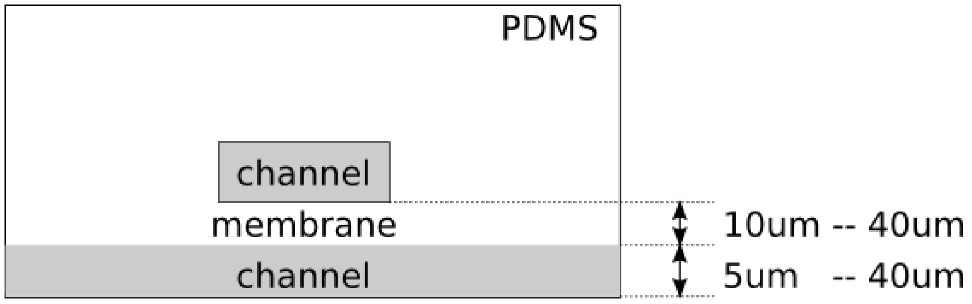


Figure 3.6: Elevation of the channel (front view, in the middle). A channel of certain types (e.g. a control one in push-down technology) has to be elevated above the floor, so that the simulation corresponds to the actual chip. The elevation is computed as the sum of the membrane's thickness and the height of the bottom channel and is typically from 15μ to 80μ.

either by spin-coating speed vs thickness table or using a profilometer.

While the bulk of the PDMS was always a cube, the shape of the channels varied. Two

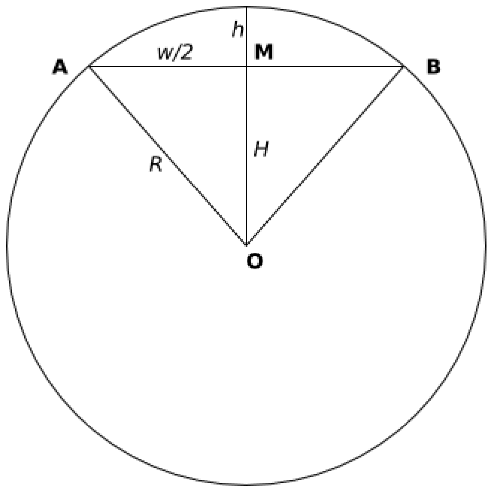


Figure 3.7: Sectioning of a cylinder of radius R with a plane at height H. The resulting section (top) has height h and width W. O is the centre of the circle, A and B are the edges of the channel and M is its middle point.

types of channel shapes were modelled: rectangular and tear-drop shaped. Channels with a rectangular cross-section were trivial to generate, since it was a geometry primitive in COMSOL. We modelled tear drop-shaped channels as a section of a cylinder (see Figure 3.7). The radius of the cylinder R and the height of the section plane H were derived from the following expressions:

$$\begin{aligned} H^2 + \frac{W^2}{4} &= R^2 \text{ (triangle OMB)} \\ h + H &= R, \end{aligned}$$

where h is the height of the channel and W is its width. Equivalently,

$$\begin{aligned} H &= \frac{W^2}{8h} - \frac{h}{2}, \\ R &= \frac{h}{2} + \frac{W^2}{8h}. \end{aligned}$$

The resulting cross-section shape doesn't differ much from the actual profile of a tear drop-shaped channel measured by a profilometer.

Properties

The PDMS bulk and the channel were set to solid and fluid domains respectively. Hyper-elastic PDMS was modelled as a neo-Hookean solid. The channel was assigned with water as the material.

The six external surfaces of the bulk were assigned with the following boundary conditions: the top was always a free surface, as were the left and right sides; Front and back surfaces were fixed in the X direction (the one the channel ran along), so that the channel could deform in the $Y - Z$ plane only. Finally, the bottom surface could be fixed, in case the chip was bond to the glass, or it could be free.

The channel's top and side surfaces were coupled with the PDMS bulk, while the bottom was either coupled or fixed if the channel had a glass floor. The back and forward surfaces were fixed in X direction to be consistent with the bulk.

Since both the channel and bulk have extruded geometry, a swept mesh (see Figure 3.8) was the natural meshing algorithm to use. At first front surfaces of the bulk and the channel were 2d-meshed and then the 3d mesh was constructed as a set of prisms, based on the 2d triangles (see Figure 3.9). This kind of mesh allowed us to significantly reduce the number of nodes and, as a consequence, memory footprint

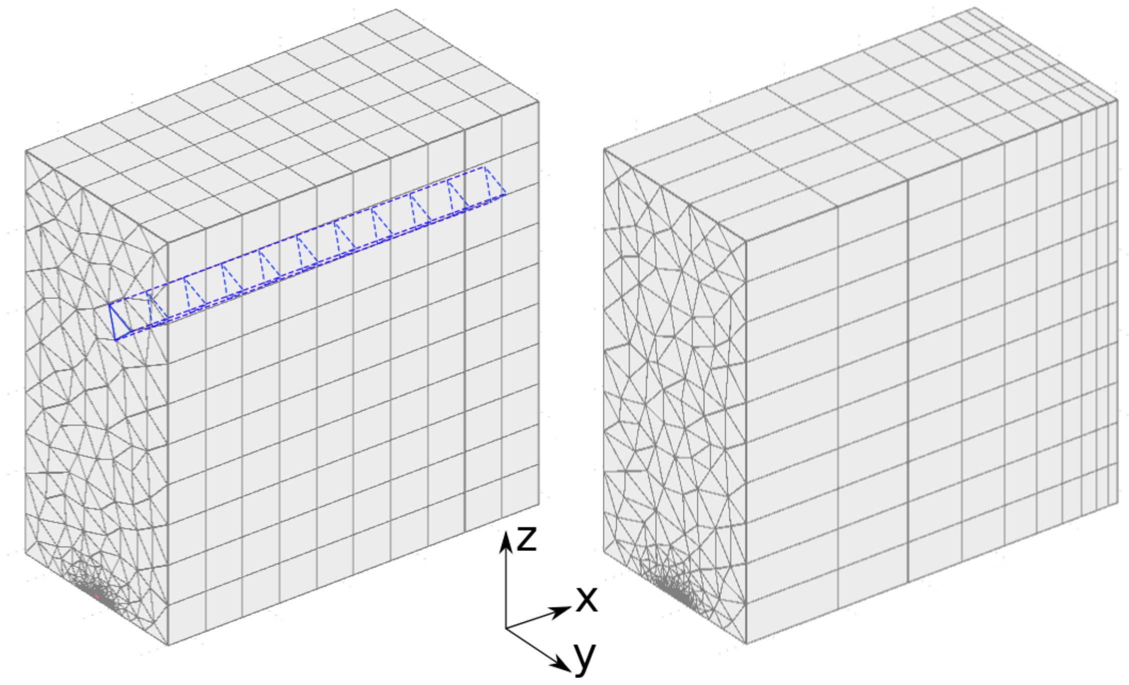


Figure 3.8: Uniformly (left) and exponentially (right) distributed swept meshes of the bulk. Swept mesh, unlike tetrahedral one, consists of series triangular prisms (depicted in blue). It is a good choice for problems with extruded geometry, since it requires less mesh primitives while achieving the same simulation quality. Non-uniform swept mesh allows one to increase the simulation quality while keeping the same number of nodes.

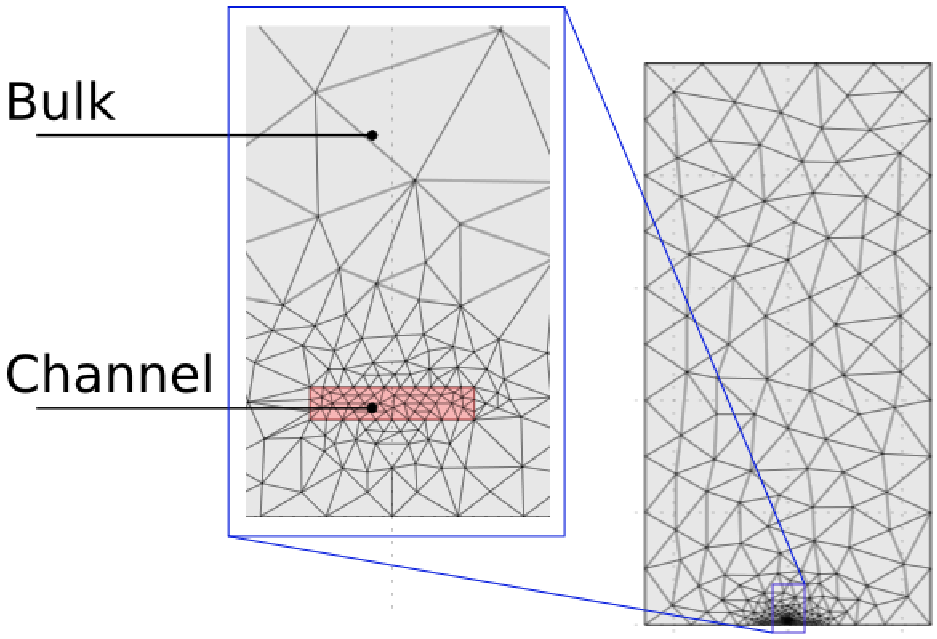


Figure 3.9: 2d mesh of the front surface of the bulk and the channel: overall view (right) and a close view on the channel (left). The mesh is triangular; a minimum triangle size in the channel is 5μ. The bulk is adaptively meshed, so that the mesh is fine grained close to the channel, but is coarse otherwise.

and the execution time.

The size of the minimal element of the 2d mesh was chosen such that the results of the simulation were insensitive to this size: about 20 triangles for the channel's surface. The 2d mesh of the PDMS bulk was automatically (by the tool) adapted to the mesh of the channel.

The number of prisms (or the number of sections) varied from 10 to 50, depending on the pressure drop across the channel. For smaller number of sections the accuracy of the simulation was increased by a non-uniform partitioning of the channel (see Figure 3.8); the mesh was finer where larger pressure drops (and therefore deformations) were expected.

Stimuli and Simulation

We conducted two types of simulations; steady state (i.e. constant flow) and transient. The first type was used to extract parameters, while the second helped to validate the model's results.

For stationary simulations, the front and the back surfaces of the channel were its inlet

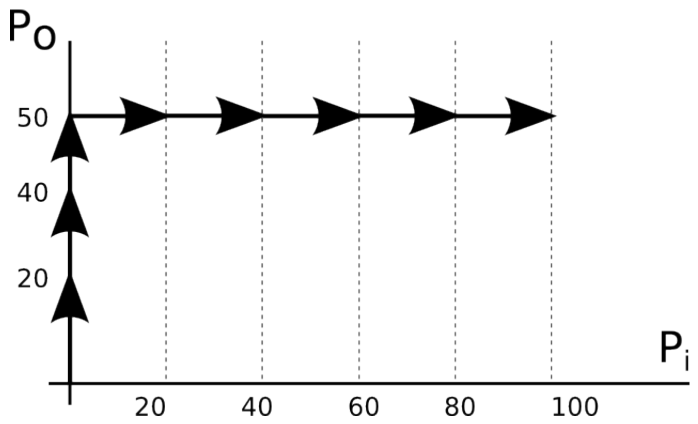


Figure 3.10: A simulation protocol for inlet and outlet pressures of 100kPa and 50kPa respectively. The values of the steps is less than $20kPa$, thus all the steps converge. This simulation schedule isn't unique, but it's the easiest one to implement in COMSOL.

and outlet respectively. Both terminals were assigned with pressures p_i and p_o ($p_i \geq p_o$). The pressure values ranged from 0 to $200kPa$. For lower pressures ($\leq 20kPa$), the simulation converged unconditionally. For higher absolute pressures or pressure differences ($p_i - p_o$) simulations, we needed to conducted a series of simulations, each time increasing the value of the pressures by a small step ($10 - 20kPa$), and use the results of the previous simulation as a starting point for the next one. For example, a simulation protocol for $p_i = 100kPa$ and $p_o = 50kPa$ is shown in Figure 3.10.

In transient simulations, we sealed the back surface of the channel (making it a wall) and applied an approximate pressure pulses to the front surfaces (the inlet). The attack time of the pulse was typically about 10^{-3} sec and the amplitude varied from 0 to 200kPa. The simulated time ranged from $1s$ to $15s$. To achieve convergence at higher pressures ($> 100kPa$), we used longer attack times (up to $10^{-2}s$).

Data Processing

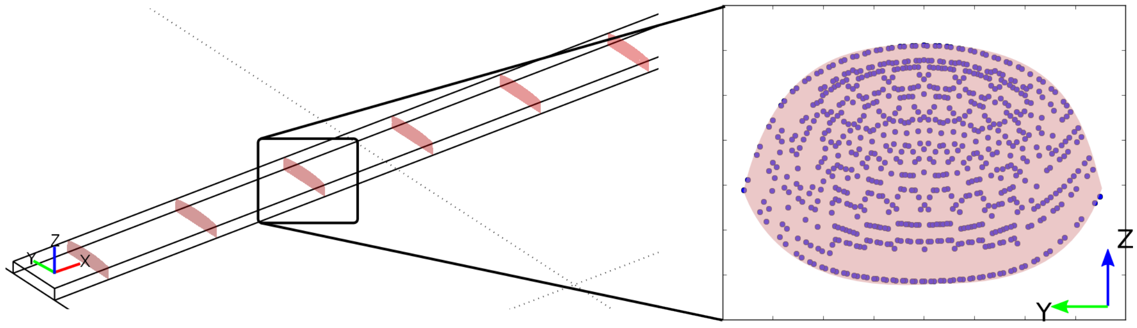


Figure 3.11: Slices of a channel (left) and a 2d point cloud of a single slice (right). Each blue node corresponds to a data point (x, y, z, p) extracted from the simulation's results. To obtain the area-vs-pressure curve, a convex hull of each slice (right) is being created and its area computed.

After the simulation finished, we used COMSOL to extract certain data as 2d and 3d point sets. For stationary simulations, we extracted the flow rate and the channel's volume vs inlet and outlet pressures curves, pressure distribution along the x axis, and the channel's cross-section area vs x. For all, but the last curves we used standard COMSOL tool set. To obtain the cross-section area vs x, we extracted pressure vs (x, y, z) coordinate for a number of channel's slices. Then, we applied a convex hull algorithm to each slice to find the contours of the slice and computed the area (see Figure 3.11). For transient simulations, we evaluated pressure vs time and volume vs time curves using the COMSOL tool set.

3.2.3 Compact Model Fitting

For each channel-bulk configuration, we needed to extract the potential function $U(P)$ and the capacitance C_{ch}. We employed two different fitting methods; one is faster, the other is more accurate.

The first fitting methods relies on a linear drop property of the potential function, i.e. $U(P(x)) = kx + b$. Moreover, if we set the outlet pressure to zero, we'll arrive at:

$$U(P(x)) = kx = \frac{Q(P_{inlet})}{L_{channel}}x, \tag{3.1}$$

where $Q(P_{inlet})$ is the flow rate given the inlet pressure and $L_{channel}$ is the length of the channel. The second quantity is easy to check by assigning x to 1: we'll get the definition of the potential function.

From the simulation, we obtain the $p(x)$ curve and the value of Q. Since the $p(x)$ is monotone, we can invert the function to get $x(p)$. If we substitute x with $x(p)$ in (3.1), we arrive at:

$$U(p(x(p)) = U(p) = \frac{Q(P_{inlet})}{L_{channel}}x(p).$$

The equation provides a straightforward way for fitting the potential function:

1. Simulate a long channel with some P_{max} as the inlet pressure and 0 as the outlet pressure. The value of P_{max} is the maximum pressure to be used in the design (e.g. $20psi$).

2. Extract the values of Q and the $p(x)$ curve as a set of (x, p) data points.

3. Do a polynomial fit $\hat{x}(p)$ of $x(p)$ function given the data points. Then $\frac{Q(P_{inlet})}{L_{channel}}\hat{x}(p)$ will be the polynomial approximation of the potential function.

While the result of the procedure is an approximation, by selecting the degree of the polynomial, we can achieve any given accuracy. Thus, we used the terms potential function and polynomial approximation of the potential function interchangeably.

To fit the C_{ch}, the channel's linear capacitance, we extract the cross-section area vs the x coordinate and substitute x with $p(x)$ to get the (S, p) data points. Then we fit the points with a linear function, and the resulting slope is C_{ch}.

While this simulation method is fast (the simulations typically take about 10 minutes), it suffers from simulation inaccuracy; at high pressures the deformations of the inlet surface is significantly different from that of the outlet. This deformation causes the simulator to violate the mass conservation law: difference between the inlet and the outlet flow rates may be up to 25%. This effect is especially strong in simulations of rectangular shaped channels, since higher pressures round the channels. While the flow mismatch can be diminished by refining the channel's mesh, we weren't able to achieve significant reductions because the simulation quickly becomes memory-bound.

A method that addresses the flow mismatch issue consists of measuring the flow rate at a small fixed pressure difference, but at various pressure offsets:

$$P_{inlet} = P_{offset} + \delta P \text{ and } P_{outlet} = P_{offset},$$

where $\delta P << P_offset$. Under these boundary conditions, the flow rate will be:

$$Q = U(P_{offset} + \delta P) - U(P_{offset}) \approx U'_P(P_{offset})\delta P.$$

In other words, we measure the derivative of $U(P)$, rather than $U^{-1}(P)$ as in the other method. In this way, the fitting procedure is:

1. Run a set of simulations with the P_{offset} parameter ranging from 0 to some P_{max} (e.g. 200 kPa).

2. Extract the Q vs P_{offset} curve as a set of (Q, P) data points.

3. Fit the polynomial approximation $\sum_{i=0}^{i=n} k_i x^i$ to the data points. Then, $\frac{1}{\delta P} \sum_{i=0}^{i=n} \dfrac{k_i}{i+1} x^{i+1}$ will be the desired polynomial approximation of the potential function.

Fitting of the C_{ch} can be done in the same way as before or one can extract the $V vs P$ curve and perform a linear fitting.

3.2.4 Channel's characteristics

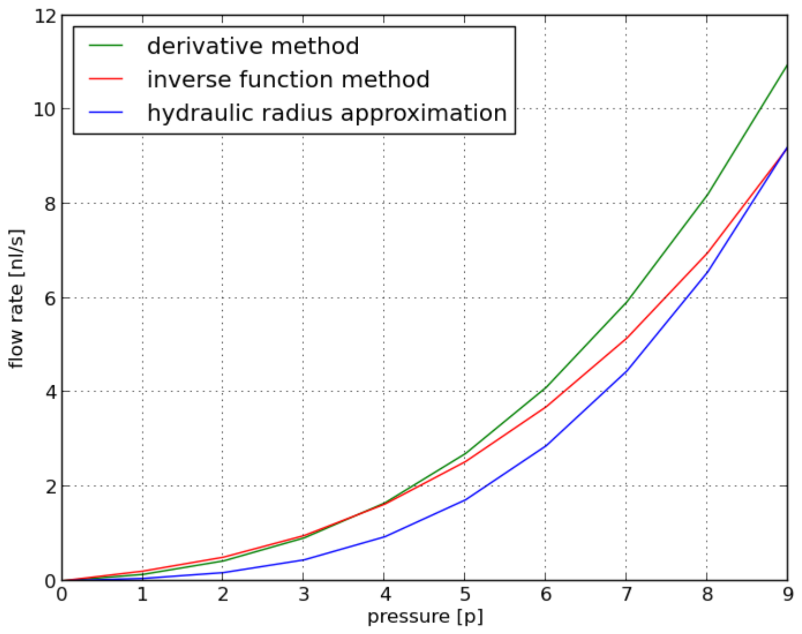


Figure 3.12: Potential function of a 100μ wide and 10μ tall rectangle flow channel fitted by three different methods. Inverse function and derivative methods both give similar results; the hydraulic radius approximation is pretty close too.

Besides the described fitting methods, we also used Poiseuille's law with an approximated hydraulic radius:

$$R_h = \frac{2A}{P},$$

where A and P are area and perimeter of the channel's cross-section. Resulting curves for the three methods are shown in Figure 3.12. Both fitting methods produce similar results, which are not far from the hydraulic radius approximation. The discrepancy between the two methods increases at higher pressure and/or lower shear moduli, i.e.

where the deformation of the cross-section becomes significant and the mass is no longer conserved in the simulation. The Poiseuille's approximation works better for tear drop shaped channels, since their cross-section is closer to a circle. Same goes for rectangular channels at higher pressures (see also Figure 3.11). [3]

For an $80\mu \times 10\mu$ (W x H) rectangular channel, a potential function looks like:

$$U(p) = 0.4\Big(\frac{p}{p_{max}}\Big)^3 + 1\Big(\frac{p}{p_{max}}\Big)^2 + 0.6\Big(\frac{p}{p_{max}}\Big)\Big[\frac{nl}{s}\Big],$$

where $p_{max} = 100kPa$. It is clear, that the function is monotone and is non-linear; the 2nd and the 3rd coefficients are significantly greater than zero.

Another interesting property of the potential function is its convexity. Consider the derivative $U'_P(P)$ of the potential function. Its physical meaning is the flow rate between pressures $P + dP$ and P. As we increase the pressure offset P, the cross-sectional area increases, so does the flow rate; therefore the $U''_P \geq 0$ and $U(P)$ is convex.

Figure 3.13 demonstrates the behaviour of the potential function for different shear moduli of the PDMS bulk. Lower shear modulus corresponds to softer material, and therefore a more non-linear potential function. The degree of the polynomial approximation can be up to 5 for lowest shear moduli. Rigid channels' have quadratic potential functions, or equivalently linear $U'_P(P)$, which is the small deformation case we mentioned in the introduction (see Figure 3.1). Unlike shear modulus, height and other parameters defined chip-wide (or region-wide, but at a cost), width can vary segment-wise, since masks are 2d drawings. This fact makes width a parameter that requires deeper investigation. Figure 3.14(a) depicts how $U(P)$ changes with the width W of the channel. When normalized to the maximum flow rate, potential functions don't differ significantly from each other (while the maximum flow itself varies). This fact gives rise to a question, whether it is possible to decouple the

[3]When we used a special hydraulic radius formula for the rectangular shaped channel, the mismatch between the Poiseuille's approximation and the simulation was less than 2%. Unfortunately, the approximation holds for pressures close to 0 only.

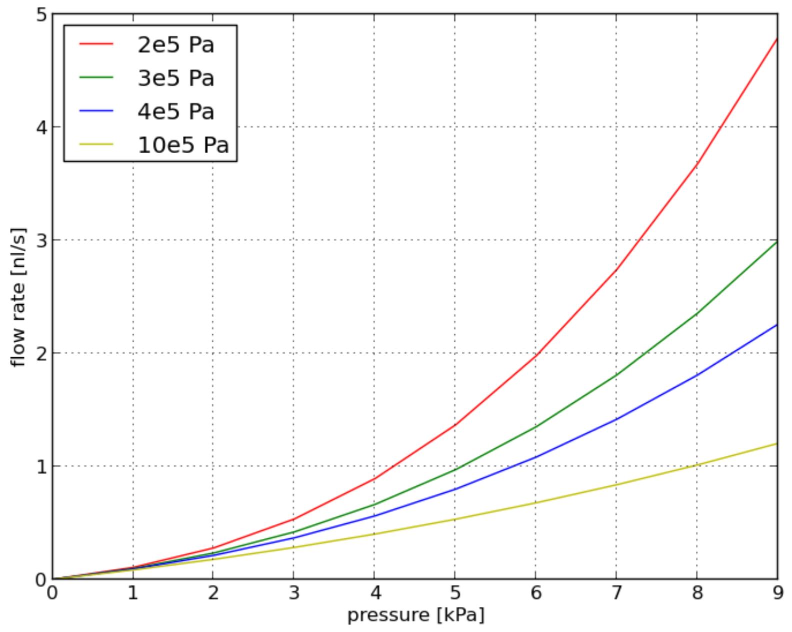


Figure 3.13: Potential functions for different values of the initial shear modulus. The higher is the modulus, the more rigid is the PDMS, the more linear is the potential function (absolutely rigid channels have linear potential functions).

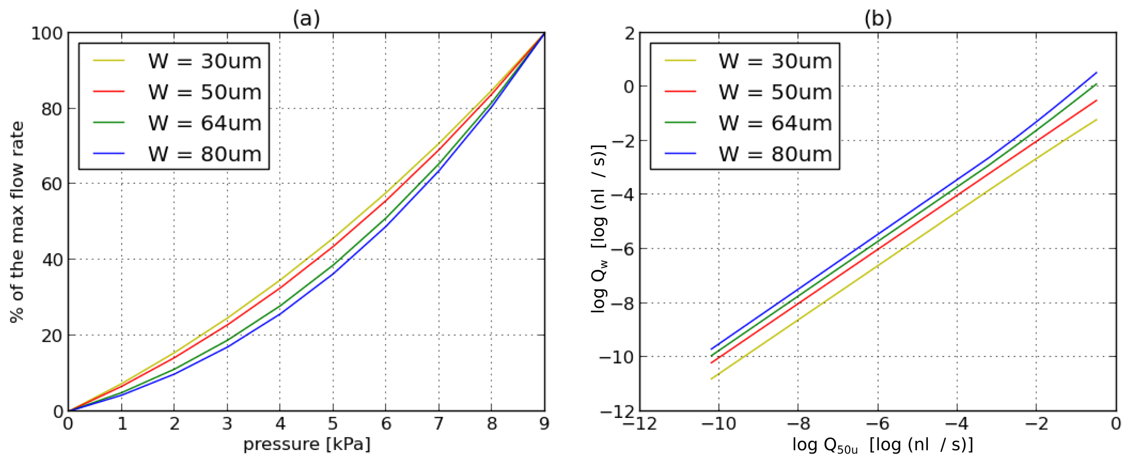


Figure 3.14: Normalized to the maximum flow rate potential functions for different channel widths (a) and a log-log plot of the 50μ function vs other potential functions (b). The log-log plot shows how $U(P)$ scales with the width.

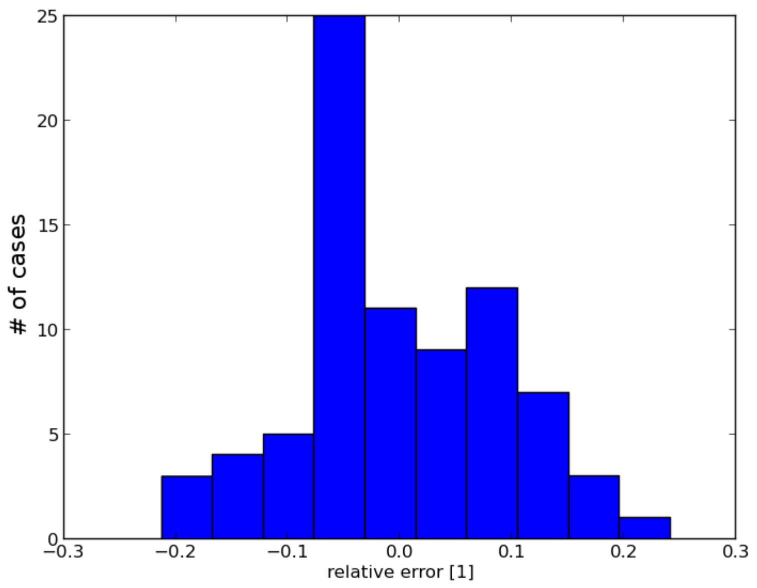


Figure 3.15: Plot of the errors resulting from using a single potential function ($U_{50\mu}$) approximation for all widths.

potential function from the width:

$$U(P,W) = \hat{U}(P)\hat{Q_{max}}(W),$$

where $Q_{max}(W)$ is the maximum flow rate. Using this form, we can rewrite the flow rate equation:

$$QR(W,L) = \Delta U,$$

where $R(W,L) = L/Q_{max}(W)$. This transformation allows us to model a network of channels of different widths via a set of linear equations by a simple change of coordinates $P \to U(P)$.

First, consider a plot of $\log(U_{50}(P))$ vs $\log(U_W(P))$ (see Figure 3.14(b)), where $W \neq 50\mu$. The straight lines on the plot can be expressed in the following way:

$$\log(U(P,W)) = k(w)\log(U_{50}(P)) + b(w),$$

where $k(w)$ and $b(w)$ are the slope and the offset of the corresponding lines. Fitting the curves showed that $k(w) = 1 \pm 0.03\%$ and $b(w) = \alpha \log(w)$, where α depends on other parameters of the channel. If we exponentiate both sides of the equality, we arrive at:

$$U(P, W) \approx U_{50}(P) w^{\alpha}.$$

In Figure 3.15, relative approximation errors are shown to be less than 25%. However, the distribution is skewed, suggesting there's a better choice of parameters. Besides α, we can also choose which $U_W(P)$ to use as the baseline potential function. Instead of selecting the W, we'll fit a $\hat{U}(P)$ function such that:

$$U(W, P) \approx \hat{U}(P) * w^{\alpha}.$$

First we sample the equality at some pressure set $\{P_i\}$ and take logarithms of the

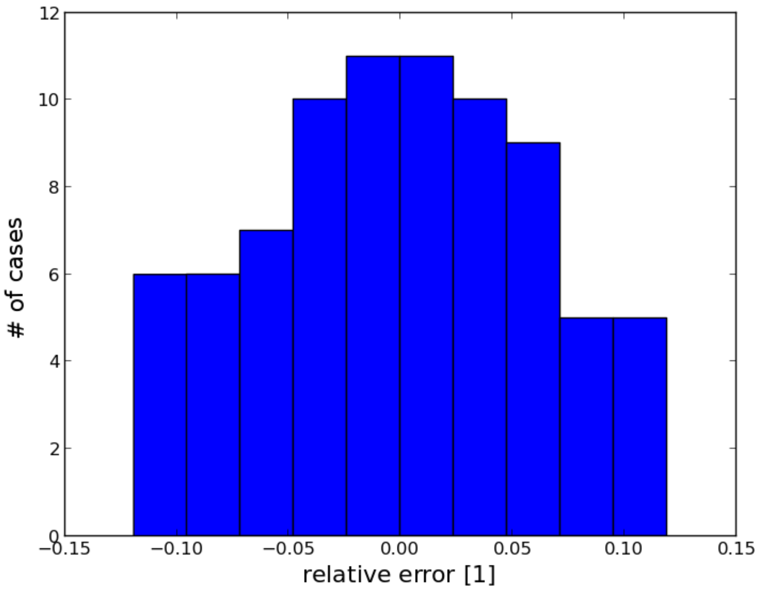


Figure 3.16: Histogram of the relative errors of the potential function approximation

both sides yielding:

$$log(U(W,P_i)) \approx \log \hat{U}(P_i) + \alpha \log(W).$$

The minimization of the error problem has a closed form solution, since the approximation is linear. The resulting error distribution shown in Figure 3.16 isn't skewed and the error is less than 11%.

This approximation has two important implications. Since we can approximately decouple P and W, the flow rate equation is transformed into:

$$\Delta U(P) = Q\frac{L}{W^{\alpha}} = QR(L,W),$$

where $U(P)$ is some base-line potential function. Therefore, for a network of channels with the same global properties (i.e. height, shear modulus, shape and type of channel), we can fully substitute P with $U(P)$ and treat it as a network of constant resistors under potential U.

Another important consequence of this approximation is that the expression of the resistance $R(L,W) = LW^{-\alpha}$ is a monomial. This fact enables us to use a geometric programming framework help optimize designs.

In contrast to the resistive characteristics, the capacitive ones are much simpler. A typical cross-section area vs pressure curve is shown in Figure 3.17. The near linearity of the curve (with a small exception at the origin) justifies the constant capacitor approximation in our compact model; since the area is linear in pressure, so is the volume V_{seg} of the cone section. The $V_P'(P)$ derivative equals to the product of segment's length and linear capacitance of the channel, i.e. the slope of the $S(P)$ curve.

The linear capacitance C_{ch} isn't sensitive to the height of the channel, since the side surfaces of channel have small area (compared to the top and bottom ones) and don't deform that much. The C_{ch} vs shear modulus is shown in Figure 3.17. Naturally, lower moduli result in higher capacitances due to higher deformations.

For the same reasons as in the resistive analysis, we're interested in the $C(w)$ function.

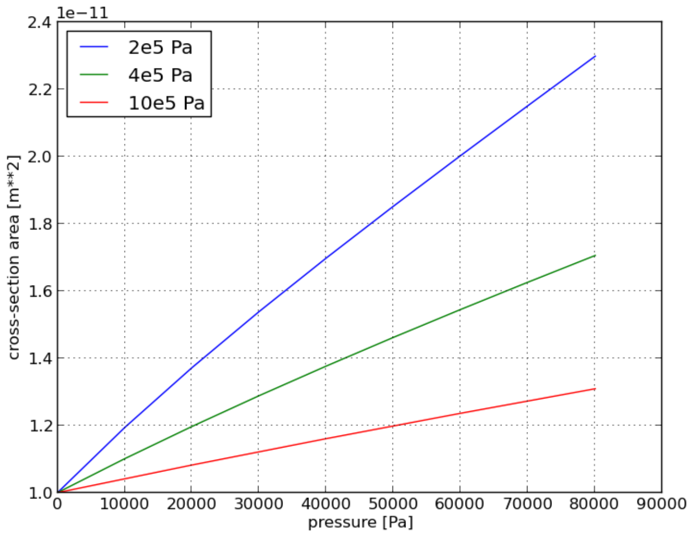


Figure 3.17: Cross-section area vs pressure vs shear modulus family of curves. More rigid channels deform less and therefore have lower area vs pressure curves.

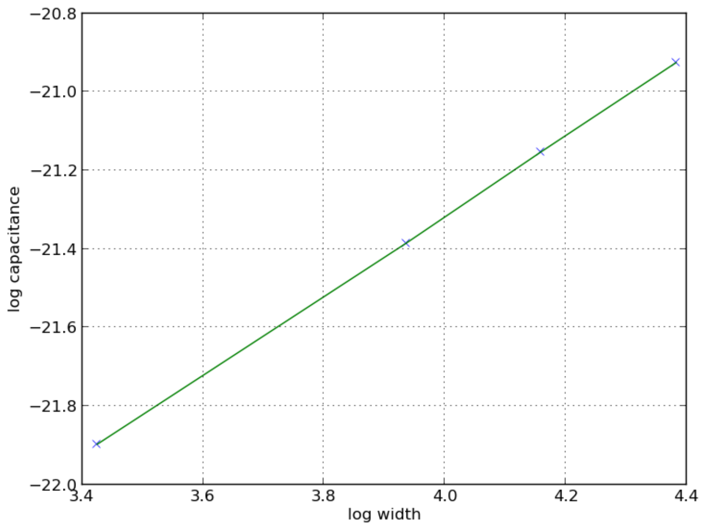


Figure 3.18: The log capacitance is linear in log width, therefore capacitance is a power function of the channel's width

The $\log C$ vs $\log w$ in Figure 3.18 is a straight line, therefore $C(w) = C_0 w^{\beta}$, where β is a characteristic of the channel. The $C_0 w^{\beta}$ expression is a monomial and so is the expression of the timing constant $\tau(w, l) = RC = w^{\beta-\alpha}L$. We will use this equation to model timing in the RC-networks using the geometric programming framework.

3.2.5 Transient analysis

So far, we have talked about modelling channels in a stationary regime, i.e. where pressure and flow rates do not change in time. While being very important for flow channels, it is not sufficient for control channels. These channels have blocked ends, so have only trivial stationary regimes, constant pressure and no flow. For these channels, we are interested in the transient response, i.e. when pressure and flow rates change in time. In this section, we'll validate the derived compact model of the channel using COMSOL, then analyze and simplify the model.

As it was mentioned earlier, the equivalent circuit of a channel is a series of Π-sections (see Figure 3.4). The capacitor is the ingredient that enables transient modelling (w/o capacitors, all changes are immediate).There are still a few questions we need to answer for our compact model. First, we need to determine the number of sections sufficient to accurately model timing. Being irrelevant for stationary regimes, the number of sections defines the order of the volume approximation (see Figure 3.3). The other question is whether an RC network is a sufficient model for transient behaviour? Capacitors model the ability of the channel to accumulate volume, but there are other physical effects, like the inertia of the fluid[4], that may influence the channel's transient behaviour.

To answer these questions, we simulated switching of a long control channel using both FEM and our compact model. A natural choice for circuit simulation is any tool from the SPICE family[5]. In the SPICE language, the model of the channel took

[4]In particular, fluid's inertia prohibits instantaneous changes in flow rate. An appropriate way of modelling inertia is adding a inductance.

[5]In our research, we used an open source simulator called ng-spice.

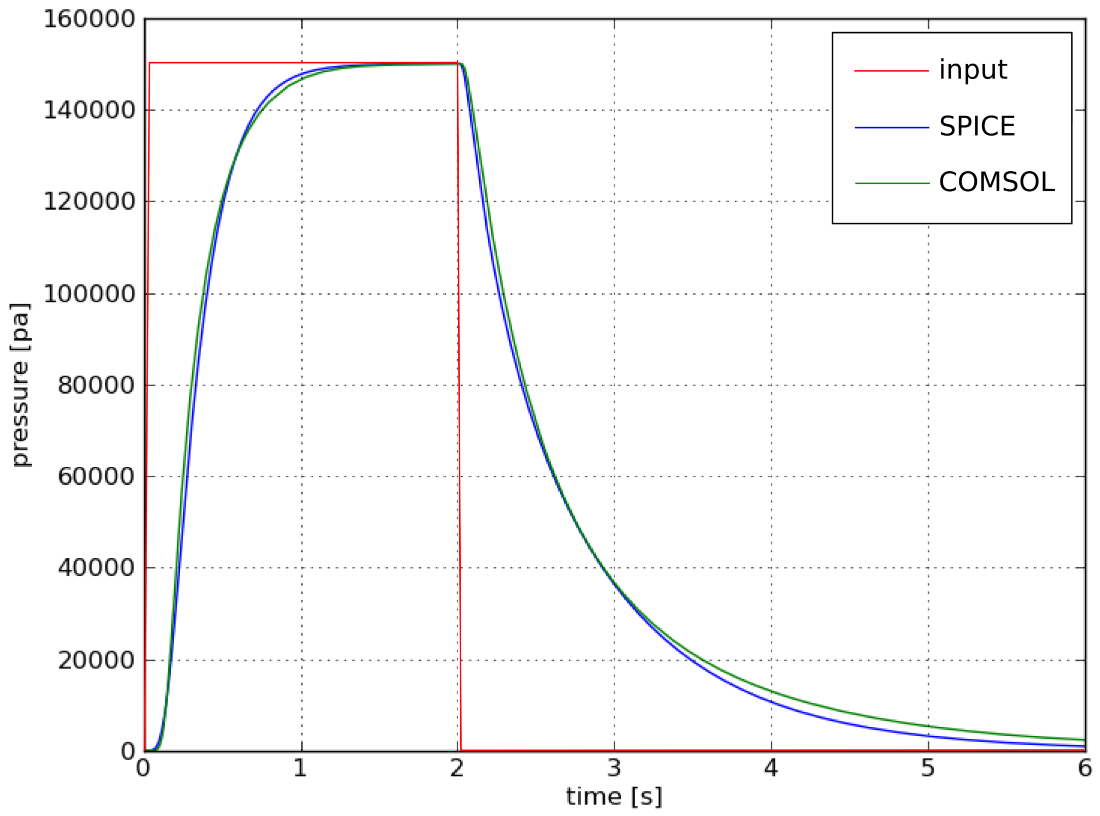


Figure 3.19: Pressure response to a square pulse (red) of the finite element and compact models of the channel simulated in COMSOL and SPICE respectively. The inlet pressure was almost instantly ($\Delta t = 10^{-3}s$) turned on at $t = 0s$ and turned off at $t = 2s$. The relative error between the curves is less than 2%. One of the prominent features of the transient behaviour is charge-discharge asymmetry.

a dozen of lines of code. Although, the simulation require solving non-linear differential equations, the potential function form of the non-linear part resulted in a small run time; typically less than 10 seconds for a single channel, while it took hours for COMSOL to converge.

The simulation set up consisted of the same long channel (see Figure 3.5) having one inlet and sealed at the end. In the equivalent circuit, the inputs were connected to the voltage source and the output was open. We have also chosen a large number of Π-sections (≥ 50). In the experiment, we studied the response of the channel to switching the pressure on and then, after the flow stops, switching it off. We measured the pressure at the sealed end as the function of time. The resulting curves (see Figure 3.19) show a very small mismatch (below 2%) between the models. The result implies the absence of the inertia effect[6] as well as validates the constant capacitance approximations for large number of segments.

An interesting effect revealed by the simulation is the charge-discharge asymmetry;

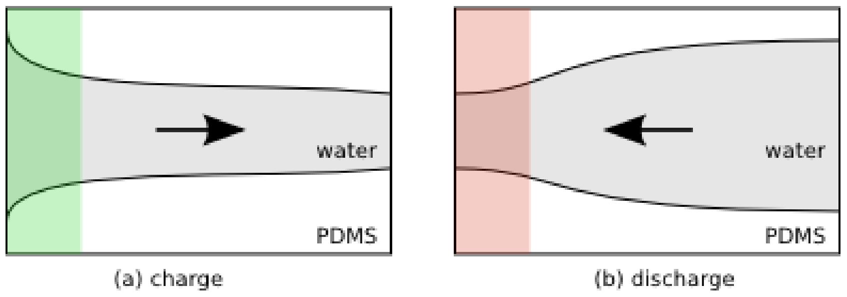


Figure 3.20: Schematic explanation of the charge-discharge asymmetry. The channel charges through a low-resistance segment (green, a) and discharges through high-resistance segment (red, b).

the process of pressurizing the channel to some high pressure takes significantly smaller amount of time than the depressurizing process. This effect is caused by the non-linearity of the resistor. Moments (about 1/20 of the charging time) after the inlet pressure is turned on, the sealed end is still unpressurized, but the region close to the inlet is already deformed and has reduced resistance (see Figure 3.20). The region of lower resistance is increasing as more fluid is coming into the channel. Therefore the charging of the channel happens through a constantly decreasing resistance.

An opposite behaviour is observed in a depressurizing channel see Figure 3.20). The first region to shrink after the pressure is turned off is the inlet section. The fluid has to escape through an increasing resistance of the inlet, making the discharge time

[6]In cases of very small ramp time $< 10^{-5}$, we've seen pressure oscillations that can't be explained by RC network alone. However, both their amplitude and duration were insignificant compared to the maximum pressure and charging time respectively. Finally, such switching times are also hard to achieve in real settings.

higher than the charge time.

Although the simulation of a single channel was pretty fast, the runtime increases

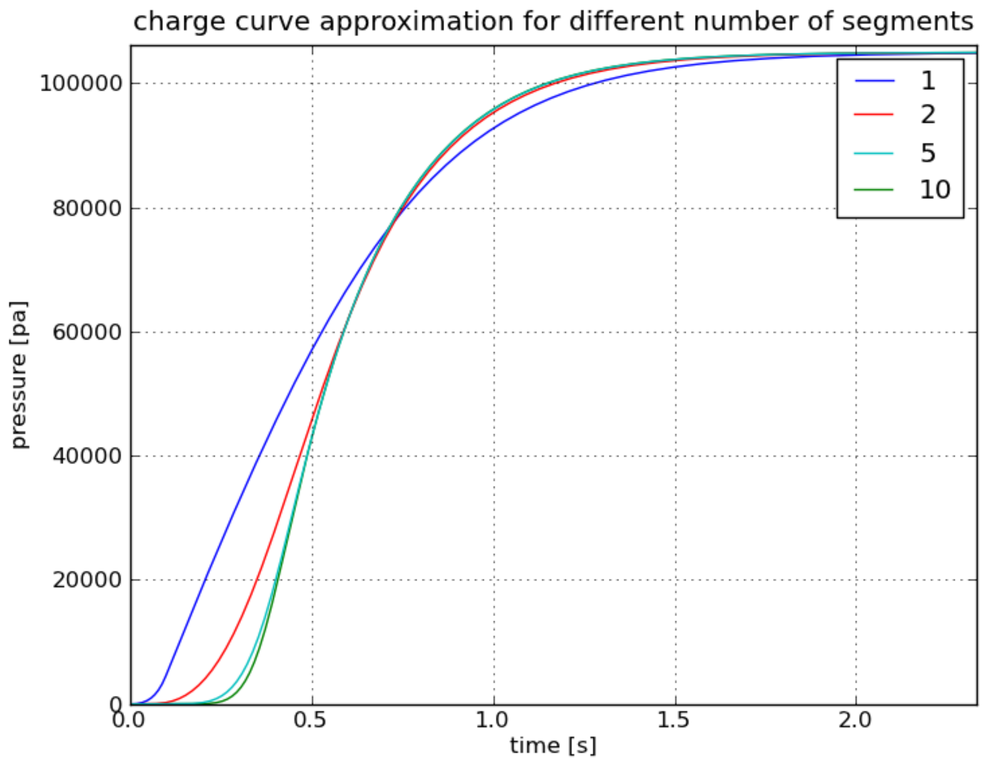


Figure 3.21: Simulated charge curves of a control channel for different number of Π-sections. While 10 sections are sufficient for accurate modelling, 2 sections provide a good approximation and even 1 sectioned model isn't terribly off.

dramatically with the number of channels in case of a microfluidic network simulation. To establish the dependence between the approximation error and the number of segments, we ran a series of simulations varying this number.

Figures 3.21 and 3.22 show the effect of the reduction of the number of segments on the model's quality. For the switch-on curve, 10 sections are as accurate as 50 sections. A 5 section model introduces very little error, on the order of 5%. While a single section model gives a significant mismatch in the first half of the curve, it is pretty adequate in the second half. The main difference between the curves is the

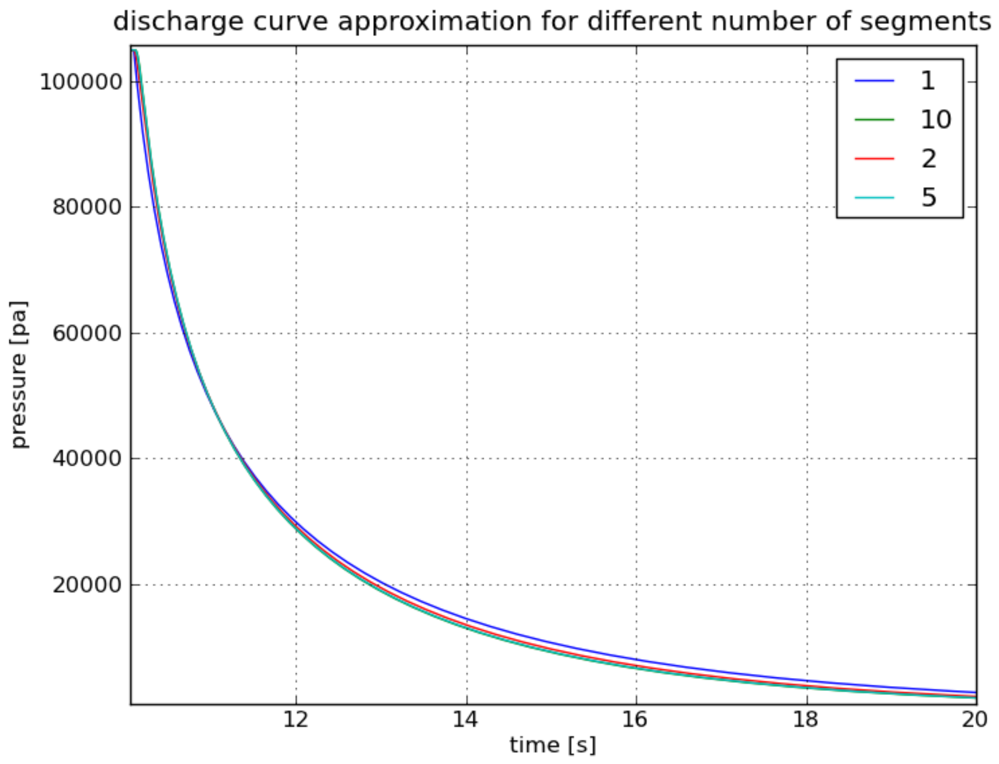


Figure 3.22: Simulated discharge curves of a control channel for different number of Π-sections. Single sectioned model is almost as accurate as the 10-sectioned one.

length of the initial zero pressure region, where all the flow charges the intermediate capacitors, not the last one.

The switch-off experiment (see Figure 3.22) demonstrates negligible difference between models with different number of sections. The reason for that is the discharge process is much less abrupt than the switch-on, therefore it doesn't require as much distributed capacitors for accurate modelling.

While we know that a 1cm long tube doesn't require more than 10 section model, the question is, how does this number scale with the length of the channel. The answer to the question is it doesn't; given the number of section, the relative mismatch will be the same regardless of the length. This property is a result of the fact that potential function scales linearly with channels length, therefore the cone section approximation of the channel's volume will have a constant relative error.

In practice, this fact means that as long as the operating time scale is bigger than the

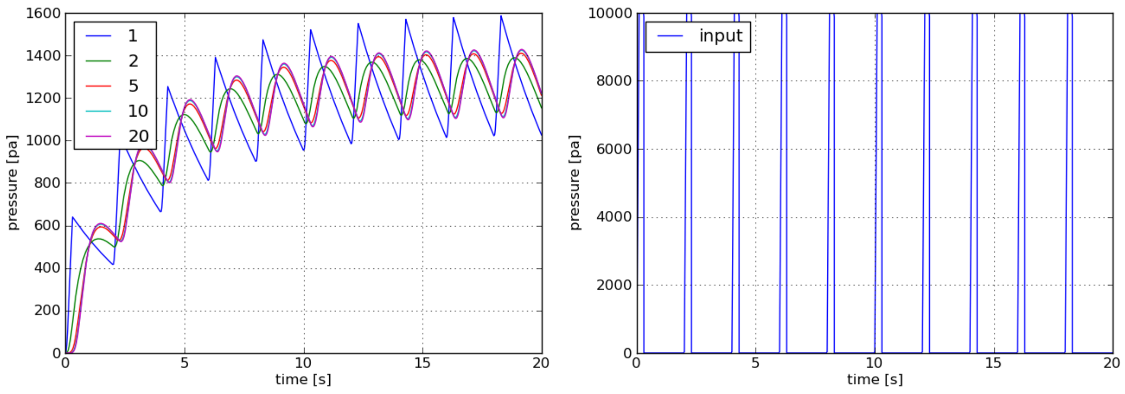


Figure 3.23: A large mismatch between a 1-section and 10-section models (left) induced by a fast modulated pulse input (right). The error is caused by a poor accuracy of the single section model at the moments after the pressure is turned on.

charge-discharge time constants, it's safe to use a 2 or even a single section model. However, due to the limited accuracy of these extremely reduced models at t's close to zero (see Figure 3.21), there are ways of constructing a pathological input signal that results in a large mismatch. Such a signal is shown in Figure 3.23, where a fast and asymmetric pulse causes significant mismatch between a single sectioned and a full model. A circuit with 5 sections, however, produces very accurate results, which is a very typical for all input signals and channel configurations we've tested.

So far we have learned how to build accurate channel's models for both stationary and transient regimes. In some cases, however, we don't need to know the exact $p(t)$ but only some of its properties. One such case is timing analysis. After a pressure is turned on, some amount of time is required for pressure to propagate through the control channel to a valve and valve doesn't seal the flow channel until it's pressurized at some p_{seal} pressure. Let's call the time it takes the channel to deliver the required pressure T_{close}. Similarly, T_{open} is an amount of time the valve is still closed after the pressure is released. Turns out, to get these timing numbers, you don't even need to run the circuit simulation. You can estimate these delays from the RC time constants of a linear RC network.

One can understand why the linear model works through a simple dimensional anal-

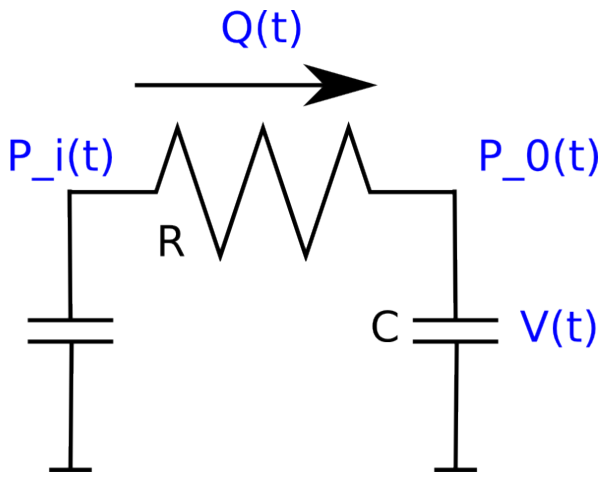


Figure 3.24: A single Π-section model of a control channel. R and C are its resistance and capacitance respectively. $Q(t)$ is the current flow through the channel and $V(t)$ its current volume. $P_i(t)$ and $P_o(t)$ are the inlet and sealed end pressures respectively.

ysis of the problem. Consider a single section model of a channel, consisting of a resistor and a capacitor (see Figure 3.24). Kirchhoff's laws applied to the flow circuit yield:

$$\begin{aligned} Q(t)R = \dot{V}(t)R &= U(P_o(t)) - U(P_i(t)), \\ V &= CP_o, \end{aligned}$$

where V is volume of the channel, $Q = \dot{V}$ is the flow rate, a time derivative of the volume function, P_i and P_o are inlet and sealed end pressures respectively, and C and R are channel's capacitance and resistance. When combined, the two equations yield:

$$RC\frac{dP_o(t)}{dt} = U(P_o(t)) - U(P_i(t)).$$

Since we're interested in modelling switching responses, $P_i(t)$ is the Heaviside function and so is $U(P_i(t))$. Suppose, we know the solution $P^*(t)$ to this differential equation. What happens if we time scale it by a factor of γ? Its derivative will be:

$$\frac{dP^*(\gamma t)}{dt} = \gamma\frac{dP^*(\gamma t)}{d(\gamma t)},$$

but since P^* is a solution,

$$\frac{dP^*(\gamma t)}{dt} = \gamma\frac{dP^*(\gamma t)}{d(\gamma t)} = \gamma\frac{1}{RC}(U(P^*(\gamma t)) - U(P_i(\gamma t))),$$

where $U(P_i(\gamma t)$ is still a Heaviside function, because it's invariant to time stretching. Putting it all together, we arrive at:

$$\frac{\gamma}{RC}\frac{dP^*(\gamma t)}{dt} = U(P^*(\gamma t)) - U(P_i(\gamma t)),$$

i.e. $P^*(\gamma t)$ is a solution to the original problem, where RC is scaled by a factor of $1/\gamma$. Finally, because the $P^*(t)$ is a monotone function, the threshold hitting times T_{close} and T_{open} are both linear in RC. Moreover, the same argument can be extended [28] to a series of Π-sections, meaning that we can use a linear resistor (with an appropriately chosen resistance) to model the switching process in a control channel[7].

For an arbitrary network of channels, however, the time scaling approach becomes

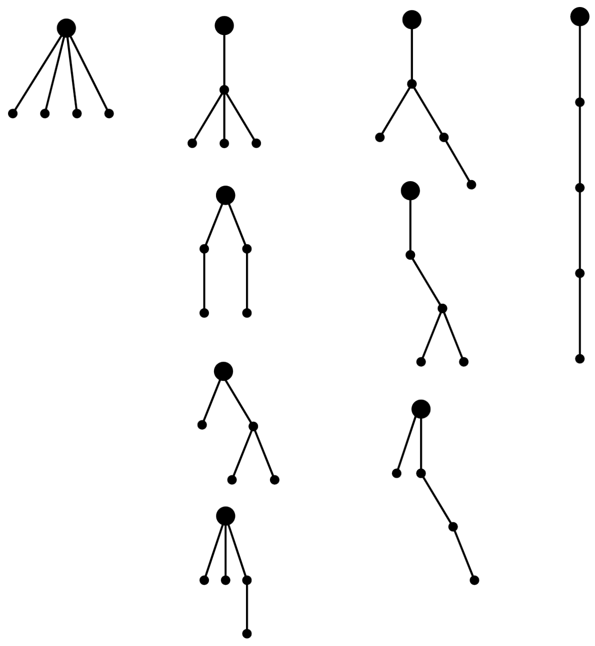


Figure 3.25: All possible topologies of a 4-edged tree with a dedicated inlet node (big dot). They include (left to right) one single tier, four 2-tier, three 3-tier and one 4-tier tree.

an approximation (so called single time constant approximation). To investigate the approximation error, we generated all possible control tree structures with a fixed number of segments (see Figure 3.25 for an example of a 4 edged tree). Then we simulated switching on and off processes for both compact and linearized compact models of all tree topologies.

The resulting histogram of relative mismatches is shown in Figure 3.26. For the vast majority of cases, the linearization error is less than 5%, and more than half the cases have about 1% mismatch. The maximum error is, however, large – about 50%. Such errors occur in significantly skewed trees (see Figure 3.27). The explanation of the behaviour is similar to that of charge-discharge asymmetry; the smaller branch

[7]The values of R and C would be different for the positive and negative edges due to charge-discharge asymmetry.

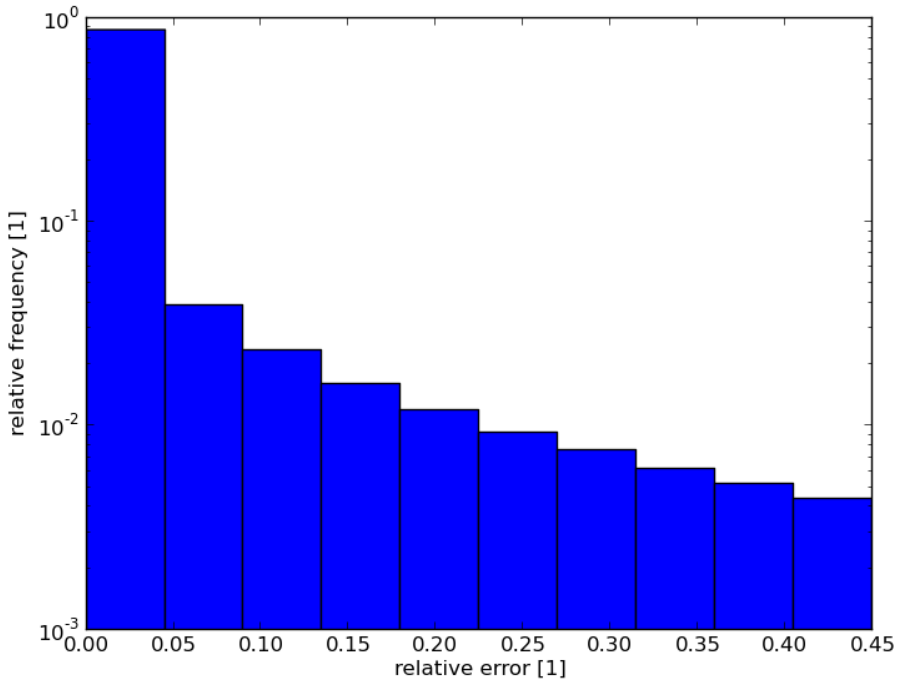


Figure 3.26: Histogram (on a log scale) of relative mismatches between non-linear and linearized models of all possible tree structures of length 8. Most of the cases have $< 5\%$ linearization error. The maximum error is however as much as 50%.

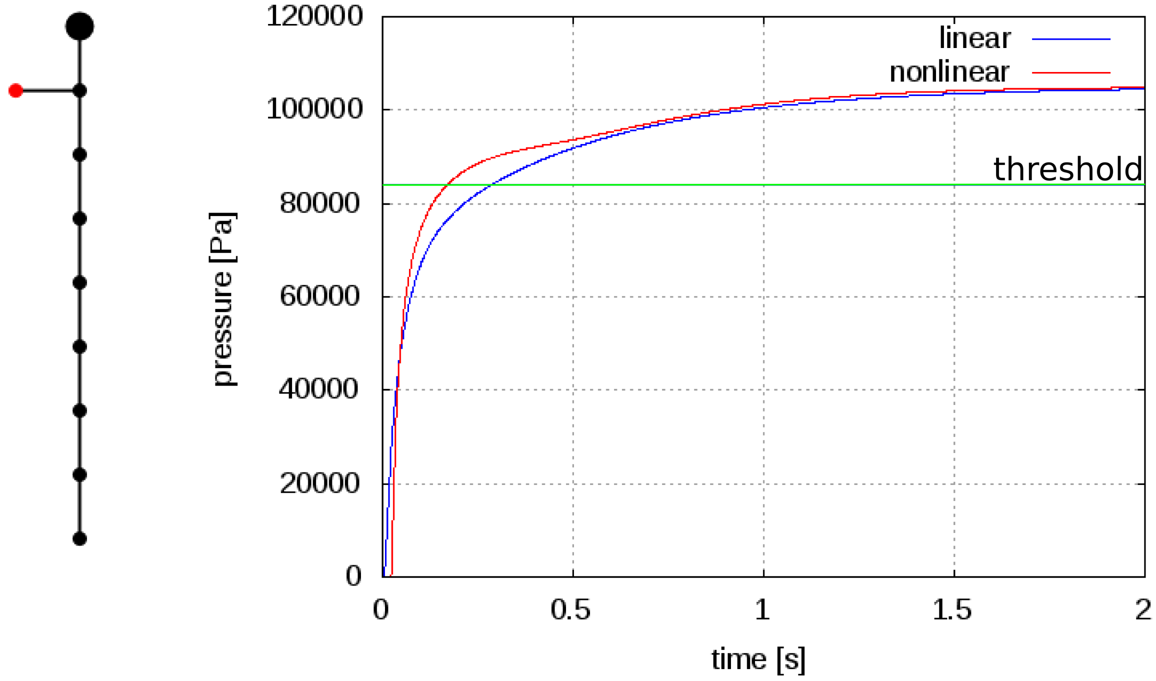


Figure 3.27: A skewed (or unbalanced) tree (left) and its charging curves (right) simulated with non-linear and linearized models. The charging curves are pressure vs time measured at the output shown by the red dot. The non-linear compact model shows 2x smaller threshold hitting time (blue line).

has lower resistance and gets charged first, but the channel's expansion amplifies the effect.

Several aspects of the microfluidic design, however, can mitigate the severity of the maximum mismatch. First, the linearized model always overestimates the rise time, i.e. the timing is conservative. Next, we're typically interested in state switching times, i.e. max T_i over all closing (or opening) times T_i. Linearization doesn't introduce significant ($> 5\%$) errors to this timing constant. Finally, such unbalanced control structures are rarely used and can be avoided by various balancing design techniques, such by moving the inlet closer to the tree's center.

A channel is the most used device in microfluidics (some chips even consists of channels only). In this section, we learned a number of approaches to the analysis of the channel. We characterized the channel using finite element modelling, we constructed a very simple, fast and accurate circuit-based compact model. We reduced the model even further for simple and insightful analysis of networks of channels; flow networks can be modelled via a set of linear equations and timing of the control trees can be modelled analytically. In the next section, we'll develop a compact model for valves, which will allow us to analyze and synthesize complex microfluidic structures up to whole chips.

3.3 Valve

In a general sense, a valve is a microfluidic device that controls the flow through channels. The control ranges from altering the flow rate to sealing the channel completely. While there are many types of microfluidic valves[8], we'll focus on push-down valves only, since it shares the same basic principle with the others.

A push-down valve is a structure that occurs on an intersection[9] between a tear-drop shaped flow channel and a control channel (see Figure 3.28). It is called push-down, since the excess of the control pressure pushes the ceiling of the flow channel against

[8]Although we leave hybrid devices, like electro-fluidic valves outside the scope, the equivalent circuit modelling approach that we use is applicable to them as well.

[9]one channel goes above or below the other one.

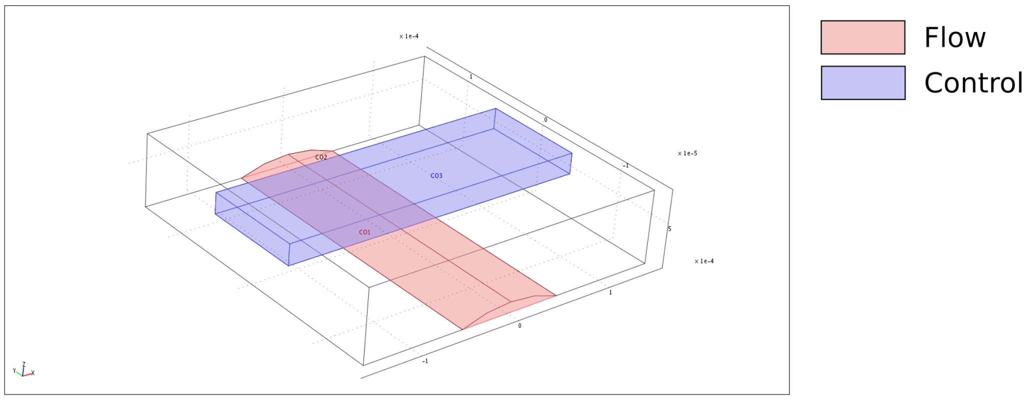


Figure 3.28: A push-down valve as an intersection between a flow (red) and a control (blue) channels. The flow channel has a tear-drop shape and is below the control line. A reversed channel order results in a push-up valve, where the floor of the flow line is pushed towards its ceiling by the control pressure.

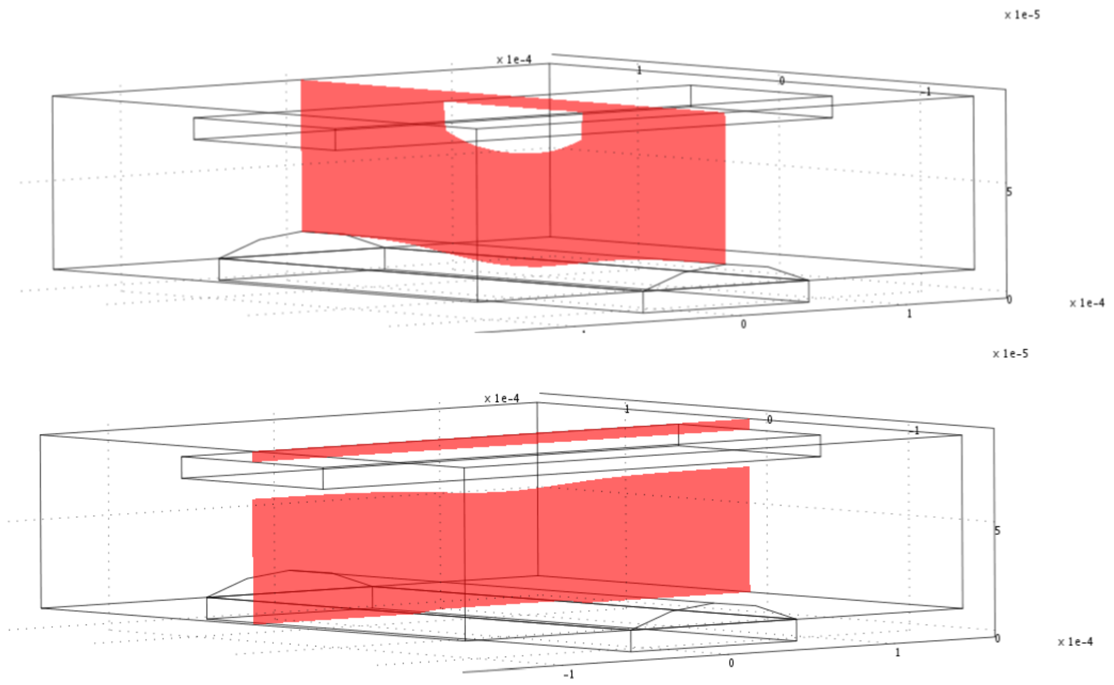


Figure 3.29: Slices of a pressurized push-down valve along X (top) and Y (bottom) axis. The flow channel is almost completely blocked. Maximum deformation of the channel is achieved closer to the center of the control channel.

its floor either partially obstructing or completely blocking the flow (see Figure 3.29). Since a valve is basically a segment of a channel with adjustable (and controllable) cross-section, we can use the same modelling approach we used for channels; again, we started with COMSOL-based characterization. The simulation set up didn't deviate from that of the channel, except for few details. First, we didn't model long channels, since the valve's dimensions are bounded unlike the channel, which can be arbitrary long. Therefore, the chunk of PDMS was only $300\mu \times 300\mu$ for a $100\mu \times 100\mu$ valve. Similarly, the chunk wasn't as tall as in the channel's model for the main process happens between the channels.

Next, we used a generic tetrahedral mesh instead of a simplified swept one, since the geometry of the chunk isn't extruded (it loses the property when we add an orthogonal control channel to the existing flow one). Although this resulted in a more complex mesh, the simulation speed wasn't significantly compromised due to a smaller overall size of the model.

Finally, in addition to all the constraints from the channel's model, we introduced the contact pair constraint between the surfaces of the flow channel, prohibiting the propagation of the channel's ceiling beyond its floor. While making the model consistent, the constraint has its toll on the simulation speed; the larger is the contact area (see Figure 3.30), the longer it takes to converge. In practice, it resulted in an exponential slowdown.

Another issue with the contact pairs was inaccuracy at higher pressures caused by the dead volume near the edges of the flow channel. This side effect was responsible, for example, for non-monotonicity of the flow rate vs control pressure curve (see Figure 3.32). As the result, there was no use in running the simulation beyond a certain control pressure threshold. In practice, the stopping criterion was the 100x drop in the flow rate.

The equivalent circuit of the valve's compact model is shown in Figure 3.31. As in the channel's model, the circuit reflects both conductive and capacitive phenomena, modelled by capacitors and a resistor respectively. The former effect is the drop of the valve's conductance caused by the increase of the control pressure. The capacitive

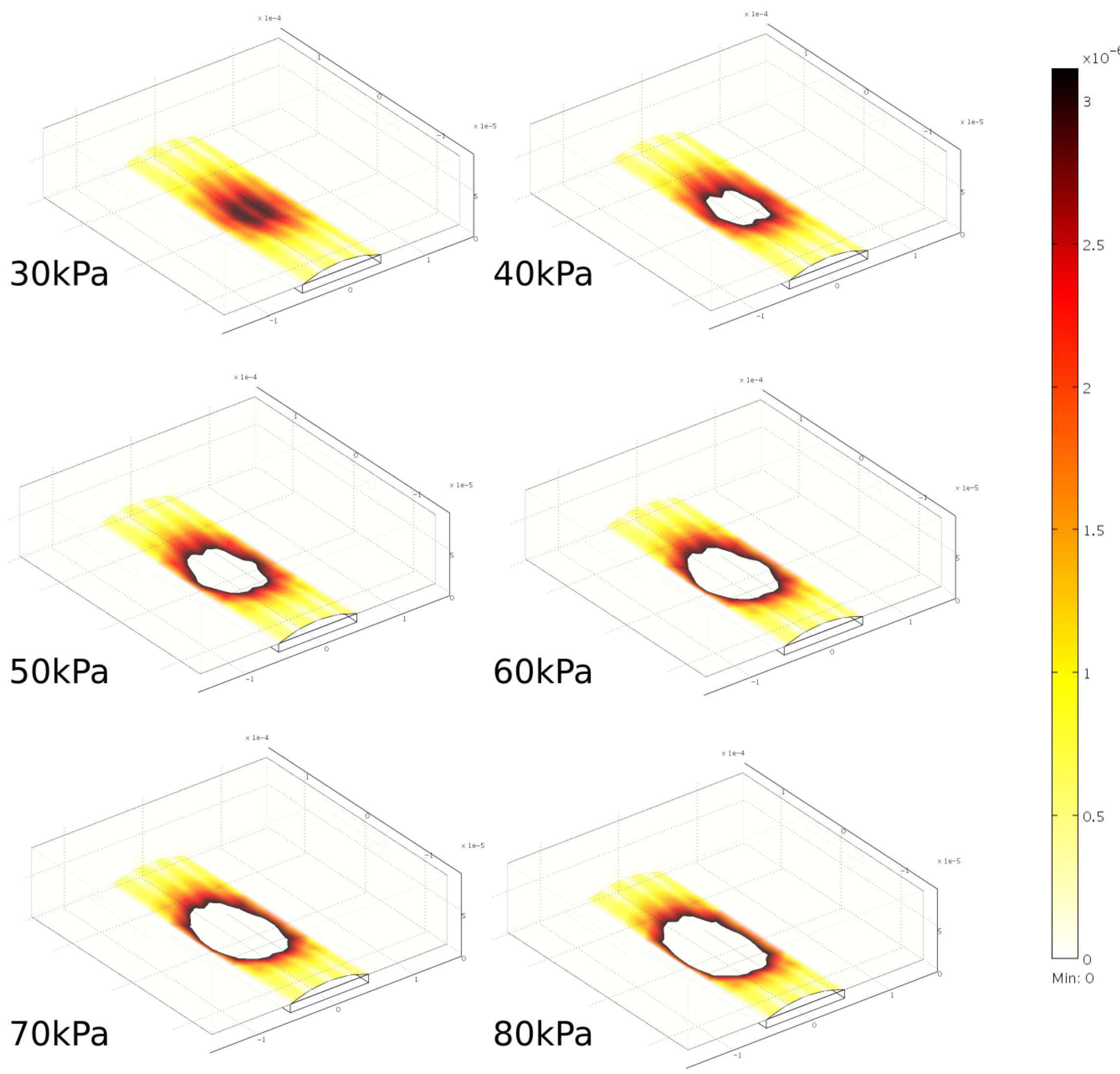


Figure 3.30: 3D displacement map of a 55μ-thick valve for a range of control pressures. At $30kPa$, the membrane is significantly deformed, however, the ceiling doesn't touch the floor of the flow channel yet. At $40kPa$ the contact has already occurred, the white region depicts the contact area. Next, the contact area expands towards the sides of the channel, and causes its complete obstruction at the threshold pressure of $60kPa$. After $60kPa$ the expansion of the contact area is aligned with the flow channel direction. It doesn't cause any changes in valve's resistance, but still affects its volume.

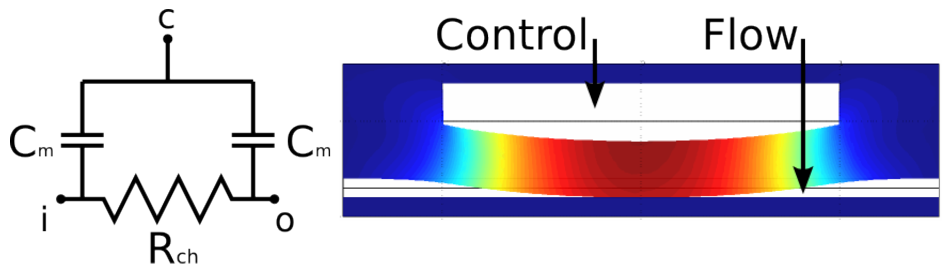


Figure 3.31: Equivalent circuit of a valve (left) and the valve's cross-section (right). The circuit consists of two non-linear capacitors, representing the parts of the membrane, and a resistor, controlled by the pressure at point c. The flow channel is connected to terminals i and o, and the control line is connected to c.

effect is the volume displacement induced by the excess of the pressure.

Clearly, both effects are non-linear; while increasing the control pressure, the flow stops at some point, so does the expansion of the valve's membrane. Therefore, both the resistance R_{ch} and the capacitance C_m are function of the pressure P_c.

At higher control pressures, the resistance is effectively infinite, and the flow part of the valve is divided by the membrane wall into two independent parts (see Figure 3.31, right). The displaced volume therefore can be different, depending of the pressures P_i and P_o. To model this effect, we introduced two capacitors that share the charge at lower control pressures (i.e. when R_{ch} is a short) and are decoupled at higher pressures (when R_{ch} breaks the circuit).

In this way, the compact model of a valve is defined by the $R_{ch}(P_c)$ and $C_m(P_c)$ functions. To characterize a valve, we simulated the valve varying the control pressure. For each pressure value, we extracted the hydraulic conductance of the flow channel as well as its volume.

Figure 3.32 shows the cross-section area in the middle of the valve (i.e. in its narrowest part) vs the control pressure. It can be partitioned into three nearly linear regions. In the green one, the membrane expands inside the flow channel, but doesn't touch its floor. In the yellow region, there's a contact between the channel's surfaces that blocks the flow completely in the red region.

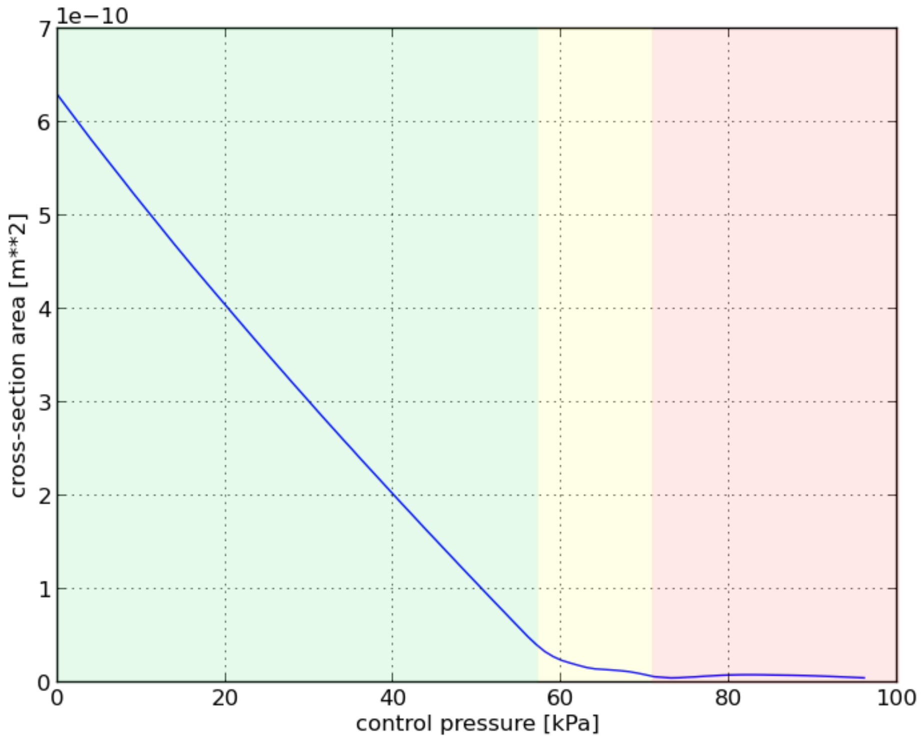


Figure 3.32: Cross-section area of the narrowest part of the flow channel vs control pressure of the valve. The curve shows three distinct linear regimes; no contact between the floor and the ceiling (green), partial contact (yellow) and full contact (red), i.e. full flow obstruction.

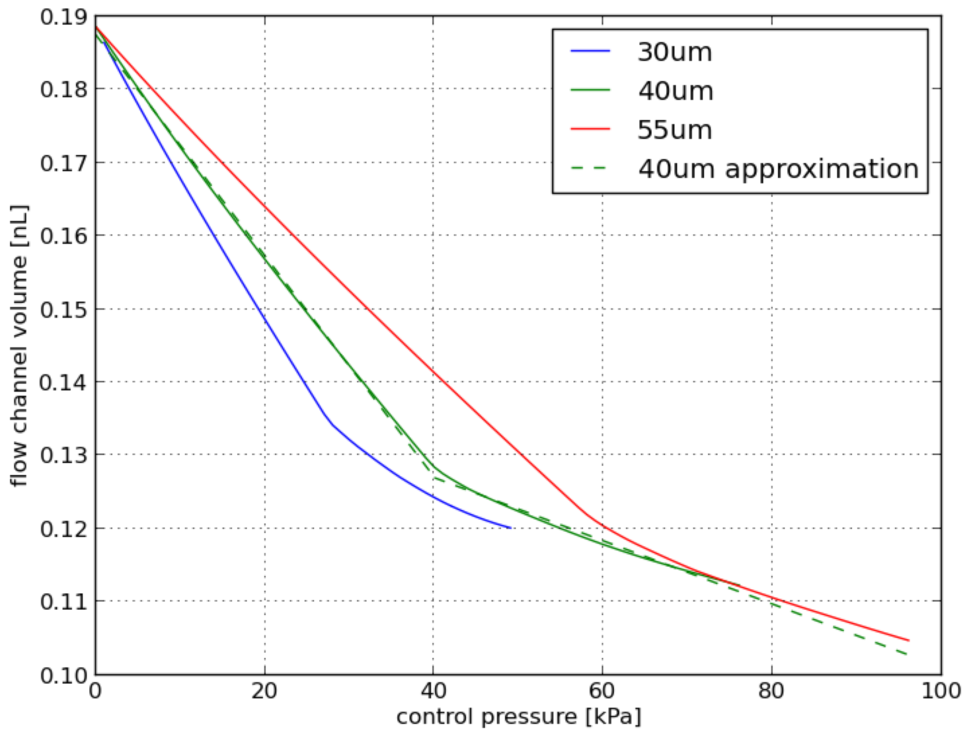


Figure 3.33: Volume of the flow channel under a valve vs control pressure for membranes of different thickness. The dependence is almost piece-wise linear with two linear segments. The increase in the membrane's thickness moves the knee point to the right. At higher pressures, however, the curves get close to each other, since the valve closes and the expansion of the membrane doesn't depend on its thickness.

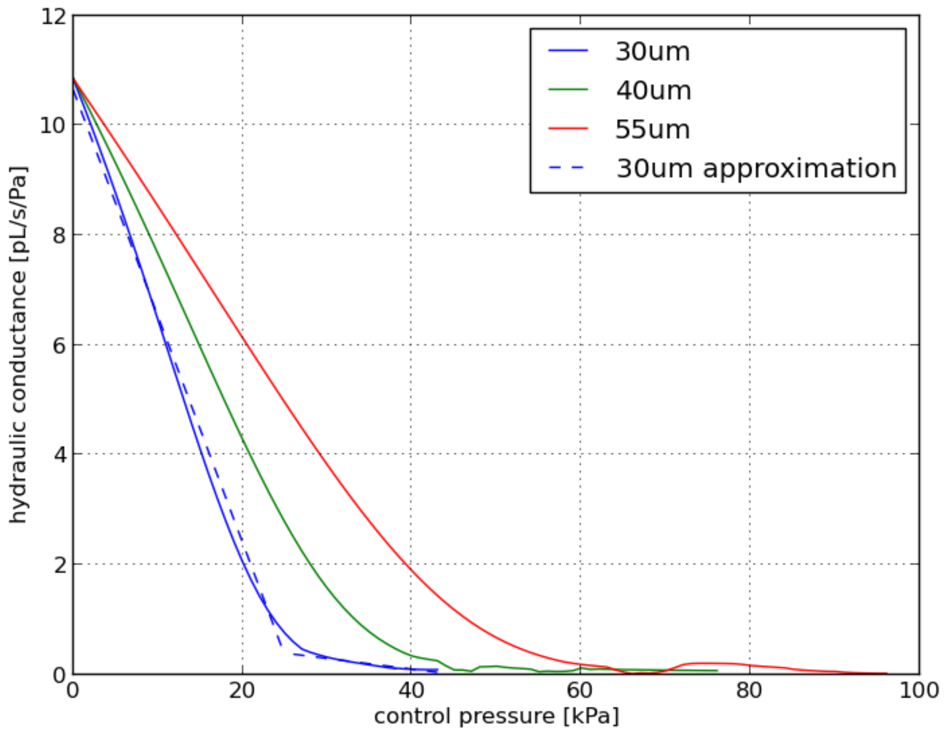


Figure 3.34: Channel's hydraulic conductance vs control pressure for membranes of different thickness. The curve is well-approximated by a 3 stage piece-wise linear function (blue dotted line). Thicker membrane makes it harder to seal the valve, scaling the curve to the right.

The hydraulic conductance vs control pressure curve shown in Figure 3.34 is consistent with the cross-section one. It also has three regions corresponding to the similar pressure ranges. As the result, the $(R_{ch}(P_c))^{-1} = G_{ch}(P_c)$ can be approximated with a piece-wise linear function (see a blue dotted line in Figure 3.34):

$$G(P) = \alpha\Big[1 - \frac{P}{P_0}\Big]_+ + \beta\Big[1 - \frac{P}{P_1}\Big]_+,$$

where

$$[x]_+ = \begin{cases} 0, \text{ if } x \leq 0 \\ x, \text{ otherwise} \end{cases},$$

P_0 is the pressure of the contact (1st knee of the curve) and P_1 is the pressure at which the channel is blocked (when the curve hits x axis).

Figure 3.34 shows the dependence of the hydraulic conductance curve on the thickness of the membrane (i.e. the distance from the ceiling of the flow channel and the floor of the control channel). Thicker membrane requires more control pressure for the same displacement, therefore the 55μ curve is strictly to the right of the 40μ and 30μ ones. Accordingly, the cut-off pressure is higher for valves with thicker membranes.

In Figure 3.33, the volume of the flow channel is drawn against the control pressure of the valve. Similar to the conductance curve, the dependence can be approximated by a piece-wise linear function with two regions. In the first one, the membrane goes all the way to the floor of the flow channel, and expands to its sides (see Figure 3.30, $30 - 60kPa$). In the second region, the contact area reached the sides, it expands in one direction only, reducing the slope of the $v(p)$ curve (see Figure 3.30, $60 - 90kPa$). Although not shown on the plot, at higher pressures the curve will converge to a value, corresponding to the full displacement of the valve's volume. The capacitance of the membrane, a derivative $v'(p)$ is, therefore, a piecewise constant function:

$$C_m(P) = \begin{cases} C_0, \text{ if } P \leq P_m \\ C_1, \text{ otherwise} \end{cases},$$

where P_m is the control pressure at which the contact area reaches the sides of the flow channel, and C_0 and C_1 are the slopes of the function in the first and the second regions respectively.

The $v(p)$ dependence on the membrane thickness is also shown in Figure 3.33. Similar to the $g(p)$ curves, the $v(p)$ functions for 55μ are to the right of the others. The knee point moves down and right as the thickness increases; by the time the contact area expanded to the sides of the channel, more volume is displaced by the valve.

So far, we learned how to construct compact models of microfluidic channels and valves derived from a detailed FEM simulation. Having an electrical circuit form, these models can be easily combined into networks to model a real chip. The approach, however, has a number of potentially weak points; first, the derivation was based on a set of assumptions that might or might not hold in reality. And secondly, the charge-based model of the interactions between the basic devices can be insufficient. The compact modelling approach, therefore, requires an experimental validation in real life settings.

3.4 Validation

To validate our compact modelling approach, we used two sets of experiments. In each set, we measured the pressure or flow rate response of a chip to pressure stimuli and compared to the response of the corresponding model. We assess the simulation error as the difference between the two responses. Although we didn't aim at an absolutely correct model, the resulting error didn't exceed 15% in both sets of experiments.

In the validation process, we used two microfluidic chips. The first one, a mammalian cell culture chip used in Stephen Quake's lab at Stanford University, featured long control channels. We used it to check the compact model of the channels. We designed the second chip to validate the peristaltic pump model, a device that contains both channels and valves.

3.4.1 Long channels

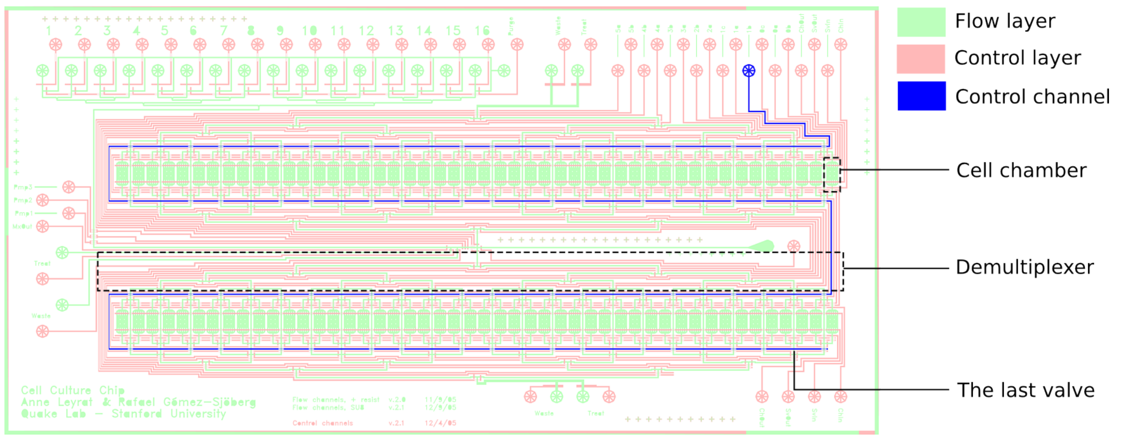


Figure 3.35: Mammalian cell culture chip used for validation of the channel's model. The chip uses a demultiplexer to route cells into individual chambers. Due to large chip dimensions, the multiplexer has very long (up to $25mm$) control channels, one of which is highlighted with a blue color. During the measurement, we monitored the last channel of each valve to assess the pressure at the end of the channel.

The mammalian cell culture chip is shown in the Figure 3.35. Its main function is to build certain cell concentrations in each of the 96 chambers. The chip utilizes a demultiplexer that routes the cells into the chambers. Due to the larger size of mammalian cells, the reaction chambers are big and so is the chip. The demultiplexer's control channels are long (up to $25mm$) and are perfect candidates for the validation process, since on one hand the slow pressure changes are easy to capture, on the other hand we can learn the model's accuracy on bigger scales.

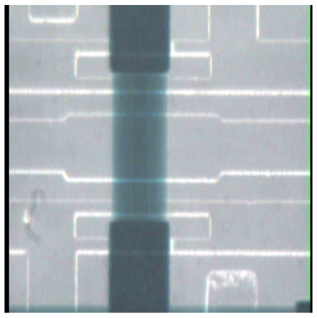
Similarly to the transient validation in COMSOL, in this experiment, we aimed at measuring the step response of the channel. The pressure in a particular point of a channel can be measured in an invasive manner, e.g. using an external pressure sensor, or in a non-invasive one, e.g. by embedding such sensor in the chip. We chose to use the latter method, making valve play the role of the sensor.

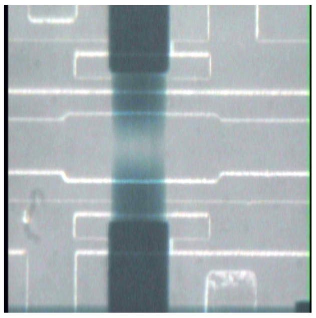
By design, each of the demultiplexer's channels ends with a valve. We used the valve's deformation image as a proxy to the control pressure. First, we filled the flow channel with a dyed fluid (water mixed with food coloring) and the control channel with transparent one. Then, we fixed the viewing window of the microscope on the valve (see Figure 3.36, top row) and took its pictures at different control pressures. Next, we calculated the total intensity (sum of all color components) of each image.

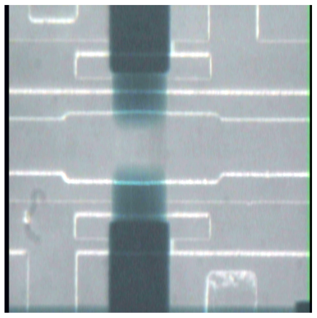
The resulting intensity vs pressure curve is shown in Figure 3.36. The plot has two important properties; first, the data points are strictly monotone in pressure, meaning that the $I(P)$ function can be inverted into $P(I)$ and used to assess the pressure. Secondly, if we approximate the data points with a piece-wise linear function, it will be similar to the displaced volume of the valve (see Figure 3.33), indicating that the total image intensity is proportional to the volume of the dyed fluid.

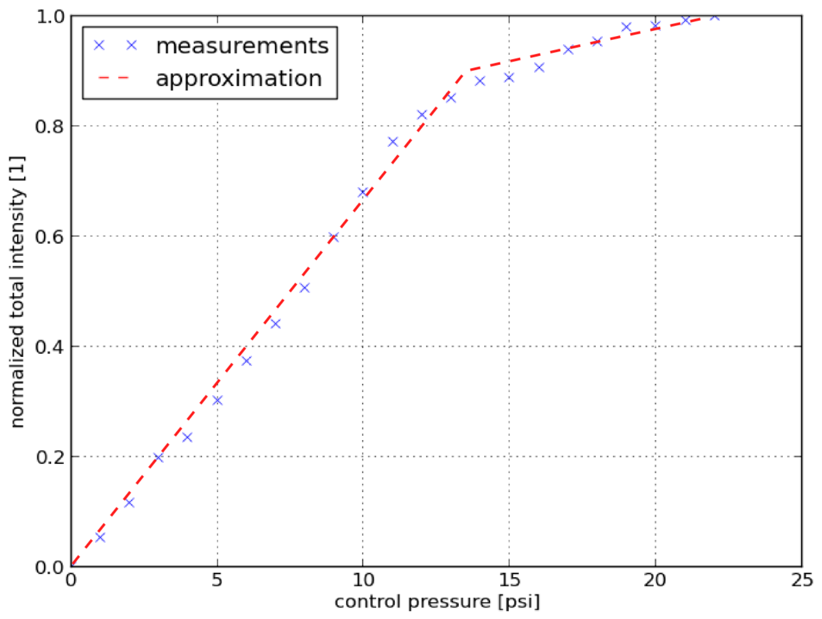
The final step is the actual measurement. During this step, we applied low frequency square pulses to the inlet and recorded the total intensity of the image acquired from the microscope camera. Then, we applied the $P(I)$ function to the intensities $I(t)$ and obtained the response function $P(I(t)) = P(t)$.

Figure 3.37 shows the case of the worst mismatch between the simulation and the ex-


(a)


(b)


(c)



(d)

Figure 3.36: Top row: microscope acquired images of the valve under 0psi (a), 10psi (b) and 20psi (c). Flow channel contains dyed fluid and a control channel is filled with transparent one. The valve's membrane displaces the dyed fluid from the flow channel, increasing the total intensity of the image. Bottom row: normalized total intensity vs control pressure plot (d). The curve has a 2-stage piece-wise linear structure that is typical for a volume curve of a valve.

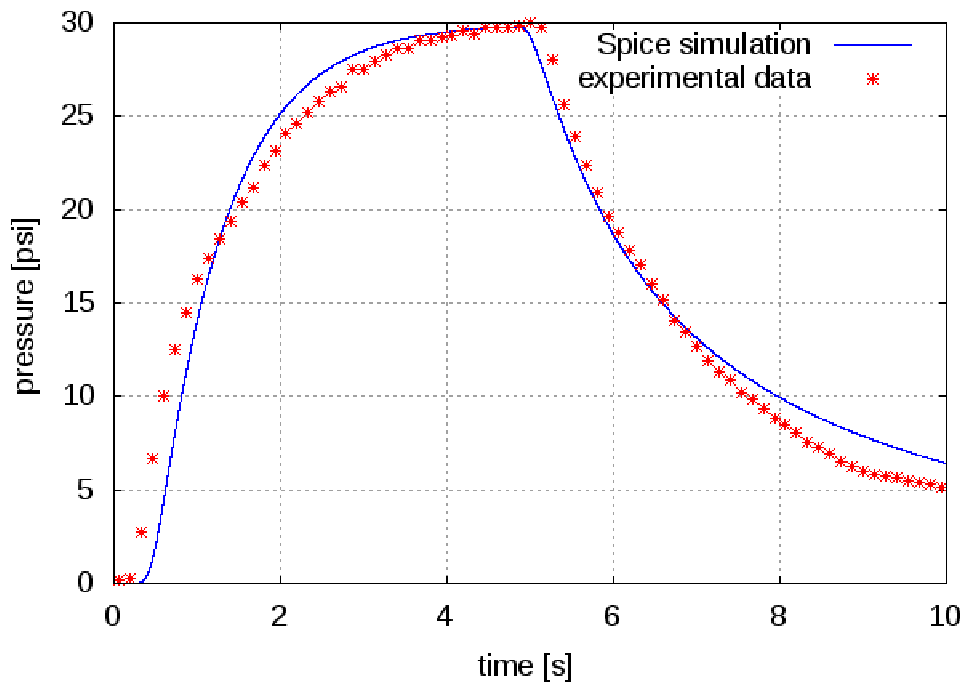


Figure 3.37: Case of the worst mismatch between the experimental and the modelled pressure impulse response of a long control channel.

perimental data, with an error of about 15%. Expectedly, the mismatch was observed on the chip's longest channel. Although the error isn't negligible, the difference between the curves have both positive and negative parts of approximately same area, meaning that the error doesnt accumulate over time.

Now that we've tested the channel's model, we can proceed to a more complex device comprised of channels and valves, the peristaltic pump.

3.4.2 Peristaltic Pump

Overview

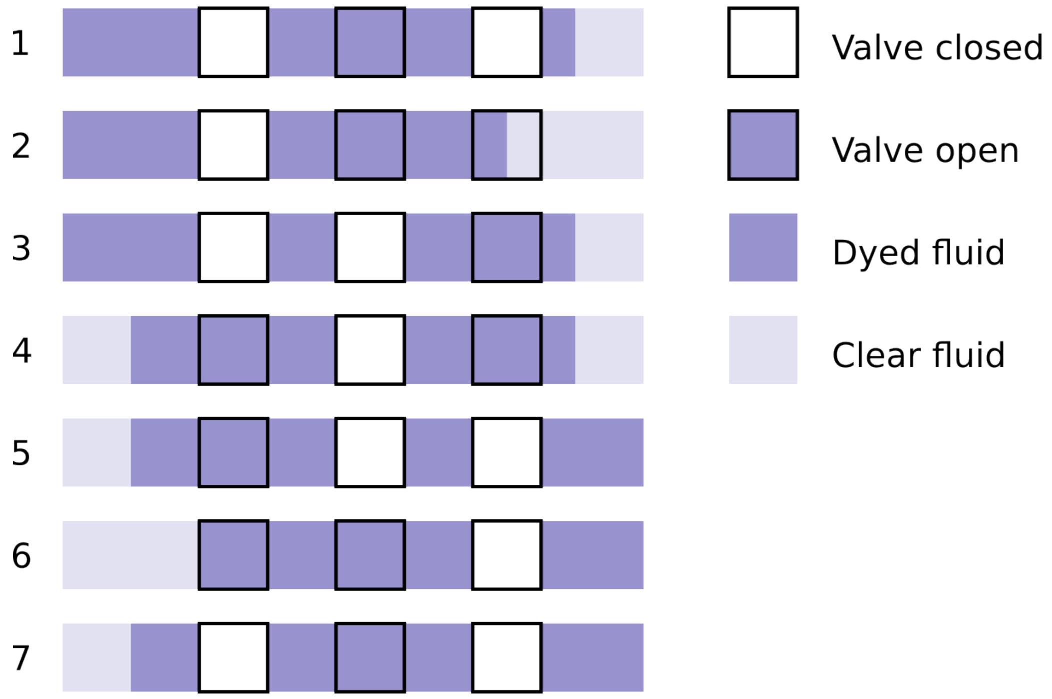


Figure 3.38: Peristaltic pump in action: a portion of dyed fluid (dark blue) is pushed from the right to the left by the peristaltic pump. The sequence starts from row 1, goes through row 7 and loops to the second row. In steps 4-6, a portion of fluid is pulled from the right side by the first and the second valves. In step 7, the portion is trapped inside the pump and is released to the left in step 3.

The microfluidic peristaltic pump is a device that consists of a flow line with 3 or more valves, each connected to a separate control line (see Figure 3.39). The

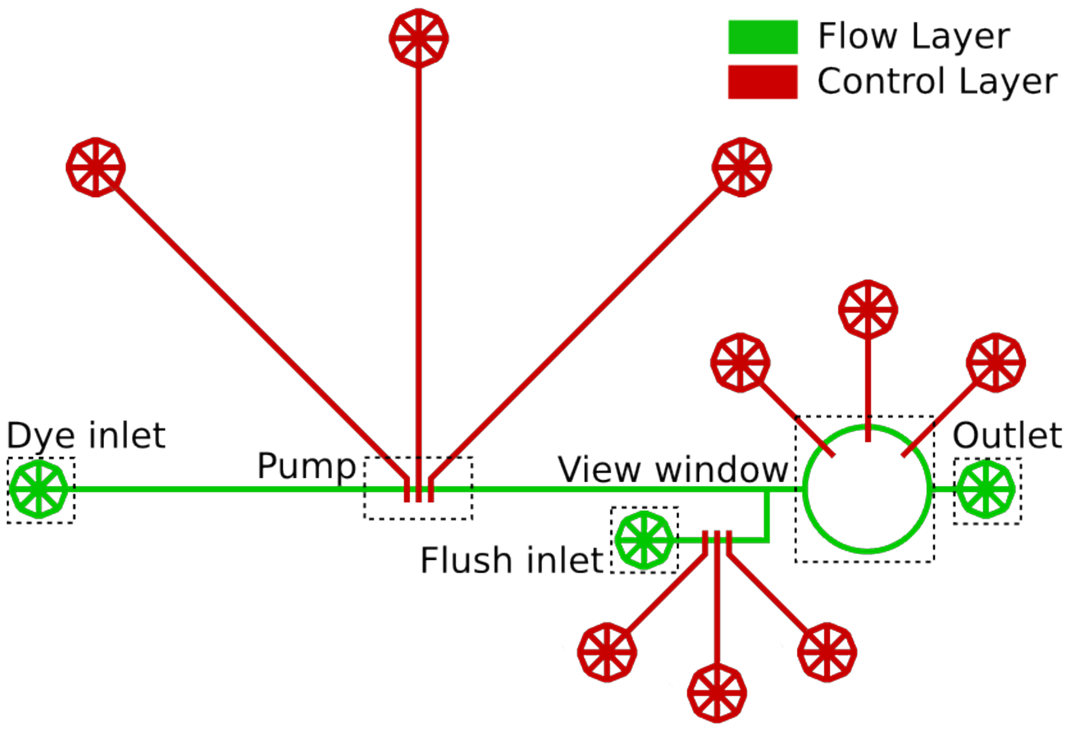


Figure 3.39: Drawing of the peristaltic pump test chip. It has two inlets, one for dyed fluid and one for transparent flush, and one outlet. The pump pushes coloured fluid into the round chamber which is observed via a microscope camera. The chamber filling process is filmed and processed to assess the flow rate.

device pumps the fluid through the flow channel, mimicking natural peristalsis with its valves. Although peristaltic pumps are slow (compared to pressure driven flows), due to discrete control, they allow to accurately measure the fluid. This enables a chip, for example, to mix reagents in certain ratio and volume with low amounts of waste.

To make the pump work, the valves open and close in a special order, one of which is shown in Figure 3.38. The main idea of the sequence is that in each step the flow channel is divided into two parts by at least one valve. This in turn makes sure that if another valve opens, it causes an influx from one side only; Suppose, the middle valve is closed. If we open the first valve (see Figure 3.39,3), the right side of the channel supplies the necessary fluid to fill the valve's volume. If we close the third valve (see Figure 3.38,5), the fluid is pushed to the left.

The pump isn't very efficient; the fastest it can do is one valve's volume (on the order of $0.1nl$) per cycle [10]. Therefore, the only way to achieve higher flow rates is to increase the pumping rate (i.e. number of cycles per second). Unfortunately, as the pumping rate increases, the efficiency of the pump drops.

The decrease in efficiency is caused by delays in control channels; at higher pumping rates the pressure at the end of the channel doesn't reach its maximum, resulting in an incomplete sealing of the valves. This in turn has two consequences. First, the volume displaced by the valve decreases since the membrane doesn't go all the way into the flow channel. Second, a partially closed valve conducts fluid, and therefore doesn't fully decouple the flow channel; e.g. if in step 5 (see Figure 3.38, row 5) the middle valve isn't fully sealed, the third valve will push the fluid both forward and backward.

The flow rate should increase with pumping rate at lower frequencies and converge to zero at higher ones. However, it is non-trivial to reason about the behaviour of the rest of the curve, given that all elements of the pump have non-linear characteristics.

From the compact modelling perspective, the pump is a very simple device; it consists

[10] A tight (about 90%) upper bound

of 4 channels (3 control and 1 flow) and 3 valves. This makes a pump a perfect candidate for model validation experiments.

Experiment

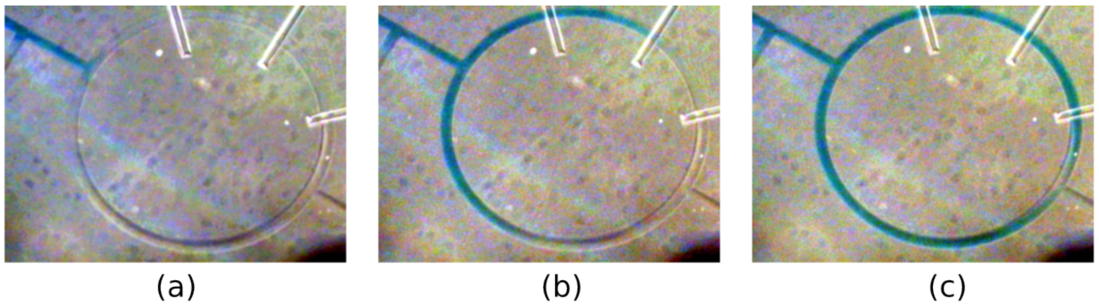
For the peristaltic pump experiment, we design and manufactured a simple microfluidic chip (see Figure 3.39) consisting of a pump, a mixing chamber, 2 inlets and 1 outlet. In the experiment, we assessed the flow rate through the pump vs pumping frequency. The measurements were done in a fashion similar to the long channel experiments; first we flushed the mixing chamber with clear water from inlet 2. Then we started filming the chamber while pumping dyed fluid into it. Finally, we computed the flow rate based on the intensity of the image (which we already know to be proportional to the volume).

Figure 3.40 shows a snapshot of the mixing chamber, as well as the computed intensity vs time curve. This curve has 3 distinct linear regions. The first one, a constant region, corresponds to the fully transparent mixing chamber. The second segment is linear in time, indicating the propagation of the dyed fluid into the chamber. After the chamber is filled with dyed fluid, the intensity stabilizes resulting in the third, constant segment.

The plot, however, has some subtle details. First, the curve's second knee is much more round than the first one. The asymmetry is caused by the non-flat velocity profile of the fluid (and consequently the dyed front). When the dyed fluid reaches the end of the chamber, its front is stretched much more compared to that at the time of entry (see Figure 3.40 (a) and (c)).

Second, the third region is much more variative compared to the first one. While the chamber is fully filled with dye, the pump is still working causing the small variations in the volume of the mixing chamber (which as we know has some non-zero capacitance). This in turn causes the oscillations of the total intensity of the image.

Putting it all together, the flow rate assessment procedure goes as following; first we evaluate intensities I_{min} and I_{max} that correspond to an empty and a full chamber respectively, by averaging the intensities of the first and the third regions. Next,



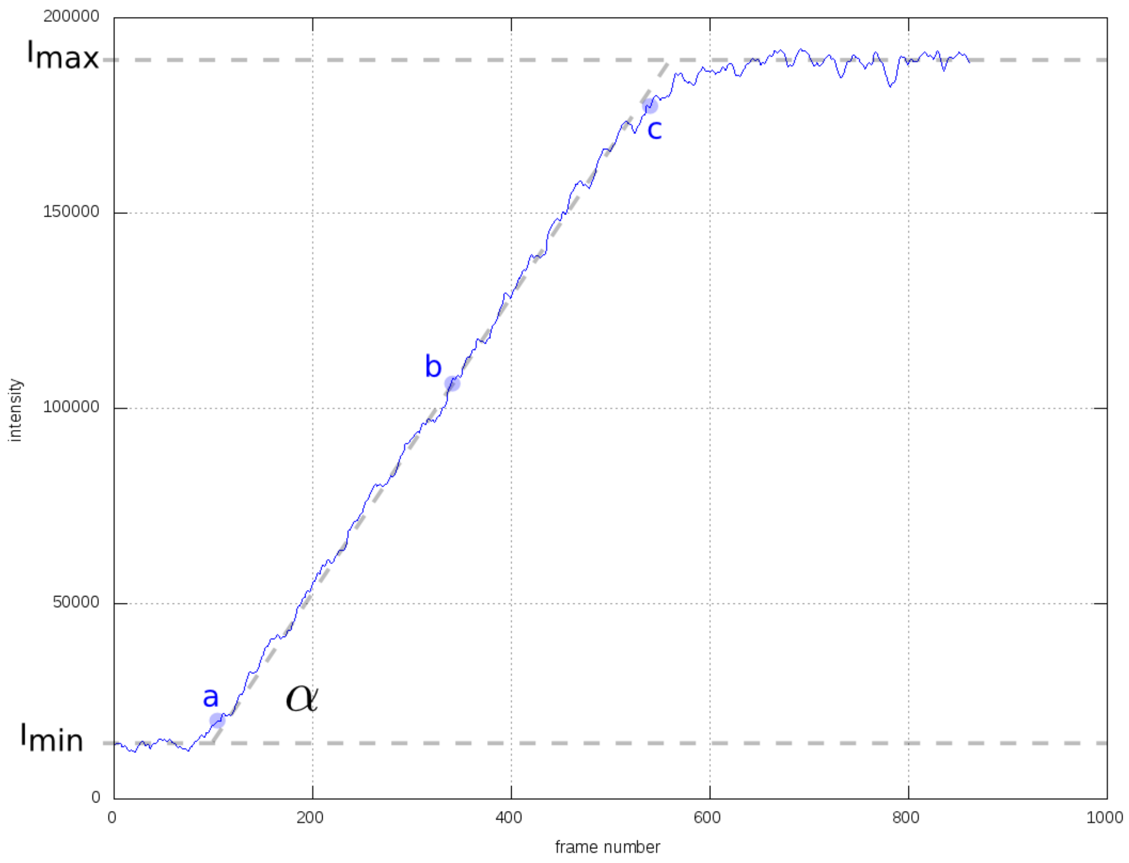


Figure 3.40: Top row: raw captured images of the mixing chamber during the pumping experiment. In (a) the dyed fluid had just entered the chamber. In (b) it's half way through and in (c) it had just reached the chamber's outlet. Bottom row: total image intensity vs time. Dotted lines indicate max and min intensity levels as well as the linear increase of the intensity over time. Blue letters *a*, *b* and *c* correspond to the images shown in the top row. At the time point *c*, the colored fluid escaped the chamber, but hasnt filled it completely yet.

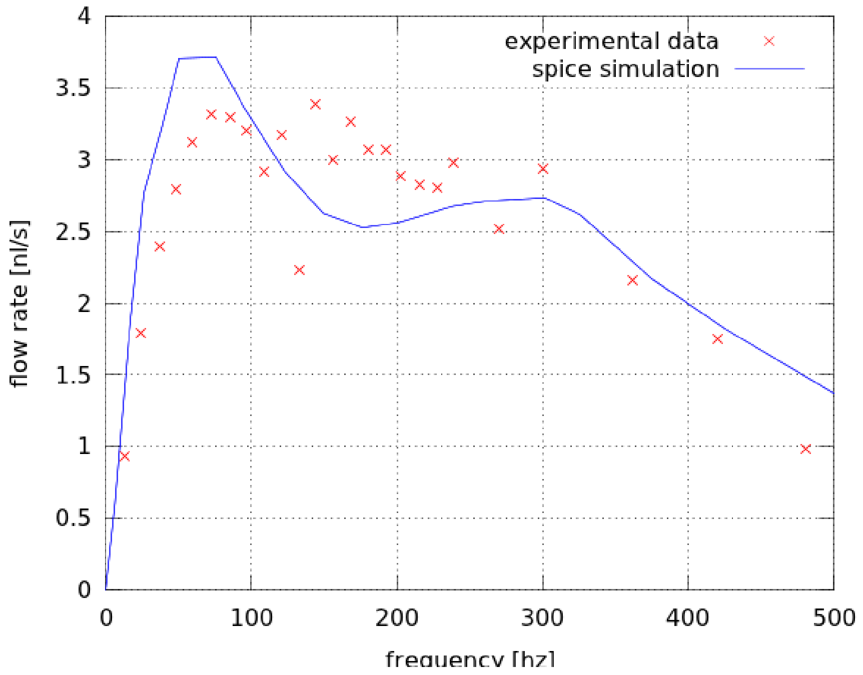


Figure 3.41: Simulated and experimental throughput vs pumping rate curves. Simulation error is less than 15%. The curve's shapes is typical for microfluidic peristaltic pumps and features 4 regions: attack $(0 - 80hz)$, decay$(80 - 110hz)$, plateau $(110-300hz)$ and release$(300hz$ and on). The flow rate converges to zero as frequency goes to infinity.

we (manually) locate a linear (i.e. with minimal curving) region in the middle and determine its slope coefficient α. Since we know the volume $V_{chamber}$ of the mixing chamber, the flow rate can be calculated as:

$$Q_f = \alpha \frac{V_{chamber}}{I_{max} - I_{min}},$$

where Q_f is the flow rate at some pumping frequency f. The resulting $Q(f)$ curve overlaid with simulated results is shown in Figure 3.41. The simulation is accurate with an error below 15%. While missing some local spikes, the simulated curve captures the overall behaviour of the pump.

Having an attack, decay, plateau and release regions, the characteristic curve shown

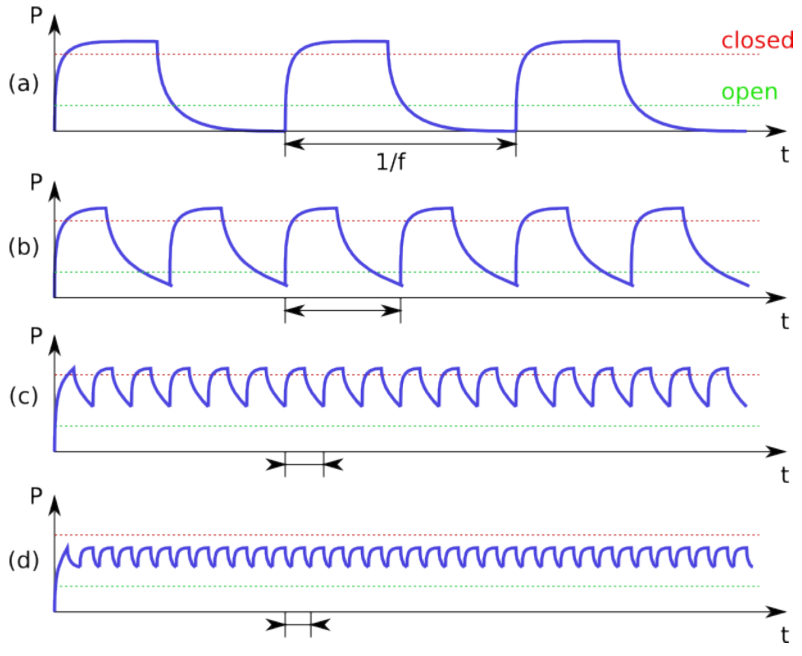


Figure 3.42: Pressure at a valve as a function of time for different pumping frequencies. (a) Normal operation of the pump, valves close and open all the way through. (b) Valves open and close properly, but the pressure range is reduced. (c) The valve closes, but doesn't open completely. (d) Degenerate pump's operation, worthless oscillations of the membrane.

in Figure 3.41 is typical for peristaltic pumps. In the attack range of frequencies, the pump operates in the proper way (as previously described), i.e. all valves close and open completely (see Figure 3.42). In this regime, the flow rate increases linearly with the pumping rate.

As the pumping frequency increases, the charge-discharge asymmetry of the control channel introduces a non-zero bound on the valve's control pressure (see Figure 3.41 (b)). As a result, valves fully close and open, but the displaced volume decreases. At first (before the throughput peak) the effect is compensated by increasing pumping frequency, making the throughput grow, at lower rates though.

As we further accelerate the pumping, the displaced volume starts to drop much faster, due to the change in the $V_{membrane}(P_{ctrl})$ slope (see Figure 3.33). Although the valves are still open and close properly, this negative effect overwhelms the increase in pumping speed, making the flow rate go down.

The next frequency region changes the operating regime of the pump, in which the valves do not fully open (see 3.41 (c)). In addition to reducing the volume displacement even more, the effect boosts valve's resistance, affecting the distribution of the fluid's volume inside the pump. The behaviour of the curve in this frequency range depends on the pump's parameters and can be pretty complex [11]. In general, the $Q(f)$ function oscillates around a plateau.

Finally, as the pumping rates go even higher, the pump's performance drops drastically because of the non-closing valves (see Figure 3.41 (d)). This causes a backflow (i.e. the fluid goes backwards when it shouldn't), which increases with the pumping speed and ultimately negates the flow rate. It should be noted, that in this regime the pump has the closest behaviour to a natural peristalsis, i.e. when fluid is driven forward by a smooth wave of channel's deformation.

The good agreement between our simple compact model and the measured results, gives us confidence in the model prediction. Thus the compact modelling approach becomes a powerful instrument for simulation and analysis of microfluidic devices (made of valves and channels). The parameterization of the model allows us to do sensitivity analysis as well as local optimizations of a device or a whole chip. Turns

[11] The region also has the highest experiment-simulation mismatch, most likely due to second order effects not included into the model.

out, because of the analytical simplifications done to the channel's model, it is possible to do efficient and provable global optimizations via geometric programming framework, which are described in the next chapter.

Chapter 4

Synthesis

4.1 Optimization background

The vast majority of design problems involve constrained optimization, i.e. reducing cost (time, space, energy) subject to certain requirements, in one way or another. In VLSI design, such optimizations include minimizing the area of a chip subject to timing and energy constraints or minimizing total wire-length subject to design rules, to name a few. It is not surprising, that many of the microfluidic design problems can be framed this way.

Microfluidic chips are area constrained (they should fit on a die), and time constrained (experiments should be done in a reasonable time). Design rules, like minimal distance between the adjacent channels, introduce constraints on valid solutions. Depending on the chip goals, any of the said parameters can play a role of a cost to be minimized. For example, if we want to have as much chips on a die as possible, the area becomes such a cost. Or if we are to perform as many experiments as possible, we'll minimize the operation time of the chip.

Regardless of the origin and application area, any optimization problem can be framed

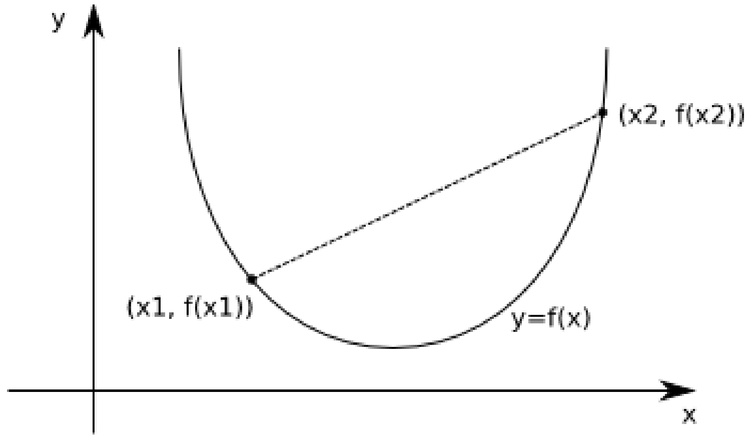


Figure 4.1: A convex function $f(x)$. A segment connecting two arbitrary points on the $(x, f(x))$ curve, $(x_1, f(x_1))$ and $(x_2, f(x_2))$, is above the curve itself.

as:

$$\begin{aligned} \text{minimize} \quad & f(x), \\ \text{subject to} \quad & x \in G, \end{aligned}$$

where $f(x)$ is a cost function, x is a vector of design variables and parameters, and G is a set of feasible design points (e.g. where design variables satisfy all the constraints).

In general, optimization problems are hard to solve, requiring an exhaustive search through the feasible set, ending up with an exponential execution time. Although certain optimization problems can be solved via heuristic algorithms in reasonable time, such an approach generally doesn't guarantee optimality. However, there is a class of optimization problems, called convex problems, that allows for relatively fast convergence to a provably optimal solution. [29]

One of the possible ways to define a convex optimization problem is:

$$
\begin{array}{ll}
\text{minimize} & f(x), \\
\text{subject to} & \\
g_0(x) \leq 0, & g_1(x) \leq 0, \cdots, g_n(x) \leq 0, \\
h_0(x) = 0, & h_1(x) = 0, \cdots, h_m(x) = 0,
\end{array}
$$

where the functions $f(x)$ and $g_i(x)$ are convex, and all $h_i(x)$ are linear. A function $f(x)$ is convex if for any two points x_1 and x_2 a segment $(x_1, f(x_1)) - (x_2, f(x_2))$ lies above the function $f(x)$ (see Figure 4.1).

Although it is trivial to prove convexity of functions of one variable like $f(x) = x^2$ or $f(x) = e^x$, in general, convexity checking of functions of multiple variables using a definition alone requires significant effort. Fortunately, there is a set of functions composition rules that preserve convexity, e.g. a sum $f(x) + g(x)$ is convex if both $f(x)$ and $g(x)$ are convex. This is also true for $\max(g(x), f(x))$ and $f(kx+b)$, however convexity of $f(x)g(x)$ or $f(g(x))$ isn't guaranteed.

This idea is used in disciplined convex programming (DCP), a framework, that automatically recognizes and solves a convex optimization problem, if the functions are constructed of some primitive convex functions (like x^2 or e^x) abiding the said rules. Solvers based on the framework (e.g. CVX) are very efficient and robust. Moreover, besides the optimal solution, they also produce a certificate of global optimality. In terms of design, it means that if we manage to formulate a design problem in a symbolic way following DCP rules, we'll be able to obtain a provably optimal design that satisfies all the constrains.

Unfortunately, the analytical simplification of the microfluidic compact model involves non-convex functions, (e.g. Elmore's delay isn't convex). However, an extension of the convex programming called geometric programming [30] is a perfect fit for all the functions.

A general definition of a geometric optimization problem is:

$$\begin{array}{ll} \text{minimize} & f(x), \\ \text{subject to} & \\ g_0(x) \leq 1, & g_1(x) \leq 1, \cdots, g_n(x) \leq 1, \\ h_0(x) = 1, & h_1(x) = 1, \cdots, g_m(x) = 1, \\ x > 0, & \end{array}$$

where $f(x)$ and $g_i(x)$ are posynomials and $h_i(x)$ are monomials. A monomial function is:

$$h_i(x) = k \prod_j x_j^{\alpha_j},$$

where k is a non-negative coefficient and α_j's are arbitrary numbers and a posynomial is a sum of monomials. The reason why a geometric program can be efficiently solved is that it can be transformed into a convex program by a logarithmic change of variables: $y_i = \log x_i$. For example, an $h_i(x) = 1$ constraint will turn into a linear one:

$$1 = k \prod_j x_j^{\alpha_j} = k \exp(\sum_j \alpha_j \log x_j) = k \exp(\sum_j \alpha_j y_j),$$

which, in turn, is equivalent to

$$\sum_j \alpha_j y_j = -\log(k),$$

a linear constraint. In a similar way, it can be shown that the posynomial functions will turn into convex ones.

For example, Elmore's delay of a channel as a function of lengths (l_j) and widths (w_j) of its segments is a posynomial:

$$D_{Elmore} = \sum_j k_j l_j^2 w_j^{\alpha_j},$$

where k_j's are constant non-zero coefficients. The delay therefore can be minimized and/or constrained by some non-zero value.

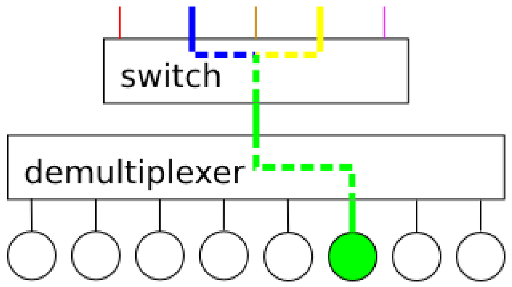


Figure 4.2: Block diagram of the bacteria culture chip: a set of reagent inputs are connected through a switch to a demultiplexer that can route its input into any round cell chamber. Colored lines represent a sample path of the reagents through the chip.

4.2 Bacteria culture chip

The bacteria culture chip (BCC) is an array chip for massively parallel study of bacteria colonies in various environments. The chip consists of a number of fully addressable cell chambers, capable of holding cell colonies and a set of reagent inputs connected to a switch that allows combining the reagents in arbitrary ways (see Figure 4.2).

While the range of experiments enabled by the chip is quite large, most of them share the same schema: first, the cells of different bacteria and concentrations are fed into each of the cell chambers. Second, they are fed with or exposed to different solutions and the consequences are being studied. For example, the array is populated with a drug-resistant strain of pathogenic bacteria. Then, different combinations and/or sequence of antibiotics are introduced into the array to find out the best way to suspend the growth of the colonies. Another experiment aims at separating bacteria in a mixed colony by using various feeding solutions.

Initially, such bacteria studies were done using mammalian cell culture chip that has the same structure, but features larger cell chambers. While it makes sense for experiments involving large mammalian cells, it is inconvenient to study bacteria colonies in such chambers. The BCC was intended to use much smaller chambers (200μ) that can fully fit into microscope's view window at high magnification levels, allowing observation of the whole colony.

Such a reduction in chambers' dimensions allowed us to have many more chambers on a single die, extending the parallelism capabilities of the chip. This increase, however, comes at a price; more chambers mean larger demultiplexers (that might or might not fit on a die) and, therefore, longer control and flow channels, resulting in higher operational delay.

Not only the number of chambers can vary, but also their layout on the chip; for example, in the mammalian culture chip, the array is divided into two blocks of 48 chambers and 2 demultiplexers each (see Figure 3.35). The bacteria culture chip, however, isn't restricted to a particular implementation, making it possible to reduce the spatial and timing overhead by choosing the right design.

Finally, even when we fix the design of the chip, the dimensions of the channels aren't trivial to set in an optimal way. To understand the trade-offs of the design, consider an instance of the chip with 16 cell chambers (see Figure 4.3). We'll simplify the design by assigning the same widths W_f and W_c to flow and control channels respectively. The delay associated with the propagation of the fluid through the flow channels is a non-linear function of W_f and L_f, total length of a demultiplexer path:

$$T_{flow} = D_f(W_f, L_f) = D_f(W_f, L_0 + W_{demux} + L_{demux}),$$

where L_{demux} and W_{demux} are the length and the width of the demultiplexer, and L_0 is some constant (see Figure 4.3).

The D_f function increases in $L_f = L_0 + W_{demux} + L_{demux}$ and decreases in W_f. The delay caused by the demultiplexer switching is another non-linear function of the width and the length of the control channels:

$$T_{control} = D_c(W_c, L_c) = D_f(W_c, L_1 + W_{demux} + 2L_{demux}),$$

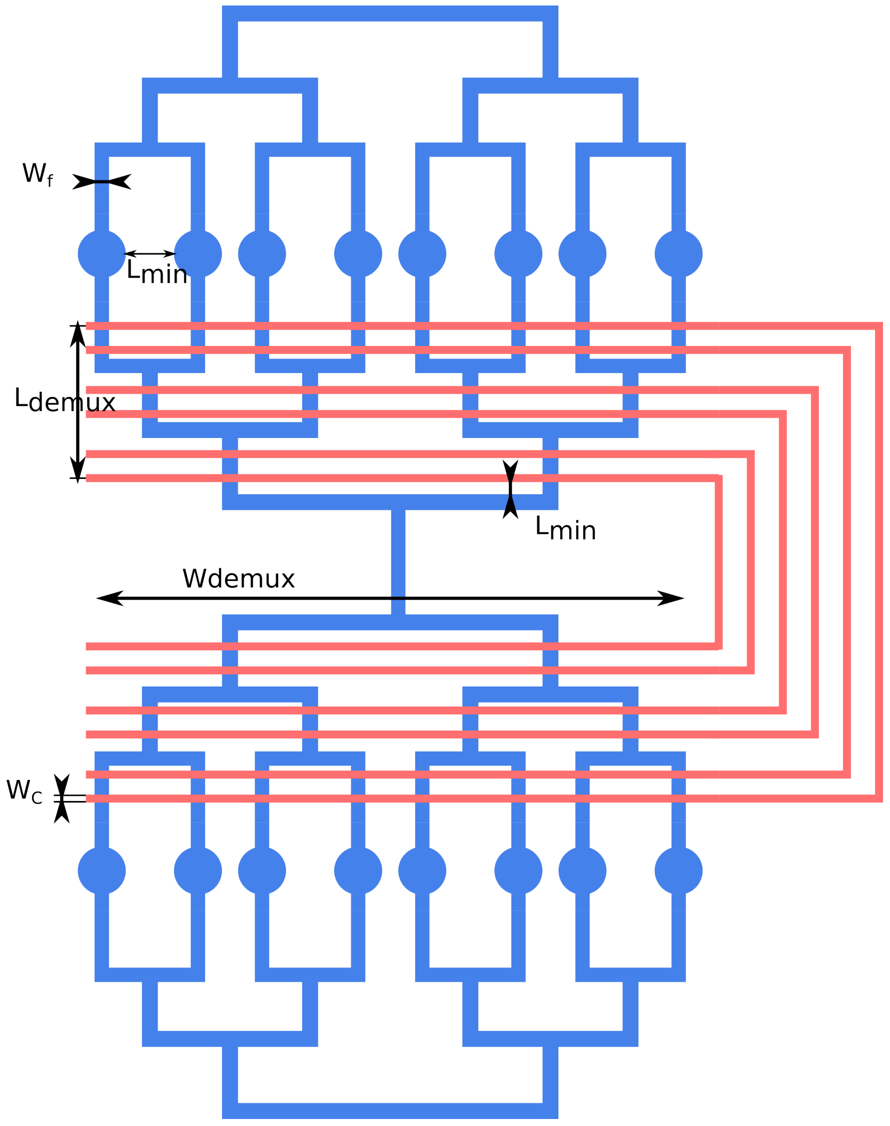


Figure 4.3: Sample implementation of an 8 chamber bacteria culture chip. Blue and red lines are flow and control channels respectively. W_c is the width of the control channels. W_f is the width of the channels directly connected to the chambers, L_{min} is the minimal spacing distance, L_{demux} and W_{demux} are the total length and width of the demultiplexer block.

where L_1 is a constant overhead caused by the minimal spacing requirements. The D_c function increases in L_c and decreases in W_c.

Finally, the width and length of the multiplexer depend both on the widths of control and flow channels:

$$\begin{aligned} W_{demux} &\geq 8W_f + 7L_{min}, \\ L_{demux} &\geq 3W_f + 6W_c + 8L_{min}. \end{aligned}$$

Putting it all together, some of the simple trade-offs are as follows. If we increase the width of the control channels W_c, the T_c will go down, but the T_f will go up. If we increase the W_f, the width of the flow channels, we'll definitely increase the T_c, while the impact on the T_f will depend on the structure of the D_f function.

In the real setting, the trade-offs become much more complex, since each of the channels can have distinct widths, many more constraints apply (e.g. minimum and maximum allowed widths) and the real chip has significantly more cell chambers.

To address the complexity of the design space, we developed a synthesis and optimization tool for microfluidic chips that enables us to quickly construct a design out of parameterized building blocks and to globally optimize its parameters using the geometric programming framework.

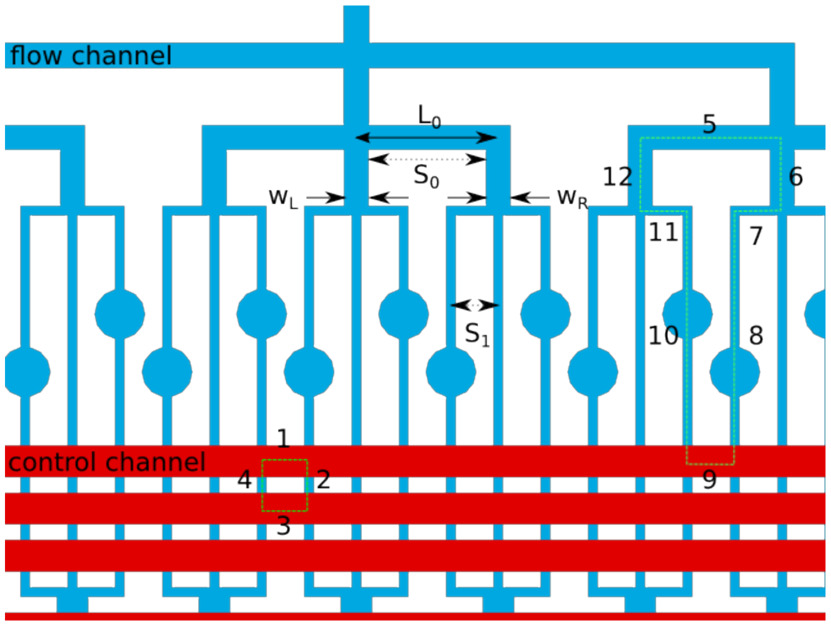


Figure 4.4: Design rules. S_0 is a spacing that is expressed through L_0, W_L and W_R. S_1 is another spacing expressed the same way. Segments labelled 1-4 and 5-12 are examples of loops.

4.3 GP formulation

The GP model used for design consists of variables, constraints and the cost function. Variables and constraints are grouped according to their physical meaning. The groups are geometrical, fluidic flow, fluidic control and delay group. The cost function is a combination of the delays and/or flow rates and/or volumes.

Geometrical groups represent the layout of the design and deals with dimensions only. We partitioned the network of channels into straight segments with their lengths (L_i) and widths (W_i) as variables. For example in Figure (4.4), L_0 is a segment's length, W_L and W_R are widths, some of the segments are enumerated from 1 to 12.

There are three types of constraints in the group: symmetry, spacing and consistency. Symmetry constraints are of the form: $w_i = w_j$ or $L_i = L_j$. In the design, we enforce alignment of the chambers by making all levels of the distribution tree and the demux of the same height, e.g. $L_{12} = L_6$ (see Figure 4.4). To speed up the optimization, we also enforced equal width along a level, e.g. $W_{12} == W_6 == W_5$. While being optional, the simplification leads to a logarithmic decrease in the number of variables.

Spacing constraints ensure there is proper spacing (typically 50μ) between the edges of the adjacent channels. In Figure 4.4, S_0 is a spacing and it is expressed through the variables:

$$S_0 = L_0 - W_L/2 - W_R/2 \geq 50 \times 10^{-6}.$$

Moving all widths to the right hand side makes the expression a valid GP constraint, a posynomial is less or equal to a monomial:

$$L_0 \geq 50 \times 10^{-6} + W_L/2 + W_R/2.$$

The operation is used for all Π shaped structures, like (8,9,10) or (11,12,5) in Figure 4.4.

Consistency constraints make sure all the loops in the network translate into geometric loops. In a simple case of a rectangle loop (1,2,3,4) (see Figure 4.4), the constraints are:

$$\begin{aligned} L_1 &= L_3 && (4.1) \\ L_2 &= L_4. && (4.2) \end{aligned}$$

For a more complex (5-10) loop, the constraints should be:

$$\begin{aligned} L_{12} &= L_6 && (4.3) \\ L_{10} &= L_8 && (4.4) \\ L_7 + L_9 + L_{11} &= L_5. && (4.5) \end{aligned}$$

While the equations 4.1 to 4.4 are valid GP equalities (monomial equals to monomial), the last one is not. It can be relaxed, however, to the form:

$$L_7 + L_9 + L_1 1 \leq L_5$$

And although the inequality doesn't guarantee the loop is closed, it actually turns into equality when optimized. The reason for that is the pressure applied to all of the lengths to be as small as possible (within the constraints) and keeping L_{10} larger than $L_7 + L_8 + L_9$ is suboptimal. Consistency constraints are produced for all simple loops, i.e. the loops that geometrically don't contain other loops within their bounds.

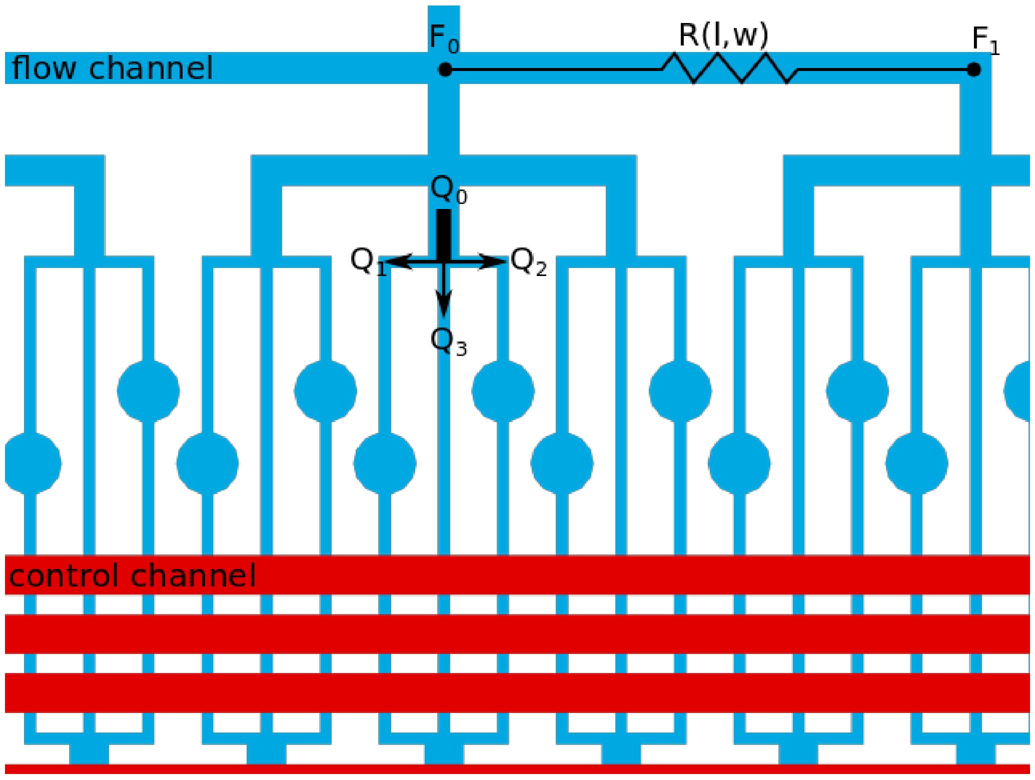


Figure 4.5: Fluidic Kirchhoff's laws. F_0 and F_1 are values of the potential function. $R(l, w)$ is the channel's resistance, such that $F_0 - F_1 = R(w, l) \times Q$. Q_0, Q_1, Q_2 and Q_3 are the flow rates such that: $Q_0 = Q_1 + Q_2 + Q_3$

.

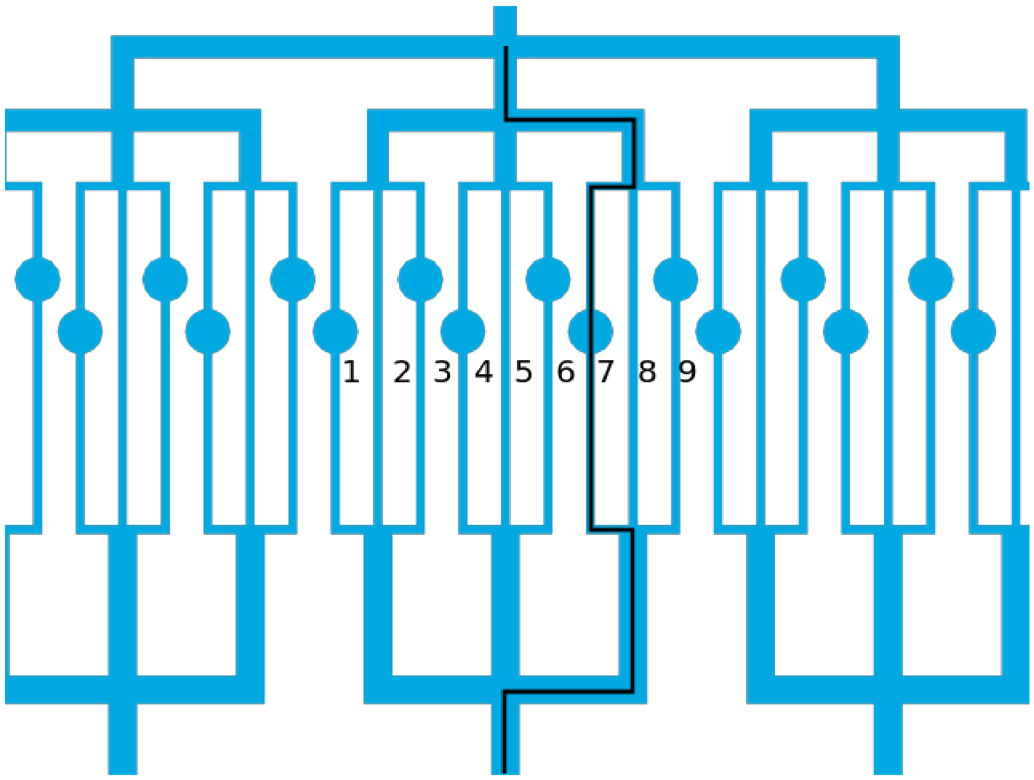


Figure 4.6: Simple path from the inlet to the outlet. The columns are enumerated from 1 to 9 with respective flow rates Q_1 to Q_9.

.

The purpose of the fluidic flow group is to model the way fluid flows through the channels, and therefore currents are group variables. The constraints of the group should represent fluidic analogs to Kirchhoff's laws:

$$\begin{aligned} \Delta U_i &= R_i(l_i, w_i) \times Q_i && (4.6)\\ \sum_{at\ forks} Q_i &= 0, && (4.7) \end{aligned}$$

where ΔU_i is the potential difference between the ends of the channel i, R_i is the channel's resistance and Q_i's are flow rates (see Figure 4.6). Both set of equations are not valid GP constraints, but can be transformed to become valid.

First, consider an activated demux. In this case, there is only one path from the inlet to the outlet (see Figure 4.6). So, we can rewrite the set equations (4.6-4.7):

$$U_{inlet} = \sum_{i \in path} R_i(l_i, w_i) \times Q_{path}, \tag{4.8}$$

where F_{inlet} is the potential at the inlet (a fixed quantity) and Q_{path} is the path's current. The equation is not a valid GP constraint, but again, it can be relaxed:

$$U_{inlet} \geq \sum_{i \in path} R_i(l_i, w_i) \times Q_{path}. \tag{4.9}$$

Since $R_i(l_i, w_i)$'s are monimials (typically $\beta l_i w_i^{\alpha}$), the whole right hand side expression is a posynomial. Analogous to consistency constraint case, the equation is suboptimal in the inequality form (as we want Q's to be as big as possible) and will be turned into an equality by the optimizer.

Now, consider a distribution tree, when all the flow channels are open and the current flows through all of them. In this case, we can deduce the direction of the flow topologically. This can be done by drawing a path from the inlet to the outlet through the channel segment in question (see Figure 4.6). For any such path, the flow will have the same direction (from the inlet to the outlet). Therefore, we can use positive Q_i's and transform equations 4.7 into:

$$Q_j = \sum_{at\ forks} Q_i, \tag{4.10}$$

where Q_j is a flow rate before a fork and Q_i's are flow rates after it (see Figure 4.5). After simplification, each Q_i can be expressed:

$$Q_i = \sum_j A_{ij} Q_j^c, \tag{4.11}$$

where Q_j^c is a chamber column current (e.g. see Figure 4.6 from 1 to 9) and A_{ij} are positive coefficients. Finally, the set of equations 4.6 can be rewritten and relaxed as:

$$U_{inlet} \geq \sum_{i \in path} R_i(l_i, w_i) \times Q_i = \sum_{i \in path} R_i(l_i, w_i) \sum_j A_{ij} Q_j^c \quad \forall paths \tag{4.12}$$

This in turn is a valid GP constraint, since all A_{ij}'s are positive, and it will result in an equality after optimization for the same reasons as in the demux case.

The fluidic control group models the Elmore's delay of the control channels. For each connected network of channels, the relaxed form of the constraint is:

$$T_{Elmore} \geq \sum_{i \in network} \sum_{j \in network} B_{ij} C_i(w, l) R_j(w, l), \tag{4.13}$$

where all the B_{ij} are non-negative coefficients and $C(w, l)$ is a capacity of the ith segment. The right hand side is a posynomial, since C_i and R_j are monomials. The relaxation is valid, since we want smaller delays and T_{Elmore} is suboptimal unless equation 4.13 is an equality.

A flow delay group models the propagation delay of the flow channels. For the demux case, the filling happens path-wise, so the delay is:

$$T_{i \in path}^{demux} \geq Q_{path}^{-1} \sum_{i \in path} V_i(w_i, l_i),$$

where $V_i(w_i, l_i)$ is a volume of the ith segment and is a monomial (e.g. $height \times w_i \times l_i$

for a square channel). The inequality is a valid GP constraint and will turn into equality for the same reasons as in the Elmore's delay case.

In the case of a general distribution tree, it's not possible to express the propagation delay via a posynomial; however it's possible to bind it above:

$$\begin{aligned} Q_{min} &\leq Q_i^c \forall i, && (4.14)\\ T_{total}^{tree} &\geq (NQ_{min})^{-1} \sum_i V_i(w_i, l_i), && (4.15) \end{aligned}$$

where N is the total number of chambers columns. In other words, this is the time it will take to fill the whole tree with a smallest current. However, if the tree is a symmetrical binary one, the upper bound is exactly the propagation delay, since at all forks all currents are the same.

Now that we modelled all the physics, we can construct a cost function as a posynomial of any of the group variables. In our case, we wanted to minimize the operational delay of the chip during the initial cells injection process. For each of the columns, we repeat the following steps:

1. Switch the demux to address chamber i.
2. Wait until the fluid propagates to the chamber.
3. Switch the demux to the nearest flush line.
4. Wait until the line is flushed.

The resulting cost function therefore looks like this:

$$minimize \sum_{i \in controls} E_i T_{elmore}^i + \sum_{i \in chambers} G_i T_{pathi}^{demux},$$

where E_i and G_i are integers that depend on the configuration of the chip.

4.4 Synthesis tool

The optimization problem used for cell culture chip features nearly 2000 expressions and more than 400 variables. Descriptions of such scale are tedious to write, non-trivial to modify and hard to comprehend. Bringing automation to the process can make the approach usable.

Fortunately, microfluidic chips quite often have a structure that allows us to assemble a design using a small number of building blocks and a limited number of operations on those blocks. For example, the distribution tree of the cell culture chip is composed of tree segments (or forks), connected to each other in a hierarchical manner. Assay chips, as another example, have an array structure, i.e. there is a single core block which is duplicated and stacked. Finally, we can take an assay, connect it to two distribution trees and get almost a full design.

To produce an optimization problem out of a design, we introduce symbolically parameterized blocks, i.e. block with symbols instead of real numbers. Symbols can be operated with like normal numbers, but an operation will result in an expression rather than a number. This allows us to generate constraints and cost functions of the optimization problem in a relatively straightforward way. For example, if we need to bind the length of a certain path, the summation of all the channels' lengths along the path will give us the expression of the total length.

In this section, we will describe the tool that synthesizes a symbolic design from a high level description, generates an optimization problem in a symbolic form and produces a drawing based on the solution of the problem.

4.4.1 Graph representation

The main data structure used to represent a design is a directed graph. In the simplest case, edges are straight segments of the channels and the nodes are junctions of 2, 3 or 4 edges. Both nodes and edges have attributes. A typical edge has *length*, *width*, *layer* and *direction* attribute. The former two can be either a non-negative number or a symbol; *layer* is an arbitrary string (e.g. `flow1` or `control`) and *direction* is one

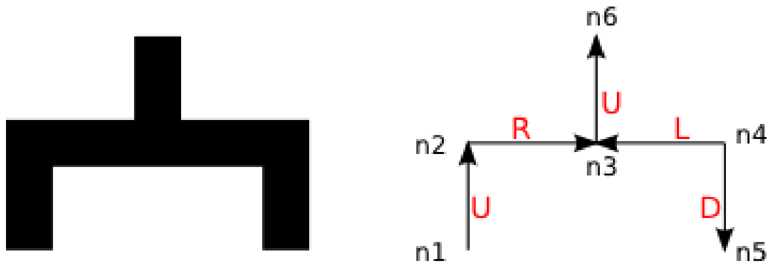


Figure 4.7: A simple channel fork; layout drawing (left) and internal graph representation (right). n1 to n5 are the names of the nodes, arrows are edges and the directions are shown in red. For example, the (n2,n3) edge represents the left horizontal segment and (n3,n6) is the top vertical one.

of the `u`, `d`, `l` or `r`. The latter attribute in conjunction with the target node of the edge determines the relative positions of the nodes.

Consider a simple fork design in Figure 4.7. Commonly, it used to either split a flow

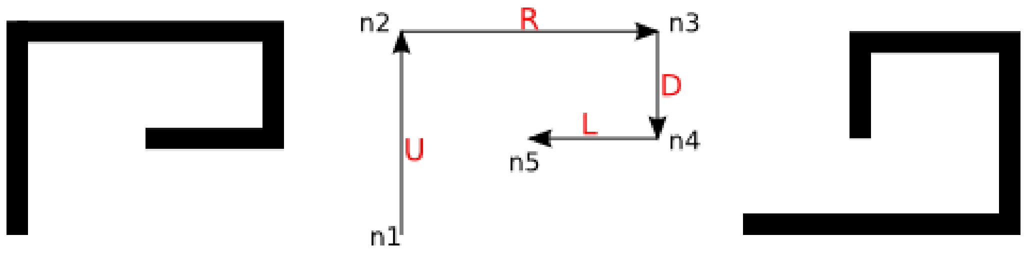


Figure 4.8: Two possible (right and left) layouts of a graph (in the middle). The layout depends on the lengths of the channels, specifically, $L(n2, n3) > L(n4, n5)$ for the left layout, and $L(n2, n3) < L(n4, n5)$ for the right one.

into two, or merge a couple of flows together. The directions of the edges in the graph, however, do not represent the flow, but the relative positioning only. For example, the $(n1 \rightarrow n2$, `U`$)$ edge means the segment goes up from $n1$ to $n2$, or alternatively, $n2$ is on top of the $n1$. The meaning doesn't change with full direction reversal: $(n1 \leftarrow n2$, `D`$)$ maps to the exact same thing. In the same fashion, $n3$ is to the right of the $n2$ and to the left of $n4$. Moreover, we can deduce that $n1$ is to the left of $n5$ and $n6$ is on top of all the nodes.

While in this example, relative locations of all nodes are uniquely determined, it is not true in general. In Figure 4.8, the layout of the channel graph depends on the

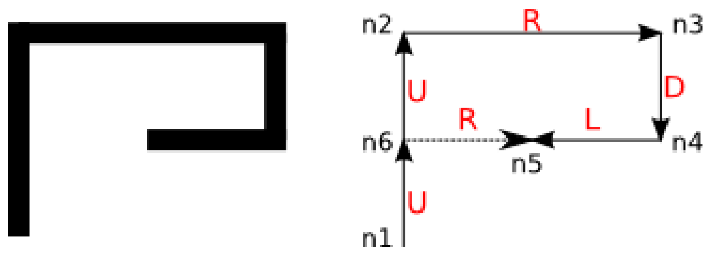


Figure 4.9: A virtual segment (a dotted line) is used to disambiguate the layout of the graph. In this case, it is clear that $L(n2, n3) > L(n4, n5)$, since $n6, n2, n3, n4$ is a cycle, and, therefore $L(n2, n3) = L(n4, n5) + L(n5, n6)$.

lengths of the wires. To combat ambiguity, we use virtual segments, edges with zero width and on a special layer (see Figure 4.9). Layout disambiguation is not the only purpose of the virtual edges; they can supply constraints, e.g. minimal spacing between edges $n5$ and $n6$ in Figure 4.9, or speed up the processing by providing layout hints to the tool. Virtual edges are also used as 'pins', i.e. they aid in merging of two (or more) graphs into a single design.

Similar to the edges, nodes can also carry special attributes modifying their interpretation. One example is a splitter node ($n1$ in Figure 4.10, top). It translates into a non-valve intersection of the control and the flow channels. In this case, the node also has *width* and *length* attributes, defining the dimensions of the splits. Another example is a chamber node ($n2$ in Figure 4.10, bottom). Its purpose is to represent a round chamber at the location of the node $n2$ with a radius, defined by an attribute.

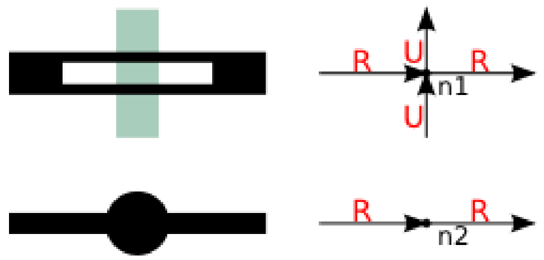


Figure 4.10: Special nodes (n1,n2) and corresponding layouts. A splitter (top, $n1$) and a reaction chamber (bottom, $n2$).

Besides the design graph, the tool also utilizes a layer table and a technology file. The first table is used for rendering purposes, e.g. which color to use for a certain layer. The technology file contains design rules information (e.g. minimum width or spacing) on a per-layer basis. It also holds the symbolic models for the resistances and the capacitances of the channels.

4.4.2 Design composition

While it is possible to build a design relying on the graph representation alone (the tool is agnostic about the source of the graph, as long as it abides by the rules), for larger designs, it makes more sense not to work on a node-edge level, but use a higher level of abstraction. In this section, we will show a simple approach to creating large structured designs using a small set of primitives and instructions.

On this level of abstraction, a primitive is a *block*, a graph with pins. Pins are edges

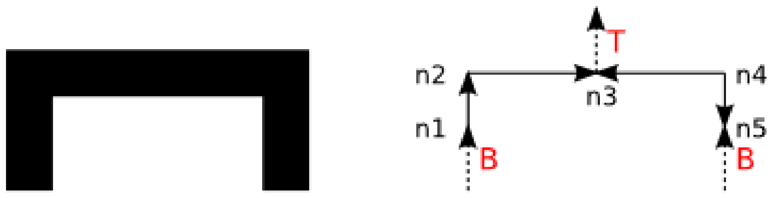


Figure 4.11: Binary fork block layout(left) and graph representation(right). Dotted edges are pins with red labels showing the *side* attribute: nodes $n1$ and $n5$ have a bottom pins and $n3$ has a top pin.

with `side` (indicating the layout side the pin is on), `direction` and `layer` attribute. The purpose of pins is to aid in blocks' merging, so they don't have widths or lengths. The user can apply unitary transformations to the blocks, as well as replicate and combine blocks together. A binary fork block is shown in Figure 4.11. It has three pins: one on top and two on the bottom side.

Unitary block operations are rotation, mirroring, filtering and copying. The first two are geometric transformations, which require modification of the `direction` and `side` (for pins) attributes only (see Table 4.1), keeping the structure of the graph unchanged. Filtering removes edges (and nodes) that satisfy certain conditions, e.g. they are on a certain layer. Finally, copying makes a copy of a block keeping the

	U	**D**	**L**	**R**
CW rotation	L	R	U	D
CCW rotation	R	L	D	U
x-mirror	U	D	R	L
y-mirror	D	U	L	R

Table 4.1: Translation of direction of an edge for different transformations. E.g. for an x-mirroring, *left* and *right* will swap and both *up* and *down* remain the same.

attributes but renaming the node names. Typically, the renaming is just adding a prefix or a postfix to the name.

Combination of the blocks is a graph join operation, which in general, requires specifying the nodes to be merged. In our case, however, it is possible to use topology information and pins to significantly simplify the process. Combination of the blocks in our framework is done by stacking them side-by-side or on top of each other.

Consider an example shown in Figure 4.12. We start with a binary fork, copied it,

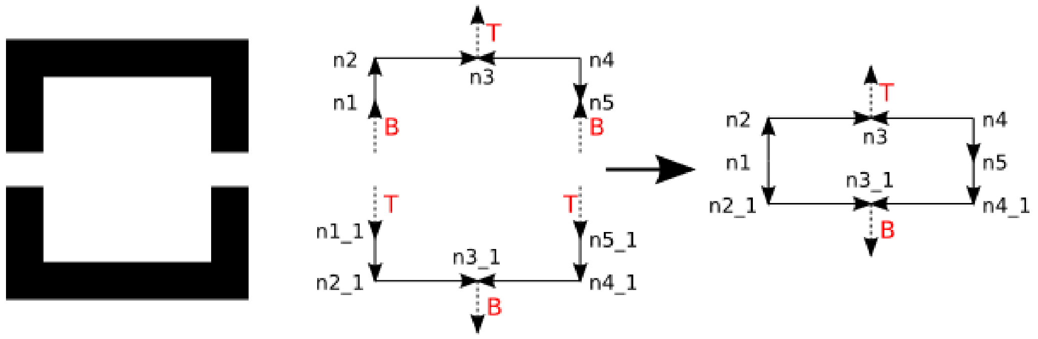


Figure 4.12: Vertical stacking of the two binary forks. One fork is a flipped copy of the other one. Bottom pins of the top block are matched with the top pins of the bottom block. Corresponding nodes are merged and redundant pins eliminated.

mirrored it and relabelled the nodes by adding a `_1` postfix. Now, we want to connect the two forks to make a rectangle. On a block level, it is achieved by a single command `vstack`, which does the following.

First, it decides which pins on which side are to be connected, e.g. if block `A` is on top of block `B`, bottom pins of `A` should be connected to top pins of `B`. In our example, these are `n1`, `n5` and `n1_1`, `n5_1`. Next, the tool determines the correspondence between the pins by sorting them either topologically or geometrically.

Topological sorting of the pins produces an ordering by relying on the topology information only. Consider a path from `n1` to `n5`; the directions of the edges are: up, right, right, down. Obviously, `n1` is to the left of `n5`. Similarly, the tool deduces that `n1_1` is to the right of `n5_1` and, therefore, they should be connected to `n1` and `n5` respectively. Then, the pin edges are deleted and the corresponding nodes are merged; `n1_1` and `n5_1` become `n1` and `n5` respectively.

Topological ordering, however, isn't always possible, e.g. in Figure 4.13 the path be-

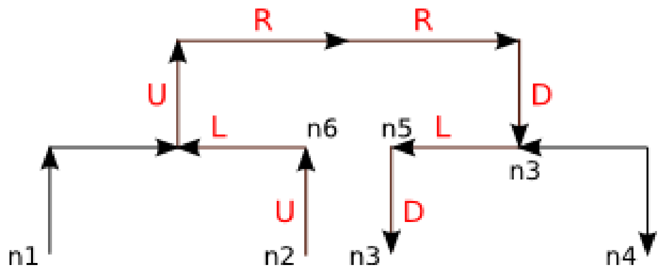


Figure 4.13: It isn't possible to sort the nodes n1, n2, n3 and n4 topologically; the only path between n2 and n3 contains both right and left turns, making the relative placement of the two nodes ambiguous. To enable the ordering, one can add a virtual edge between n5 and n6 or from n1 to n4 through n2 and n3.

tween nodes `n2` and `n3` is: up, left, up, right, right, down, right, left. The ambiguity can be resolved by adding a virtual edge between `n5` and `n6`; the path turns into up, right, down. Another option is to add a series of virtual edges from `n1` to `n4` through `n2` and `n3`. If the pins can't be sorted topologically, the tool creates a temporary feasibility problem for block using design rule check and consistency constraints only. The solution is then used to determine the ordering of the nodes.

Building an array of blocks is merely a repetition of `stack` commands. To create an $M \times N$ array, the tool makes a row block of M blocks by copying and horizontally stacking them $M - 1$ times. The row block is then copy-stacked vertically $N - 1$

times.

Simple yet powerful, the operations showed in this section can produce a wide variety of structures. In the next section, we'll use these functions to create more complex building blocks, trees and demuxes.

4.4.3 Dynamic blocks

A distribution tree is a structure that has an outlet on one side and multiple outlets on the other side. Its purpose is either to feed a lot of outputs with a single source (e.g. same reagent into multiple reaction chambers) or provide a path for a fluid to the outlet (if the chambers are addressable). One of the benefits of using a tree instead of connecting all channels in parallel to a single line is a much more balanced performance (e.g. flow rate deviation among the output channels).

The most common types of trees in microfluidic design are binary, trinary (or ternary)

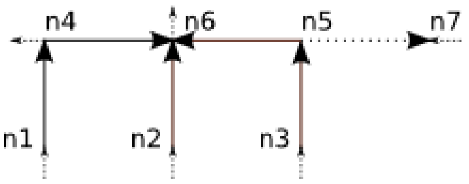


Figure 4.14: Trinary fork block. All the nodes except for $n5$ have pins attached to them (dotted edges with small arrows). Pins $n7$ and $n4$ are used to assemble a row. The rest are to stack the rows. ($n5$, $n7$) is a virtual edge.

and the combination of the two. Assembly of the distribution trees is an easy task for the tool; it is done by building tiers as rows of forks and then stacking them vertically. Let's assemble a simple trinary tree with $N = 3^k$ outputs. We begin with a single trinary fork (see Figure 4.14 block as `core_block` and perform the following sequence of operations:

```
// initialize with the last tier
tree = array(core_block , 3^k,1)
for i=k−1 to 1:
```

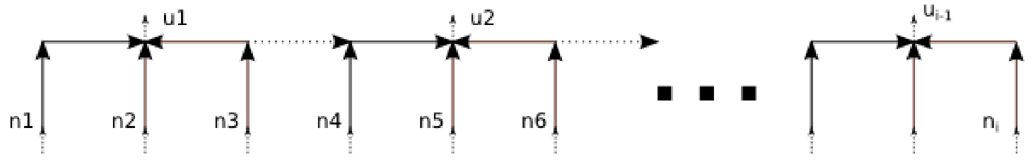


Figure 4.15: Trinary tree tier. Pins n_1 to n_k will be connected to the existing *tree* block. Pins u_1 to $u_{k/3}$ will be the new pins of the block.

```
row = array(core_block, 3^i,1)
// placing row on top of tree
tree = vstack(tree, row)
```

During each iteration of the loop, a `row` is generated by stacking 3^i forks together (see Figure 4.15). The loop repeats $k-1$ times and produces a tree block with one pin on the top and N pins on the bottom.

In the similar way, the tool can build a multiplexor-demultiplexer, which is essentially

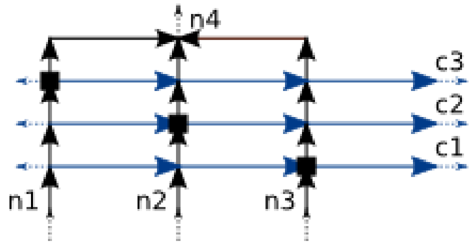


Figure 4.16: Trinary mux core block. Black and blue edges are flow and control channels respectively. $n1$ to $n4$ are flow pins for vertical stacking. $c1$ to $c3$ are control pins to assemble a row. Squares denote valve nodes.

a distribution tree with control channels added to it. To assemble the mux, we perform the same sequence of actions, but this time with a different core block (see Figure 4.16). Again, each iteration of the loop produces a row of trinary forks with three control lines across the tier. As the output, we get a block with one top pin, N bottom pins, and $3\log_3 N$ left and right pins (3 for each side of a tier).

Now, we can compose a sample design, similar to the one of the cell culture chip:

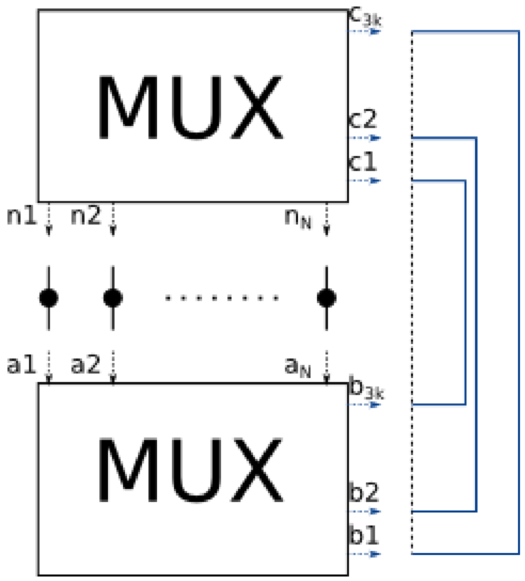


Figure 4.17: Addressable chambers chip. MUX blocks are multiplexors. Lines with a circle in the middle are flow channels with a reaction chamber.

```
chambers = array(rotate(chamber, 90), N)
mux1 = mux(N)
mux2 = mirror_x(mux1)
connector = bracket(3*k)
design = vstack(mux1,chambers)
design = vstack(design, mux2)
design = hstack(design, connector)
```

First we create an array of N chambers by using a chamber block (see Figure 4.10) rotated by 90 degrees. Then, we create a top multiplexer `mux1` with N inputs and a mirrored one, `mux2` to be placed at the bottom. Next, we create a connector, which is a set of channel brackets (see Figure 4.17, on the right). Finally, we stack `mux1`, `chambers` and `mux2` vertically. The resulting block is then stacked with the `connector` to produce the design.

4.4.4 Geometrical constraints

Geometrical constraints restrict the design in a way that guarantees a valid and manufacturable layout. They deal with width and length variables only and can be divided into 3 categories: consistency, spacing and width constraints. The purpose of the first set of constraints is to ensure consistency among the values of the lengths, spacing constraints set the minimal distance between neighbouring channels and the latter category establishes the lower and upper boundaries on the widths.

In our approach, we don't deal with absolute coordinates, but with increments as variables. To obtain (x_n, y_n) coordinates of a node n, we need to summation all the increments along a path from the zero node (a node with $(0,0)$ coordinates) to n:

$$\begin{bmatrix} x_n \\ \\ y_n \end{bmatrix} = \sum_{e \in path(0,n)} \begin{bmatrix} \Delta x_e \\ \\ \Delta y_e \end{bmatrix},$$

where $(\Delta x_e, \Delta y_e)$ is either $(\pm l_e, 0)$, or $(0, \pm l_e)$ and l_e is the length of the edge e. For

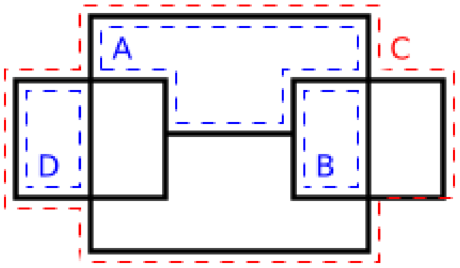


Figure 4.18: Cycles in a layout. A, B and D are minimal cycles, since they don't contain any other cycles. C isn't a minimal cycle, since it contains cycles A, B and D.

a nontrivial design, $path_n$ is non-unique and if the lengths are consistent, we will get the same node coordinates regardless of the chosen path:

$$\sum_{e \in path_a(0,n)} \begin{bmatrix} \Delta x_e \\ \\ \Delta y_e \end{bmatrix} = \sum_{e \in path_b(0,n)} \begin{bmatrix} \Delta x_e \\ \\ \Delta y_e \end{bmatrix} \quad \forall n, a, b$$

In this form, the equation almost certainly won't be suitable for the GP framework, since both of the sides contain more than one addend. To address the issue, we'll transform the system into:

$$\sum_{e \in c} \begin{bmatrix} \Delta x_e \\ \Delta y_e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \forall cycles\ c \in S,$$

where c is a cycle in a topological sense, i.e. a path that starts and ends at the same node and S is a set of all minimal cycles, i.e. cycles that won't contain any other cycles in a layout (see Figure 4.18). To find such cycles, we use a wall follower maze traversal algorithm. If we think of channels as walls (black lines in Figure 4.18) and start the traversal in any of the loops, we'll eventually return to the starting point (otherwise, it wouldn't be a planar layout). Thus, the algorithm is to go through all the edges and do the maze traversal for both sides of the edge.

Two types of the consistency equations can be transformed into GP constraints:

$$\begin{aligned} l_i &= l_j,\ or \\ \sum_i l_i &= l_j. \end{aligned}$$

While the former equation is fully compatible with GP, the latter will work only if changed into:

$$\sum_i l_i \leq l_j.$$

The constraint will turn into an equality, since the l's satisfying the inequality are

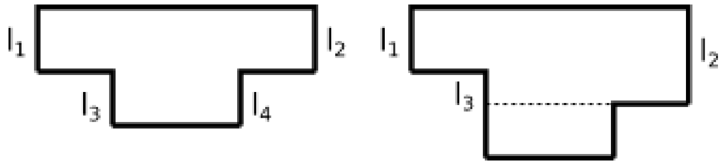


Figure 4.19: Breaking a cycle (left) into 2 simple ones (right) with a virtual edge to produce GP compatible consistency constraints.

suboptimal.

Consider the loops in Figure 4.18. The cycle D will produce two equalities (one for the x and y axis). The consistency constraints for B will contain one equality (for x) and one inequality (for y). Finally, the cycle A will have one inequality for the y axis. For the x axis, however, the appropriate constraint is: $l_1 + l_2 = l_3 + l_4$ (see Figure 4.19, left). The problem comes from the relative placement ambiguity, which we talked about in the Graph Representation section. Here, the general solution is the same: use a virtual edge for disambiguation purposes (see Figure 4.19, right) or, in other words, to partition a cycle into two simpler ones. In practice, however, we often reuse blocks and/or variables to produce a more aligned/symmetrical layout, e.g. we use l_1 instead of l_2 and l_3 instead of l_4, resulting in a GP valid constraints for the loop A.

Next type of the geometrical constraints is spacing constraints. Their purpose is to

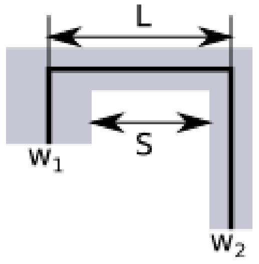


Figure 4.20: Spacing S of a Π-shaped channel equals to $L - w_1/2 - w_2/2$.

ensure proper spacing between the channels and unlike consistency equations; they deal with both w's and l's. The tool searches for all Π-shaped (see figure 4.20) configurations of the wires and produces the spacing inequality

$$L - w_1/2 - w_2/2 >= S_{min},$$

which is clearly GP compatible: $L >= S_{min} + w_1/2 + w_2/2$. The value of the S_{min} depends on the layers the channels are in and is obtained from the design rules table.

To find all the Π-shaped paths, one can go through all the nodes and check all the paths that start at this node. This naive approach is, however, very slow due to the large number of forks in a typical design. However, it is clear that Π structures can only occur in a minimal loop and, luckily, we already have a collection of such loops from the consistency constraints generator. Therefore, the algorithm reduces to checking the paths inside all the minimal loops.

Finally, the width constraints are merely lower and upper bound on the w's of the channels. The tool goes through all the edges of the design graph and produces the inequalities:

$$W_{min} \leq w_i \leq W_{max},$$

where the values of W_{min} and W_{max} depend on the edge's layer and are obtained from the design rules table.

4.4.5 Flow constraints

In this section, we will describe the way the tool generates constraints for the current rates Q's through the flow channels, i.e. the GP equivalent of the hydraulic Kirchhoff's laws:

$$\sum_{i \in fork} Q_i = 0 \tag{4.16}$$

$$\sum_{i \in io\ path} Q_i \times R_i(w,l) = U_{inlet}\ \forall io\ path, \tag{4.17}$$

where Q_i is the flow rate through the channel i, $R_i = \alpha w_i^{\beta} \times l_i$ is its resistance, U_{inlet} is the potential function at the inlet and *io path* is a path between the inlet and the outlet. Flow constraints, unlike geometric ones, aren't applied to the whole design graph, but to its *configurations*. A configuration is a network of flow channels, when some (if any) valves are sealed. For example, a typical configuration of a MUX is when there's only one path from the inlet to the outlet. In contrast, the only configuration of the distribution tree is when all channels are open. From the tool's perspective, a configuration is a graph, where blocked valve nodes are removed (see Figure 4.21). By design, flow directions in a configuration don't depend on the dimensions of the graphs; current goes from inlet to the outlet for any path.

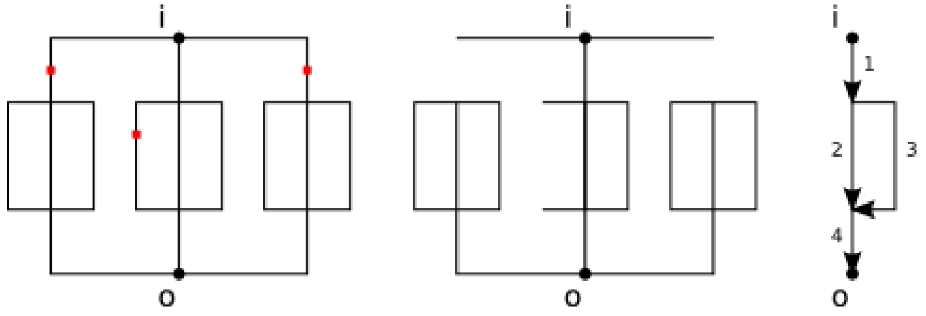


Figure 4.21: From a configuration to a flow multigraph. Left: initial configuration, red nodes are sealed valves. Letters 'i' and 'o' mean the inlet and the outlet respectively. Middle: blocked nodes are removed. Right: flow multigraph; edges without flow are removed, edges without forks are merged.

This property is used to construct the flow multigraph (see Figure 4.21, right), i.e. a graph where more than one edge is allowed between a pair of nodes. First, a set of all paths from the inlet to the outlet is generated. Then, edges not touched by these paths are removed (along with the nodes). Finally, each path is traversed from the inlet to the outlet, and edges are directed the same way along the path. A conflict between directions indicates a wrong configuration and an error message is produced. Finally, nodes that have only 2 neighbours are removed, their adjacent edges merged and resistances summed.

At this point, we have a multigraph with edges' directions corresponding to the flow directions and nodes corresponding to forks. Now, we assign each edge with a unique non-negative symbol Q_i, representing the flow rate through this edge. We apply Equation 4.16 to all the nodes n and obtain the following system:

$$Q_n = \sum_{j \in n} Q_j \ \forall nodes\ n. \tag{4.18}$$

Although all Q's are non-negative, this system isn't GP-compatible. Neither can we relax the system by substituting $=$ with $\geq$, since larger Q's are better. To enforce the constraints, we'll treat Equations 4.18 as a set of substitution rules. By recursively applying these rules to them, we arrive at:

$$Q_i = \sum_{j \in atoms} A_{ij} Q_j \tag{4.19}$$

where *atoms* is a set of Q's that can't be expressed as a sum of other flow rates and A_{ij} are non-negative coefficients. Finally, we combine Equations (4.17) and (4.19) into:

$$\sum_{i \in io\ path} \sum_{j \in atoms} A_{ij} Q_j^c R_i(w,l) \leq U_{inlet}, \tag{4.20}$$

a valid (relaxed) GP constraint, equivalent to the Kirchhoff's laws for a configuration c (superscript of Q_j^c). Now we can use Q's in the cost function, e.g. to maximize the minimum current among all configurations:

$$\begin{aligned} Q_{min} &\leq Q_i\ \forall i && (4.21) \\ maximize &\quad Q_{min}. && (4.22) \end{aligned}$$

Or we can use Q in propagation delay constraints.

4.4.6 Propagation delay

Consider a mux configuration with a single flow path from the inlet to the outlet. How long does it take the fluid to reach the outlet (or the reaction chamber) after the pressure is applied? The answer is $T_{ch} = V_{ch}/Q_{ch}$, where V_{ch} is the volume (either up to the chamber of the whole way to the outlet) of the channel. The propagation delay is easily expressed through the GP variables:

$$T_{ch} = \frac{1}{Q_{ch}} \sum_{i \in ch} S_i(w_i) l_i,$$

where S_i is the cross-section of the i-th channel that depends on its width w_i and its shape. To construct the expression, the tool merely traverses the channel and uses appropriate $S(w)$ from the technology file.

In case of multiple currents, the total flow delay can be expressed in the following form:

$$T_{ch} = \max_{path \in paths} T_{path},$$

where *paths* is the set of all paths from the inlet to the outlet with non-zero currents and T_{path} is defined as:

$$T_{path} = \sum_{i \in path} \frac{S_i(w_i) l_i}{Q_i},$$

where i is a segment in the *path* and Q_i is the flow rate in this segment. Unlike the pressure drop case, we cannot substitute Q_i's with sums of currents, since it breaks the GP rules. However, if the tree is *balanced*, all forks and merges happen synchronously and the delay time can be simplified into:

$$T_{conf} = V_{total}/Q_{inlet},$$

i.e. the total volume divided by the inlet flow rate. We'll use this fact to upper bound the propagation delay for a non-balanced configuration:

$$Q_{inlet} = \sum_{j \in atoms} A_{ij} Q_j \leq Q_{min} \sum_{j \in atoms} A_{inlet,j},$$

where Q_{min} is defined by Equation (4.22). Therefore:

$$T_{conf} \leq \frac{V_{total}}{k Q_{min}},$$

where $k = \sum_{j \in atoms} A_{inlet,j}$. The tool constructs the expression in the exact same way as in the single current case.

Propagation delay isn't the only source of the operational delay. Change of network configurations requires pressurising and de-pressurising the control channels. Since it takes time for the pressure to propagate to the valve, the control channel's delay should be considered as well.

4.4.7 Pressure propagation delay

By definition, the pressure propagation delay is the time from the moment pressure is applied to the inlet and the moment the valve is sealed (i.e. the pressure at the valve reaches a certain level). We estimate the value using Elmore's delay formula:

$$T_{Elmore} = \sum_{j \in G} C_j \sum_{i \in path_j} R_i,$$

where j's are edges in the control graph G, C_j and R_j are the capacitance and the resistance of the edge j, and $path_j$ is the path from the inlet to the edge j.

The construction of the expression is straightforward. For each control edge j, the tool computes the path from the inlet and derives its resistance R_{path} multiplied by C_j. The final expression is the sum: $\sum_i C_i R_{path_i}$. Due to the charge-discharge asymmetry, expressions of the closing time T_{on} and the opening time T_{off} use different expressions of the R (taken from the technology file).

At this point, we've got all the variables (and the respective constraints) to produce the cost function.

4.4.8 Cost function, solver and rendering

The user has full control over the cost function; they can make and posynimal combinations of the design variables, such as flow rates Q's, delays T's, lengths, widths and so forth. For example, we have a mux and we want to minimize the maximum flow delay among all mux configurations:

$$minimize \ \max(T_{ch1}, T_{ch2}, \cdots, T_{chn}).$$

Or we can minimize the total time it takes to go through all the reaction chambers sequentially, i.e. switch the mux, fill the channel, switch the mux again, fill another channel and so forth:

$$minimize \ \sum_c (T^c_{flow} + T^c_{Elmore}),$$

where c is a mux configuration.
In case of the cell culture chip, the sequence also contains flushes between the mux switches:

$$minimize \sum_c (T^c_{flow} + T^c_{Elmore} + T^c_{flush}).$$

After the cost function is defined, the optimization problem is ready to be solved. Since all the constraints and the cost function have symbolic representation, it is straightforward to construct the solver's input by simply printing all the expressions into one text file. After the file is generated, it is given to the CVX solver that does all the computations and returns the optimal values of all the variables as well as the value of the cost function.

Once we have the values of lengths and widths, the tool needs to reconstruct the absolute coordinates of the nodes, so that the design layout can be rendered. It is done by executing breadth-first graph traversal from the node zero (the one that has $(0, 0)$ coordinates). For each graph node, the operation produces a path from it to the node zero and the summation of the Δx and Δy increments along the path provides the coordinates. When the absolute coordinates are derived, the tool renders each edge i as a rectangle with length l_i, width w_i and appropriate corner coordinates (x_i, y_i).

4.5 Synthesized chip

We used the developed tool to produce an optimized design of the bacteria culture chip. The block representation of the design is shown in Figure 4.22. By using only 2 demultiplexers instead of 4, we saved about 15% of space and reduced the control delay by about 50%. To maximize space utilization, we used 1-to-3 demultiplexers only (as opposed to both 1-to-2 and 1-to-3 demuxes in the mammalian culture chip). As a result, we were able to fit 324 cell chambers and 162 bypass channels on the die.

The optimization of the design via geometric programming was done in CVX[31] with MOSEK[32] as the back-end solver. The solver took about 10 minutes to converge to a globally optimal solution. In the optimized design, all the control channels

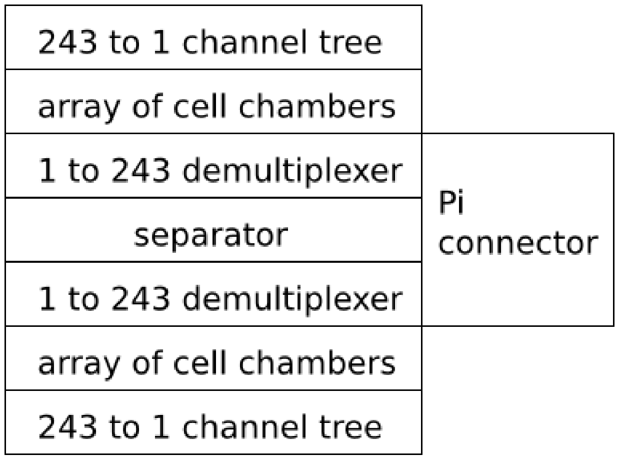


Figure 4.22: The block diagram of the synthesized part of the bacteria culture chip. 7 blocks are vertically stacked: trees, arrays of chambers and demultiplexers. The Pi-connector connects the control channels of the demultiplexers. The separator is a blank space needed to fit the rest of the chip's channels.

were set to the maximum allowed width (100μ), while the flow channels varied in width. Figure 4.24 depicts the width of a channel depending on its position in the demultiplexers hierarchy: the root of the demux has the highest possible width and the closer is the channel to the chamber array, the narrower it becomes. Leaf channels are 30μ wide. The sizing of the flow channels didn't introduce much of a gain in performance; the reduction of the propagation delay was on the order of 10% compared to the case of equally wide channels.

The bacteria culture chip (see Figure 4.23) manufactured in the microfluidics foundry at Stanford University using the synthesized design was topologically correct: there were no short-cuts and no breaks. The result confirms the ability of the synthesis tool to abide by the design rules. We also compared the flow propagation delays of the optimized version of the chip vs a version with equally wide flow channels. The optimized chip showed a marginal (2%) decrease in delay, which isn't surprising given the model's accuracy of 15%.

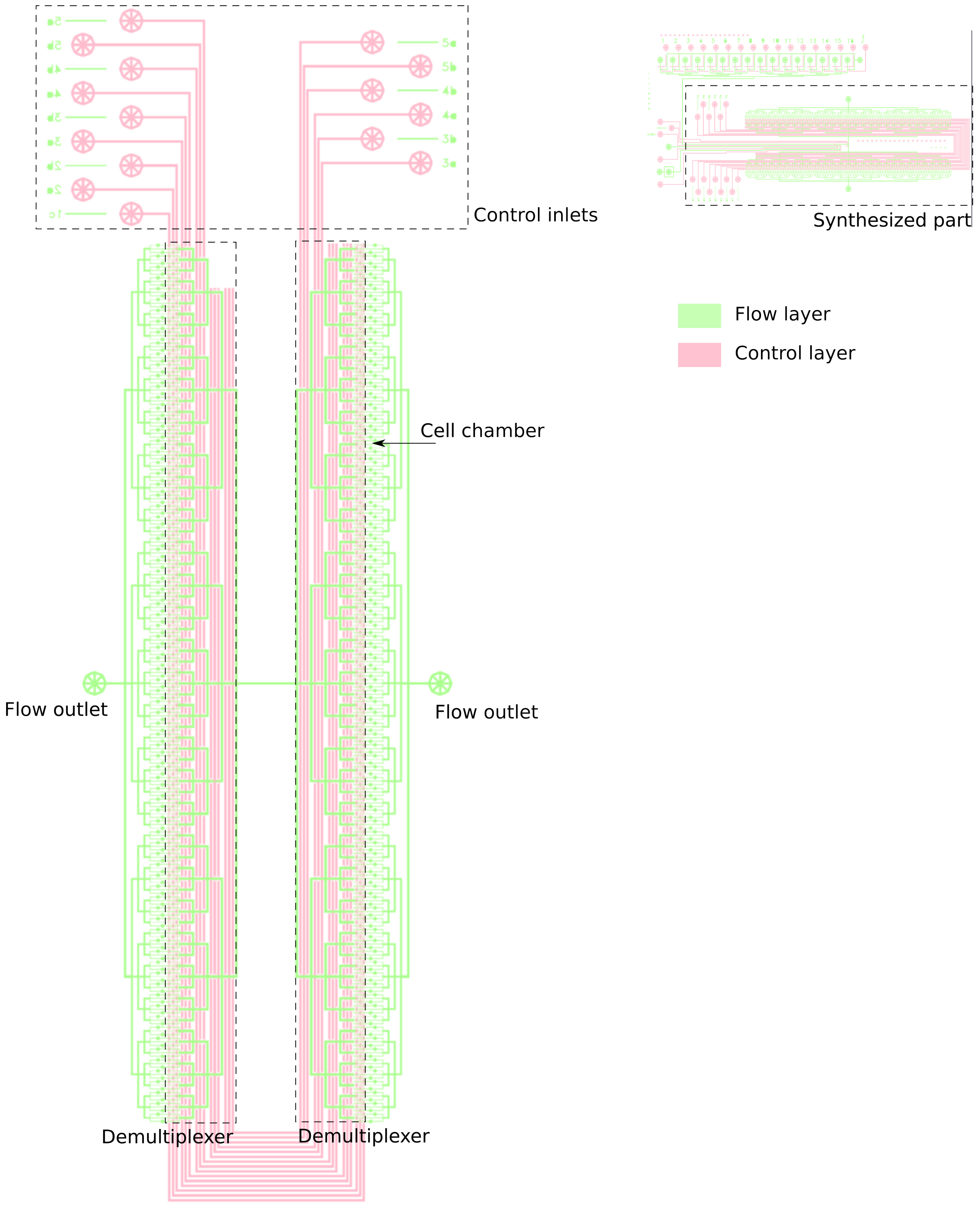


Figure 4.23: The full bacteria culture chip (right) and its synthesized part (left). Major components are outlined: control and flow terminals, 1-to-243 demultiplexers and 324 cell chambers.

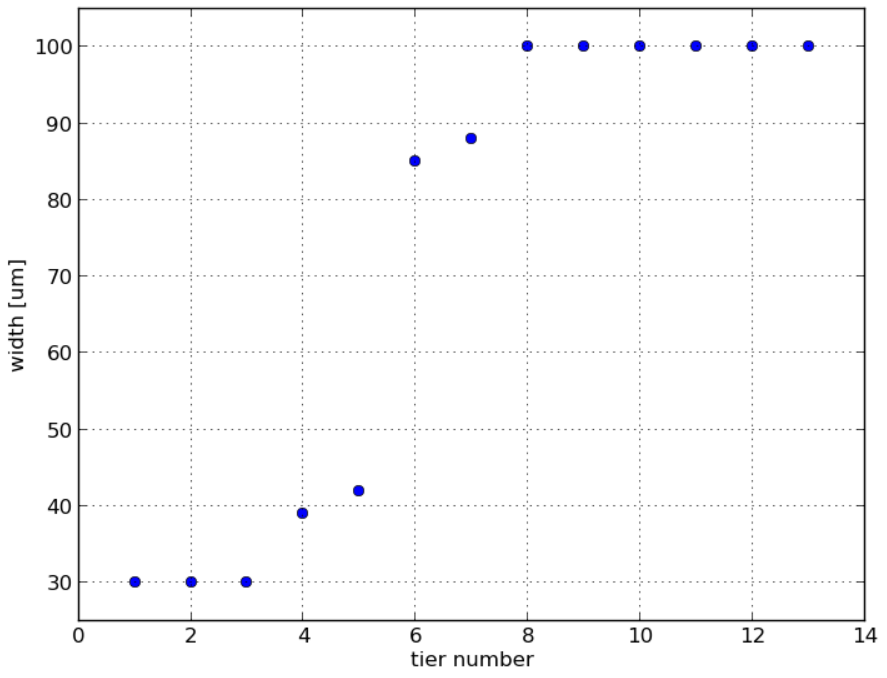


Figure 4.24: Widths of the synthesized demultiplexers' tiers. The root of the tree has the widest possible channels (100μ). The channels become narrower with proximity to the cell chambers, hitting the lowest width of 30μ.

Chapter 5

Conclusion

Soft-litho microfluidics and integrated circuits have much in common. In our research, we were guided by these similarities in every stage. Hydraulic-electric analogy is a tool known and used for a while in both directions; intuitive fluid behaviour is used to explain electricity and well-studied Kirchhoff's laws are applicable to microfluidic networks.

Surprisingly, a microfluidic channel was also analogous to a MOS transistor; a potential function approach used to model a non-linear I-V relationship inside a transistor turned out to be applicable to the non-linear flow-pressure dependence inside a channel. Compact models that we derived using this idea gave us a good explanation of a charge-discharge asymmetry of a long channel and the non-trivial flow-frequency relation of a peristaltic pump. For both MOS and microfluidic channels, this potential function approach enabled Elmore's delay approximations through RC time scaling idea.

From the validating experiments, we learned how to measure on-chip flow rates and pressures using just existing microfluidic structures, dyed fluids and image processing techniques. We also learned how to calibrate the model to address material properties' uncertainties.

A problem of optimal channel sizing for microfluidic networks didn't seem to have a good solution at first. However, after looking at solution of the similar wire sizing problem in IC design, we started pushing in the same direction and ultimately were able to frame our problem as a geometric program. We learned that in optimization problems, certain Kirchhoff's laws and geometry-related equations can be relaxed without side effects. Experiments on design optimization revealed deeper relations between design dimensions and its performance, such as the effect of minimal spacing requirement on the flow rate of a demultiplexer.

Finally, the concept of parametric building blocks used in circuit design was applicable to the microfluidic as well. Typical IC abstraction, like pins, bounding boxes and virtual layers found their way into the microfluidic synthesis tool, as well as block compositions rules. Using these concepts we were able to construct complex microfluidic chips from a high level abstraction, and have the resulting chip be well optimized.

Bibliography

[1] Marc A. Unger, Hou-Pu Chou, Todd Thorsen, Axel Scherer, and Stephen R. Quake. Monolithic microfabricated valves and pumps by multilayer soft lithography. *Science*, 288(5463):113–116, 2000.

[2] Zhanhui Wang, Min-Cheol Kim, Manuel Marquez, and Todd Thorsen. High-density microfluidic arrays for cell cytotoxicity analysis. *Lab Chip*, 7:740–745, 2007.

[3] Rafael Gmez-Sjberg, Anne A. Leyrat, Dana M. Pirone, Christopher S. Chen, and Stephen R. Quake. Versatile, fully automated, microfluidic cell culture system. *Analytical Chemistry*, 79(22):8557–8563, 2007. PMID: 17953452.

[4] Polly M. Fordyce, Doron Gerber, Danh Tran, Jiashun Zheng, Hao Li, Joseph L. DeRisi, and Stephen R. Quake. De novo identification and biophysical characterization of transcription-factor binding sites with microfluidic affinity analysis. *Nature Biotechnology*, 28(9):970–975, September 2010.

[5] Mingzhe Gan, Jing Su, Jing Wang, Hongkai Wu, and Liwei Chen. A scalable microfluidic chip for bacterial suspension culture. *Lab Chip*, 11:4087–4092, 2011.

[6] Peter A. Sims, William J. Greenleaf, Haifeng Duan, and X. Sunney Xie. Fluorogenic DNA sequencing in PDMS microreactors. *Nature Methods*, 8(7):575–580, July 2011.

[7] Jinyi Wang, Guodong Sui, Vani P. Mocharla, Rachel J. Lin, Michael E. Phelps, Hartmuth C. Kolb, and Hsian-Rong Tseng. Integrated microfluidics for parallel screening of an in situ click chemistry library. *Angewandte Chemie International Edition*, 45(32):5276–5281, 2006.

[8] Luis M. Fidalgo and Sebastian J. Maerkl. A software-programmable microfluidic device for automated biology. *Lab Chip*, 11:1612–1619, 2011.

[9] Stephen R Quake Jong Wook Hong. Integrated nanoliter systems. *Nature Biotechnology*, (10):11791183, 2003.

[10] Todd Thorsen, Sebastian J. Maerkl, and Stephen R. Quake. Microfluidic large-scale integration. *Science*, 298(5593):580–584, 2002.

[11] Xia Lou, Jaehoon Chung, Young-Ji Kim, Il-Joo Cho, and Euisik Yoon. A fully addressable micro-array chip integrated with cascade multiplexors for selective cell loading and retrieval. In *Solid-State Sensors, Actuators and Microsystems Conference, 2009. TRANSDUCERS 2009. International*, pages 1301 –1304, june 2009.

[12] Jaehoon Chung, Young-Ji Kim, and Euisik Yoon. Highly-efficient single-cell capture in microfluidic array chips using differential hydrodynamic guiding structures. *Applied Physics Letters*, 98(12):123701, 2011.

[13] Jitendra S Kanodia Emily Gong Stanislav Y Shvartsman Hang Lu Kwanghun Chung, Yoosik Kim. A microfluidic array for large-scale ordering and orientation of embryos. *Nature Methods*, (2):171176, 2010.

[14] Vincent Studer, Giao Hang, Anna Pandolfi, Michael Ortiz, W. French Anderson, and Stephen R. Quake. Scaling properties of a low-actuation pressure microfluidic valve. 95(1):393–398, 2004.

[15] I. Klammer, A. Buchenauer, G. Dura, W. Mokwa, and U. Schnakenberg. A novel valve for microfluidic pdms-based systems. In *Micro Electro Mechanical Systems, 2008. MEMS 2008. IEEE 21st International Conference on*, pages 626–629, 2008.

[16] Nikolas Chronis, Gang Liu, Ki-Hun Jeong, and Luke Lee. Tunable liquid-filled microlens array integrated with microfluidic network. *Opt. Express*, 11(19):2370–2378, Sep 2003.

[17] A. Sarkar and G. Jayaraman. Nonlinear analysis of oscillatory flow in the annulus of an elastic tube: Application to catheterized artery. *Physics of Fluids (1994-present)*, 13(10), 2001.

[18] Bozhi Yang and Qiao Lin. A compliance-based microflow stabilizer. *Microelectromechanical Systems, Journal of*, 18(3):539–546, 2009.

[19] Laurence W. Nagel. *SPICE2: A Computer Program to Simulate Semiconductor Circuits.* PhD thesis, EECS Department, University of California, Berkeley, 1975.

[20] K.W. Oh, K. Lee, B. Ahn, and E.P. Furlani. Design of pressure-driven microfluidic networks using electric circuit analogy. *Lab Chip*, 12(3):515–45, 2012.

[21] Marek Turowski, Zhijian Chen, and Andrzej Przekwas. Automated generation of compact models for fluidic microsystems. *Analog Integr. Circuits Signal Process.*, 29:27–36, October 2001.

[22] Y. Hsu and N. Le. Equivalent electrical network for performance characterization of piezoelectric peristaltic micropump. *Microfluidics and Nanofluidics*, 7:237–248, 2009. 10.1007/s10404-008-0380-7.

[23] Ling-Sheng Jang, Kuan Shu, Yung-Chiang Yu, Yuan-Jie Li, and Chiun-Hsun Chen. Effect of actuation sequence on flow rates of peristaltic micropumps with pzt actuators. *Biomedical Microdevices*, 11:173–181, 2009. 10.1007/s10544-008-9222-3.

[24] A.N. Chatterjee and N.R. Aluru. Combined circuit/device modeling and simulation of integrated microfluidic systems. *Microelectromechanical Systems, Journal of*, 14(1):81–95, feb. 2005.

[25] S. Burgmann, S. Groe, W. Schrder, J. Roggenkamp, S. Jansen, F. Grf, and M. Bsen. A refractive index-matched facility for fluidstructure interaction studies of pulsatile and oscillating flow in elastic vessels of adjustable compliance. *Experiments in Fluids*, 47(4-5):865–881, 2009.

[26] Philippe Marmottant. A wikibook of microfluidics. `http://en.wikibooks.org/wiki/Microfluidics/Hydraulic_resistance_and_capacity`.

[27] Comsol multiphysics. `http://www.comsol.com`.

[28] Mark A. Horowitz. *Timing Models for MOS Circuits*. PhD thesis, Stanford, CA, USA, 1983.

[29] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[30] Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi. A tutorial on geometric programming, 2007.

[31] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.0 beta. `http://cvxr.com/cvx`, September 2013.

[32] Mosek. `http://mosek.com/products/mosek/`.