Shape turnpike, numerical control and optimal design for PDEs

Gontran Lance

Shape turnpike, numerical control and optimal design for PDEs

THÈSE

présentée et soutenue publiquement le 26 Novembre 2021 pour l'obtention du

Doctorat de Sorbonne Université

Spécialité "Mathématiques appliquées"
École Doctorale Sciences Mathématiques de Paris Centre (ED 386)

par

Gontran LANCE

devant le jury constitué de :

Grégoire ALLAIRE	*Rapporteur*
Lars GRÜNE	*Rapporteur*
Olivier PIRONNEAU	*Examinateur*
Yannick PRIVAT	*Examinateur*
Hasnaa ZIDANI	*Examinateur*
Emmanuel TRÉLAT	*Directeur*
Enrique ZUAZUA	*Co-directeur*

au *Laboratoire Jacques-Louis Lions*
CNRS UMR 7598, 4 place Jussieu, 75005 France

Table of contents

Table of contents 3

Table of figures 5

Listings 7

Introduction and short description of the main contributions 9
1 Turnpike phenomenon in optimal control . . . 9
1.1 State of the art . . . 11
1.2 Contributions : shape turnpike . . . 12
2 Numerical PDE constrained optimization . . . 15
2.1 Context and objectives . . . 15
2.2 Contributions . . . 17
2.3 Applications to shape turnpike . . . 19

1 Shape turnpike for linear parabolic PDEs 21
1.1 Shape turnpike for linear parabolic equation . . . 22
1.1.1 Setting . . . 23
1.1.2 Preliminaries . . . 23
1.1.3 Main results . . . 25
1.2 Proofs . . . 27
1.2.1 Proof of Theorem 1.1 . . . 27
1.2.2 Proof of Theorem 1.4 . . . 30
1.2.3 Proof of Theorem 1.7 . . . 31
1.2.4 Proof of Theorem 1.8 . . . 31
1.3 Numerical simulations : optimal shape design for the 2D heat equation . . . 34
1.4 Further comments . . . 40
Appendices . . . 40
1.A Energy inequalities . . . 40
1.B Bathtub principle . . . 43

2 PDE-constrained optimization with `FreeFEM` and `IpOpt` 45
2.1 Preliminaries . . . 47
2.1.1 The `FreeFEM` software . . . 47
2.1.2 PDE constrained optimization . . . 50
2.1.3 Optimization and discretization strategies . . . 59

2.1.4 The optimization routine IpOpt . . . 61
2.1.5 Automatic differentiation . . . 63
2.2 Linear quadratic PDE constrained optimization . . . 67
2.2.1 Derivatives of discretized functions (*FDTO*) . . . 69
2.2.2 Discretization of continuous derivatives (*FOTD*) . . . 71
2.2.3 Inhomogeneous Dirichlet boundary conditions . . . 74
2.2.4 Automatic differentiation alternative . . . 75
2.3 Extension to time-dependent problems . . . 78
2.3.1 Implicit Euler scheme. . . . 78
2.3.2 Time discretization with FreeFEM. . . . 79
2.4 Optimization under semilinear PDE constraints . . . 81
2.5 Optimal shape design problems . . . 86
2.6 Boundary shape optimization . . . 87
2.6.1 Boundary and domain parametrization . . . 87
2.6.2 Shape optimization problem . . . 91
2.6.3 Sensitivity analysis . . . 92
2.6.4 Codes and results . . . 96
2.6.5 Further comments . . . 100
Appendices . . . 101
2.A Some FreeFEM functions . . . 101
2.B Semi-automatic differentiation and adjoint method . . . 102
2.C PDE Optimization with Python or Matlab . . . 105

3 Numerical solving of time-varying shape design problems 107
3.1 Numerical shape design for parabolic PDE models . . . 108
3.1.1 Particular case of dimension 1 . . . 108
3.1.2 Level-set method in dimension $d \geqslant 2$ . . . 110
3.1.3 Convexification method . . . 113
3.1.4 Discussion : comparison of implemented methods . . . 115
3.2 Numerical examples illustrating the turnpike phenomenon . . . 121
3.3 Further prospects . . . 125
3.3.1 Symmetries of solutions . . . 125
3.3.2 Numerical extension to semilinear PDEs . . . 125

Conclusion and perspectives 127

Bibliography 131

Table of figures

1 Turnpike illustration 10
2 Control of a heat equation by a shape acting as a source term 13
3 Sperm's motion through its oscillating flagellum 18

1.1 Relaxation occurence and control viewed as level-set 30
1.2 1D time-varying shape for Mayer case - $T = 3$ 35
1.3 Error between time and static optimal shapes with respect to the Hausdorff distance $t \mapsto d_{\mathscr{H}}(\omega_T(t), \bar{\omega})$ introduced in Theorem 1.8 35
1.4 Optimal shape's time evolution cylinder - $T = 2$ 37
1.5 Time optimal shape $T = 5$ - Static shape 37
1.6 Error between dynamical optimal triple and static one 38
1.7 Error between dynamical optimal triple and static one (1D) 38
1.8 Relaxation phenomenon : (a) $t = 0$; (b) $t = 0.5$; (c) $t \in [1,4]$; (d) $t = 4.5$; (e) $t = T$; (f) static shape 39
1.9 Error between dynamical optimal triple and static one (Relaxation case) 39

2.1 Example (2.1) : (a) L-mesh; (b) solution y of (2.2) for $u = 1$ 48
2.2 Convergence curves : (a) objective function; (b) convergence criterion 63
2.3 Convergence curves of several methods 77
2.4 (a) initial square mesh; (b) 3D mesh with `buildlayers` 80
2.5 Convergence curves for semilinear case : (a) objective function; (b) convergence criterion 85
2.6 Fluid domain : (a) initial square; (b) torus after periodization. 88
2.7 Fluid domain mesh : (a) initial; (b) modified for $f(x) = 0.1\sin(3\pi x)$. 89
2.8 Velocity of Stokes fluid u for $f(x) = 0.1\sin(3\pi x)$ and $v_D = -1$ 90
2.9 Optimal solution of (2.52,2.53,2.54,2.55,2.57) for $M_1 = 0.4$ and $M_2 = 5.0$ on : (a) rough mesh; (b) fine mesh. 99
2.10 Optimal solution of (2.52,2.53,2.54,2.55,2.57) for $M_1 = 0.4$ and : (a) $M_2 = 2$ and $J \approx 0.07$; (b) $M_2 = 10.0$ and $J \approx 0.16$; (c) $M_2 = 50.0$ and $J \approx 0.19$; 100
2.11 Solution with no curvature constraint ($M_2 = +\infty$) and $J \approx 0.196$ 101

3.1 Numerical turnpike in dimension 1 109
3.2 Optimal static shape : level-set versus convexification 115
3.3 Optimal solution when relaxation : level-set versus convexification 116
3.4 Optimal shape for several elliptic operators 117
3.5 3D optimal static shape : level-set versus convexification 118

3.6 Inverse problem optimal solution : level-set versus convexification 119
3.7 Time-varying optimal shape at several time 122
3.8 Error between both static and time optimal triples 122
3.9 Time-varying optimal shape - Example 1 123
3.10 Time-varying optimal shape - Example 1 123
3.11 Time-varying optimal shape - Example 4 124
3.12 Time-varying optimal shape - Example 3 124
3.13 Symmetries of static optimal shapes 125
3.14 Symmetries of static optimal designs for semilinear models 126
3.15 Symmetries of time-varying optimal designs 126

Listings

2.1 Mesh generation with buildmesh . . . 48
2.2 Poisson solution by solving the variational formulation . . . 48
2.3 **varf** command . . . 49
2.4 Poisson solution with finite element matrices . . . 49
2.5 Calling IpOpt in FreeFEM . . . 61
2.6 Automatic differentiation in direct mode . . . 64
2.7 Automatic differentiation in reverse mode . . . 66
2.8 Finite element matrices involved in (2.30,2.31) . . . 69
2.9 LQ state equation . . . 71
2.10 LQ adjoint equation . . . 72
2.11 LQ gradient's interpolation . . . 72
2.12 LQ cost function . . . 73
2.13 Derivative of the LQ cost function . . . 73
2.14 State equation . . . 74
2.15 Dirichlet map . . . 74
2.16 AMPL : "file.mod" . . . 76
2.17 AMPL : "file.dat" . . . 76
2.18 AMPL : "file.run" . . . 77
2.19 3D mesh cylinder with 2D mesh basis . . . 80
2.20 Semilinear state equation with a fixed point method . . . 82
2.21 Semilinear state equation with Newton method . . . 83
2.22 Semilinear adjoint equation . . . 84
2.23 AMPL : "file.mod" - semilinear case . . . 85
2.24 1D finite element space and movemesh command . . . 88
2.25 Stokes state equation . . . 91
2.26 Regularization of f' with L^2 projection with $\mathbb{P}_1$ finite elements . . . 91
2.27 Volume constraint . . . 92
2.28 Stokes adjoint system . . . 93
2.29 Matrices of first and second derivatives of f . . . 95
2.30 L^2 regularization for ϕ_1 and ϕ_2 . . . 95
2.31 Stokes gradient's interpolation . . . 96
2.32 $H^2(0,1)$-inner product's construction . . . 96
2.33 Cost function for Stokes problem . . . 97
2.34 Derivative of the cost function for Stokes problem . . . 98
2.35 Length and curvature constraints of the boundary . . . 98

2.36 Calling `IpOpt` in the Stokes problem 99
2.37 Matrix of derivative's jumps 101
2.38 Hot restart routine 102
2.39 Sparse matrices importation in Python 105
2.40 `CasADi` template for LQ PDE Optimization 105

Introduction and short description of the main contributions

Control theory can be defined as the way of acting on systems in order to make them go from an initial state to a final state. To park a car, to heat a room, etc., the possibilities of applications are at least as numerous as the ways to describe things that happen in real life. This makes it a field at the interface of several and different scientific fields (aeronautics, biology, structural calculation, economics, etc.) by mixing *fundamental* and more *applied* mathematics. In a world that is increasingly competitive and in search of performance, the question of optimizing the action on a system naturally arises and the theory of optimal control aims to provide precise answers. In parallel with the progress made in functional and numerical analysis, it is becoming easier to describe mathematically what we observe in reality. This thesis is therefore part of the broad field of optimal control of partial differential equations, both from a theoretical and numerical point of view, and more precisely in the context of optimal design, and is divided into two main contributions. First, we focus on the turnpike phenomenon that appears in optimal control and more specifically in the case of shape optimization. Second, the difficult question of numerical methods for designing time-varying optimal shapes has led to the emergence of a generalizable methodology for PDE-constrained optimization. Finally, the last part is devoted to some numerical results to give possible directions for the extension of the turnpike results.

1 Turnpike phenomenon in optimal control

Let $T > 0$ be the final time fixed and let consider a general time dependent optimal control problem

$$\begin{aligned} &\min \frac{1}{T}\int_0^T f^0(y(t),u(t))\,dt + g(T,y(T),u(T)) \\ &\text{subject to : } \begin{cases} \partial_t y = f(y(t),u(t)) \quad \forall t \in (0,T) \\ h(y(0),y(T)) = 0. \end{cases} \end{aligned} \tag{1}$$

with both *Lagrange* (integral) and *Mayer* (final) cost to minimize and some additional terminal constraints. The turnpike theory aims to reveal a quasi-stationary structure of solutions of optimal control problems (1). Moreover, we write a *stationary* problem, setting aside integrals, time derivatives and terminal cost as well as constraints, which results in an instantaneous cost

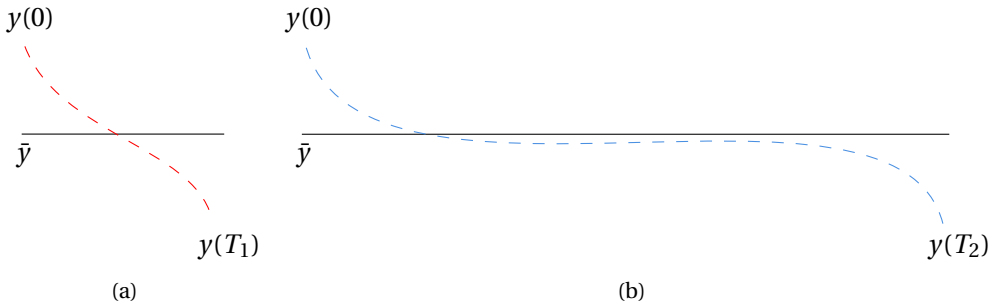


FIGURE 1 – State turnpike illustration : (a) small time behavior ; (b) large time behavior.

minimization problem under the constraint of being an equilibrium of the control dynamics :

$$\min\ f^0(y,u) \quad \text{subject to : } f(y,u)=0. \tag{2}$$

The Turnpike phenomenon was first observed and studied by economists for discrete-time optimal control problems (see [29; 71]) and is roughly explained by the following definition.

Definition 1. *An optimal control problem* (1) *is said to verify a turnpike property if, when the final time T is sufficiently large, an optimal solution remains most of the time close to an optimal stationary state which is in turn a solution of an associated stationary problem* (2).

The importance of long-term behavior is illustrated in Figure 1 where $\bar{x}$ represents a steady state solution of 2 and two trajectories are compared. This highlights the essential notion that if the initial and final states are not far apart, the optimal trajectory can ignore the turnpike (Figure 1(a)), but conversely, if they are far enough apart, it will always pay to stay close to the turnpike for a long time (Figure 1(b)). A brief analogy is possible by looking at the word *turnpike*, or in other words a highway, which describes very well this quasi-stationary state (one changes very little in direction and speed) on which one can stay for hours and hours to connect quite distant starting and ending points.

The study of the turnpike phenomenon may most of the time involve the first-order optimality conditions of (1), more standardly known as the *Pontryagin maximum principle* (see [89] for the finite dimension version) that introduce the adjoint variable p and the *Hamiltonian*

$$H(y,u,p,p^0) = \langle p, f(y,u)\rangle + p^0 f^0(y,u)$$

so that for any optimal solution (y_T,u_T), when controls are free of constraints, the corresponding extremal lift (y_T,p_T,p^0,u_T) should verify the following extremal equations

$$\begin{aligned}
\partial_t y_T(t) &= \frac{\partial H}{\partial p}(y_T(t),p_T(t),p^0,u_T(t)) \quad \forall\, t\in(0,T),\\
\partial_t p_T(t) &= -\frac{\partial H}{\partial y}(y_T(t),p_T(t),p^0,u_T(t)) \quad \forall\, t\in(0,T),\\
\frac{\partial H}{\partial u}(y_T(t),p_T(t),p^0,u_T(t)) &= 0 \quad \forall\, t\in(0,T).
\end{aligned}$$

We usually have additional terminal conditions on the adjoint vector, which are better known as transversality conditions. The adjoint vector (p_T, p^0) can be interpreted as the *Lagrange* multiplier associated with the dynamics constraint in the optimal control problem (1) and is defined to within one multiplicative scalar. Throughout this thesis, we assume that (1) will have a normal extremal lift, i.e., the real number p^0 is set to -1 so that the study will reduce to the optimal triples (y_T, p_T, u_T). Moreover, the first-order optimality conditions of the stationary problem (2) also introduce a *Lagrange* multiplier $\bar{p}$ so that an optimal stationary triplet $(\bar{y}, \bar{p}, \bar{u})$ satisfies the extremal stationary equations

$$\begin{aligned} f(\bar{y}, \bar{u}) &= 0 \\ -\frac{\partial f^0}{\partial y}(\bar{y}, \bar{u}) + \langle \bar{p}, \frac{\partial f}{\partial y}(\bar{y}, \bar{u}) \rangle &= 0 \\ -\frac{\partial f^0}{\partial u}(\bar{y}, \bar{u}) + \langle \bar{p}, \frac{\partial f}{\partial u}(\bar{y}, \bar{u}) \rangle &= 0. \end{aligned}$$

Insofar as $(\bar{y}, \bar{u})$ appears to be an equilibrium point of the dynamics involved in (1) (more broadly $(\bar{y}, \bar{p}, \bar{u})$ is an equilibrium point of the extremal equations), we can expect to see the optimal triple (y_T, p_T, u_T) converge to the optimal stationary triple $(\bar{y}, \bar{p}, \bar{u})$ as the final time T tends to infinity. More precisely, we normally expect a three-piece trajectory (see Figure 1) :

- on the first instants, on a short interval $[0, \tau]$ close to 0, the optimal triple (y_T, p_T, u_T) starts from $(y_T(0), p_T(0), u_T(0))$ until it approaches the stationary state $(\bar{y}, \bar{p}, \bar{u})$;
- on the second interval, much larger, $[\tau, T-\tau]$, the optimal solution remains close to the optimal stationary state. This part of the trajectory has an essential consequence on the minimization of the cost function when T tends to infinity ;
- at the end of the trajectory, we expect to observe on $[T-\tau, T]$ a behavior similar to the one on $[0, \tau]$ so that the (y_T, p_T, u_T) leaves the stationary state $(\bar{y}, \bar{p}, \bar{u})$ quite quickly to reach the final state $(y_T(T), p_T(T), u_T(T))$.

The main researches consist on the one hand in finding on which variables the turnpike phenomenon is perceptible (state, adjoint, control, or three of them) and on the other hand in quantifying this proximity to the stationary state along the time interval. In finite dimension, any chosen norm is suitable, while in the context of infinite dimension and especially in the context of shape optimization, the choice will be important. We briefly present in the following section most of the recent results on turnpike properties in optimal control.

1.1 State of the art

Over the past few decades, turnpike questions have seen increasing interest, with several notable variations in multiple areas of optimal control, some of which are stronger than others. The main questions are : does the turnpike occur on the control, the state, the adjoint variables, or even on the three of them, and how close are the time evolving and the stationary optimal solutions ? For example, a general and strong result for a finite dimensional continuous time optimal control problem is stated in [93] where the authors manage to show an exponential turnpike property for both the state and the control as well as for the adjoint vector resulting from the application of Pontryagin's maximum principle. The main result states that under certain controllability and observability assumptions, one can find two positive constants C_1, C_2

independent of the final time and at least one optimal solution of (1) such that the optimal triple (y_T, p_T, u_T) is exponentially close to an optimal triple of (2), this by checking the following inequality

$$\|y_T(t) - \bar{y}\| + \|p_T(t) - \bar{p}\| + \|u_T(t) - \bar{u}\| \leqslant C_1(e^{-C_2 t} + e^{-C_2(T-t)}) \quad \forall t \in (0, T).$$

Such a property induces a precise knowledge of the optimal triplet in the middle of the trajectory, which can then be suitably used for the initialization of numerical methods. Other exponential turnpike properties for infinite dimensional optimal control problems have been established in [80; 81; 92] and have in common the study of extremal equations arising from the application of Pontryagin's maximum principle, which brings out some hyperbolicity properties of the Hamiltonian flow. We further refer to [53] for a turnpike analysis of general linear evolution equations and to [38] for an adjoint turnpike result for general nonlinear optimal control problems. Weaker notions of turnpike can be found in [41; 91], where the authors introduce the notions of *integral* and *measure* turnpike and highlight the intimate connection of the latter with the notion of strict dissipativity introduced in [99]. Roughly speaking, the measure-turnpike property means that any optimal solution remains most of the time close to an optimal solution of an associated static optimal control problem, except for a sub-interval of time having a bounded Lebesgue measure when the final time goes to infinity. The definitions of measure-turnpike depend on the optimal control problem considered and we refer to [19; 26; 41; 48; 49; 51; 91] for works referring to these notions of measure-turnpike and strict dissipativity.

More generally, the solutions of optimal control problems are not generally reduced to a single one and the turnpike phenomenon is therefore not necessarily limited to staying around a single stationary state. Competition between several turnpikes can be observed (see [84]) and, for some classes of optimal control problems for periodic systems, a turnpike can occur around a periodic trajectory, which is itself considered to be the optimal solution of a periodic optimal control problem (see [86; 92; 100–102]). In [36; 37], the authors introduce the concept of *velocity turnpikes* for the optimal control of mechanical systems such that a turnpike occurs around a partial steady state. Similarly, a linear turnpike theorem can be found in [90].

The search for turnpike properties is also fruitful in the case of model predictive control (MPC) problems (see e.g., [26; 33–35; 40; 41; 53; 54] and references therein) and for discrete-time problems (e.g., [26; 41; 42; 48; 50–52]). Recently, other turnpike results in the field of deep learning can be found in [30; 39] and in [77] the authors give promising results for observing the turnpike for stochastic optimal control problems.

This thesis is a continuation of the research on the turnpike phenomenon for infinite dimensional optimal control problems and is more precisely related to the field of shape optimization.

1.2 Contributions : shape turnpike

Shape optimization is a special case of optimal control problem where the way to act on a system is to modify its shape with a PDE or ODE constraint

$$\partial_t y = f\big(y(t), \omega(t)\big), \qquad h\big(y(0), y(T)\big) = 0. \tag{3}$$

and with various terminal and boundary conditions. Let Ω be a domain in $\mathbb{R}^d$ and a fixed final

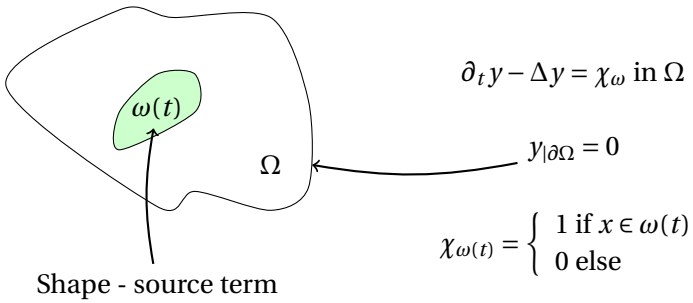


FIGURE 2 – Control of a heat equation by a shape acting as a source term

time $T > 0$, the time-varying shape $t \mapsto \omega(t)$ is for almost every t in $(0,T)$ a measurable subdomain of Ω (viewed as a control, as in [14]), and must be designed to minimize a given cost functional

$$J_T(\omega) = \frac{1}{T}\int_0^T f^0\big(y(t),\omega(t)\big)\,dt + g\big(y(T),\omega(T)\big), \tag{4}$$

where y is the solution of (3). We refer for instance to the example of the heat equation in Figure 2 where a time-dependent subdomain plays the role of a volume heat source.

Some of the most studied problems in optimal shape design are those of fluid mechanics and structures, which are mostly found in their stationary form (see for example [7] for the optimal shape of a cantilever and [73] for several examples of shape design in fluid dynamics). In addition, the industry has known for a few years the development of new very promising technologies, such as additive manufacturing (or 3D printing) as opposed to subtractive manufacturing for the design of parts and which allows more and more sophisticated structures, by allowing for example changes in topology, which brings a certain excitement. Besides, the development of numerical methods in shape optimization is already a rather difficult issue when dealing with static problems, which requires even more attention to the issue of adding a temporal dimension. For large time horizons, we could get around this difficulty by treating only the stationary case as far as a turnpike phenomenon occurs. From a numerical point of view, two main applications of the turnpike properties are of great interest to us.

First, for control problems considered in a long time horizon, the possibility that the optimal trajectories and controls converge to those of the corresponding steady-state model could justify the idea of treating numerically only the stationary problem. In practice, for a large final time, the actual computation of the control can be very expensive, especially in the nonlinear case, as it requires iterative methods to solve the coupled optimality system combining the forward controlled state equation and the backward adjoint equation. In contrast, the stationary problem does not encounter these difficulties.

Second, this asymptotic behavior can also allow for appropriate initializations for iterative optimization methods to solve the optimality system characterizing the optimal pairs, by choosing the steady-state optimal pairs, which are less costly to compute in a first attempt. For example, it has been shown in [93] that the turnpike property provides a way to successfully initialize direct or indirect (shooting) methods in numerical optimal control, by initializing them with the optimal triple of the associated static problem.

We thus associate to the dynamical problem (4,3) a *static* optimal shape design problem,

which does not depend on time,

$$\min_{\omega} f^0(y,\omega), \quad f(y,\omega)=0. \tag{5}$$

According to the turnpike phenomenon introduced in Definition 1, one expects that, for T large enough, an optimal time-varying shape of (4,3) remains most of the time "close" to an optimal stationary shape solution of the static problem (5).

As mentioned in [103], two main ingredients are essential to guarantee the turnpike results. First, the system considered in (3) must verify a controllability property. On the other hand, the cost functional (4) must be sufficiently coercive so that a partial information of the state included in this functional is sufficient to obtain a complete information, which refers to an observability property. Therefore, in accordance with previous turnpike results in [80; 81; 91] we first address the issue of the turnpike phenomenon in shape optimization by looking at a heat type partial differential equation as illustrated in Figure 2. Let Ω be a domain in $\mathbb{R}^d$ and for some target function $y_d \in L^2(\Omega)$, we introduce the time-varying shape $t \mapsto \chi_{\omega(t)}$ whose characteristic function $\chi_{\omega(\cdot)}$ acts as a distributive control on the heat equation

$$\partial_t y - \Delta y = \chi_\omega, \quad y_{|\partial\Omega}=0, \quad y(0)=y_0, \tag{6}$$

with the aim of minimizing a quadratic criterion

$$\frac{1}{2T}\int_0^T \int_\Omega (y(t,x)-y_d(x))^2\,dx\,dt + \int_\Omega (y(T,x)-y_d(x))^2\,dx. \tag{7}$$

The first observed results seemed very promising and therefore led us to theoretically address in the Chapter 1 the question of minimizing the same quadratic cost function (7) under constraint of a parabolic equation, seen as a generalization of the heat equation (6). For this purpose, we switch to a general second-order elliptic operator with a positive ellipticity constant so that the energy inequalities are verified with final time independent constants. The time-varying admissible shape $t \mapsto \omega(t)$ is, along time, a measurable subset of Ω with maximal measure constraint and verifies :

$$\text{for a.e. } t\in(0,T), \quad \omega(t)\in \mathscr{U}_L = \{\omega\subset\Omega \text{ measurable }, \quad |\omega|\leqslant L|\Omega|\}.$$

To show existence of solutions, we first perform a relaxation method by switching the characteristic functions χ_ω to a function a with values into $(0,1)$ so that admissible controls are searched into the larger space

$$\text{for a.e. } t\in(0,T), \quad a(t)\in \overline{\mathscr{U}}_L = \left\{a\in L^\infty(\Omega;[0,1]), \int_\Omega a(x)\,dx \leqslant M\right\}.$$

We then prove the existence of solutions (y_T, a_T) of the resulting relaxed optimal control problems and write optimality conditions using Pontryagin's maximum principle to qualify the optimal control at each time as level-sets of the adjoint variable

$$\text{for a.e.} t\in(0,T), \exists (s,c)\in\mathbb{R}\times(0,1), \quad a_T(t)=\chi_{\{p_T(t)>s\}} + c\chi_{\{p_T(t)=s\}}.$$

Thus, we exhibit some assumptions on the data of the problem such that we notice that the optimal solutions of the relaxed problem are solutions of the starting shape optimization problem insofar as the set $\{p_T(t)=s\}$ has zero Lebesgue measure.

Regarding the turnpike properties, we conduct the study on the relaxed problem knowing that, under the assumptions made beforehand, the optimal control is the characteristic function of some upper level-set of the adjoint variable. We finally divide our work into two parts by treating the integral and final costs separately.

In the Lagrange case, we introduce the notion of *strict dissipativity* adapted to our problem and prove that the optimal solution (state and control) satisfies both the measure and integral turnpike properties for the $L^2(\Omega)$-norm, i.e., both states and adjoints (y_T, p_T) and $(\bar{y}, \bar{p})$ of the respective optimal triples of $(\boldsymbol{DSD})_{\mathbf{T}}$ and $\boldsymbol{SSD}$ verify the following inequality for a positive constant M independent of the final time T :

$$\int_0^T \|y_T(t) - \bar{y}\|_{L^2(\Omega)} + \|p_T(t) - \bar{p}\|_{L^2(\Omega)}\, dt \leqslant M \quad \forall\, T > 0.$$

Thus, the larger T is, the more the quantity $\|y_T(t) - \bar{y}\|_{L^2(\Omega)} + \|p_T(t) - \bar{p}\|_{L^2(\Omega)}$ is close to 0 along the time interval. The main arguments of the proof are the use of optimality conditions from Pontryagin's maximum principle for time and stationary problems and energy inequalities with constants that do not depend on the final time.

In the Mayer case, we manage to prove that the Hausdorff distance between the optimal time-varying shape and some stationary shape returned as the upper level-set of the first non-zero mode of the adjoint p_T verifies the following exponential turnpike

$$d_{\mathscr{H}}\big(\omega_T(t), \bar{\omega}\big) \leqslant M e^{-\mu(T-t)} \quad \forall\, t \in (0, T).$$

In short words, the proof relies on a spectral decomposition of the adjoint variable from Pontryagin's maximum principle so that p_T is mostly exponentially close to a stationary function. Since along the time frame, the optimal control $a_T(t)$ is an upper level-set of the adjoint, we manage to quantify the proximity of the time-varying shape to a stationary one using a lemma inspired by [25].

We successively illustrate these results by numerical simulations on the heat equation. Numerical methods in shape optimization are generally applied to stationary problems and the context of time-varying shape optimization problems has raised the need for efficient numerical methods. For the problem we consider, we search for the optimal shape by treating the relaxed problem and observe well the solutions being shapes. Our problem is more broadly part of the field of PDE constrained optimization whose general numerical solving is a difficult issue. We therefore illustrate on several examples in the Chapter 2 how we can perform efficient optimization algorithms in order to propose them as further models.

2 Numerical PDE constrained optimization

2.1 Context and objectives

More generally, engineering problems or more theoretical research problems often lead to optimization problems governed by partial differential equations. Advances in computational capabilities, PDE solvers and optimization algorithms have provided accurate and efficient methods for solving difficult PDE-constrained optimization problems.

When approaching an optimal control problem governed by partial differential equations, as we will show in the illustrative examples provided in this thesis, it is particularly important to

first establish a rigorous mathematical framework in which the problem is well-posed, before deriving and designing appropriate numerical methods to solve it efficiently.

Throughout the Chapter 2, we refer to [60; 69; 89; 94] regarding classical issues on PDE control theory and general optimization. Ways of solving PDEs are numerous, finite differences, finite volumes, spectral, and general Galerkin methods. Here we will specifically focus on PDEs resolutions thanks to the finite element method and we refer to [85] for variational formulations of PDE problems. This chapter can serve as an introduction to numerical PDE optimization provided some basic knowledge is required in numerical computations as well as some basics in the `C++` language.

We look at *Optimal control problems* as optimization problems where the standard decision variables, previously called *controls*, are acting on the *state* variables through an ordinary differential equation (ODE) or a partial differential equation (PDE). Within this viewpoint, the control is the input and the resulting state is the output. The optimization problem consists of determining what is the best input, over a class of possible inputs, so that the output satisfies some constraints and minimizes a given criterion.

Most often, the objective function depends on both state and control variables and thus requires, at least from the numerical point of view, to make explicit or to compute the state's dependence on control (i.e., the input-output mapping of the system). State y and control u are respectively assumed to belong to real Banach spaces Y and U. General optimal control problems under consideration in Chapter 2 are written in the abstract form

$$\min_{(y,u)\in Y\times U} J(y,u), \qquad e(y,u)=0, \qquad c(y,u)\in K \tag{8}$$

where $J : Y \times U \mapsto \mathbb{R}$ is the objective function, $e : Y \times U \mapsto Z$ usually stands for some PDE and $c : Y \times U \mapsto K$ defines some additional constraints. Here, Z is a real a Banach space and K is a closed convex set. Well-posedness is assumed, which means that, for every $u \in U$, the equation $e(y,u) = 0$ has a unique solution $y = y(u) \in Y$ so that the reduced cost function

$$\widehat{J}(u) = J(y(u),u) \tag{9}$$

can be introduced.

The existence of solutions for somewhat general optimization problems can be a very difficult question; it often relies on functional analysis and compactness arguments. On the other hand, the uniqueness of the solution is often a consequence of the strict convexity of the problem. In the Chapter 2, we are not devoted to report existence or uniqueness problems but to show how to compute numerically in an efficient way a solution whose existence has already been proved or at least assumed. Given an optimal control problem, we will emphasize with several examples the importance of having a rigorous mathematical framework in which the problem is solved, as a guide to finding appropriate ways to discretize the problem and ensure the convergence of the resulting numerical method. Throughout the book, we focus on finite element discretization methods, particularly in 1, 2 or 3 dimensions. We present the software `FreeFEM` developed in our laboratory, a PDE solver using the finite element method and based on variational formulations with user-friendly and powerful features. We will provide some examples and details on its use and we will also refer to the documentation [56] (also available online) where most of the features are described. As far as optimization strategies are concerned, we use here differentiable methods which require the computation of derivatives of

functions. FreeFEM is flexible enough to allow users to write their own optimization algorithm (e.g. gradient, BFGS, Newton methods), or to plug it into some existing optimization routines. On our side, we focus particularly on the IpOpt routine (see [97]), which is well adapted to large-scale nonlinear optimization problems, and we show how to combine it with FreeFEM. In particular, we will show how IpOpt can be called directly from FreeFEM. Since solving PDEs requires a number of variables that increases with the size of the mesh, we have to deal with a large number of state and control variables (more than a million in the usual problems) as well as many constraints. The options available in IpOpt guarantee a good adaptability and efficiency for general convex and non-convex optimization of large size.

2.2 Contributions

Through several examples, we aim to propose a general methodology to look at an optimal control problem in a discretized version to solve it numerically.

We first focus on very standard examples like the minimization of a quadratic functional subject to a linear elliptic equation

$$\begin{aligned} &\min J(y,u) = \frac{1}{2}\int_\Omega \big(y(x)-y_d(x)\big)^2\,dx + \frac{\alpha}{2}\int_\Omega u(x)^2\,dx \\ &\text{subject to } -\nabla\cdot(a\nabla y) = u \text{ in } \Omega, \quad y = 0 \text{ in } \partial\Omega \end{aligned}$$

and separate the solving according to *direct* or *indirect* methods. The first one is based on a complete discretization of the basic optimal control problem (8) and leads to a classical finite dimensional optimization problem whose derivatives of the involved functions are then computed. In this case, we choose whether we consider the discretized PDE as a constraint or as an explicit step in the computation of the cost function and compare the two alternatives. Conversely, following the Pontryagin maximum principle, we write in a second option the optimality conditions in order to express the derivative of the reduced cost function (9) by introducing the adjoint variable. We highlight the advantage of this method over a sensitivity analysis that would require the computation of directional derivatives in each direction of a finite element basis, which is prohibitive. A third method is to bypass the computation of derivatives by means of automatic differentiation via the optimization software AMPL, which is designed to solve general finite-dimensional optimization problems by means of several available solvers (see [44]). It is not usual to observe that direct and indirect methods are equivalent, but this can happen in very particular examples (see [87]). In this perspective, we highlight on a simple example how automatic differentiation can hide a discretized version of the adjoint equation resulting from the Pontryagin maximum principle. Returning to the LQ case, we then compare the convergence curves of the different optimization methods and highlight the scalability of FreeFEM to handle time-dependent problems

$$\begin{aligned} &\min J(y,u) = \frac{1}{2}\int_0^T\int_\Omega \big(y(x,t)-y_d(x)\big)^2\,dx\,dt + \frac{\alpha}{2}\int_0^T\int_\Omega u(x,t)^2\,dx\,dt \\ &\text{subject to } \partial_t y - \nabla\cdot(a\nabla y) = u \text{ in } (0,T)\times\Omega, \quad y = 0 \text{ in } (0,T)\times\partial\Omega, \quad y(0) = y^0 \end{aligned}$$

with several time discretization choices. We finally expand the range of applications with an example on a semilinear PDE and perform iterative methods for the solving of the state equation (Newton-Raphson and fixed-point algorithm) so that a future user has the main tools available for the numerical solving of a general optimal control problem with FreeFEM.

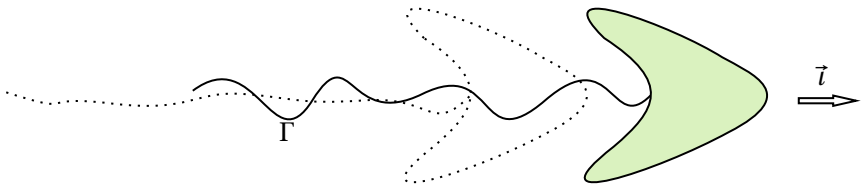


FIGURE 3 – Sperm's motion through its oscillating flagellum

We continue with a less classical example and underline how `FreeFEM` is really well suited for shape optimization problems by mesh deformations. We address the question of the optimal shape of a micro-swimmer evolving in a fluid governed by a Stokes PDE in order to maximize its speed in a given direction. The main applications are for example the design of nano-robots for medical purposes inspired by the study of chemotaxis (see for example Figure 3). Here are some works on this topic : in [11], the authors addressed the question of swimming strategies of a three-sphere body with low Reynolds number and, writing the problem in an optimal control framework, proved a global controllability result, i.e. the body can swim in any given direction. More broadly, in [70], the authors focus on an ordinary body immersed in a fluid and look for controllability properties with respect to either small deformations or displacements in a finite number of elementary deformations. For our part, we consider the Γ flagellum as an infinite oscillating curve for which we seek the shape maximizing the fluid velocity. To this end, we give a theoretical framework and a numerical methodology, which can be adapted to other similar problems, by searching for the shape of a periodic moving boundary Γ of a fluid domain Ω so that it maximizes the velocity u of the fluid set in motion at the free surface Σ_3 :

$$\min -\int_{\Sigma_3} u\cdot\vec{\imath}\,ds$$
$$\text{subject to } \begin{cases} -\mu\nabla\cdot\varepsilon(u)+\nabla p=0 & \text{in } \Omega \\ \operatorname{div}(u)=0 & \text{in } \Omega \\ \sigma(u,p)\vec{n}=0 & \text{in } \Sigma_3 \\ u=v_D\begin{pmatrix}0\\ f'\end{pmatrix} & \text{in } \Gamma. \end{cases}$$

We assimilate the shape of the boundary to the graph of a function f so that deformations of the boundary shape lead to an inhomogeneous Dirichlet boundary condition, which can be physically interpreted as friction forces. The study is therefore done in the frame of reference of the moving boundary with some specific assumptions on the acceptable boundary shapes in order to write a well-posed optimization problem insofar as the theoretical framework will have a significant influence in choosing a discretization strategy. A sensitivity analysis, which is standard in shape optimization, involves the emergence of the adjoint for the derivative calculation and we finally detail the discretization strategy.

2.3 Applications to shape turnpike

We conclude with Chapter 3 which is devoted to the description of the numerical part of results stated in Chapter 1. We present how we perform time-varying shape optimization and numerically illustrate the turnpike phenomenon. We first focus on the one-dimensional case $\Omega = [0, L]$ with L a positive constant :

$$\min\ J_T(\omega(\cdot)) = \frac{1}{2T}\int_0^T \int_0^L \big(y(t,x) - y_d(x)\big)^2\, dx dt$$
$$\text{subject to :} \begin{cases} \partial_t y - \partial_{xx} y = \chi_{\omega(\cdot)} & \forall (t,x) \in (0,T)\times(0,L) \\ y(t,0) = y(t,L) = 0, & \forall t \in (0,T) \\ y(0,x) = y_0, & \forall x \in (0,L). \end{cases}$$

We use the software `AMPL` with the combination of an implicit Euler scheme in time and finite differences centered in space. The special 1D case allows us to use the solver `CPLEX` (see [23], `CPLEX` is free to use, within memory allocation, on the NEOS website) which is normally indicated for solving linear programming problems with integer variables and can be extended to mixed integer quadratic programming. In a higher dimension, we take Ω as a subset of $\mathbb{R}^d$, and with the intention of finding an efficient method to solve the time-dependent problem, we first focus on the stationary case

$$\min_{\omega\in\mathcal{U}_L} \frac{1}{2}\|y - y_d\|^2_{L^2(\Omega)}, \qquad -\Delta y = \chi_\omega, \qquad y_{|\partial\Omega} = 0,$$

and search for a fast iterative algorithm. We first perform a *level-set* method, i.e., we look for the shape ω to be the upper level-set of a given function ϕ

$$\begin{cases} \phi(x) < 0 & \Longleftrightarrow\ x \in \omega \\ \phi(x) = 0 & \Longleftrightarrow\ x \in \partial\omega \cap \Omega \\ \phi(x) > 0 & \Longleftrightarrow\ x \in \Omega\backslash\bar{\omega}, \end{cases}$$

and employ a gradient-based algorithm inspired of [43]. We then adapt the framework introduced in Chapter 2 for the linear-quadratic case to a *convexified* version of the problem (6,7) by replacing the characteristic function χ_ω with a function $a \in \overline{\mathcal{U}}_L$. We finally compare the two methods on several design problems of shapes governed by linear elliptic equations. The convexification method seems to us to be more efficient and more easily adaptable to the time dependent problem. We therefore extend this method to the time-dependent problem so that the resulting numerical simulations highlight the phenomenon of *shape turnpike*, for several examples involving linear parabolic equations. To conclude, we finally bring some numerical results for semi-linear PDEs such as

$$\min_{\omega\in\mathcal{U}_L} \frac{1}{2}\|y - y_d\|^2_{L^2(\Omega)}, \qquad -\Delta y + f(y) = \chi_\omega, \qquad y_{|\partial\Omega} = 0,$$

as an opening for further investigations.

Chapter 1

Shape turnpike for linear parabolic PDEs

Table of contents

1.1 Shape turnpike for linear parabolic equation . . . 22
1.1.1 Setting . . . 23
1.1.2 Preliminaries . . . 23
1.1.3 Main results . . . 25
1.2 Proofs . . . 27
1.2.1 Proof of Theorem 1.1 . . . 27
1.2.2 Proof of Theorem 1.4 . . . 30
1.2.3 Proof of Theorem 1.7 . . . 31
1.2.4 Proof of Theorem 1.8 . . . 31
1.3 Numerical simulations : optimal shape design for the 2D heat equation . . . 34
1.4 Further comments . . . 40
Appendices . . . 40
1.A Energy inequalities . . . 40
1.B Bathtub principle . . . 43

Abstract We introduce and study the turnpike property for time-varying shapes from the perspective of optimal control. We focus here on linear parabolic equations of the second order where the shape acts as a source term and we look for the optimal time-varying shape that minimizes a quadratic criterion. We first establish the existence of optimal solutions under certain appropriate sufficient conditions. We then provide the necessary conditions for optimality in terms of adjoint equations and, using the concept of strict dissipativity, we prove that the state and adjoint satisfy the measure-turnpike property, which means that the extremal time-varying solution remains essentially close to the optimal solution of an associated static problem. We show that the optimal shape enjoys the exponential turnpike property in terms of Hausdorff distance for a quadratic Mayer cost. We illustrate the turnpike phenomenon in optimal design by several numerical simulations.

1.1 Shape turnpike for linear parabolic equation

Throughout the chapter, we denote by :

- $Q \subset \mathbb{R}^d$, $d \geqslant 1$ and $|Q|$ its Lebesgue measure if Q measurable subset ;
- (p,q) the scalar product in $L^2(\Omega)$ of p,q in $L^2(\Omega)$;
- $\|y\|$ the L^2-norm of $y \in L^2(\Omega)$;
- χ_ω the indicator (or characteristic) function of $\omega \subset \mathbb{R}^d$;
- d_ω the distance function to the set $\omega \subset \mathbb{R}^d$.

Let $\Omega \subset \mathbb{R}^d$ ($d \geqslant 1$) be an open bounded Lipschitz domain. We consider the uniformly elliptic second-order differential operator

$$Ay = -\sum_{i,j=1}^{d} \partial_{x_j}\big(a_{ij}(x)\partial_{x_i} y\big) + \sum_{i=1}^{d} b_i(x)\partial_{x_i} y + c(x)y$$

with $a_{ij}, b_i \in C^1(\Omega)$, $c \in L^\infty(\Omega)$ with $c \geqslant 0$. We consider the operator $(A, D(A))$ defined on the domain $D(A)$ encoding Dirichlet conditions $y_{|\partial\Omega} = 0$; when Ω is C^2 or a convex polytop in $\mathbb{R}^2$, we have $D(A) = H_0^1(\Omega) \cap H^2(\Omega)$. The adjoint operator A^* of A, also defined on $D(A)$ with homogeneous Dirichlet conditions, is given by

$$A^* v = -\sum_{i,j=1}^{d} \partial_{x_i}\left(a_{ij}(x)\partial_{x_j} v\right) - \sum_{i=1}^{d} b_i(x)\partial_{x_i} v + \left(c - \sum_{i=1}^{d} \partial_{x_i} b_i\right) v$$

and is also uniformly elliptic, see [31, Definition Chapter 6]. The operators A and A^* do not depend on t and have a constant of ellipticity $\theta > 0$ (for A written in *non-divergence form*), i.e. :

$$\sum_{i,j=1}^{d} a_{ij}(x)\xi_i\xi_j \geqslant \theta|\xi|^2 \qquad \forall x \in \Omega. \tag{1.1}$$

Moreover, we assume that

$$\theta > \theta_1 \tag{1.2}$$

where θ_1 is the largest root of the polynomial $P(X) = \frac{X^2}{4C_p} - \|c\|_{L^\infty(\Omega)} X - \frac{\left(\sum_{i=1}^d \|b_i\|_{L^\infty(\Omega)}\right)^2}{2}$ with C_p the Poincaré constant on Ω. This assumption is used to ensure that an energy inequality is satisfied with constants not depending on the final time T. We refer to appendix 1.4 for details.

We assume throughout that A satisfies the classical maximum principle (see [31, Section 6.4]) and that $c^* = c - \sum_{i=1}^d \partial_{x_i} b_i \in C^2(\Omega)$.

Let $(\lambda_j, \phi_j)_{j\in\mathbf{N}^*}$ be the eigenelements of A with $(\phi_j)_{j\in\mathbf{N}^*}$ an orthonormal eigenbasis of $L^2(\Omega)$:

- $\forall\, j \in \mathbf{N}^*, \qquad A\phi_j = \lambda_j\phi_j, \qquad \phi_{j_{|\partial\Omega}} = 0$
- $\forall\, j \in \mathbf{N}^*,\ j > 1, \qquad \lambda_1 < \lambda_j \leqslant \lambda_{j+1}, \qquad \lambda_j \to +\infty.$

A typical example satisfying all assumptions above is the Dirichlet Laplacian, which we will consider in our numerical simulations.

We recall that the Hausdorff distance between two compact subsets K_1, K_2 of $\mathbb{R}^d$ is defined by

$$d_{\mathscr{H}}(K_1,K_2) = \sup\Big(\sup_{x\in K_2} d_{K_1}(x), \sup_{x\in K_1} d_{K_2}(x)\Big).$$

1.1.1 Setting

Hereafter, we identify any measurable subset ω of Ω with its characteristic function χ_ω. Let $L \in (0,1)$. We define the set of admissible shapes

$$\mathscr{U}_L = \{\omega \subset \Omega \text{ measurable } \mid |\omega| \leqslant L|\Omega|\}.$$

Dynamical optimal shape design problem $(\boldsymbol{DSD})_{\mathbf{T}}$ Let $y_0 \in L^2(\Omega)$ and let $\gamma_1 \geqslant 0, \gamma_2 \geqslant 0$ be arbitrary. We consider the parabolic equation controlled by a (measurable) time-varying map $t \mapsto \omega(t)$ of subdomains

$$\partial_t y + Ay = \chi_{\omega(\cdot)}, \qquad y_{|\partial\Omega} = 0, \qquad y(0) = y_0. \tag{1.3}$$

Given $T > 0$ and $y_d \in L^2(\Omega)$, we consider the dynamical optimal shape design problem $(\boldsymbol{DSD})_{\mathbf{T}}$ of determining a measurable path of shapes $t \mapsto \omega(t) \in \mathscr{U}_L$ that minimizes the cost functional

$$J_T(\omega(\cdot)) = \frac{\gamma_1}{2T}\int_0^T \|y(t) - y_d\|^2\, dt + \frac{\gamma_2}{2}\|y(T) - y_d\|^2$$

where $y = y(t,x)$ is the solution of (1.3) corresponding to $\omega(\cdot)$.

Static optimal shape design problem Besides, for the same target function $y_d \in L^2(\Omega)$, we consider the following associated static shape design problem :

$$\min_{\omega\in\mathscr{U}_L} \frac{\gamma_1}{2}\|y - y_d\|^2, \quad Ay = \chi_\omega, \quad y_{|\partial\Omega} = 0. \tag{$\boldsymbol{SSD}$}$$

We are going to compare the solutions of $(\boldsymbol{DSD})_{\mathbf{T}}$ and of $(\boldsymbol{SSD})$ when T is large.

1.1.2 Preliminaries

Convexification Given any measurable subset $\omega \subset \Omega$, we identify ω with its characteristic function $\chi_\omega \in L^\infty(\Omega;\{0,1\})$ and we identify $\mathscr{U}_L$ with a subset of $L^\infty(\Omega)$ (as in [10; 82; 83]). Then, the convex closure of $\mathscr{U}_L$ in L^∞ weak star topology is

$$\overline{\mathscr{U}}_L = \left\{a \in L^\infty\big(\Omega;[0,1]\big) \mid \int_\Omega a(x)\,dx \leqslant L|\Omega|\right\}$$

which is also weak star compact. We define the *convexified* (or *relaxed*) optimal control problem $(\boldsymbol{OCP})_{\mathbf{T}}$ of determining a control $t \mapsto a(t) \in \overline{\mathscr{U}}_L$ minimizing the cost

$$J_T(a) = \frac{\gamma_1}{2T}\int_0^T \|y(t) - y_d\|^2\, dt + \frac{\gamma_2}{2}\|y(T) - y_d\|^2$$

under the constraints

$$\partial_t y + Ay = a, \qquad y_{|\partial\Omega} = 0, \qquad y(0) = y_0. \tag{1.4}$$

The corresponding convexified static optimization problem is

$$\min_{a\in\overline{\mathscr{U}}_L} \frac{\gamma_1}{2}\|y - y_d\|^2, \qquad Ay = a, \qquad y_{|\partial\Omega} = 0. \tag{$\boldsymbol{SOP}$}$$

Note that the control a does not appear in the cost functionals of the above convexified control problems. Therefore the resulting optimal control problems are affine with respect to a. Once we have proved that an optimal solution $a \in \overline{\mathscr{U}}_L$ exists, we expect that any such minimizer will be an element of the set of extremal points of the compact convex set $\overline{\mathscr{U}}_L$, which is exactly the set $\mathscr{U}_L$ (since ω is identified with its characteristic function χ_ω). If this is true, then actually $a = \chi_\omega$ with $\omega \in \mathscr{U}_L$. Here, as it is usual in shape optimization, the interest of passing by the convexified problem is to allow us to derive optimality conditions, and thus to characterize the optimal solution. It is anyway not always the case that the minimizer a of the convexified problem is an extremal point of $\overline{\mathscr{U}}_L$ (i.e., a characteristic function) : in this case, we speak of a *relaxation phenomenon*. Our analysis hereafter follows these guidelines.

Taking a minimizing sequence and by classical arguments of functional analysis (see e.g., [69]), it is straightforward to prove existence of solutions a_T and $\bar{a}$ respectively of $(\boldsymbol{OCP})_{\mathbf{T}}$ and of $(\boldsymbol{SOP})$ (see details in Section 1.2.1).

We underline the following fact : **if** $\bar{a}$ and $a_T(t)$, for *a.e.* $t \in [0,T]$, are characteristic functions of some subsets (meaning that $\bar{a} = \chi_{\bar{\omega}}$ with $\bar{\omega} \in \mathscr{U}_L$ and for almost every t in $(0,T)$ $a_T(t) = \chi_{\omega_T(t)}$ with $\omega_T(t) \in \mathscr{U}_L$), **then** actually $t \mapsto \omega_T(t)$ and $\bar{\omega}$ are optimal shapes, solutions respectively of $(\boldsymbol{DSD})_{\mathbf{T}}$ and of $(\boldsymbol{SSD})$.

Our next task is to apply necessary optimality conditions to optimal solutions of the convexified problems stated in [69, Chapters 2 and 3] or [67, Chapter 4] and infer from these necessary conditions that, under appropriate assumptions, the optimal controls are indeed characteristic functions.

Necessary optimality conditions for $(\boldsymbol{OCP})_{\mathbf{T}}$ According to the Pontryagin maximum principle (see [69, Chapter 3, Theorem 2.1], see also [67]), for any optimal solution (y_T, a_T) of $(\boldsymbol{OCP})_{\mathbf{T}}$ there exists an adjoint state $p_T \in L^2(0,T;L^2(\Omega))$ such that

$$\begin{aligned} \partial_t y_T + A y_T = a_T,\ y_{T|\partial\Omega} = 0,\ y_T(0) = y_0 \\ \partial_t p_T - A^* p_T = \gamma_1(y_T - y_d),\ p_{T|\partial\Omega} = 0,\ p_T(T) = \gamma_2\big(y_T(T) - y_d\big) \end{aligned} \tag{1.5}$$

$$\forall a \in \overline{\mathscr{U}}_L : \quad \big(p_T(t), a_T(t) - a\big) \geqslant 0 \quad \text{for a.e. } t \in [0,T]. \tag{1.6}$$

Necessary optimality conditions for $(\boldsymbol{SOP})$ Similarly, applying [69, Chapter 2, Theorem 1.4], for any optimal solution $(\bar{y}, \bar{a})$ of $(\boldsymbol{SOP})$ there exists an adjoint state $\bar{p} \in L^2(\Omega)$ such that

$$\begin{aligned} A\bar{y} = \bar{a}, \qquad \bar{y}_{|\partial\Omega} = 0 \\ -A^*\bar{p} = \gamma_1(\bar{y} - y_d), \qquad \bar{p}_{|\partial\Omega} = 0 \end{aligned} \tag{1.7}$$

$$\forall a \in \overline{\mathscr{U}}_L : \quad (\bar{p}, \bar{a} - a) \geqslant 0. \tag{1.8}$$

Using the bathtub principle (see e.g., [68, Theorem 1.14] and 1.18), (1.6) and (1.8) give

$$a_T(\cdot) = \chi_{\{p_T(\cdot) > s_T(\cdot)\}} + c_T(\cdot)\chi_{\{p_T(\cdot) = s_T(\cdot)\}} \tag{1.9}$$

$$\bar{a} = \chi_{\{\bar{p} > \bar{s}\}} + \bar{c}\chi_{\{\bar{p} = \bar{s}\}} \tag{1.10}$$

with, for a.e. $t \in [0,T]$,

$$c_T(t) \in L^\infty(\Omega;[0,1]) \text{ and } \bar{c} \in L^\infty(\Omega;[0,1]) \tag{1.11}$$

$$s_T(\cdot) = \inf\{\sigma \in \mathbb{R} \mid |\{p_T(\cdot) > \sigma\}| \leqslant L|\Omega|\} \tag{1.12}$$

$$\bar{s} = \inf\{\sigma \in \mathbb{R} \mid |\{\bar{p} > \sigma\}| \leqslant L|\Omega|\}. \tag{1.13}$$

1.1.3 Main results

Existence of optimal shapes Proving existence of optimal shapes, solutions of $(\boldsymbol{DSD})_{\mathbf{T}}$ and of $(\boldsymbol{SSD})$, is not an easy task. Indeed, relaxation phenomena may occur, i.e., classical designs in $\mathscr{U}_L$ may not exist but may develop homogeneization patterns (see [59, Sec. 4.2, Example 4.2.2]). Therefore, some assumptions are required on the target function y_d to establish existence of optimal shapes. We define :

- $y^{T,0}$ and $y^{T,1}$, the solutions of (1.4) corresponding respectively to $a = 0$ and $a = 1$;
- $y^{s,0}$ and $y^{s,1}$, the solutions of : $Ay = a, y_{|\partial\Omega} = 0$, corresponding respectively to $a = 0$ and $a = 1$;
- $y^0 = \min\left(y^{s,0}, \min\limits_{t\in(0,T)} y^{T,0}(t)\right)$ and $y^1 = \max\left(y^{s,1}, \max\limits_{t\in(0,T)} y^{T,1}\right)$.

We recall that A is said to be analytic-hypoelliptic in the open set Ω if any solution of $Au = v$ with v analytic in Ω is also analytic in Ω. Analytic-hypoellipticity is satisfied for the second-order elliptic operator A as soon as its coefficients are analytic in Ω (for instance it is the case for the Dirichlet Laplacian, without any further assumption, see [74]).

Theorem 1.1. *We distinguish between Lagrange and Mayer cases.*

1. $\gamma_1 = 0, \gamma_2 = 1$ *(Mayer case) : If A is analytic-hypoelliptic in Ω then there exists a unique optimal shape ω_T, solution of $(\boldsymbol{DSD})_{\mathbf{T}}$.*
2. $\gamma_1 = 1, \gamma_2 = 0$ *(Lagrange case) : Assuming that $y_0 \in D(A)$ and that $y_d \in H^2(\Omega)$:*
   (a) *If $y_d < y^0$ or $y_d > y^1$ then there exist unique optimal shapes $\bar{\omega}$ and ω_T, respectively, of $(\boldsymbol{SSD})$ and of $(\boldsymbol{DSD})_{\mathbf{T}}$.*
   (b) *There exists a function β such that if $Ay_d \leqslant \beta$, then there exists a unique optimal shape $\bar{\omega}$, solution of $(\boldsymbol{SSD})$.*

Proofs are given in Section 1.2. To prove existence of optimal shapes, we deal first with the convexified problems $(\boldsymbol{OCP})_{\mathbf{T}}$ and $(\boldsymbol{SOP})$ and show existence and uniqueness of solutions. Hereafter, using optimality conditions (1.5)-(1.7) and under the assumptions given in Theorem 1.1 we can write the optimal control as characteristic functions of upper level sets of the adjoint variable. In the static case, for example, one key observation is to note that, if $|\{\bar{p} = \bar{s}\}| = 0$, then it follows from (1.10) that the static optimal control $\bar{a}$ is actually the characteristic function of a shape $\bar{\omega} \in \mathscr{U}_L$. This proves the existence of an optimal shape.

Remark 1.2. *Note that in the Mayer case ($\gamma_1 = 0, \gamma_2 > 0$), $(\boldsymbol{SSD})$ is reduced to solve $Ay = \chi_\omega$, $y_{|\partial\Omega} = 0$. There is no criterion to minimize.*

Remark 1.3. *Theorem 1.1 guarantees the uniqueness of an optimal shape. We deduce from the inequality (1.38) in the appendix that we also have the uniqueness of the corresponding state and adjoint state. Thus we have uniqueness for both the dynamic and the static optimal triple.*

In what follows, we study the behavior of optimal solutions of $(\boldsymbol{DSD})_{\mathbf{T}}$ compared to those of $(\boldsymbol{SSD})$ and give some turnpike properties. In the Lagrange case, inspired by [80], [81] and [91], we first prove that state and adjoint satisfy integral and measure turnpike properties. In the Mayer case, we estimate the Hausdorff distance between dynamical and static optimal shapes and show an exponential turnpike property. We denote by :

- (y_T, p_T, ω_T) the optimal triple of $(\boldsymbol{DSD})_{\mathbf{T}}$ and
$$J_T = \frac{\gamma_1}{2T}\int_0^T \|y_T(t) - y_d\|^2\, dt + \frac{\gamma_2}{2}\|y_T(T) - y_d\|^2;$$
- $(\bar{y}, \bar{p}, \bar{\omega})$ the optimal triple of $(\boldsymbol{SSD})$ and $\bar{J} = \frac{\gamma_1}{2}\|\bar{y} - y_d\|^2$.

Integral turnpike in the Lagrange case

Theorem 1.4. *For $\gamma_1 = 1, \gamma_2 = 0$ (Lagrange case), there exists $M > 0$ (independent of the final time T) such that*
$$\int_0^T \left(\|y_T(t) - \bar{y}\|^2 + \|p_T(t) - \bar{p}\|^2\right) dt \leqslant M \qquad \forall\, T > 0.$$

Measure-turnpike in the Lagrange case

Definition 1.5. *We say that (y_T, p_T) satisfies the* state-adjoint measure-turnpike property *if for every $\varepsilon > 0$ there exists $\Lambda(\varepsilon) > 0$, independent of T, such that*
$$|P_{\varepsilon,T}| < \Lambda(\varepsilon) \qquad \forall\, T > 0$$
where $P_{\varepsilon,T} = \left\{t \in [0,T] \mid \|y_T(t) - \bar{y}\| + \|p_T(t) - \bar{p}\| > \varepsilon\right\}$.

We refer to [22; 41; 91] (and references therein) for similar definitions. Here, $P_{\varepsilon,T}$ is the set of times along which the time optimal state-adjoint pair $\left(y_T, p_T\right)$ remains outside of an ε-neighborhood of the static optimal state-adjoint pair $(\bar{y}, \bar{p})$ in L^2 topology.

Recall that a $\mathcal{K}$-class function is a continuous monotone increasing function $\alpha : [0; +\infty) \mapsto [0; +\infty)$ with $\alpha(0) = 0$. We now recall the notion of dissipativity (see [99]).

Definition 1.6. *We say that $(\boldsymbol{DSD})_{\mathbf{T}}$ is* strictly dissipative *at an optimal stationary point $(\bar{y}, \bar{\omega})$ of $(\boldsymbol{SSD})$ with respect to the* supply rate function
$$w(y, \omega) = \frac{1}{2}\left(\|y - y_d\|^2 - \|\bar{y} - y_d\|^2\right)$$
if there exists a storage function $S : E \to \mathbb{R}$ *locally bounded and bounded below and a $\mathcal{K}$*-class function $\alpha(\cdot)$ *such that, for any $T > 0$ and any $0 < \tau < T$, the strict dissipation inequality*
$$S(y(\tau)) + \int_0^\tau \alpha(\|y(t) - \bar{y}\|)\, dt \leqslant S(y(0)) + \int_0^\tau w\big(y(t), \omega(t)\big)\, dt \tag{1.14}$$
is satisfied for any pair $\left(y(\cdot), \omega(\cdot)\right)$ solution of (1.3).

Theorem 1.7. *For $\gamma_1 = 1, \gamma_2 = 0$ (Lagrange case) :*

(a) $(\boldsymbol{DSD})_{\mathbf{T}}$ is strictly dissipative in the sense of Definition 1.6.

(b) The state-adjoint pair (y_T, p_T) satisfies the measure-turnpike property.

Exponential turnpike in the Mayer case The exponential turnpike property is a stronger property and can be satisfied either by the state, by the adjoint or by the control or even by the three together.

Theorem 1.8. *For $\gamma_1 = 0, \gamma_2 = 1$ (Mayer case) : For Ω with C^2 boundary and $c = 0$ there exist $T_0 > 0$, $M > 0$, $\mu > 0$ and a shape $\bar{\omega} \in \mathscr{U}_L$ such that, for every $T \geqslant T_0$,*

$$d_{\mathscr{H}}\big(\omega_T(t), \bar{\omega}\big) \leqslant M e^{-\mu(T-t)} \qquad \forall\, t \in (0,T).$$

In the Lagrange case, based on the numerical simulations presented in Section 1.3 and Chapter 3 we conjecture the exponential turnpike property, i.e., given optimal triples (y_T, p_T, ω_T) and $(\bar{y}, \bar{p}, \bar{\omega})$, there exist $C_1 > 0$ and $C_2 > 0$ independent of T such that

$$\|y_T(t) - \bar{y}\| + \|p_T(t) - \bar{p}\| + \|\chi_{\omega_T(t)} - \chi_{\bar{\omega}}\| \leqslant C_1\Big(e^{-C_2 t} + e^{-C_2(T-t)}\Big)$$

for a.e. $t \in [0,T]$.

1.2 Proofs

1.2.1 Proof of Theorem 1.1

We first show existence of an optimal shape, solution for $(\boldsymbol{OCP})_{\mathbf{T}}$ and similarly for $(\boldsymbol{SOP})$. We first see that the infimum exists. We take a minimizing sequence $(y_n, a_n) \in L^2(0,T; H_0^1(\Omega)) \times L^\infty\big(0,T; L^2\big(\Omega, [0,1]\big)\big)$ such that, for $n \in \mathbf{N}$, for *a.e.* $t \in [0,T]$, $a_n(t) \in \overline{\mathscr{U}}_L$, the pair (y_n, a_n) satisfies (1.4) and $J_T(a_n) \to J_T$. The sequence (a_n) is bounded in $L^\infty\big(0,T; L^2\big(\Omega, [0,1]\big)\big)$, so using (1.38) and (1.39), the sequence (y_n) is bounded in $L^\infty(0,T; L^2(\Omega)) \cap L^2(0,T; H_0^1(\Omega))$. We show then, using (1.4), that the sequence $(\frac{\partial y_n}{\partial t})$ is bounded in $L^2(0,T; H^{-1}(\Omega))$. We subtract a sequence still denoted by (y_n, a_n) such that one can find a pair $(y,a) \in L^2(0,T; H_0^1(\Omega)) \times L^\infty\big(0,T; L^2\big(\Omega, [0,1]\big)\big)$ with

$$\begin{aligned} y_n &\rightharpoonup y \quad \text{weakly in } L^2(0,T; H_0^1(\Omega)) \\ \partial_t y_n &\rightharpoonup \partial_t y \quad \text{weakly in } L^2(0,T; H^{-1}(\Omega)) \\ a_n &\rightharpoonup a \quad \text{weakly * in } L^\infty\big(0,T; L^2\big(\Omega, [0,1]\big)\big). \end{aligned} \tag{1.15}$$

We deduce that

$$\begin{aligned} \partial_t y_n + A y_n - a_n &\to \partial_t y + Ay - a \quad \text{in } \mathscr{D}'\big((0,T)\times\Omega\big) \\ y_n(0) &\rightharpoonup y(0) \quad \text{weakly in } L^2(\Omega). \end{aligned} \tag{1.16}$$

We get using (1.16) that (y,a) is a weak solution of (1.4). Moreover, since $L^\infty\big(0,T; L^2\big(\Omega, [0,1]\big)\big) = \Big(L^1\big(0,T; L^2\big(\Omega, [0,1]\big)\big)\Big)'$ (see [62, Corollary 1.3.22]) the convergence (1.15) implies that for every $v \in L^1(0,T)$ satisfying $v \geqslant 0$ and $\|v\|_{L^1(0,T)} = 1$, we have $\int_0^T \Big(\int_\Omega a(t,x)\,dx\Big) v(t)\,dt \leqslant L|\Omega|$. Since the function f_a defined by $f_a(t) = \int_\Omega a(t,x)\,dx$ belongs to $L^\infty(0,T)$, the norm $\|f_a\|_{L^\infty(0,T)}$ is the supremum of $\int_0^T \Big(\int_\Omega a(t,x)\,dx\Big) v(t)\,dt$ over the set of all possible $v \in L^1(0,T)$ such that

$\|v\|_{L^1(0,T)} = 1$. Therefore $\|f_a\|_{L^\infty(0,T)} \leqslant L|\Omega|$ and $\int_\Omega a(t,x)\,dx \leqslant L|\Omega|$ for a.e. $t \in (0,T)$. This shows that the pair (y,a) is admissible. Since $H_0^1(\Omega)$ is compactly embedded in $L^2(\Omega)$ and by using the Aubin-Lions compactness Lemma (see [15]), we obtain

$$y_n \to y \quad \text{strongly in } L^2(0,T;L^2(\Omega)).$$

We get then by weak lower semi-continuity of J_T and by Fatou Lemma that

$$J_T(a) \leqslant \liminf J_T(a_n).$$

Hence a is an optimal control for $(\boldsymbol{OCP})_{\mathbf{T}}$, that we rather denote by a_T (and $\bar{a}$ for $(\boldsymbol{SOP})$). We next proceed by proving existence of optimal shape designs.

1. We take $\gamma_1 = 0, \gamma_2 = 1$ (Mayer case). We consider an optimal triple (y_T, p_T, a_T) of $(\boldsymbol{OCP})_{\mathbf{T}}$. Then it satisfies (1.5) and (1.9). It follows from the properties of the parabolic equation and from the assumption of analytic-hypoellipticity that p_T is analytic on $(0,T)\times\Omega$ and that all level sets $\{p_T(t) = \alpha\}$ have zero Lebesgue measure. We conclude that the optimal control a_T satisfying (1.5)-(1.9) is such that

$$\text{for a.e. } t \in [0,T] \quad \exists s(t) \in \mathbb{R}, \quad a_T(t,\cdot) = \chi_{\{p_T(t) > s(t)\}} \tag{1.17}$$

i.e., $a_T(t)$ is a characteristic function. Hence, for a Mayer problem $(\boldsymbol{DSD})_{\mathbf{T}}$, existence of an optimal shape is proved.

2-(a). In the case $\gamma_1 = 1, \gamma_2 = 0$ (Lagrange case), we give the proof for the static problem $(\boldsymbol{SSD})$. We suppose $y_d < y^0$ (we proceed similarly for $y_d > y^1$). Having in mind (1.7) and (1.10), we have $A\bar{y} = \bar{c}$ on $\{\bar{p} = \bar{s}\}$. By contradiction, if $\bar{c} \leqslant 1$ on $\{\bar{p} = \bar{s}\}$, let us consider the solution y^* of : $Ay^* = a^*, y^*_{|\partial\Omega} = 0$, with the control a^* which is the same as $\bar{a}$ verifying (1.10) except that $\bar{c} = 0$ ($\bar{c} = 1$ if $y_d > y^1$) on $\{\bar{p} = \bar{s}\}$. We have then $A(\bar{y} - y^*) \leqslant 0$ (or $A(\bar{y} - y^*) \geqslant 0$ if $y_d > y^1$). Then, by the maximum principle (see [31, Section 6.4]) and using the homogeneous Dirichlet condition, we get that the maximum (the minimum if $y_d > y^1$) of $\bar{y} - y^*$ is reached on the boundary and hence $y_d \geqslant y^* \geqslant \bar{y}$ (or $y_d \leqslant y^* \leqslant \bar{y}$ if $y_d > y^1$). We deduce $\|y^* - y_d\| \leqslant \|\bar{y} - y_d\|$. This means that a^* is an optimal control. We conclude by uniqueness.

We use a similar argument thanks to maximum principle for parabolic equations (see [31, Section 7.1.4]) for existence of an optimal shape solution of $(\boldsymbol{DSD})_{\mathbf{T}}$.

In view of proving the next part of the theorem, we first give a useful Lemma inspired by [66, Theorem 3.2] and from [32, Theorem 6.3].

Lemma 1.9. *Given any $p \in [1, +\infty)$ and any $u \in W^{1,p}(\Omega)$ such that $|\{u = 0\}| > 0$, we have $\nabla u = 0$ a.e. on $\{u = 0\}$.*

Proof of Lemma 1.9. A proof of a more general result can be found in [66, Theorem 3.2]. For completeness, we give here a short argument. Du denotes here the weak derivative of u. We need first to show that for $u \in W^{1,p}(\Omega)$ and for a function $S \in C^1(\mathbb{R})$ for which there exists $M > 0$ such that $\|S'\|_{L^\infty(\Omega)} < M$, we have $S(u) \in W^{1,p}(\Omega)$ and $DS(u) = S'(u)Du$. To do that, by the Meyers-Serrins theorem, we get a sequence $u_n \in C^\infty(\Omega) \cap W^{1,p}(\Omega)$ such that $u_n \to u$ in $W^{1,p}(\Omega)$ and $u_n \to u$ almost everywhere. We get by the chain rule $DS(u_n) = S'(u_n)Du_n$ and $\int_\Omega |DS(u_n)|^p\,dx \leqslant \|S'\|^p_{L^\infty(\Omega)} \|Du_n\|^p_{L^p(\Omega)}$ involving $S(u_n) \in W^{1,p}(\Omega)$. Since S is Lipschitz, we have

$\|S(u_n)-S(u)\|_{L^p(\Omega)} \leqslant \|u_n-u\|_{L^p(\Omega)} \to 0$ when $n \to 0$. We write then

$$\begin{aligned}\|DS(u_n)-S'(u)Du\|_{L^p(\Omega)} &= \|S'(u_n)Du_n - S'(u)Du\|_{L^p(\Omega)}\\ &\leqslant \|S'(u_n)(Du_n-Du)\|_{L^p(\Omega)} + \|(S'(u_n)-S'(u))Du\|_{L^p(\Omega)}\\ &\leqslant \|S'\|_{L^\infty(\Omega)}\|u_n-u\|_{W^{1,p}(\Omega)} + \|(S'(u_n)-S'(u))Du\|_{L^p(\Omega)}.\end{aligned}$$

The first term tends to 0 since $u_n \to u$ in $W^{1,p}(\Omega)$. For the second term, we use that $|S'(u_n)-S'(u)|^p|Du|^p \to 0$ a.e. and $|S'(u_n)-S'(u)|^p|Du|^p \leqslant 2\|S'\|^p_{L^\infty(\Omega)}|Du|^p \in L^1(\Omega)$. By the dominated convergence theorem, $\|(S'(u_n)-S'(u))Du\|_{L^p(\Omega)} \to 0$ which implies that $\|DS(u_n)-S'(u)Du\|_{L^p(\Omega)} \to 0$. Finally $S(u_n) \to S(u)$ in $W^{1,p}(\Omega)$ and $DS(u) = S'(u)Du$. Then, we consider $u^+ = \max(u,0)$ and $u^- = \min(u,0) = -\max(-u,0)$. We define

$$S_\varepsilon(s) = \begin{cases} (s^2+\varepsilon^2)^{\frac{1}{2}} - \varepsilon & \text{if } s \geqslant 0\\ 0 & \text{else.}\end{cases}$$

Note that $\|S_\varepsilon'\|_{L^\infty(\Omega)} < 1$. We deduce that $DS_\varepsilon(u) = S_\varepsilon'(u)Du$ for every $\varepsilon > 0$. For $\phi \in C_c^\infty(\Omega)$ we take the limit of $\int_\Omega S_\varepsilon(u)D\phi\,dx$ when $\varepsilon \to 0^+$ to get that

$$Du^+ = \begin{cases} Du & \text{on } \{u>0\}\\ 0 & \text{on } \{u\leqslant 0\}\end{cases} \quad \text{and } Du^- = \begin{cases} 0 & \text{on } \{u\geqslant 0\}\\ -Du & \text{on } \{u<0\}\end{cases}.$$

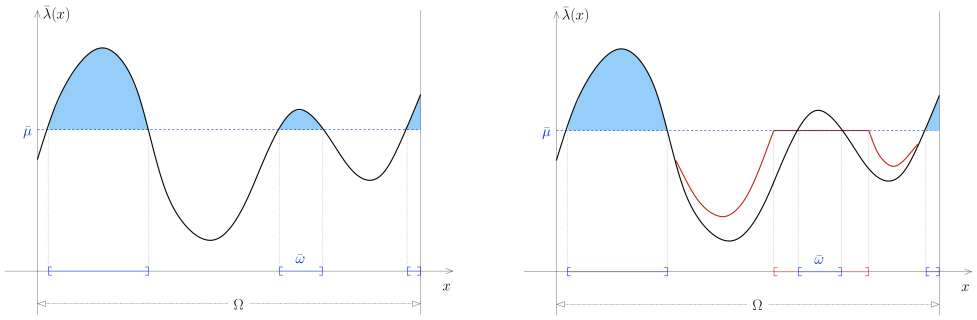
Since $u = u^+ - u^-$, we get $Du = 0$ on $\{u=0\}$. We can find this Lemma in a weaker form in [32, Theorem 6.3]. □

2-(b). We assume that $Ay_d \leqslant \beta$ in Ω with $\beta = \bar{s}Ac^*$. Having in mind (1.7) and (1.10), we assume by contradiction that $|\{\bar{p}=\bar{s}\}|>0$. Since A and A^* are differential operators, applying A^* to $\bar{p}$ on $\{\bar{p}=\bar{s}\}$, we obtain by Lemma 1.9 that $A^*\bar{p} = c^*\bar{s}$ on $\{\bar{p}=\bar{s}\}$. Since $(\bar{y},\bar{p})$ verifies (1.7) we get $y_d - \bar{y} = c^*\bar{s}$ on $\{\bar{p}=\bar{s}\}$. We apply then A to this equation to get that $Ay_d - \bar{s}Ac^* = A\bar{y} = \bar{a}$ on $\{\bar{p}=\bar{s}\}$. Therefore $Ay_d - \bar{s}Ac^* \in (0,1)$ on $\{\bar{p}=\bar{s}\}$ which contradicts $Ay_d \leqslant \beta$. Hence $|\{\bar{p}=\bar{s}\}| = 0$ and thus (1.10) implies $\bar{a} = \chi_{\bar{\omega}}$ for some $\bar{\omega} \in \mathscr{U}_L$. Existence of solution for (***SSD***) is proved.

Remark 1.10. *Condition in Theorem 1.1.2-(b) is a necessary condition. We can construct example, where $Ay_d \leqslant \beta$ is not satisfied and where we observe relaxation, which is closely related to the fact $|\{\bar{p}=\bar{s}\}|>0$.*

Indeed, we plot on Figure 1.1 the adjoint state $\bar{p}$ for the static problem in 1D. At the left-hand side, $\bar{p}$ is assumed to be analytic : in this case, all level sets of $\bar{p}$ have zero Lebesgue measure (there is no subset of positive measure on which $\bar{p}$ would remain constant). When $\bar{p}$ is not analytic and remains constant on a subset of positive measure (see Figure 1.1 in red), we do not have necessarily zero Lebesgue measure level sets, and on $\{\bar{p}=\bar{s}\}$, $\bar{a}$ can take values in $(0,1)$.

The uniqueness of the optimal controls follows from the strict convexity of the cost functions. Indeed, in the dynamic case, whatever $(\gamma_1,\gamma_2) \neq (0,0)$, J_T is strictly convex with respect to the variable y. The injectivity of the control-state mapping gives the strict convexity with respect to the variable a. Moreover, the uniqueness of $(\bar{y},\bar{p})$ emerges by application of Poincar'e's inequality and the uniqueness of (y_T,p_T) follows from Gronwall's inequality (1.39) in the Appendix.



(a) (b)

FIGURE 1.1 – Optimal shape design existence : (a) level-sets with zero Lebesgue measure, no relaxation : shape existence; (b) positive measure level-sets, relaxation

1.2.2 Proof of Theorem 1.4

For $\gamma_1 = 1, \gamma_2 = 0$ (Lagrange case), the cost is $J_T(\omega) = \frac{1}{2T}\int_0^T \|y(t) - y_d\|^2\, dt$. We consider the triples $(y_T, p_T, \chi_{\omega_T})$ and $(\bar y, \bar p, \chi_{\bar\omega})$ satisfying the optimality conditions (1.5) and (1.7). Since $\chi_{\omega_T(t)}$ is bounded at each time $t \in [0,T]$ and by application of Gronwall inequality (1.39) in the appendix to y_T and p_T we can find a constant $C > 0$ depending only on A, y_0, y_d, Ω, L such that

$$\forall T > 0 \quad \|y_T(T)\|^2 \leqslant C \quad \text{and} \quad \|p_T(0)\|^2 \leqslant C.$$

Setting $\tilde y = y_T - \bar y, \tilde p = p_T - \bar p, \tilde a = \chi_{\omega_T} - \chi_{\bar\omega}$, we have

$$\partial_t \tilde y + A\tilde y = \tilde a, \quad \tilde y_{|\partial\Omega} = 0, \quad \tilde y(0) = y_0 - \bar y \tag{1.18}$$

$$\partial_t \tilde p - A^*\tilde p = \tilde y, \quad \tilde p_{|\partial\Omega} = 0, \quad \tilde p(T) = -\bar p. \tag{1.19}$$

First, using (1.5) and (1.7) one has $\big(\tilde p(t), \tilde a(t)\big) \geqslant 0$ for almost every $t \in [0,T]$. Multiplying (1.18) by $\tilde p$, (1.19) by $\tilde y$ and then adding them, one can use the fact that

$$\big(\bar y - y_0, \tilde p(0)\big) - \big(\tilde y(T), \bar p\big) = \int_0^T \big(\tilde p(t), \tilde a(t)\big)\, dt + \int_0^T \|\tilde y(t)\|^2\, dt.$$

By the Cauchy-Schwarz inequality we get a new constant $C > 0$ such that

$$\frac{1}{T}\int_0^T \|\tilde y(t)\|^2\, dt + \frac{1}{T}\int_0^T \big(\tilde p(t), \tilde a(t)\big)\, dt \leqslant \frac{C}{T}.$$

The two terms at the left-hand side are positive and using the inequality (1.38) with $\zeta(t) = \tilde p(T - t)$, we finally obtain $M > 0$ independent of T such that

$$\frac{1}{T}\int_0^T \big(\|y_T(t) - \bar y\|^2 + \|p_T(t) - \bar p\|^2\big)\, dt \leqslant \frac{M}{T}.$$

1.2.3 Proof of Theorem 1.7

(a). Strict dissipativity is established thanks to the storage function $S(y) = \big(y, \bar{p}\big)$ where $\bar{p}$ is the optimal adjoint. Following the Gronwall inequality (1.39) in the appendix, since $\|y(t)\|^2 < M$ for every $t \in [0,T]$ with M independent of final time T, the storage function S is locally bounded and bounded below. We next consider an admissible pair $(y(\cdot), \chi_{\omega(\cdot)})$ of $(\boldsymbol{OCP})_{\mathbf{T}}$, we multiply (1.3) by $\bar{p}$ and or $\tau > 0$, we integrate over $(0,\tau)\times\Omega$ and use optimality conditions of static problem (1.7)-(1.8) combined with integration by parts to write

$$\int_0^\tau \big(y_t + Ay, \bar{p}\big)\,dt = \int_0^\tau \big(\chi_{\omega(t)}, \bar{p}\big)\,dt \leqslant \int_0^\tau \big(\chi_{\bar{\omega}}, \bar{p}\big)\,dt$$
$$\text{and so} \qquad \big(y(\tau), \bar{p}\big) - \int_0^\tau \big(y(t) - \bar{y}, \bar{y} - y_d\big)\,dt \leqslant \big(y(0), \bar{p}\big).$$

Noting that $\|y - \bar{y}\|^2 + 2\big(y - \bar{y}, \bar{y} - y_d\big) = \|y - y_d\|^2 - \|\bar{y} - y_d\|^2$ we make appear the quantity $\|y(t) - \bar{y}\|^2$ and finally get the strict dissipation inequality (1.14) with respect to the supply rate function $w(y,\omega) = \frac{1}{2}\Big(\|y - y_d\|^2 - \|\bar{y} - y_d\|^2\Big)$ and with $\alpha(s) = \frac{1}{2}s^2$:

$$(\bar{p}, y(\tau)) + \int_0^\tau \frac{1}{2}\|y(t) - \bar{y}\|^2\,dt \leqslant (\bar{p}, y(0)) + \int_0^\tau w\big(y(t), \omega(t)\big)\,dt. \tag{1.20}$$

(b). Now we prove that strict dissipativity implies measure-turnpike, by following an argument of [91]. Applying (1.20) to the optimal solution (y_T, ω_T) at $\tau = T$, we get

$$\frac{1}{T}\int_0^T \|y_T(t) - \bar{y}\|^2\,dt \leqslant J_T - \bar{J} + \frac{(y_T(0) - y_T(T), \bar{p})}{T}.$$

Considering then the solution y_s of (1.3) with $\omega(\cdot) = \bar{\omega}$ and $J_s = \frac{1}{T}\int_0^T \|y_s(t) - y_d\|^2$, we have $J_T - J_s < 0$ and we show that $J_s - \bar{J} \leqslant \frac{1 - e^{-CT}}{CT}$, then we find $M_1 > 0$ independent of T such that

$$\frac{1}{T}\int_0^T \|y_T(t) - \bar{y}\|^2\,dt \leqslant \frac{M_1}{T}. \tag{1.21}$$

Applying (1.38) to $\zeta(\cdot) = p_T(T - \cdot) - \bar{p}$, we get $M_2 > 0$ independent of T such that

$$\frac{1}{T}\int_0^T \|p_T(t) - \bar{p}\|^2\,dt \leqslant \frac{M_2}{T}\int_0^T \|y_T(t) - \bar{y}\|^2\,dt. \tag{1.22}$$

We combine (1.21) and (1.22) to finally get a constant $M > 0$ which does not depend on T such that $\forall \varepsilon > 0$, $|P_{\varepsilon,T}| \leqslant \dfrac{M}{\varepsilon^2}$.

1.2.4 Proof of Theorem 1.8

We take $\gamma_1 = 0, \gamma_2 = 1$ (Mayer case). We want to characterize optimal shapes as being the level set of some functions as in [25]. Let $(y_T, p_T, \chi_{\omega_T})$ be an optimal triple, coming from Theorem 1.1.1. Then $\zeta(t,x) = p_T(T - t, x)$ satisfies

$$\partial_t \zeta + A^*\zeta = 0, \quad \zeta_{|\partial\Omega} = 0, \quad \zeta(0) = y_d - y_T(T). \tag{1.23}$$

We write $y_d - y_T(T)$ in the basis $(\phi_j)_{j\in\mathbf{N}^*}$. There exists $(\zeta_j) \in \mathbb{R}^{\mathbf{N}^*}$ such that $y_d - y_T(T) = \sum_{j\geqslant 1}\zeta_j\phi_j$. We can solve (1.23) and get $p_T(t,x) = \sum_{j\geqslant 1}\zeta_j\phi_j(x)e^{-\lambda_j(T-t)}$. Using the Gronwall inequality (1.39) in the appendix, there exists $C_1 > 0$ independent of T such that the solution of (1.3) satisfies $\|y_T(t)\|^2 \leqslant C_1$ for every $t \in (0,T)$. Hence $|\zeta_j|^2 \leqslant C_1$ for every $j \in \mathbf{N}^*$. Let us consider the index $j_0 = \inf\{j \in \mathbf{N}, \zeta_j \neq 0\}$. Take $\lambda = \lambda_{j_0}$ and $\mu = \lambda_k$ where k is the first index for which $\lambda_k > \lambda$. We define $\Phi_0 = \sum_{\lambda_j=\lambda_{j_0}} \zeta_j\phi_j$ which is a finite sum of the eigenfunctions associated to the eigenvalue λ_{j_0}. We write, for every $x \in \Omega$ and every $t \in [0,T]$,

$$\begin{aligned}|p_T(t,x) - e^{-\lambda(T-t)}\Phi_0(x)| &= \left|\sum_{j\geqslant k}\zeta_j\phi_j(x)e^{-\lambda_j(T-t)}\right| \\ &\leqslant \sum_{j\geqslant k}\left|\zeta_j\phi_j(x)\right|e^{-\lambda_j(T-t)}.\end{aligned}$$

Since $|\zeta_j|^2 \leqslant C_1, \forall\, j \in \mathbf{N}^*$, by the Weyl Law and sup-norm estimates for the eigenfunctions of A (see [88, Chapter 3]), we can find $\alpha \in (0,1)$ such that $\alpha\mu > \lambda$ and two constants $C_1, C_2 > 0$ independent of x, t and T such that

$$|p_T(t,x) - e^{-\lambda(T-t)}\Phi_0(x)| \leqslant C_1 e^{-\alpha\mu(T-t)}\sum_{j\geqslant k} j^{\frac{N-1}{2N}}e^{-C_2 j^{\frac{1}{N}}(T-t)}.$$

Let $\varepsilon > 0$ be arbitrary. We claim that there exists $C_\varepsilon > 0$ independent of x, t, T such that, for every $x \in \Omega$,

$$|p_T(t,x) - e^{-\lambda(T-t)}\Phi_0(x)| \leqslant C_\varepsilon e^{-\alpha\mu(T-t)} \quad \forall\, t \in (0, T-\varepsilon)$$

$$|p_T(t,x) - e^{-\lambda(T-t)}\Phi_0(x)| \leqslant C_\varepsilon \qquad \forall\, t \in (T-\varepsilon, T).$$

To conclude we take an arbitrary value for ε and we write μ instead of $\alpha\mu$ but always with $\mu > \lambda$ to get

$$\|p_T(t) - e^{-\lambda(T-t)}\Phi_0\|_{L^\infty(\Omega)} \leqslant C\, e^{-\mu(T-t)} \quad \forall\, t \in [0,T] \tag{1.24}$$

with $C > 0$ not depending on the final time T. Using the bathtub principle ([68, Theorem 1.16]) and since Φ_0 is analytic, we introduce $s_0 \in \mathbb{R}$ and the shape $\omega_0 = \{\Phi_0 \geqslant s_0\} \in \mathscr{U}_L$ such that χ_{ω_0} is solution of the auxiliary problem

$$\max_{u\in\mathscr{U}_L}\int_\Omega \Phi_0(x)u(x)\,dx. \tag{1.25}$$

Let $t \in [0,T]$ fixed. For $x \in \omega_0$, we remark that (1.24) implies that $p(t,x) \geqslant s_0 e^{-\lambda(T-t)} - C^{-\mu(T-t)}$. Reminding the definition of $s_T(t)$ in (1.12) we write

$$\begin{cases} \omega_0 \subset \left\{p(t,x) \geqslant s_0 e^{-\lambda(T-t)} - C^{-\mu(T-t)}\right\} \\ |\omega_0| = L|\Omega| \quad \text{and} \quad \left|\left\{p_T(t,x) \geqslant s_T(t)\right\}\right| \leqslant L|\Omega|. \end{cases}$$

Hence $s_T(t) \geqslant s_0 e^{-\lambda(T-t)} - C^{-\mu(T-t)}$. We change the roles of ω_0 and $\omega_T(t)$ to get $s_T(t) \leqslant s_0 e^{-\lambda(T-t)} + C^{-\mu(T-t)}$ and finally obtain

$$|s_T(t) - e^{-\lambda(T-t)}s_0| \leqslant C\, e^{-\mu(T-t)} \quad \forall\, t \in [0,T]. \tag{1.26}$$

We write $\Phi = s_0 - \Phi_0$, $\psi_T(t,x) = s_T(t) - p_T(t,x)$ and $\psi_0(t,x) = e^{-\lambda(T-t)}\Phi(x)$ and using (1.24) with (1.26), we get a new constant $C > 0$ independent of T such that

$$\|\psi_T(t,x) - \psi_0(t,x)\|_{L^\infty(\Omega)} \leqslant C\, e^{-\mu(T-t)}, \quad \forall\, t \in [0,T]. \tag{1.27}$$

We now follow arguments of [25] to establish the exponential turnpike property for the control and then for the state by using some information on the control χ_{ω_T}. We first remark that for all $t_1, t_2 \in [0,T]$, $\{\psi_0(t_1,\cdot) \leqslant 0\} = \{\psi_0(t_2,\cdot) \leqslant 0\} = \{\Phi \leqslant 0\}$. Then we take $t \in [0,T]$ and we compare the sets $\{\psi_0(t,\cdot) \leqslant 0\}, \{\psi_T(t,\cdot) \leqslant 0\}$ and $\{\psi_0(t,\cdot) + Ce^{-\mu(T-t)} \leqslant 0\}$. Thanks to (1.27) we get for every $t \in [0,T]$

$$\{\Phi \leqslant -Ce^{-(\mu-\lambda)(T-t)}\} \subset \{\psi_T(t,\cdot) \leqslant 0\} \subset \{\Phi \leqslant Ce^{-(\mu-\lambda)(T-t)}\} \tag{1.28}$$

$$\{\Phi \leqslant -Ce^{-(\mu-\lambda)(T-t)}\} \subset \{\psi_0(t,\cdot) \leqslant 0\} \subset \{\Phi \leqslant Ce^{-(\mu-\lambda)(T-t)}\}. \tag{1.29}$$

We infer from [25, Lemma 2.3] that for every $t \in [0,T]$,

$$d_{\mathscr{H}}\Big(\{\psi_T(t,\cdot) \leqslant 0\}, \{\Phi \leqslant 0\}\Big) \leqslant d_{\mathscr{H}}\Big(\{\Phi \leqslant -Ce^{-(\mu-\lambda)(T-t)}\}, \{\Phi \leqslant Ce^{-(\mu-\lambda)(T-t)}\}\Big). \tag{1.30}$$

To conclude, since $d_{\mathscr{H}}$ is a distance, we only have to estimate

$$d_{\mathscr{H}}\Big(\{\Phi \leqslant 0\}, \{\Phi \leqslant \pm Ce^{-(\mu-\lambda)(T-t)}\}\Big).$$

Lemma 1.11. *Let $f : \Omega \to \mathbb{R}$ be a continuously differentiable function and set $\Gamma = \{f = 0\}$. Under the assumption (S) : there exists $C > 0$ such that*

$$\|\nabla f(x)\| \geqslant C \quad \forall\, x \in \Gamma,$$

there exist $\varepsilon_0 > 0$ and $C_f > 0$ only depending on f such that for any $\varepsilon \leqslant \varepsilon_0$

$$d_{\mathscr{H}}(\{f \leqslant 0\}, \{f \leqslant \pm\varepsilon\}) \leqslant C_f \varepsilon.$$

Proof of Lemma 1.11. We consider f satisfying *(S)* with $\Gamma = \{\Phi = 0\}$. We assume by contradiction that for every $\varepsilon > 0$, there exists $x \in \{|f| \leqslant \varepsilon\}$ such that $\|\nabla f(x)\| = 0$. We take $\varepsilon = \frac{1}{n}$ and we subtract a subsequence $(x_n) \to x \in \{|f| \leqslant 1\}$ (which is compact). By continuity of f and of $\|\nabla f\|$, we have $x \in \Gamma$ and $\|f(x)\| = 0$, which raises contradiction with *(S)*. Hence we find $\varepsilon_0 > 0$ such that $\|\nabla f(x)\| \geqslant \frac{C}{2}$ for every $x \in \{|f| \leqslant \varepsilon\}$. We apply [20, Corollary 4] (see also [20, Theorem 2]) to get

$$d_{\mathscr{H}}(\{f \leqslant 0\}, \{f \leqslant \pm\varepsilon\}) \leqslant \frac{2}{C}\varepsilon.$$

A more general statement can be found in [20; 25]. □

We infer that Φ satisfies *(S)* on $\|\nabla_x \psi_0(t,x)\| = e^{-\lambda(T-t)}\|\nabla_x \Phi(x)\|$ for $x \in \Omega$. We first remark that Φ_0 satisfies $A\Phi_0 = \lambda_{j_0}\Phi_0, \Phi_{0|\Gamma} = s_0$ and that the set $\Gamma = \{\Phi_0 = 0\}$ is compact. Since Ω has a C^2 boundary and $c = 0$ the Hopf lemma (see [31, Section 6.4]) gives

$$x_0 \in \Gamma_0 \implies \|\nabla_x \Phi(x_0)\| = \|\nabla_x \Phi_0(x_0)\| > 0.$$

Hence there exists $C_0 > 0$ not depending on t, T such that for every $x \in \Gamma_0$, $\|\nabla_x \Phi(x_0)\| \geqslant C_0 > 0$. We take $\nu > 0, e^{-\mu\nu} \leqslant \varepsilon_0$. We remark that $e^{-\mu(T-t)} \leqslant \varepsilon_0, \forall\, t \in (0, T-\nu)$ and we use Lemma 1.11 combined with (1.30) to get that, for every $t \in (0, T-\nu)$,

$$d_{\mathscr{H}}\Big(\{\psi_T(t,\cdot) \leqslant 0\}, \{\Phi \leqslant 0\}\Big) \leqslant C_0 e^{-(\mu-\lambda)(T-t)}.$$

We adapt the constant C_0 such that on the compact interval $t \in (T-\nu, T)$ the sets are the same whatever $T \geqslant T_0 > 0$ may be, to get that, for every $t \in (0, T)$,

$$d_{\mathscr{H}}\Big(\{\psi_T(t,\cdot) \leqslant 0\}, \{\Phi \leqslant 0\}\Big) \leqslant C_0 e^{-(\mu-\lambda)(T-t)}.$$

We obtain therefore an exponential turnpike property for the control in the sense of the Hausdorff distance

$$d_{\mathscr{H}}(\omega_T(t), \omega_0) \leqslant C_0 e^{-(\mu-\lambda)(T-t)} \quad \forall\, t \in [0, T]. \tag{1.31}$$

Here is a possible way to find a further turnpike property on state and adjoint. We could use a similar argument (valid only for convex sets) as in [47, Theorem 1-(a)] : $\|\chi_{\omega_T(t)} - \chi_{\omega_0}\| \leqslant C d_{\mathscr{H}}(\omega_T(t), \omega_0)$. Denoting by $b_\omega = d_\omega - d_{\omega^c}$ the oriented distance, we follow [28, Theorem 4.1-(b)] and [28, Theorem 5.1-(iii)(iv)] and we use the inequality $\|\chi_{\overline{A}_1} - \chi_{\overline{A}_2}\| \leqslant \|d_{A_1} - d_{A_2}\|_{W^{1,2}(\Omega)} \leqslant \|b_{A_1} - b_{A_2}\|_{W^{1,2}(\Omega)} = \|b_{A_1} - b_{A_2}\| + \|\nabla b_{A_1} - \nabla b_{A_2}\|$ to try to make the link between $\|\chi_{\omega_T(t)} - \chi_{\omega_0}\|$ and $d_{\mathscr{H}}(\omega_T(t), \omega_0)$. Afterwards, applying Gronwall inequality (1.39), we get

$$\|y(t) - \bar{y}\|_{L^2(\Omega)} \leqslant C_0 e^{-\frac{(\mu+\lambda)}{2}(T-t)} \quad \forall\, t \in (0, T) \tag{1.32}$$

with $\bar{y}$ solution of $Ay = \chi_{\omega_0}, y_{|\partial\Omega} = 0$. Taking $\kappa = \frac{\mu+\lambda}{2} > 0$ and by application of Gronwall inequality (1.39) for the adjoint, we finally get the exponential turnpike property for the state, adjoint and control.

1.3 Numerical simulations : optimal shape design for the 2D heat equation

Mayer Case. We illustrate the exponential turnpike phenomenon involved in the theorem 1.8 with a simulation in dimension 1. We still focus on the heat equation seen on the domain $\Omega = [-1, 1]$ with a constant final target function $y_d = 0.1$ and try to solve the minimization problem

$$\min J_T(\omega(\cdot)) = \int_{-1}^{1} \big(y(T,x) - 0.1\big)^2\, dx$$
$$\text{subject to : } \begin{cases} \partial_t y - \partial_{xx} y = \chi_{\omega(\cdot)} & \forall (t,x) \in (0,T) \times (-1,1) \\ y(t,0) = y(t,1) = 0, & \forall\, t \in (0,T) \\ y(0,x) = y_0, & \forall\, x \in (-1,1). \end{cases}$$

We do not concentrate here on how to treat such a problem numerically, which will be presented in Chapter 3. To do so, we use the `AMPL` software combined with the `IpOpt` method and plot in Figure 1.2 the shape evolving in time. We try to show that the inequality of the Theorem 1.8 is well verified. At the beginning the optimal shape in time remains stationary until the final

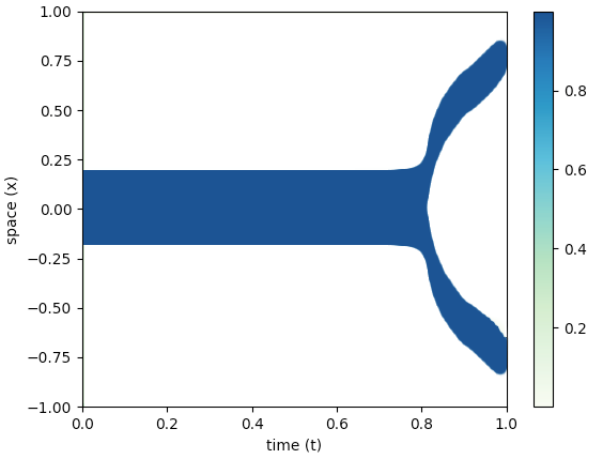


FIGURE 1.2 – 1D time-varying shape for Mayer case - $T = 3$

time T is almost reached. Once T is almost reached, the optimal shape evolves to minimize the final cost. The constant M implied in the Theorem 1.8 is independent of the final time T, which we verify by plotting for different values of T the behavior of the error $t \mapsto d_{\mathscr{H}}(\omega_T(t), \bar{\omega})$. We plot the time evolution of the Hausdorff distance between the optimal shape found and the stationary shape $t \mapsto d_{\mathscr{H}}(\omega_T(t), \bar{\omega})$ for several final times $T \in \{1, 3, 5\}$. From one side, we observe

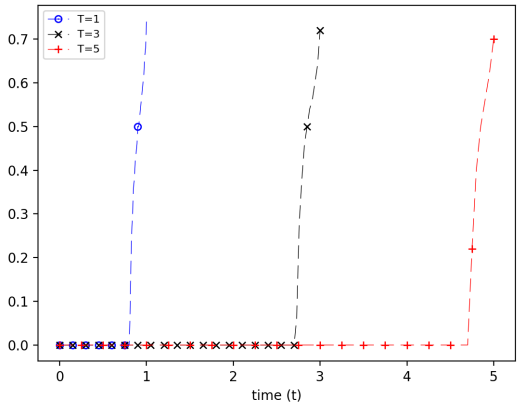


FIGURE 1.3 – Error between time and static optimal shapes with respect to the Hausdorff distance $t \mapsto d_{\mathscr{H}}(\omega_T(t), \bar{\omega})$ introduced in Theorem 1.8

in Figure 1.3, that the biggest is T, the more often is $d_{\mathscr{H}}(\omega_T(t), \bar{\omega})$ near to 0. On the other side, whatever the final time T choosen, the behavior of $t \mapsto d_{\mathscr{H}}(\omega_T(t), \bar{\omega})$ stays unchanged close to the final time T. Moreover, close to T, the way in which the error quickly leaves the 0 value to reach its final value highlights the exponential character of the turnpike phenomenon, and in the same way for T which takes larger and larger values.

Lagrange Case. Numerical simulations will be detailed in Chapter 3 where several methods are introduced. The one choosen to obtain figures below is based on the numerical solving of PDE constrained optimization problems introduced in Chapter 2. From the time being, we take $\Omega = [-1,1]^2$, $L = \frac{1}{8}$, $T \in \{1,\dots,5\}$, $y_d = \text{Cst} = 0.1$ and $y_0 = 0$. We focus on the heat equation and consider the minimization problem

$$\min_{\omega(\cdot)} \int_0^T \int_{[-1,1]^2} |y(t,x) - 0.1|^2 \, dx \, dt \tag{1.33}$$

under the constraints

$$\partial_t y - \Delta y = \chi_\omega, \qquad y(0,\cdot) = 0, \qquad y_{|\partial\Omega} = 0. \tag{1.34}$$

We compute numerically a solution by solving the equivalent convexified problem using a direct method in optimal control (see MR2224013). We discretize here with an implicit Euler method in time and with a decomposition on a finite element mesh of Ω using `FreeFEM` (see [56]). We finally express the optimal control problem as a quadratic programming problem in finite dimension. We then use the routine `IpOpt` (see [97]) on a standard office machine. The numerical solution of such problems has been shown to require efficient optimization methods which we present in the next Chapter 2 and whose applications to our problem are later illustrated in the Chapter 3.

We plot on the Figure 1.4 the evolution in time of the optimal shape $t \mapsto \omega(t)$ which appears as a cylinder whose section at time t represents the shape $\omega(t)$. At the beginning ($t = 0$) we notice that the shape concentrates in the middle of Ω in order to heat up as fast as possible and approach the state y close to $\bar{y}$. Once this is acceptable, the shape is almost stationary for a long time. Finally, since the target y_d is taken here as a constant, the optimal final state $y_T(T)$ must be as flat as possible. Indeed, for $t < T$ and by plotting the state curve, we observe that $y_T(t)$ is much larger in the center of Ω than near the boundary. Thus, at the final moment, the shape approaches the boundary of Ω so that $y_T(T)$ becomes larger near it and smaller at the center of Ω. We thus observe that $y_T(T)$ is almost constant in Ω and very close to y_d.

We plot in Figure 1.5 a comparison between the optimal shape at several times (in red) and the optimal static shape (in yellow) and notice the same design when $t = \frac{T}{2}$.

Now, in order to highlight the turnpike phenomenon, we plot the time evolution of the distance between the optimal dynamic triple and the optimal static triple $t \mapsto \|y_T(t) - \bar{y}\| + \|p_T(t) - \bar{p}\| + \|\chi_{\omega_T(t)} - \chi_{\bar{\omega}}\|$. In the figure 1.6, we notice that this function is most of the time very close to 0 with terminal arcs which seems to be exponential. This behavior leads us to conjecture that the exponential property of the turnpike should be satisfied. We notice on Figures 1.6 and 1.9 that the initial arcs all start from the same point while the final arcs close to the final time do not end at the same point depending on the final time concerned while we expect it to end at the same final point when T is large enough. For the sake of clarity, we represent on Figure 1.7 this error in dimension 1, because we are limited in the number of time steps in dimension 2 (with one hundred time steps, the optimization algorithm would process more than ten million variables). We therefore observe that the endpoints are always identical and conclude that in dimension 2, this difference in the final time could be due to the limited number of time steps in dimension 2 and therefore to a lack of numerical precision in the final time when computing the error between the two optimal triples.

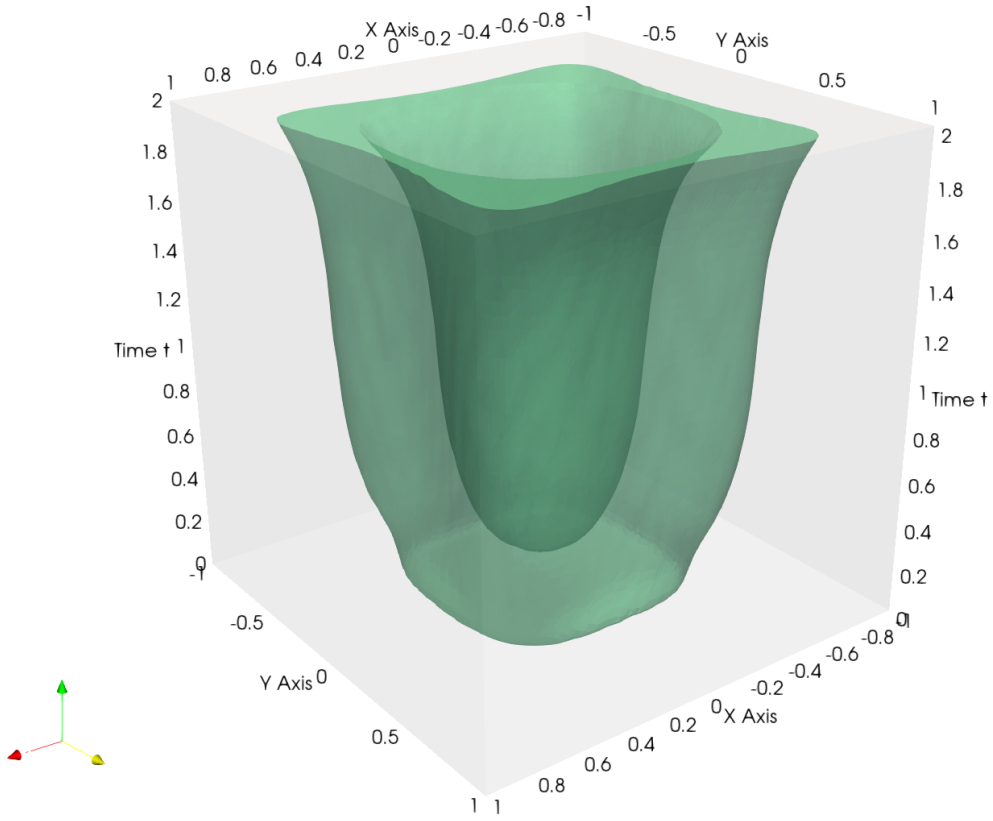


FIGURE 1.4 – Optimal shape's time evolution cylinder - $T = 2$

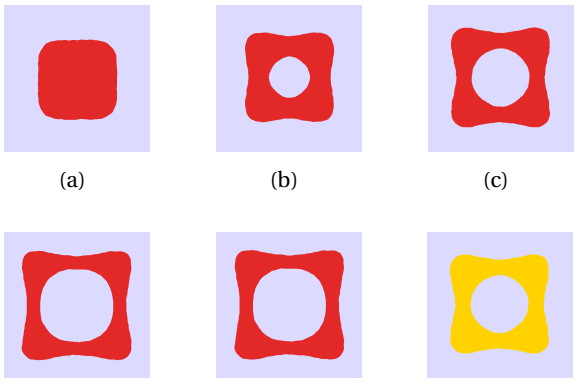


FIGURE 1.5 – Time optimal shape $T = 5$ - Static shape : (a) $t = 0$; (b) $t = 0.5$; (c) $t \in [1,4]$; (d) $t = 4.5$; (e) $t = T$; (f) static shape

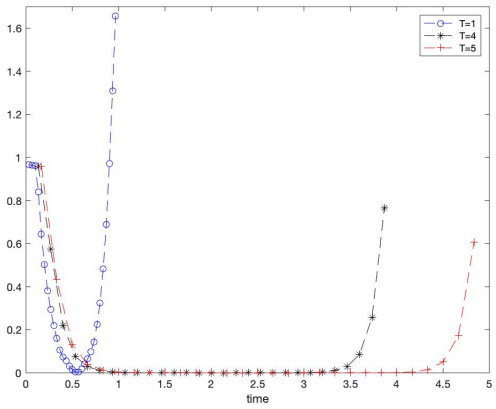


FIGURE 1.6 – Error between dynamical optimal triple and static one

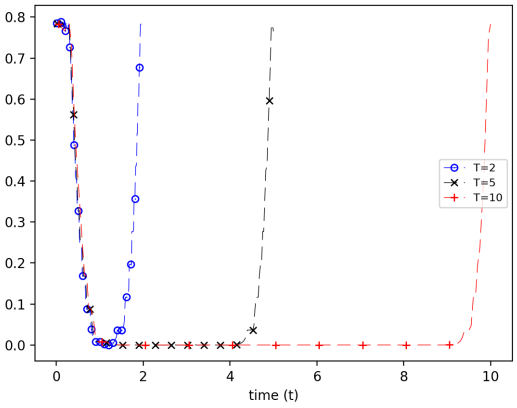


FIGURE 1.7 – Error between dynamical optimal triple and static one (1D)

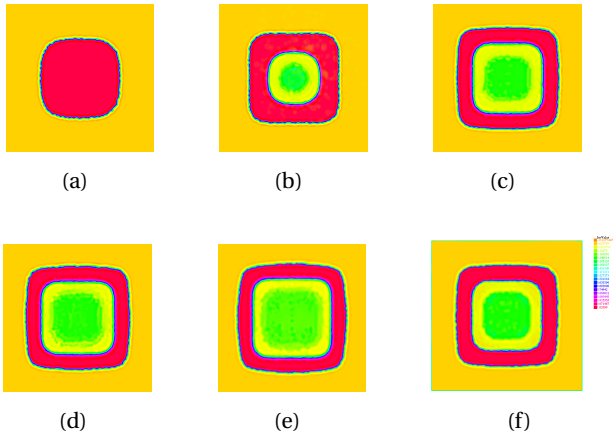


FIGURE 1.8 – Relaxation phenomenon : (a) $t = 0$; (b) $t = 0.5$; (c) $t \in [1,4]$; (d) $t = 4.5$; (e) $t = T$; (f) static shape

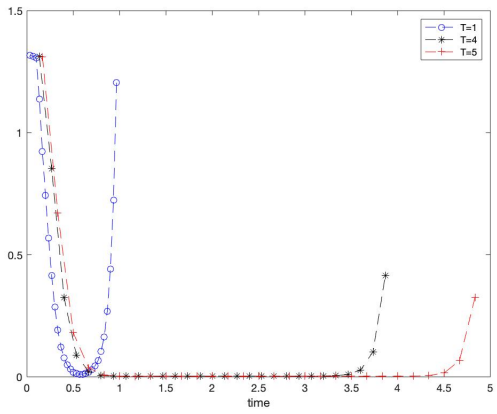


FIGURE 1.9 – Error between dynamical optimal triple and static one (Relaxation case)

To complete these numerical remarks, we must specify that the existence of optimal forms for y_d seems to be acquired when y_d is convex. We see indeed on Figure 1.5 the existence of the time optimal shape for y_d convex on Ω. In the contrary case, we can sometimes observe a relaxation phenomenon due to the presence of $\bar{c}$ and $c_T(\cdot)$ in the optimality conditions (1.5) - (1.7).

Indeed, we consider the same problem $(\boldsymbol{OCP})_{\mathbf{T}}$ in 2D with $\Omega = [-1,1]^2$, $L = \frac{1}{8}$, $T = 5$ and the static one associated $(\boldsymbol{SOP})$ with the nonconvex function $y_d(x,y) = -\frac{1}{20}(x^2 + y^2 - 2)$.

In Figure 1.8 we see that optimal control $(a_T, \bar{a})$ of $(\boldsymbol{OCP})_{\mathbf{T}}$ and $(\boldsymbol{SOP})$ take values in $(0,1)$ in the middle of Ω, which illustrates that relaxation occurs for some nonconvex y_d. Here, y_d was chosen such that $-\Delta y_d \in (0,1)$. We have tuned the parameter L to observe the relaxation phenomenon, but for same y_d and smaller L, optimal solutions may be shapes. We notice that despite the relaxation, turnpike still occurs in Figure 1.9.

1.4 Further comments

Numerical simulations when $\Delta y_d > 0$ lead us to conjecture the existence of an optimal shape for $(\boldsymbol{DSD})_{\mathbf{T}}$, as we have not yet observed any relaxation phenomenon in this case. The question of existence is nevertheless very difficult and could be proved thanks to arguments like maximal regularity properties and Hölder estimates for solutions of parabolic equations.

Moreover, still on the basis of our simulations and in particular of Figure 1.6, we conjecture the exponential turnpike property.

The work we have presented here concerns the second order parabolic equations and in particular the heat equation. Concerning the Mayer case, we have used in our arguments Weyl's law, sup-norm estimates of the eigenelements (see [88]) and the analyticity of the solutions (analytic-hypoelliptic operator). Nevertheless, concerning the Lagrange case and having in mind [81; 91], it seems reasonable to extend our results to general local parabolic operators which satisfy an energy inequality (1.38) as well as the maximum principle to ensure the existence of solutions, at least for the relaxed problem. However, some results such as the theorem 1.1.2-(b) must be adapted. Moreover, we consider a linear partial differential equation that gives uniqueness of the solution due to the strict convexity of the criterion. On the contrary, if we do not have uniqueness, as in [91], the notion of measure-turnpike seems to be a relevant way to get turnpike results.

Appendix

1.A Energy inequalities

We recall some useful energy inequalities for elliptic operators A defined in Section 1, requisite to study the existence of solutions and turnpike phenomenon. In that aim, we first introduce Poincaré's inequality

Theorem 1.12. *Let Ω be an open bounded subset of $\mathbb{R}^d$ with smooth boundary $\partial\Omega$, then it exists $C_p > 0$ such that, for any $y \in H_0^1(\Omega)$*

$$\|y\|^2_{L^2(\Omega)} \leqslant C_p \|\nabla y\|^2_{L^2(\Omega)}. \tag{1.35}$$

We then prove the following results

Theorem 1.13. *If $\theta > \theta_1 > 0$ in (1.2), where θ_1 is the largest root of the polynomial $P(X) = \frac{X^2}{4C_p} - \|c\|_{L^\infty(\Omega)} X - \frac{\left(\sum_{i=1}^d \|b_i\|_{L^\infty(\Omega)}\right)^2}{2}$, we can find $\gamma > 0$ such that and*

$$(Ay, y)_{L^2(\Omega)} \geqslant \gamma \|y\|^2_{H_0^1(\Omega)} \quad \forall y \in H_0^1(\Omega). \tag{1.36}$$

Proof. Let $y \in H_0^1(\Omega)$ and write

$$(Ay, y)_{L^2(\Omega)} = \int_\Omega \left(\sum_{i,j=1}^d \left(a_{ij}(x) \partial_{x_i} y \partial_{x_j} y \right) + \sum_{i=1}^d b_i(x) y \partial_{x_i} y + c(x) y^2 \right) dx$$

so that, by means of the ellipticity condition (1.1) the following inequality holds

$$\theta\|\nabla y\|^2_{L^2(\Omega)} \leqslant (Ay,y)_{L^2(\Omega)} + \sum_{i=1}^{d} \|b_i\|_{L^\infty(\Omega)} \int_\Omega |y|\,|\nabla y|\,dx + \|c\|_{L^\infty(\Omega)}\|y\|^2_{L^2(\Omega)}. \tag{1.37}$$

We introduce a positive parameter $\epsilon > 0$ and write Cauchy's inequality to delink the term $|y|\,|\nabla y|$ in $\int_\Omega |y|\,|\nabla y|\,dx$ as

$$\int_\Omega |y|\,|\nabla y|\,dx \leqslant \epsilon \int_\Omega |\nabla y|^2\,dx + \frac{1}{4\epsilon}\int_\Omega y^2\,dx$$

and introduce it in the previous inequality so that

$$\left(\theta - \epsilon \sum_{i=1}^{d} \|b_i\|_{L^\infty(\Omega)}\right)\|\nabla y\|^2_{L^2(\Omega)} \leqslant (Ay,y)_{L^2(\Omega)} + \left(\|c\|_{L^\infty(\Omega)} + \frac{1}{4\epsilon}\sum_{i=1}^{d}\|b_i\|_{L^\infty(\Omega)}\right)\|y\|^2_{L^2(\Omega)}.$$

We almost get (1.36) insofar $\theta - \epsilon\sum_{i=1}^{d}\|b_i\|_{L^\infty(\Omega)}$ must be positive. In that aim, we choose $\epsilon = \frac{\theta}{2\sum_{i=1}^{d}\|b_i\|_{L^\infty(\Omega)}}$ and we decompose the left term $\|\nabla y\|^2_{L^2(\Omega)}$ by means of Poincaré's inequality (1.35) so that $\|\nabla y\|^2_{L^2(\Omega)} \geqslant \frac{1}{2}\|\nabla y\|^2_{L^2(\Omega)} + \frac{1}{2C_p}\|y\|^2_{L^2(\Omega)}$ to reword (1.37) as

$$\frac{\theta}{4}\|\nabla y\|^2_{L^2(\Omega)} \leqslant (Ay,y)_{L^2(\Omega)} + \left(\|c\|_{L^\infty(\Omega)} + \frac{1}{2\theta}\left(\sum_{i=1}^{d}\|b_i\|_{L^\infty(\Omega)}\right)^2 - \frac{\theta}{4C_p}\right)\|y\|^2_{L^2(\Omega)}.$$

It only remains to show that the right term $\left(\|c\|_{L^\infty(\Omega)} + \frac{1}{2\theta}\left(\sum_{i=1}^{d}\|b_i\|_{L^\infty(\Omega)}\right)^2 - \frac{\theta}{4C_p}\right)$ is negative. We recognize the quantity $-\theta P(\theta)$, with P the polynomial defined in Theorem 1.13 and whose discriminant is strictly positive. Since θ verifies $\theta > \theta_1$, $\theta P(\theta) > 0$ (since $P(0) < 0$, we obviously have $\theta_1 > 0$) and thus

$$(Ay,y)_{L^2(\Omega)} \geqslant \frac{\theta}{4}\|\nabla y\|^2_{L^2(\Omega)} + \theta P(\theta)\|y\|^2_{L^2(\Omega)}.$$

Therefore, we take $\gamma = \min(\frac{\theta}{4}, \theta P(\theta))$ and (1.36) holds. □

Remark 1.14. *We claim that the condition $\theta > \theta_1$ is not optimal since rough inequalities have been used (especially by using Poincaré's inequality and by putting $\epsilon = \frac{\theta}{2\sum_{i=1}^{d}\|b_i\|_{L^\infty(\Omega)}}$) and is most likely refinable.*

Theorem 1.13 guarantees the existence of a unique solution of state equation in problem (***SOP***) by Lax-Milgram Theorem. From Theorem 1.13 follows energy inequalities for parabolic operators in Theorem 1.15.(a) used to show turnpike properties. With the aim to have both existence and energy inequalities for PDE (1.4), we remind the following result based on [31].

Theorem 1.15. *Let A be an elliptic operator verifying the ellipticity condition (1.2) in the framework of Theorem 1.13.*

(a) One can find a constant $C > 0$, independent of the final time T, such that for any solution y of (1.4) the following inequality holds

$$\|y(t)\|^2_{L^2(\Omega)} + \int_0^t \|y(s)\|^2_{H_0^1(\Omega)}\,ds \leqslant C\left(\|y_0\|^2_{L^2(\Omega)} + \int_0^t \|a(s)\|^2_{L^2(\Omega)}\,ds\right) \quad \forall\, t \in (0,T). \tag{1.38}$$

(b) One can find constants $C_1, C_2 > 0$, independent of the final time T, such that for any $y \in$ solution of (1.4), we have

$$\|y(t)\|^2 \leqslant \|y_0\|^2 e^{-C_1 t} + C_2 \int_0^t e^{-C_1(t-s)} \|a(s)\|^2 \, ds \quad \forall t \in (0,T). \tag{1.39}$$

Proof. According to [31, Section 7 - Theorem 5], let $y \in L^2(0,T;H^2(\Omega)) \cap L^\infty(0,T;H_0^1(\Omega))$ such that $\partial_t y \in L^2(0,T;L^2(\Omega))$ be the unique solution of (1.4).

Remark 1.16. *The proof for such a setting is essentially based on* Galerkin's approximation, *by bringing in the complete set of appropriately normalized eigenfunctions of the Dirichlet Laplacian as smooth functions $(\phi_k)_{k\in 1..\infty}$ such that*

$$\begin{aligned} (\phi_i,\phi_j)_{H_0^1(\Omega)} = 0 \text{ if } i \neq j: &\quad \text{orthogonal basis of } H_0^1(\Omega), \\ (\phi_i,\phi_j)_{L^2(\Omega)} = \delta_{ij}: &\quad \text{orthonormal basis of } L^2(\Omega), \end{aligned}$$

and by decomposing the PDE solution onto the sequential resulting finite dimensional spaces.

We apply [31, Section 5 - Theorem 3] to get that $y \in C(0,T;L^2(\Omega))$ such that the mapping $t \mapsto \|y(t)\|_{L^2(\Omega)}$ is absolutely continuous and its derivative is $\frac{d}{dt}\|y(t)\|_{L^2(\Omega)} = \langle \partial_t y, y\rangle_{H^{-1}(\Omega),H_0^1(\Omega)}$. Thus, for any t in $(0,T)$, let multiply (1.4) by $y(t)$ and integrate over Ω to get

$$\frac{d}{dt}\|y(t)\|^2_{L^2(\Omega)} + (Ay(t),y(t))_{L^2(\Omega)} = (a(t),y(t))_{L^2(\Omega)} \quad \forall t \in (0,T).$$

Let us notice that for almost every t in $(0,T)$, $y(t)$ lies in $H_0^1(\Omega)$ such that, by applying Theorem 1.13 we get

$$(Ay(t),y(t))_{L^2(\Omega)} \geqslant \gamma \|y(t)\|^2_{H_0^1(\Omega)} \quad \text{a.e. } t \in (0,T),$$

and finally write by means of Cauchy's inequality with $\epsilon = \frac{\gamma}{2}$

$$\frac{d}{dt}\|y(s)\|^2_{L^2(\Omega)} + \frac{\gamma}{2}\|y(s)\|^2_{H_0^1(\Omega)} \leqslant \frac{1}{2\gamma}\|a(s)\|^2_{L^2(\Omega)} \quad \text{a.e. } s \in (0,T), \tag{1.40}$$

with $C = \frac{\gamma}{2} > 0$ independent of the final time T.

(a). Related to the first part of the Theorem 1.13, we easily integrate (1.40) over $(0,t)$ for every $t \in (0,T)$ such that we finally write

$$\|y(t)\|^2_{L^2(\Omega)} + \frac{\gamma}{2}\int_0^t \|y(s)\|^2_{H_0^1(\Omega)}\, ds \leqslant \|y_0\|^2_{L^2(\Omega)} + \frac{1}{2\gamma}\int_0^t \|a(s)\|^2_{L^2(\Omega)}\, ds \quad \forall t \in (0,T).$$

Thus, (1.38) holds by taking $C = \dfrac{\max(1,\frac{1}{2\gamma})}{\min(1,\frac{\gamma}{2})}$.

(b). We improve previous inequality by using the following result.

Lemma 1.17. *Let $\eta : \mathbb{R}^+ \mapsto \mathbb{R}^+$ be an absolutely continuous function such that there exist $C > 0$ and $h : \mathbb{R}^+ \mapsto \mathbb{R}^+$ measurable verifying*

$$\eta'(t) + C\eta(t) \leqslant h(t) \quad a.e\ t \in (0,T).$$

Thus, one can get the following estimate

$$\eta(t) \leqslant \eta(0)e^{-Ct} + \int_0^t e^{-C_1(s-t)} h(s)\, ds \quad \forall t \in (0,T).$$

Proof. We introduce the ODE

$$\eta'(t) + C\eta(t) = \alpha(t), \quad \forall t \in (0,T),$$

whose solution by common ordinary differential equations theory reads

$$\eta(t) = \eta(0)e^{-Ct} + \int_0^t e^{-C(s-t)}\alpha(s)\,ds \quad \forall t \in (0,T).$$

Let us then notice that for almost every s in $(0,T)$, $\alpha(s) \leqslant h(s)$ such that, by integrating over $(0,t)$ for any t in $(0,T)$, we deduce

$$\eta(t) \leqslant \eta(0) + \int_0^t e^{-C(s-t)}h(s)\,ds \quad \forall t \in (0,T).$$

□

We then apply the comparison lemma 1.17 to $\eta : t \mapsto \|y(t)\|_{L^2(\Omega)}$ with $C = \frac{\gamma}{2}$ and $h : t \mapsto \frac{1}{2\gamma}\|a(t)\|_{L^2(\Omega)}$ that are well verifying assumptions needed and so that we deduce

$$\|y(t)\|^2 \leqslant \|y_0\|^2 e^{-\frac{\gamma}{2}t} + \frac{1}{2\gamma}\int_0^t e^{-\frac{\gamma}{2}(t-s)}\|a(s)\|^2\,ds \quad \forall t \in (0,T),$$

with $C_1 = \frac{\gamma}{2} > 0$ and $C_2 = \frac{1}{2\gamma} > 0$ independent of the final time T. □

1.B Bathtub principle

We remind the following result, in an alternative form than the one stated in [68] that we use to characterize the optimal solutions of relaxed problems involved in Section 1.

Theorem 1.18. *Let (Ω,Σ,μ) be a measurable space and a function $f : \Omega \mapsto \mathbb{R}$ whose any sub-level sets has finite Lebesgue measure $\mu\{x \in \Omega, f(x) < t\}$ for all t in $\mathbb{R}$. Besides, for G stricly positive, we introduce the class of measurable functions*

$$\mathscr{C} = \left\{ g : \Omega \mapsto \mathbb{R}, \quad 0 \leqslant g \leqslant 1 \text{ and } \int_\Omega g(x)\mu(dx) \leqslant G \right\}.$$

Thus, the maximization problem

$$\sup_{g\in\mathscr{C}} \int_\Omega f(x)g(x)\mu(dx)$$

is solved by function of the form

$$g(x) = \chi_{\{f>s\}}(x) + c\chi_{\{f=s\}}(x)$$

with a level $s \in \mathbb{R}$ and some constant $c \in (0,1)$.

Proof. We first introduce the sets $\Omega^+ = \{f > 0\}$ and $\Omega^0 = \{f = 0\}$ and distinguish between cases according to whether $\mu(\Omega^+) \leqslant G$.

In a first attempt, we assume that $\mu(\Omega^+) \leqslant G$. Thus, without any effort, we notice that $\int_\Omega f(x)g(x)\mu(dx)$ is going to be maximal when $g(x) = 1$ if $f(x) > 0$ and $g(x) = 0$ when $f(x) < 0$ and its value is $\int_{\Omega^+} f(x)\mu(dx)$. Nevertheless, solution is not unique and g can take several value $c \in (0,1)$ on Ω^0 such that $g = \chi_{\Omega^+} + c\chi_{\Omega^0}$ insofar $\mu(\Omega^+) + c\mu(\Omega^0) \leqslant G$.

We now assume that $\mu(\Omega^+) > G$. This means that f is positive on a set whose measure is larger than G. Therefore, it is intuitive to notice that the constraint $\int_\Omega g(x)\mu(dx) \leqslant G$ will consequently be saturated such that $\int_\Omega g(x)\mu(dx) = G$. We thus introduce

$$s = \inf\big\{t \in \mathbb{R}, \mu\{f > t\} \leqslant G\big\}$$

and the function

$$g(x) = \chi_{\{f>s\}}(x) + c\chi_{\{f=s\}}(x)$$

such that $\mu\{f > s\} + c\mu\{f = s\} = G$. Let h be any other function in $\mathscr{C}$ and let us verify that

$$\int_\Omega f(x)g(x)\mu(dx) \geqslant \int_\Omega f(x)h(x)\mu(dx).$$

We decompose the integral as follows

$$\int_\Omega f(x)(g(x) - h(x))\mu(dx) = \int_{\{f<s\}\cup\{f=s\}\cup\{f>s\}} f(x)(g(x) - h(x))\mu(dx)$$

and notice that

$$g - h \geqslant 0 \text{ on } \{f > s\} \implies \int_{\{f>s\}} f(x)(g(x) - h(x))\mu(dx) \geqslant s\int_{\{f>s\}} (g(x) - h(x))\mu(dx)$$
$$g - h \leqslant 0 \text{ on } \{f < s\} \implies \int_{\{f<s\}} f(x)(g(x) - h(x))\mu(dx) \geqslant s\int_{\{f<s\}} (g(x) - h(x))\mu(dx)$$

such that

$$\int_\Omega f(x)(g(x) - h(x))\mu(dx) \geqslant s\int_\Omega (g(x) - h(x))\mu(dx) = sG - s\int_\Omega h(x)\mu(dx) \geqslant 0.$$

□

Chapter 2

PDE-constrained optimization using `FreeFEM` combined with `IpOpt` and automatic differentiation

Table of contents

2.1 Preliminaries . . . 47
2.1.1 The `FreeFEM` software . . . 47
2.1.2 PDE constrained optimization . . . 50
2.1.3 Optimization and discretization strategies . . . 59
2.1.4 The optimization routine `IpOpt` . . . 61
2.1.5 Automatic differentiation . . . 63
2.2 Linear quadratic PDE constrained optimization . . . 67
2.2.1 Derivatives of discretized functions (*FDTO*) . . . 69
2.2.2 Discretization of continuous derivatives (*FOTD*) . . . 71
2.2.3 Inhomogeneous Dirichlet boundary conditions . . . 74
2.2.4 Automatic differentiation alternative . . . 75
2.3 Extension to time-dependent problems . . . 78
2.3.1 Implicit Euler scheme. . . . 78
2.3.2 Time discretization with `FreeFEM`. . . . 79
2.4 Optimization under semilinear PDE constraints . . . 81
2.5 Optimal shape design problems . . . 86
2.6 Boundary shape optimization . . . 87
2.6.1 Boundary and domain parametrization . . . 87
2.6.2 Shape optimization problem . . . 91
2.6.3 Sensitivity analysis . . . 92
2.6.4 Codes and results . . . 96
2.6.5 Further comments . . . 100
Appendices . . . 101
2.A Some `FreeFEM` functions . . . 101
2.B Semi-automatic differentiation and adjoint method . . . 102
2.C PDE Optimization with `Python` or `Matlab` . . . 105

Abstract In this chapter, we show how the partial differential equation solver FreeFEM can be combined with efficient optimization tools to numerically solve difficult optimal control problems constrained by partial differential equations. As an optimization solver, we use the interior point routine IpOpt and show how it can be branched to FreeFEM as well as to automatic differentiation approaches like AMPL or CasADi. We illustrate the effectiveness of the different methods with several representative examples, ranging from classical linear quadratic problems to the non-trivial problem of the optimal shape of a micro-swimmer in a fluid. We address the issue of choosing direct or indirect methods and explain how automatic differentiation can be an effective alternative. We also show the potential of FreeFEM for mesh management and modification in optimal shape problems.

2.1 Preliminaries

2.1.1 The FreeFEM software

FreeFEM (see [56]) is a software developed in C++ to solve PDEs with the finite element method in 1, 2 and 3 dimensions. The meshes are generated thanks to an advanced automatic mesh generator and the choice of the triangular finite element space is so varied that most of the PDEs considered can be quickly solved via the discretization of the associated variational formulation. The documentation contains a complete introduction for a quick start as well as many classical examples such as heat conduction, elasticity system, Navier-Stokes equations. Users who are not familiar with the language are invited to browse the section [3, Learning by examples] where multiple examples will allow to advance step by step. We will see that not only does FreeFEM have a user-friendly interface but also that its syntax is very similar to the mathematical problems considered. Mathematically, the PDE problems considered must be solved in the context of an appropriate variational formulation (see, for example, [85]). Indeed, FreeFEM was developed to reformulate the PDE in its weak form, in order to discretize the resulting variational formulation according to a well suited choice of finite elements. As a first basic example, let us consider the Poisson equation

$$-\Delta y = u \text{ in } \Omega, \qquad y \in H_0^1(\Omega), \tag{2.1}$$

in some domain Ω, for some u in $L^2(\Omega)$. Its variational formulation is :

$$\text{find } y \in H_0^1(\Omega), \quad \int_\Omega \nabla y \cdot \nabla v \, dx = \int_\Omega uv \, dx \qquad \forall v \in H_0^1(\Omega) \tag{2.2}$$

$$\text{i.e.,} \qquad a(y,v) - b(u,v) = 0, \qquad \forall v \in H_0^1(\Omega), \tag{2.3}$$

where

$$a(y,v) = (\nabla y, \nabla v)_{L^2(\Omega)}$$

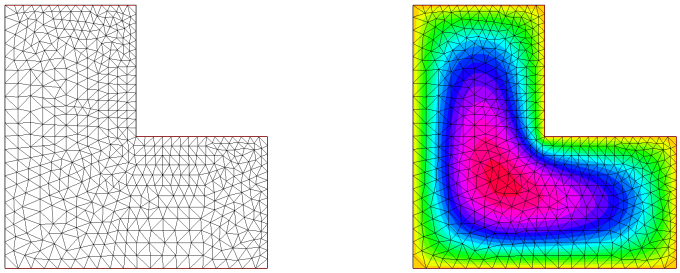
is a continuous and coercive bilinear form and

$$b(u,\cdot) = (u,\cdot)_{L^2(\Omega)}$$

is a continuous linear form, both being defined on the Hilbert space $H_0^1(\Omega)$. The Lax-Milgram theorem guarantees the existence of a weak solution of (2.2).

In order to discretize (2.2), we introduce a triangulation of the domain Ω (i.e. a mesh) as well as an appropriate finite element space which guarantees that the finite dimensional linear system resulting from the discretization of (2.3) on the basis of the finite element space is well-posed (i.e. invertible).

As an example, we solve (2.1) on a 2D L-shaped domain Ω (see Figure 2.1). The script FreeFEM for generating the mesh is given in the Code 2.1.



(a) (b)

FIGURE 2.1 – Example (2.1) : (a) L-mesh; (b) solution y of (2.2) for $u = 1$

```
border a(t=0,1){x=t; y=0; label=1;};
border b(t=0,0.5){x=1; y=t; label=2;};
border c(t=0,0.5){x=1-t; y=0.5; label=3;};
border d(t=0.5,1){x=0.5; y=t; label=4;};
border e(t=0.5,1){x=1-t; y=1; label=5;};
border g(t=0,1){x=0; y=1-t; label=6;};

mesh Th = buildmesh(a(20) + b(15) + c(15) + d(15) + e(15) + g(20));
```

Code 2.1 – Mesh generation with buildmesh

Here, `a(20)` means that the boundary involved and defined by the border `a` is split into 20 parts. The finite element space V_h is

$$V_h = \left\{ v \in H^1(\Omega), \quad \forall K \in T_h \;\; v_{|K} \in P_1 \right\} = \text{Vect}(\phi_i)_{i \in \{1..n_d\}}$$

where $\mathbb{P}_1$ is the space of continuous piecewise linear functions and whose basis is (ϕ_i). One of the big advantages of `FreeFEM` is that the user does not need to code the specificities of the mesh and the finite element functions. This is done automatically by the command **`fespace`** The access to the different data of the mesh and the finite element functions is then direct. Moreover, we will see that the use of macros allows to greatly reduce the lines of code to finally have a very purified script. The numerical solution of (2.2) is finally achieved by :

```
fespace Vh(Th,P1);
Vh Y,V,U=1; //finite element functions

macro grad(Y) [dx(Y),dy(Y)] // //macro ended by //

solve Poisson(Y,V) = int2d(Th)(grad(Y)'*grad(V))
                   - int2d(Th)(U*V)
                   + on(1,2,3,4,5,6,Y=0); // y∈ H_0^1(Ω)
plot(Th,Y); // Figure 2.1
```

Code 2.2 – Poisson solution by solving the variational formulation

Instead of solving the variational form (2.2) in a single line of code, we can also define the matrices from the discretization of the bilinear form a and the linear form b in (2.3), and solve the equivalent linear system. We introduce the finite element subspace

$$Y_h = \{ v \in V_h, \;\; v_{|\partial\Omega} = 0 \}$$

including the homogeneous Dirichlet boundary condition stated in (2.1) and let us note its basis

$$\mathrm{Vect}(\phi_i)_{i\in\{1..n\}} \quad 0 < n < n_d.$$

The stiffness and mass matrices are respectively given by

$$(A_{h,ij})_{(i,j)\in\{1..n\}} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j \, dx, \qquad (M_{h,ij})_{(i,j)\in\{1..n_d\}} = \int_\Omega \phi_i\phi_j \, dx.$$

The solution of (2.3) discretized by finite elements $\mathbb{P}_1$ finally consists in solving the linear system

$$A_h y_h - M_h u_h = 0.$$

Since the mesh T_h and the finite element space V_h are generated as in Code 2.1, FreeFEM thus only requires the command **varf** to define the matrices A_h and M_h. It is not necessary to introduce the space of zero finite elements on the edge Y_h in the code. Indeed, FreeFEM handles the Dirichlet boundary conditions (homogeneous or inhomogeneous) by specifying them directly in the variational formulation.

```
Vh Y,V,U=1; // finite element functions

macro grad(Y) [dx(Y),dy(Y)] //
varf stiffness(Y,V) = int2d(Th)((grad(Y)'*grad(V)))
                    + on(1,2,3,4,5,6,Y=0); // y ∈ H_0^1(Ω)
varf mass(Y,V) = int2d(Th)(Y*V);

matrix Ah = stiffness(Vh,Vh,solver=sparsesolver); // with Dirichlet B.C.
matrix Mh = mass(Vh,Vh,solver=sparsesolver);
```

Code 2.3 – **varf** command

Finally, **solve** Poisson(Y,V) in Code 2.2 is equivalent to solve $A_h y_h = M_h u_h$ via

```
V[] = Mh*U[];
Y[] = Ah^-1*V[];
```

Code 2.4 – Poisson solution with finite element matrices

In both cases, it is not necessary to use the characteristics of the mesh in detail, which greatly facilitates the overall implementation. Although it is very close to writing the variational formulation, the **solve** command is most often not as fast as dealing directly with the matrices which are usually sparse. When only one PDE needs to be solved, the computational time savings may not be obvious, but when in addition an optimal control problem needs to be solved, the optimization process may require many calls to the state and adjoint PDE equations. There is therefore a real advantage to working with sparse matrices as soon as the number of PDEs to be solved increases. In the rest of Section 2.2, we will highlight the advantages of using sparse matrices for calculating derivatives (especially for Jacobian and Hessian matrices). Suppose that the equation (2.1) is a constraint, we now want to minimize the functional J, depending on both y and u,

$$J(y,u) = \frac{1}{2}\int_\Omega (y-y_d)^2 \, dx + \frac{\alpha}{2}\int_\Omega u^2 \, dx, \tag{2.4}$$

whose computation requires solving (2.1) in a first step. Moreover, we will see that computing the derivative of J with respect to control may require solving the state and adjoint PDEs arising from the necessary first-order optimality conditions, as explained in Section 2.1.2 below.

2.1.2 PDE constrained optimization

In differentiable optimization problems, numerical methods are generally based on first order optimality conditions and thus on the computation of derivatives. Let W be a Banach space, we denote its topological dual by $W' = \mathscr{L}(W,\mathbb{R})$ (space of continuous linear functionals on W) and by

$$\langle z, w\rangle_{W',W} = z(w) \quad \forall z \in W'$$

the dual pairing. Given a linear operator A, its adjoint is denoted by A^*. For the special case of the Hilbert space H, the dual H' can be directly identified with H and the dual pairing is equated to the inner product of H $(\cdot,\cdot)_H$. A differentiable optimization problem is generally written as

$$\min_{w\in\mathscr{U}_{ad}} J(w) \tag{2.5}$$

where $J : W \mapsto \mathbb{R}$ is Gateaux differentiable and $\mathscr{U}_{ad} \subset W$ is a nonempty, closed and convex subset. We denote by DJ the (Gateaux) derivative of J. First-order optimality conditions are then written in the following form.

Theorem 2.1. *Let $\bar{w} \in W$ be a (local) optimal solution of (2.5). Then*

$$\bar{w} \in \mathscr{U}_{ad}, \quad \langle DJ(\bar{w}), w - \bar{w}\rangle_{W',W} \geqslant 0, \qquad \forall w \in \mathscr{U}_{ad}. \tag{2.6}$$

Classical differentiable optimization strategies involve the computation of (at least) first-order derivatives. Indeed a Taylor expansion returns at some iterate point x_k

$$J(x) = J(x_k) + \langle DJ(x_k), x - x_k\rangle_{W',W} + o(\|x - x_k\|_W)$$

and the next iterate x_{k+1} is searched so that

$$f(x_{k+1}) \leqslant f(x_k)$$

and thus, assuming that x_{k+1} is close enough to x_k, so that

$$\langle DJ(x_k), x - x_k\rangle_{W',W} \leqslant 0.$$

In the case where $W = \mathbb{R}^N$, a particular descent direction is usually given by the opposite of the gradient $-\nabla J(x_k)$ (see Algorithm 1). However, the framework provided by (2.5) and the Theorem 2.1 is rather limited when one has to look at an optimal control problem because the objective function and its derivative are not easy to compute at once. Moreover, such an elementary algorithm will not perform well for large-scale optimization problems involving a PDE constraint. A new framework is therefore needed with the first constraint of decoupling state and control variables. Then, once the existence of solutions is acquired or assumed, the

Algorithm 1 Gradient descent algorithm

initialization x_0, stop criterion ϵ
while $\|\nabla f(x_k)\| \leqslant \epsilon$ **do**
  compute α_k with linear search methods (in the direction $-\nabla f(x_k)$)
  compute $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$
  compute $\nabla f(x_{k+1})$
end while

computation of the derivatives of the functions involved should be easy and we should be able to move easily to a powerful numerical framework. We therefore write below the general minimization problem of a function J depending on both *state* and *control* variables subject to a PDE embedded in the operator e with some additional constraints encoded in $\mathscr{U}_{ad}$:

$$\min_{(y,u)\in Y\times U} J(y,u) \quad \text{subject to} \quad e(y,u)=0, \quad u\in\mathscr{U}_{ad}. \tag{2.7}$$

Assumption 2.2 below guarantees existence of solutions of (2.7) usually denoted by $(\bar{y},\bar{u})$ in $Y\times U$.

Assumption 2.2.

1. *$\mathscr{U}_{ad}\subset U$ is nonempty, closed and convex.*
2. *The mappings*
$$J: Y\times U\mapsto\mathbb{R} \text{ and } e: Y\times U\mapsto Z$$
are continuous with Z a Banach space and Y, U reflexive Banach spaces.
3. *For every $u\in V$ in a neighborhood V of $\mathscr{U}_{ad}$, the state equation*
$$e(y,u)=0$$
has a unique solution $y(u)\in Y$ and the mapping
$$u\in\mathscr{U}_{ad}\mapsto y(u)\in Z$$
is continuous.
4. *The mapping*
$$(y,u)\in Y\times U\mapsto e(y,u)\in Z$$
is weakly continuous.
5. *J is sequentially lower semi continuous.*

Some additional state constraints can be added by means of a set $\mathscr{Y}_{ad}\subset Y$ assumed to be nonempty, convex and closed. We introduce the reduced cost function of the problem (2.7)

$$u\in\mathscr{U}_{ad}\mapsto\widehat{J}(u)=J(y(u),u)$$

so that (2.7) is reformulated as

$$\min_{u\in\mathscr{U}_{ad}}\widehat{J}(u).$$

If we want to use the algorithm 1 to compute an optimal solution, we have to compute the derivative of $widehatJ$. But this requires to obtain the derivative of the operator $u \in \mathscr{U}_{ad} \mapsto y(u) \in Y$, which is not explicit. The assumption 2.3 below provides a general framework that ensures the differentiability of the input-output mapping $u\mathscr{U}_{ad} \mapsto y(u) \in Y$ (by the implicit function theorem) and at the same time allows us to compute $D\widehat{J}$, which is necessary to write the first order optimality conditions.

Assumption 2.3.

1. *$\mathscr{U}_{ad} \subset U$ is nonempty, closed and convex.*
2. *The mappings*
$$J : Y \times U \mapsto \mathbb{R} \text{ and } e : Y \times U \mapsto Z$$
are continuously Fréchet differentiable and U, Y, Z are Banach spaces.
3. *For all $u \in V$ in a neighborhood V of $\mathscr{U}_{ad}$, the state equation*
$$e(y,u) = 0$$
has a unique solution $y(u) \in Y$.
4. *The partial derivative*
$$\partial_y e(y(u),u) \in \mathscr{L}(Y,Z)$$
has a bounded inverse for all $u \in V \subset \mathscr{U}_{ad}$.

Applying Theorem 2.1 to $u \in \mathscr{U}_{ad} \mapsto \widehat{J}(u)$, we get the following first-order optimality conditions in terms of the reduced cost function $\widehat{J}$.

Theorem 2.4. *Under Assumption 2.3, if $\widehat{u} \in \mathscr{U}_{ad}$ is a (local) optimal solution of the reduced problem then*
$$\left\langle D\widehat{J}(\widehat{u}), u - \widehat{u} \right\rangle_{U',U} \geqslant 0 \qquad \forall u \in \mathscr{U}_{ad}.$$

At this step, a direction of descent can be found by following the information provided by the continuous derivative $D\widehat{J}$. Unfortunately, the Theorem 2.4 does not provide an easy way to compute it numerically since, according to the sensitivity analysis developed below, the numerical computation of the derivative of the mapping $u \in \mathscr{U}_{ad} \mapsto y(u) \in Y$ requires the computation of "too many" directional derivatives.

Sensitivity approach

Indeed, let $s \in \mathscr{U}_{ad}$. We compute $\widehat{J}(u + \epsilon s)$ for ϵ small enough. In the setting of Assumption 2.3, the chain rule gives

$$\left\langle D\widehat{J}(u), s \right\rangle_{U',U} = \left\langle \partial_y J(y(u),u), Dy(u)s \right\rangle_{Y',Y} + \left\langle \partial_u J(y(u),u), s \right\rangle_{U',U}. \tag{2.8}$$

The partial derivatives $\partial_y J$ and $\partial_u J$ are easy to compute since J explicitly depends on y and u. In contrast, computing $Dy(u)s$ is related to solving $e(y,u) = 0$ and is not immediate. The state constraint
$$e(y(u),u) = 0$$

is derivated in the direction s to make appear $\delta y_s = Dy(u)s$ as the solution of the new linear PDE

$$\partial_y e(y(u),u)\delta y_s = -\partial_u e(y(u),u)s. \tag{2.9}$$

The calculation of

$$\langle D\widehat{J}(u), s\rangle_{U',U}$$

thus requires to compute the solution δy_s of (2.9) for each direction s. Thus, the differential $D\widehat{J}(u)$ is numerically difficult to access if U has a large dimension since it is necessary to compute the directional derivative in each direction of a given basis of the vector space spanned by U. A numerical method based on this sensitivity approach is therefore not feasible in high dimension because it would be too computationally demanding. The same problem is encountered for automatic differentiation in direct mode (as explained in Section 2.1.5), where the Jacobian is not necessarily needed because we generally need a descent direction which is given by the Jacobian applied in a well-chosen direction. On the same principle as automatic differentiation in reverse mode (see [31]), the gradient can be found with much less effort by introducing an adjoint variable.

Adjoint approach

Equation (2.8) is equivalently written as

$$\left\langle D\widehat{J}(u), s\right\rangle_{U',U} = \left\langle Dy(u)^*\partial_y J(y(u),u), s\right\rangle_{U',U} + \left\langle \partial_u J(y(u),u), s\right\rangle_{U',U}$$

and thus

$$D\widehat{J}(u) = Dy(u)^*\partial_y J(y(u),u) + \partial_u J(y(u),u).$$

Moreover, (2.9) gives

$$\partial_y e(y(u),u)Dy(u) = -\partial_u e(y(u),u). \tag{2.10}$$

It is not required to know the whole matrix $Dy(u)$ but only the vector

$$Dy(u)^*\partial_y J(y(u),u).$$

Item *4* of Assumption 2.3 ensures the existence of the inverse

$$-\partial_y e(y(u),u)^{-1}$$

and (2.10) gives

$$Dy(u) = -\partial_y e(y(u),u)^{-1}\partial_u e(y(u),u)$$

and

$$\begin{aligned} Dy(u)^*\partial_y J(y(u),u) &= \left(-\partial_y e(y(u),u)^{-1}\partial_u e(y(u),u)\right)^*\partial_y J(y(u),u) \\ &= -\partial_u e(y(u),u)^* \underbrace{\left(\partial_y e(y(u),u)^*\right)^{-1}\partial_y J(y(u),u)}_{\text{adjoint } -p(u)}. \end{aligned}$$

The adjoint vector $p = p(u) \in Z'$ is thus defined as the solution of the linear equation

$$\partial_y e(y(u), u)^* p = -\partial_y J(y(u), u). \tag{2.11}$$

Finally,

$$D\widehat{J}(u) = \partial_u e(y(u), u)^* p(u) + \partial_u J(y(u), u).$$

From a numerical point of view, compared to the sensitivity approach which thus requires solving as many PDEs as U has degrees of freedom to express the derivative of $\widehat{J}$, the adjoint approach only requires solving the equation of state in (2.7) and the adjoint equation (2.11). This brings a significant advantage in that the resolution of the PDEs requires more time and more computation when the mesh is finer. Thus, the calculation of the first derivative of the objective function of the problem (2.7) at a point $u \in \mathcal{U}_{ad}$ follows the following steps :

S.1 Compute the partial derivatives

$$\partial_y J(y, u), \quad \partial_u J(y, u), \quad \partial_y e(y, u), \quad \partial_u e(y, u),$$

and the adjoint operators

$$\partial_y e(y, u)^*, \quad \partial_u e(y, u)^*.$$

S.2 Solve the state equation

$$e(y, u) = 0,$$

which gives $y(u)$ (input-output mapping).

S.3 Solve the adjoint equation

$$\partial_y e(y(u), u)^* p = -\partial_y J(y(u), u),$$

which gives the adjoint $p = p(u)$.

S.4 Finally, compute

$$D\widehat{J}(u) = \partial_u e(y(u), u)^* p(u) + \partial_u J(y(u), u).$$

The adjoint variable p can be interpreted as the Lagrange multiplier corresponding to the constraint $e(y, u) = 0$. The Lagrangian $L : Y \times U \times Z' \mapsto \mathbb{R}$ of the problem (2.7) is defined by

$$L(y, u, p) = J(y, u) + \langle p, e(y, u) \rangle_{Z',Z}.$$

The partial derivatives of the Lagrangian with respect to the adjoint variable p and the state variable y give the equation of state in (2.7) and the adjoint equation (2.11), respectively, while the partial derivative with respect to the control variable u gives the derivative of the reduced cost function $\widehat{J}$. Under the assumption 2.3, the Theorem 2.4 is reformulated as follows. This is the Pontryagin maximum principle well known in the control theory of PDEs.

Corollary 2.5. *Under Assumption 2.3, let $(\bar{y},\bar{u}) \in Y \times U$ be an optimal solution of (2.7). Then there exists $\bar{p} \in Z'$ such that*

$$e(\bar{y},\bar{u}) = 0, \tag{2.12}$$

$$\partial_y e(\bar{y},\bar{u})^* \bar{p} = -\partial_y J(\bar{y},\bar{u}), \tag{2.13}$$

$$\left\langle \partial_u J(\bar{y},\bar{u}) + \partial_u e(\bar{y},\bar{u})^* \bar{p}, u-\bar{u} \right\rangle_{U',U} \geqslant 0 \qquad \forall u \in \mathcal{U}_{ad}, \bar{u} \in \mathcal{U}_{ad}. \tag{2.14}$$

The Lagrangian formulation of the optimality conditions is

$$\langle q, \partial_p L(\bar{y},\bar{u},\bar{p}) \rangle_{Z',Z} = 0 \qquad \forall q \in Z', \tag{2.15}$$

$$\langle \partial_y L(\bar{y},\bar{u},\bar{p}), v \rangle_{Y',Y} = 0 \qquad \forall v \in Y, \tag{2.16}$$

$$\left\langle \partial_u L(\bar{y},\bar{u},\bar{p}), u-\bar{u} \right\rangle_{U',U} \geqslant 0 \qquad \forall u \in \mathcal{U}_{ad}, \bar{u} \in \mathcal{U}_{ad}. \tag{2.17}$$

The Corollary 2.5 will therefore be used here most often to find the adjoint equation and the derivative of $\widehat{J}$. Note that this requires to identify precisely the spaces involved Y, U and Z and the dual pairings must be chosen accordingly. When U is a Hilbert space, $D\widehat{J}$ can be identified with the gradient $\nabla \widehat{J}(u)$ corresponding to the chosen inner product of U.

Remark 2.6. *The Corollary 2.5 provides necessary first-order optimality conditions for an optimization problem constrained by a PDE. Such conditions are known to be sufficient when the problem (2.7) is convex. Otherwise, the sufficient conditions given by the second order optimality conditions (and the Hessian) are necessary to characterize the locally optimal solutions. However, from a numerical point of view, the numerical computation of the Hessian of the Lagrangian can be a heavy operation (the matrix is not necessarily sparse) and is commonly replaced by an approximated matrix (BFGS and quasi-Newton method for instance).*

Remark 2.7. *The introduction of the adjoint is an efficient way to calculate the derivative of the objective function. The additional control constraints included in the $\mathcal{U}_{ad}$ set can also be handled. However, it is much more difficult to account for potential additional state constraints included in $\mathcal{Y}_{ad}$ in that it may require modifying the adjoint equation to account for these constraints into an equation that is more difficult to solve. In this case, the adjoint method may not be advisable. We will give some alternatives in Chapter 2.*

Poisson example

Let Ω be an open subset of $\mathbb{R}^N$ with Lipschitz boundary. We have defined in Section 2.1.1 the problem of minimizing the cost function (2.4) subject to a Poisson PDE with homogeneous Dirichlet boundary condition (2.1), that we reformulate in the framework of (2.7) by introducing the sets $Y = H_0^1(\Omega)$ and $U = L^2(\Omega)$. For the moment, we do not consider any additional control constraints included in the set $\mathcal{U}_{ad} \subset U$. We thus denote the cost function and PDE constraints by the functions J and e in a weak form :

$$J : (y,u) \in Y \times U \mapsto \frac{1}{2}\int_\Omega (y-y_d)^2\,dx + \frac{\alpha}{2}\int_\Omega u^2\,dx \in \mathbb{R}$$

$$e : (y,u) \in Y \times U \mapsto a(y,\cdot) - b(u,\cdot) \in Z$$

where a and b are the bilinear forms defined by (2.3). Considering the Gelfand triple (see [60, Definition 1.26])

$$H_0^1(\Omega) \subset L^2(\Omega) \subset H^{-1}(\Omega),$$

$H^{-1}(\Omega)$ is identified with $H_0^1(\Omega)$ so that the dual pairing $\langle\cdot,\cdot\rangle_{Y',Y}$ is compatible with the $L^2(\Omega)$-inner product. Since $U = L^2(\Omega)$, the dual pairing $\langle\cdot,\cdot\rangle_{U',U}$ is the $L^2(\Omega)$-inner product. Finally, we set $Z = H^{-1}(\Omega) = Y'$ so that the dual pairing $\langle\cdot,\cdot\rangle_{Z',Z}$ is compatible with the $L^2(\Omega)$-inner product. Assumption 2.3 is verified, indeed Items *1.* and *2.* are straightforward while Items *3.* and *4.* are due to properties of elliptic operators stated in [31]. For $p \in Z' = H_0^1(\Omega)$, the Lagrangian is thus given by

$$L(y,u,p) = \int_\Omega \left(\frac{1}{2}(y-y_d)^2 + \frac{\alpha}{2}u^2 + \nabla y \cdot \nabla p - up\right) dx.$$

Remark 2.8. *The identification of the dual pairings involved by the $L^2(\Omega)$-inner product gives a Lagrangian that we can later manipulate easily. Nevertheless, this compatibility depends on the sets Y, Z and U chosen to verify the 2.3 hypothesis and is not always straightforward.*

We finally apply the Corollary 2.5 to express the weak formulation of the adjoint equation and exhibit the variational inequality which gives the first derivative of the reduced cost function $\widehat{J}$ which we then identify with the gradient associated with the $L^2(\Omega)$-inner product.

$$\begin{aligned}
&\partial_p L(y,u,p) = 0 \iff \int_\Omega \left(\nabla y \cdot \nabla v - uv\right) dx = 0 \qquad \forall v \in H_0^1(\Omega),\\
&\partial_y L(y,u,p) = 0 \iff \int_\Omega \left(\nabla p \cdot \nabla v + (y-y_d)v\right) dx = 0 \qquad \forall v \in H_0^1(\Omega),\\
&\partial_u L(y,u,p) : v \in L^2(\Omega) \mapsto \int_\Omega (\alpha u - p)v\, dx,
\end{aligned}$$

i.e., in the strong form for the $L^2(\Omega)$-inner product,

$$\begin{aligned}
&y \in H_0^1(\Omega) \text{ solution of } -\Delta y = u \qquad \text{in } \Omega,\\
&p \in H_0^1(\Omega) \text{ solution of } \Delta p = y - y_d \qquad \text{in } \Omega,\\
&D\widehat{J}(u) \text{ identified with } \nabla \widehat{J}(u) = \alpha u - p.
\end{aligned}$$

Dirichlet boundary control example

We focus here on a boundary control problem. We further assume that Ω has either a $\mathscr{C}^2$ boundary or is a convex polytope. In the previous example, the variational formulation is written by introducing the set $Y = H_0^1(\Omega)$ for a homogeneous Dirichlet boundary condition. For Neumann or Robin boundary conditions, we take $Y = H^1(\Omega)$ instead. For inhomogeneous Dirichlet boundary conditions, the standard variational formulation must be reformulated using some alternatives.

Given $f \in L^2(\Omega)$, we modify the previous example by adding a Dirichlet boundary condition $u \in L^2(\partial\Omega)$ so that the new problem is as follows

$$\min J(y,u) = \frac{1}{2}\int_\Omega \left(y(x) - y_d(x)\right)^2 dx + \frac{\alpha}{2}\int_{\partial\Omega} u(x)^2\, dx \tag{2.18}$$

$$\text{subject to } \begin{cases} -\Delta y = f & \text{in } \Omega,\\ y = u & \text{in } \partial\Omega. \end{cases} \tag{2.19}$$

We cannot write directly the weak formulation as in the Poisson example. To overcome this difficulty, one possibility may be to first introduce a small parameter δ so that the Dirichlet boundary condition in (2.19) becomes a Robin boundary condition

$$\delta\partial_n y + y = u \text{ on } \partial\Omega$$

and to write the variational formulation by introducing the space $Y = H^1(\Omega)$. Here, we consider instead the way FreeFEM handles the Dirichlet boundary conditions. Indeed, following [95, Section 10.6], we denote by

$$A_0 : \mathscr{D}(A_0) \mapsto L^2(\Omega)$$

the Dirichlet Laplacian (we have $\mathscr{D}(A_0) = H_0^1(\Omega) \cap H^2(\Omega)$ because of the assumption on Ω) and γ_0, γ_1 respectively the *Dirichlet* and *Neumann* traces. We introduce $\mathbf{D}$ the Dirichlet map such that for any $u \in L^2(\partial\Omega)$ we can find $\mathbf{D}u \in L^2(\Omega)$ so that

$$\Delta \mathbf{D}u = 0 \text{ on } \Omega \text{ and } \gamma_0(\mathbf{D}u) = \mathbf{D}u_{|\partial\Omega} = u$$

(actually, $\mathbf{D}u \in \mathscr{C}^\infty(\Omega)$ and the operator $\mathbf{D}$ is bounded from $L^2(\partial\Omega)$ to $L^2(\Omega)$). Moreover, the adjoint operator of $\mathbf{D}$ is

$$\mathbf{D}^* = -\gamma_1 A_0^{-1}$$

and for all $v \in L^2(\Omega)$

$$(\mathbf{Du}, v)_{L^2(\Omega)} = -(u, \partial_n \phi)_{L^2(\partial\Omega)} \text{ with } A_0\phi = v.$$

We thus seek the solution $y(u)$ of (2.19) in the affine space $H_0^1(\Omega) + \mathbf{D}u$ and we define $z \in H_0^1(\Omega) \cap H^2(\Omega)$ the solution of

$$\begin{aligned} -\Delta z &= f \quad \text{in } \Omega \\ z &= 0 \quad \text{in } \partial\Omega, \end{aligned}$$

so that $y = z + \mathbf{D}u$ with $z = A_0^{-1} f$, whose variational formulation is the following : find $z \in H_0^1(\Omega)$ such that

$$\int_\Omega \nabla z \cdot \nabla v = \int_\Omega f v \, dx \quad \forall v \in H_0^1(\Omega).$$

The state space Y has to be defined in order to find a unique weak solution $y \in Y$ of (2.19) when solving the resulting *very weak* variational formulation

$$\int_\Omega -y\Delta v \, dx = \int_\Omega f v \, dx - \int_\Omega \mathbf{D}u \Delta v \, dx \quad \forall v \in H_0^1(\Omega) \cap H^2(\Omega).$$

Since the Dirichlet Laplacian A_0 induces an isomorphism from $H_0^1(\Omega) \cap H^2(\Omega)$ to $L^2(\Omega)$ and A_0^{-1} is also selfadjoint in $L^2(\Omega)$, one can take $\phi \in L^2(\Omega)$ such that

$$v = A_0^{-1}\phi$$

and the previous variational formulation is equivalent to :

$$\text{find } y \in Y \text{ s.t. } \int_\Omega y\phi \, dx = \int_\Omega \left(A_0^{-1} f\right)\phi \, dx + \int_\Omega \mathbf{D}u\phi \, dx \qquad \forall \phi \in L^2(\Omega).$$

Therefore, we set $Y = L^2(\Omega)$, $U = L^2(\Gamma)$ and $Z = L^2(\Omega)$ so that

$$y = z + \mathbf{D}u \in L^2(\Omega)$$

is the unique solution of (2.19) (uniqueness of solution is straightforward by putting $(f,u) = 0$). Then we define the operator

$$e(y,u) = y - A_0^{-1} f - \mathbf{D}u$$

so that Assumption 2.3 is satisfied in that setting and optimality conditions yield the existence of $\phi \in L^2(\Omega)$ such that

$$\begin{aligned} \phi &= y_d - y \\ \left(\mathbf{D}^*\phi, v-u\right)_{L^2(\partial\Omega)} &\geqslant 0 \quad \forall v \in U. \end{aligned}$$

Introducing $p \in H_0^1(\Omega) \cap H^2(\Omega)$ solution of $p = A_0^{-1}\phi$, the adjoint equation now reads

$$\begin{aligned} \Delta p &= y - y_d \quad \text{in } \Omega \\ p &= 0 \qquad\quad \text{in } \partial\Omega, \end{aligned}$$

so that $D\widehat{J}(u)$ is identified with the gradient

$$\nabla \widehat{J}(u) = \alpha u + \partial_n p$$

for the $L^2(\partial\Omega)$-inner product.

Remark 2.9. *If none of the assumptions made on Ω hold, we have to modify the trial space functions $H_0^1(\Omega) \cap H^2(\Omega)$ accordingly, since existence and uniqueness of the solution to the very weak variational formulation relies on the isomorphism $A_0 \in \mathscr{L}\left(H_0^1(\Omega) \cap H^2(\Omega), L^2(\Omega)\right)$. This issue is more generally treated in [95, Section 13].*

Remark 2.10. *From the numerical point of view, u is usually smooth enough to get that $y \in H^1(\Omega)$ so that we solve (2.19) in `FreeFEM` by searching $y \in \{w \in H^1(\Omega), w_{|\partial\Omega} = u\}$ verifying*

$$\int_\Omega \nabla y \cdot \nabla v \, dx = \int_\Omega f v \, dx \qquad \forall v \in H_0^1(\Omega).$$

This is numerically carried out by specifying the boundary conditions thanks to the command **`on`**`(IndexBoard, Y=U)`.

```
Vh Y,V; // finite element functions
macro grad(Y) [dx(Y),dy(Y)] //
solve Poisson(Y,V) = int2d(Th)((grad(Y)'*grad(V)))
                   + on(1,2,3,4,5,6,Y=U); // y∈{w∈H¹(Ω), w_|∂Ω = u}
```

We mention this inhomogeneous Dirichlet limit problem to highlight the difficulties that can arise with the choice of Y, Z and U spaces to theoretically find the adjoint and derivative. From the numerical point of view, although we generally do not need to make these sets explicit since we often assume that all the data involved are sufficiently smooth, a good understanding of the mathematical framework and in particular knowledge of the discretized spaces as well as the inner products is crucial in the calculations.

When we move to the numerical framework and have in mind the algorithm 1, the numerical calculation of the derivative of the objective function involves the adjoint variable that will be found by solving the adjoint equation according to a chosen scheme. Alternatively, one can directly override the adjoint equation and directly derive a discretized version of the optimal control problem. The two approaches are respectively called *First Optimize Then Discretize (FOTD)* and *First Discretize Then Optimize (FDTO)*.

2.1.3 Optimization and discretization strategies

Given a general optimal control problem, we have given in Section 2.1.2 ways to compute the continuous derivatives of the involved functions. This numerically implies to get a suitable approximation of both functions and their derivatives that allow their numerical computation. Effectiveness of the procedure is directly imputed to handling both discretization and optimization. Given discretization parameters $0 < h < h_0$ and some finite-dimensional discretization spaces families $(Y_h)_{0<h<h_0}$, $(U_h)_{0<h<h_0}$, $(Z_h)_{0<h<h_0}$ and $(\mathscr{U}_{ad}^h)_{0<h<h_0}$ of the spaces Y, U, Z and $\mathscr{U}_{ad}$, the optimal control problem (2.7) is discretized as

$$\min_{(y_h,u_h)\in Y_h\times U_h} J_h(y_h,u_h) \quad \text{subject to :} \quad e_h(y_h,u_h)=0, \quad u_h\in\mathscr{U}_{ad}^h. \tag{2.20}$$

where

$$J_h : Y_h\times U_h \mapsto \mathbb{R} \text{ and } e_h : Y_h\times U_h \mapsto Z$$

are discretized versions of the continuous functions. Without loss of generality, the finite-dimensional spaces Y_h and U_h are identified with $\mathbb{R}^N$, $N \geqslant 1$. The problem (2.20) is expressed in a finite-dimensional framework and is then numerically solved by means of usual tools (KKT condition, differentiable optimization algorithms) by computing derivatives J_h and e_h. This approach is usually called *First Discretize Then Optimize (FDTO)* or *direct* method. In contrast, in the *First Optimize Then Discretize (FTDO)* (or *indirect*) approach, the first-order optimality condition of the continuous problem (2.7) is first derived by applying Corollary 2.5 such that all functions sets and operators involved are then discretized accordingly

$$e_h(y_h,u_h)=0, \tag{2.21}$$

$$(\partial_y e_h(y_h,u_h))^* p_h = -\partial_y J_h(y_h,u_h), \tag{2.22}$$

$$\big(\partial_u J_h(y_h,u_h)+(\partial_u e_h(y_h,u_h))^* p_h, u-u_h\big)_{U_h} \geqslant 0, \quad u_h\in\mathscr{U}_{ad}^h, \forall\, u\in\mathscr{U}_{ad}^h. \tag{2.23}$$

Both methods may not be mathematically equivalent since the partial derivatives of the discretized functions J_h and e_h may differ from the discretizations of the partial derivatives $\partial_y e_h$, $\partial_u e_h$, $\partial_y J_h$, $\partial_u J_h$.

Note that the *FOTD* approach does not give the true numerical derivative of the discretized function (rather obtained with the *FDTO* approach). This may affect the algorithm's convergence. To our knowledge, the question of the convergence of optimal solutions of the discretized problem to the solution of the continuous one is a challenging issue that deserves further consideration. In [87], under some appropriate assumptions on optimal control problems in finite dimension, it is proved that direct *FDTO* and indirect *FOTD* approaches are mathematically equivalent when the discretization is performed with a *symplectic partitioned Runge-Kutta integrator*. This reference also contains interesting issues related to automatic differentiation, that we illustrate in Appendix 2.B. Indeed, in this exemple, when the state equation is

discretized according to an implicit scheme, automatic differentiation in reverse mode hides an explicit scheme for the adjoint equation. When optimization is made first, we have to pay a special attention to the choices of discretization for state and adjoint variables. Nevertheless the freedom of discretization's choice for the adjoint variable sometimes implies that *FOTD* is more relevant since the derivative makes appear the adjoint which is, without state constraints, usually more regular than the state variable. The adjoint can thus be approximated more accurately thanks to a well-fitting discretization. Conversely, the main advantages of *FDTO* approach is its ability to allow the use of large-scale optimization methods, state constraints and automatic differentiation.

Remark 2.11. *In this remark, we mention a general rigorous mathematical framework for discretizations. Under Assumption 2.3, let Y, U and Z be separable Banach spaces. Let $\tilde{Y}$ be a separable Banach space such that the embedding $Y \hookrightarrow \tilde{Y}$ is continous. Given a discretization parameter h and a family of finite-dimensional spaces $(Y_h)_{0<h<h_0}$ assumed to be uniformly continuously embedded $Y_h \hookrightarrow \tilde{Y}$, we assume the existence of projection and injection operators*

$$P_h^Y : \tilde{Y} \mapsto Y_h \text{ and } \tilde{P}_h^Y : Y_h \mapsto \tilde{Y}$$

such that

$$P_h^Y \tilde{P}_h^Y = id_{Y_h}.$$

The numerical scheme is furthermore assumed to be convergent, i.e.,

$$\lim_{h \mapsto 0} \|\tilde{P}_h^Y P_h^Y y - y\|_{\tilde{Y}} \mapsto 0 \qquad \forall y \in Y.$$

Note that $\|\tilde{P}_h^Y\|_{\mathscr{L}(Y_h,\tilde{Y})} = 1$ and, by the Uniform Boundedness Principle, that $\|P_h^Y\|_{\mathscr{L}(\tilde{Y},Y_h)} \leqslant \mathrm{Cst}$ (uniform constant). The same setting is established for U and Z'. The mappings J and e involved in (2.7) are approximated by

$$J_h : (y_h, u_h) \in Y_h \times U_h \mapsto J(\tilde{P}_h^Y y_h, \tilde{P}_h^U u_h) \in \mathbb{R}$$

and

$$e_h : (y_h, u_h) \in Y_h \times U_h \mapsto e(\tilde{P}_h^Y y_h, \tilde{P}_h^U u_h) \in Z.$$

On the other side, according to the FOTD *approach, the functions involved are expressed as*

$$\partial_\# e^h(y_h, u_h) = (\tilde{P}_h^{Z'})^* \partial_\# e(\tilde{P}_h^Y y_h, \tilde{P}_h^U u_h) \tilde{P}_h^\star$$

and

$$\partial_\# J^h = \partial_\# J(\tilde{P}_h^Y y_h, \tilde{P}_h^U u_h) \tilde{P}_h^\star$$

where $(\#, \star) = \{(y, Y), (u, U)\}$. Such a framework encompasses most of the usual discretization strategies (finite differences, finite elements, Galerkin). The particular case of a linear PDE is addressed in [5].

For the Poisson example (2.1) and having in mind its variational formulation (2.2), we can take

$$Y = \tilde{Y} = H_0^1(\Omega), \quad U = \tilde{U} = L^2(\Omega), \quad Z' = \tilde{Z}' = H_0^1(\Omega)$$

so that $Y \hookrightarrow U \hookrightarrow Z$. The finite element space U_h consists of $\mathbb{P}_1$ Lagrange elements, Y_h is the subspace of U_h such that we have 0 on the boundary and $(Z_h)' = Y_h$. Injection operators are thus induced by the several canonical injections above and projections via the $L^2(\Omega)$-inner product.

A general framework for conformal transformations and especially Galerkin discretization methods in the context of PDE optimization is stated in [64; 65].

2.1.4 The optimization routine IpOpt

IpOpt (for **I**nterior **P**oint **Opt**imizer) is an open source software package for large-scale differentiable optimization problems. First developed in Fortran, the C++ version allows to IpOpt to be more easily interfaced with Matlab, Python, R, Julia, etc. IpOpt is already included (with most of available linear solvers and options) when installing FreeFEM. We just have to call the library by putting in the head of the code the command : **load** *"ff-Ipopt"*. We previously highlighted the ability of FreeFEM to construct and manage sparse matrices. This is crucial for the optimization processing effectiveness since IpOpt gathers many of the most powerful linear solvers (MUMPS, Pardiso, WSMP, HSL routines). The last and significant asset of IpOpt is its large panel of options (choice of linear solver, multipliers updating, adjustment of line search, BFGS or Newton method, etc). A full description of the interior point method can be found in [97] and options are available at [4, Ipopt Options Tab]. IpOpt is designed to find *local* solutions of mathematical optimization problems of the form

$$\min_{x\in\mathbb{R}^n} \quad f(x) \tag{2.24}$$

$$\text{s.t.} \begin{cases} g_L \leqslant g(x) \leqslant g_U \\ x_L \leqslant x \leqslant x_U \end{cases} \tag{2.25}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is the objective function and $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ stands for equality and inequality constraints. Here, g_L, g_U and x_L, x_U respectively refer to the lower and upper bounds of constraints and variables. The functions f and g can be nonlinear and nonconvex, but are at least assumed to be twice differentiable. With this in mind, we aim to show how to transcribe a PDE optimization problem like (2.30,2.31) to a finite-dimensional optimization problem like (2.24,2.25). Once this transcription has been made, the most usual way to call IpOpt is :

```
IPOPT(f,df,d2f,C,jacC,x0,ub=xub,lb=xlb,cub=CUB,clb=CLB,optfile="ipopt.opt");
```

Code 2.5 – Calling IpOpt in FreeFEM

where df and d2f are respectively the gradient (an array) and the Hessian (a matrix in the Triplet Format, hopefully highly sparse) of the objective function ; C includes the constraints g and jacC its Jacobian (a matrix in the Triplet Format, hopefully highly sparse too) ; ub and lb are respectively the upper and lower bounds for x, cub and clb are the upper and lower bounds of the constraints g, and x0 is an initialization point ; optfile="*ipopt.opt*" incorporates all options (maximum number of iteration, convergence tolerance threshold, choice of linear solver and so on). Difficulties for the computation of the Hessian d2f usually happen (too much memory greedy, needs the inverse of a matrix or slowing down too much the code), which is not disabling. IpOpt offers many options, we can bypass the problem by choosing a quasi-Newton method, which is the default option if IpOpt is called without specifying the Hessian.

```
IPOPT(f,df,C,jacC,x0,ub=xub,lb=xlb,cub=CUB,clb=CLB); // no Hessian ↦ BFGS
```

Despite its potential more expensive cost computation, the Newton method is usually converging to an optimal solution with less iterations than a quasi-Newton method. Nevertheless the latter sometimes offers more flexibility. We advise at least to specify the cost function f, the constraints C and their derivatives df and jacC, knowing that the more arguments we give, the more efficient IpOpt is expected to be. We refer the reader to [97] to understand the particularities and how IpOpt works.

We mention two significant points : first, all `IpOpt` options are callable from `FreeFEM`. Second, the data to be given to `IpOpt` (`f`, `df`, `d2f`, `C`, `jacC`) need to respect a precise type : `df`,`C`,`x0`,`xub`,`xlb`,`CUB`,`CLB` have to be arrays and the matrices `d2f` and `jacC` have to be expressed in the Triplet Format for sparse matrices.

As an example, consider the problem in $\mathbb{R}^2$

$$\min_{x\in K} f(x) = x_1x_2(1-x_1-x_2), \quad \text{with } K = \left\{(x_1,x_2)\in\mathbb{R}^2 | x_1,x_2 \geqslant 0, x_1+x_2 \leqslant 1\right\}$$

which is formulated in the template (2.24,2.25) with

$$\min_{x_1,x_2} f(x) = x_1x_2(1-x_1-x_2)$$
$$\begin{cases} g_L \leqslant g(x) = x_1+x_2 \leqslant g_U \\ x_L \leqslant x \leqslant x_U \end{cases}$$

with $x_L = \begin{pmatrix}0\\0\end{pmatrix}$, x_U large enough, $g_U = 1$ and g_L small enough. Derivatives are

$$\nabla f(x) = \begin{pmatrix} x_2(1-2x_1-x_2) \\ x_1(1-x_1-2x_2)\end{pmatrix},\ \nabla g(x) = \begin{pmatrix}1 & 1\end{pmatrix},\ \nabla^2 f(x) = \begin{pmatrix} -2x_2 & 1-2x_1-2x_2 \\ 1-2x_1-2x_2 & -2x_1\end{pmatrix}.$$

Therefore, the cost function is :

```
func real f(real[int] &X) //returns a real
{
    return X[0]*X[1]*(1-X[0]-X[1]);
}
```

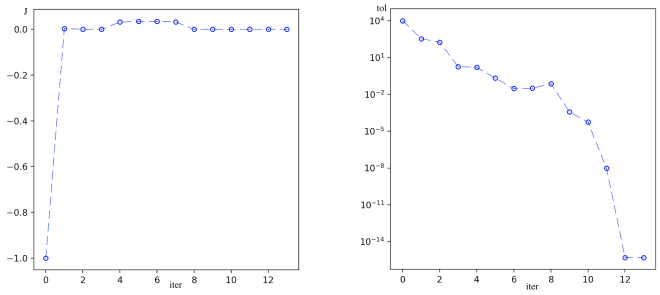
and its gradient and Hessian are :

```
func real[int] df(real[int] &X) // returns an array
{
    real[int] dJ(X.n); // size of X
    dJ = [ X[1]*(1-2*X[0]-X[1]), X[0]*(1-2*X[1]-X[0])];
    return dJ;
}
matrix hess; // matrix has to be declared outside
func matrix d2f(real[int] &X) // returns a matrix
{
    hess = [ [ -2*X[1] , 1-2*X[0]-2*X[1] ],
             [ 1-2*X[0]-2*X[1] , -2*X[0] ] ];
    return hess;
}
```

On the other side, the constraint function g is :

```
func real[int] C(real[int] &X) // returns an array
{
    real[int] cont(1); // array of size 1
    cont[0] = X[0]+X[1];
    return cont;
}
```

If only `X[0]+X[1]` is returned instead of the 1-size array containing this value, `IpOpt` will return an error. Like the Hessian of f, the Jacobian of g is :



(a) (b)

FIGURE 2.2 – Convergence curves : (a) objective function; (b) convergence criterion

```
matrix dc; // to be declared outside
func matrix jacC(real[int] &X) // returns a matrix
{
    dc = [[1,1]]; // double array of size (1,2)
    return dc;
}
```

We finally have to declare lower and upper bounds, an initialization point and then call `IpOpt` :

```
real[int] start = [1,1]; // Initialization

real[int] Xub = [10000,10000]; // free x_U
real[int] Xlb = [0,0]; // x_L

real[int] Cub = [0]; // g_U with array type
real[int] Clb = [-10000]; // free g_L

IPOPT(f,df,d2f,C,jacC,start,ub=Xub,lb=Xlb,clb=Clb,cub=Cub); // no optfile
cout << "(x,y) = " << "(" << start[0] << "," << start[1] << ")" << endl;
```

which returns the optimal solution $x = (0.5, 0.5)$ in less than 10 iterations. We plot convergence curves in Figure 2.2 for a desired convergence tolerance equal to 10^{-15}. The output of `IpOpt` is included in the `start` variables and returns the optimal solution if it is found and the Lagrange multipliers too. This example can utterly be used as a template for other problems by adapting the formulation inside the functions (`f`, `df`, `d2f`, `C`, `jacC`). If `IpOpt` meets any difficulty to converge, or if it ends to a locally infeasible point, adjusting the starting point may sometimes help. In some cases, difficulties may occur when `IpOpt` requires to compute function derivatives, for instance for nonlinear PDEs or when state constraints are additionally given. An alternative is to compute derivatives by automatic differentiation. We may also implement finite differences but this requires a very accurate approximation that may induce a crippling computation time.

2.1.5 Automatic differentiation

The leading principle of automatic differentiation is that a function $f : \mathbb{R}^p \mapsto \mathbb{R}^q$ given by a sequence of elementary numerical functions whose derivatives are already known can be differentiated by differentiating each line of the code step by step. This allows the chain rule to

be used directly in the code and thus to compute the true derivative of the numerical function. A standard framework consists in considering a function $f : \mathbb{R}^n \to \mathbb{R}$ thus mentioned in [17; 46] and a succession of calculation steps

$$(\phi_k, I_k)_{k\in\{0..N\}} \in \mathscr{E}_{k-1}$$

with $\mathscr{E}_{k-1} = \mathscr{C} \cup \mathscr{U}_k \cup \mathscr{B}_k$ where

$$\mathscr{C} = \mathbb{R} \times \{\emptyset\}, \quad \mathscr{U}_k = U \times \{-n \cdots k\}, \quad \mathscr{B}_k = B \times \{-n \cdots k\}^2.$$

Here, U is a set whose elements are functions already known and implemented in the language and whose derivatives are still in the set U. Thus, each step of the computation introduces a new variable and calls a known function of U according to a previous variable or a binary operation between two previous variables. Most of the time, this only allows classical analytic functions that we know how to derive analytically and whose derivatives are indeed combinations of elements of U (cos and sin for example). The set B gathers most of the binary operations $\{+, -, \times, \div\} \subset B$. We now suppose that we can find a finite sequence of operations

$$(\phi_k, I_k)_{k\in\{0..N\}}$$

such that all intermediate iterates $(x_{-n}, ..., x_{-1}) \in \mathbb{R}^n$ lead to the final iterate

$$x_N = f(x_{-n}, ..., x_{-1}).$$

The automatic differentiation will use the chain rule and the knowledge of the derivatives of the functions involved in ϕ_k to compute the final gradient. Two methods can be distinguished. Either the chain rule is applied according to the calls of functions in series line by line from the first line, it is the direct method, or from the last line, it is the reverse method. The first one calculates the Jacobian of the function (so a matrix) while the last one gives the Jacobian applied in a given direction (so an array). The direct method gives directly the Jacobian matrix but is expensive because it is necessary to calculate all the intermediate derivatives which are not necessarily needed for the final direction.

Direct mode

We give in the following lines a way to compute the function

$$f : (u, v) \mapsto \left((u+v)^2 + u\cos(u)\sin(v)\right)^2$$

as well as its derivative using automatic differentiation in direct mode

```
def f(u,v):
    x = u*cos(u)*sin(v)
    y = (u+v)**2 + 1
    return (x+y)**2
```

```
def df(u,v,du,dv):
    dx = u*sin(u)*du*sin(v)+cos(u)*du*sin(v)
        + u*sin(u)*cos(v)*dv
    x = u*cos(u)*sin(v)
    dy = 2*du*(u+v) + 2*dv*(u+v)
    y = (u+v)**2 + 1
    return 2*(x+y)*dx + 2*(x+y)*dy
```

Code 2.6 – Automatic differentiation in direct mode

This example shows the complete computation of the matrix 1×2 which is necessary to compute the derivative in a given direction (du, dv). To obtain the derivative, we must then evaluate

the matrix obtained in each free direction (here $(1,0)$ and $(0,1)$). This makes a complete matrix to calculate and then evaluate several times. The size of the matrix is not prohibitive in this case but we can also imagine a function $f:\mathbb{R}^p \mapsto \mathbb{R}^q$ with (p,q) very large (which happens when discretizing the weak form of a PDE). We then have to compute a matrix $p \times q$ and evaluate it p times, which is now disabling. Fortunately, we do not usually need the full Jacobian, but only the Jacobian applied in a given direction. The reverse mode allows to compute the gradient without having to evaluate the Jacobian in each free direction. A parallel with the backpropagation method in machine learning is drawn in [18].

Reverse mode

In order to write the adjoint (or reverse) code of a function that we have written numerically thanks to a given evaluation sequence $(\phi_k, I_k)_{k\in\{0..N\}}$ and such that

$$x_N = f(x_{-n}, ..., x_{-1}),$$

we associate to all intermediate operations

$$x_i = \phi_i(x_{-n}, ..., x_{i-1})$$

a quantity $(\lambda_i)_{i\in n...N\}}$, whose update follows the following rule

$$\lambda_N = 1, \quad \forall i \in \{-n...N-1\} \quad \lambda_i = \sum_{k>i} \lambda_k \partial_i \phi_k \tag{2.26}$$

The successive quantities λ_i thus depend on all $\lambda_{k\in\{i+1...N\}}$ insofar as ϕ_k implies variables $x_{j\in\{i+1...N\}}$. In practice, if the numerical code is not too long, we can proceed manually and line by line starting from the end of the implementation of the cost function. We then derive the calculation

$$x_k = \phi_k(x_{-n}, ..., x_{k-1})$$

by the involved variables $x_{-n}, ..., x_{k-1}$ and update λ variables following the rule

$$\lambda_i \mathrel{+}= \lambda_k \partial_i \phi_k(x_{-n}, ..., x_{k-1}).$$

We highlight this method on the previous example of the direct mode. We must first rewrite f with a sequence of elementary operations and then apply (2.26). However, a more usual way of doing this is to differentiate each line of the code, from the last line to the first line, with respect to the variables involved and then update the corresponding adjoining variables.

```
1  def f(u,v):
2      x = u*cos(u)*sin(v)
3      y = (u+v)**2 + 1
4      cost = (x+y)**2
5      return cost
```

```
1  def df(u,v):
2      x = u*cos(u)*sin(v)  # adjoint l_x
3      y = (u+v)**2 + 1 # adjoint l_y
4      cost = (x+y)**2 # adjoint l_c
5
6      lc = 1,lx = 0,ly = 0 # initialization
7      lx += 2*(x+y)*lc # line 4: ∂_x
8      ly += 2*(x+y)*lc # line 4: ∂_y
9      lu += 2*(u+v)*ly # line 3: ∂_u
10     lv += 2*(u+v)*ly # line 3: ∂_v
11     lu +=cox(u)*sin(v)*lx
12         -u*sin(u)*sin(v)*lx # line 2: ∂_u
13     lv += u*cos(u)*cos(v)*lx # line 2: ∂_v
14     return lu,lv
```

Code 2.7 – Automatic differentiation in reverse mode

The reader can see that all operations

$$x_k = \phi_k(x_{-n}, \cdots, x_{k-1})$$

must be computed before the calculation of the quantities λ_i because they are required to update them. The price to pay is to write the code with a well-defined sequence of operations $(\phi_k, \mathscr{I}_k)$ and to save all the intermediate variables $(x_1, \cdots, x_{N-1})$, which can be memory heavy if the function f to be calculated uses a very large number of intermediate variables. It is therefore advisable to have a code that is "optimized". The generalization to a function with value in $(\mathbb{R}^q, q > 1)$ is done by permuting the quantities λ with q vectors fixed on the canonical basis.

Let us comment on the direct and reverse modes. The reverse mode is well suited when the function $f : \mathbb{R}^p \mapsto \mathbb{R}^q$ to differentiate takes values in $\mathbb{R}^q$ with q small and especially when $p \gg q$. On the contrary, the direct mode is preferred when we differentiate with respect to few variables ($\mathbb{R}^p$ with p small and moreover when $p \ll q$). When considering optimal control problems, the choice of discretization can be different for functions and derivatives (indeed, an implicit scheme for the state usually implies an explicit scheme for the adjoint). The automatic differentiation therefore dispenses with this choice because both the direct and reverse methods return the true numerical derivative of the implemented function, and is therefore in line with a purely numerical approach.

A last important point that we would like to address is the close relationship between the inverse mode and the introduction of the adjoint variable in optimal control (see [87]). Let us take the example of the classical problem of controlled predator-prey equations (see [89, Ex. 4.10]),

$$\min J_T(u,v) = \frac{1}{2}\int_0^T (x(t)-1)^2\, dt, \tag{2.27}$$

$$\text{subject to } \begin{cases} \dot{x} = x + y + u & x(0) = 1 \\ \dot{y} = x - y + v & y(0) = 1, \end{cases} \tag{2.28}$$

$$(u,v) \in \mathscr{U}_{ad} = \left\{ f \in L^\infty(0,T), \quad \forall t \in (0,T) \;\; -1 \leqslant f(t) \leqslant 1 \right\}^2, \tag{2.29}$$

we compute the derivative of the numerical implementation of the functional cost (the inverse mode is well suited since $J : \mathbb{R}^p \mapsto \mathbb{R}$ with p large enough) and show that there appears a dis-

cretization of the adjoint equation coming from the Pontryagin maximum principle in finite dimension. The gradient computed by means of automatic differentiation gives a discretized adjoint equation whose implementation is described in Appendix 2.B. Automatic differentiation also allows the calculation of the Hessian, but the resulting matrix is usually not sparse and the optimization process thsu becomes less efficient. Furthermore, automatic differentiation is difficult to implement with respect to mesh variations (hence, for general optimal shape design problems), for which the adjoint method will prove more appropriate.

Finally, in this chapter, we have provided most of the tools necessary for us to write a general PDE optimization problem in the form (2.7), to discretize it with well-chosen finite elements, and to numerically find an optimal solution using the interior point method `IpOpt`. In the following section, by taking several examples, we will show different ways to discretize an optimization problem governed by a partial differential equation.

2.2 Linear quadratic PDE constrained optimization

The first problem we address to illustrate the numerical solution of constrained PDE optimization is the minimization of a quadratic criterion subject to a linear elliptic equation. Some additional constraints are written in a convex set $\mathscr{U}_{ad}$. The weak formulation of the PDE is introduced in order to have a well adapted framework for its numerical solution as well as the writing of the derivatives and to design an appropriate numerical strategy.

Let Ω be a bounded open Lipschitz domain, let y_d be $L^2(\Omega)$ and let u_a, u_b be $L^\infty(\Omega)$. We look for an optimal solution $u \in L^2(\Omega)$ of

$$\min J(y,u) = \frac{1}{2}\int_\Omega \big(y(x)-y_d(x)\big)^2\,dx + \frac{\alpha}{2}\int_\Omega u(x)^2\,dx \tag{2.30}$$

$$\text{subject to } \begin{cases} -\nabla\cdot(a\nabla y) = u & \text{in } \Omega, \\ y = 0 & \text{in } \partial\Omega. \end{cases} \tag{2.31}$$

$$\text{and } u_a \leqslant u \leqslant u_b. \tag{2.32}$$

To formulate (2.30,2.31) in the form of (2.7), the set of admissible controls is defined by

$$\mathscr{U}_{ad} = \{u \in L^2(\Omega), u_a \leqslant u \leqslant u_b\} \subset U = L^2(\Omega).$$

The variational formulation of (2.31) consists in finding $y \in H_0^1(\Omega)$ solution of

$$\int_\Omega a\nabla y\cdot\nabla v\,dx - \int_\Omega uv\,dx = 0 \qquad \forall v \in H_0^1(\Omega). \tag{2.33}$$

We take $Y = H_0^1(\Omega)$. The Lax-Milgram Theorem implies that, for any $u \in \mathscr{U}_{ad}$, there is a unique solution $y \in H_0^1(\Omega)$ of (2.33). Since $U = L^2(\Omega)$, the dual pairing is

$$\langle\cdot,\cdot\rangle_{U',U} = (\cdot,\cdot)_U.$$

Defining the operators

$$A \in \mathscr{L}\big(H_0^1(\Omega), H^{-1}(\Omega)\big) \quad \text{s.t.} \quad Ay : v \in H_0^1(\Omega) \mapsto \int_\Omega a\nabla y\cdot\nabla v\,dx,$$

$$B \in \mathscr{L}\big(L^2(\Omega)\big) \quad \text{s.t.} \quad Bu : v \in H_0^1(\Omega) \mapsto \int_\Omega uv\,dx,$$

the set Z has to be defined so that the operator

$$e : (y,u) \in H_0^1(\Omega) \times L^2(\Omega) \mapsto Ay - Bu \in Z$$

satisfies Assumption 2.3. The Gelfand triple

$$H_0^1(\Omega) \hookrightarrow L^2(\Omega) = L^2(\Omega)' \hookrightarrow H^{-1}(\Omega)$$

leads to set $Z = H^{-1}(\Omega)$ and the dual pairings $\langle \cdot, \cdot \rangle_{Y',Y}$ and $\langle \cdot, \cdot \rangle_{Z',Z}$ are thus compatible with the $L^2(\Omega)$-inner product. The last items of Assumption 2.3 follow from the Lax-Milgram Theorem (see [60, Lemma 1.8]). We have $A^* = A$ and $B^* = B$ and the adjoint p evolves in $Z' = H_0^1(\Omega)$. The partial derivatives of the functions under consideration are

$$\begin{aligned} \partial_y J(y,u) &= (y - y_d, \cdot)_U \\ \partial_u J(y,u) &= (\alpha u, \cdot)_U \\ \partial_y e(y,u) &= A \\ \partial_u e(y,u) &= -B. \end{aligned} \tag{2.34}$$

Now, from the numerical point of view, we have two main options : either the state equation $e(y,u) = 0$ is seen as a constraint to be checked and is considered, like the objective function J, to depend on both the state and the control (y,u). The optimization is then performed with respect to two optimization variables (state and control), this is option 1 ; or else, the only optimization variable is the control, and in this case $e(y,u) = 0$ is preliminarily solved to compute $y(u)$ as a function of u, in order to express the reduced cost function $\widehat{J}(u)$, this is option 2. Options 1 and 2 are also respectively called *simultaneous* and *sequential* methods.

Option 1 : Unknowns $(y,u) \in Y \times U$

Cost : $J(y,u)$
Constraints : $e(y,u) = 0$ and $(y,u) \in Y_{ad} \times \mathcal{U}_{ad}$

Option 2 : Unknowns $u \in U$

Cost : $J(y(u),u) = \widehat{J}(u)$
Constraints : $u \in \mathcal{U}_{ad}$

Option 2 brings up the reduced cost function $\widehat{J}$ whose derivatives with respect to the control variable u are computed using the adjoint representation presented in Section 2.1.2. The PDE constraint is thus implicitly contained in the numerical implementation of the reduced cost function. In contrast, Option 1 keeps the cost function dependent on the state and control variables (y,u) and the PDE constraint is an explicit equality constraint. Although option 1 seems more memory greedy, it is generally more efficient, when we can write it, than option 2 because we can more easily compute the Hessian of the cost function as a sparse matrix. Moreover, a notable advantage of option 1 is its ability to handle potential constraints on the state included in $\mathcal{Y}_{ad}$ whereas the adjoint equation and option 2 are not well suited in this case. This being said, for the numerical part, let T_h be a triangulation of Ω :

```
Th = square(50,50) //we take Ω = [0,1]^2,
```

and the finite element space

$$V_h = \left\{ v \in H^1(\Omega), \quad \forall K \in T_h \;\; v_{|K} \in \mathbb{P}_1 \right\} = \text{Vect}(\phi_i)_{i\in\{1..n_d\}},$$

with $\mathbb{P}_1$ elements that guarantee the resulting linear system of (2.31) to be invertible (see [85]).

```
Vh = fespace(Th,P1) // with P1 Lagrange finite elements
nd = Vh.ndof        // n_d degrees of freedom for V_h
```

Its basis is denoted by $(\phi_i)_{1\leqslant i\leqslant n_d}$ so that the sets Y_h, U_h and $\mathcal{U}_{ad}^h$ are

$$\begin{aligned} Y_h &= \{v \in V_h, v_{|\partial\Omega} = 0\} = \text{Vect}(\phi_i)_{i\in\{1..n\}} \quad \text{for } 0 < n < n_d, \\ \mathcal{U}_{ad}^h &= \{u \in V_h, u_a \leqslant u \leqslant u_b\} \subset U_h = V_h. \end{aligned}$$

We introduce $u_a^h = \left((u_a,\phi_i)_U\right)_{i\in\{1..n_d\}}$ and $u_b^h = \left((u_b,\phi_i)_U\right)_{i\in\{1..n_d\}}$ and the stiffness and mass matrices coming out from the operators A and B read

$$A_{h,ij} = \left(a\nabla\phi_i, \nabla\phi_j\right)_{(i,j)\in\{1..n\}^2}, \quad M_{h,ij} = \left(\phi_i,\phi_j\right)_{(i,j)\in\{1..n_d\}^2}.$$

One can notice that normally M_h is larger than A_h because the latter does not take into account the finite element functions which have non-zero values on the boundary. Fortunately, FreeFEM handles the Dirichlet boundary conditions very well so that we can consider numerically that $Y_h = V_h$ and write the stiffness matrix by specifying the Dirichlet boundary conditions directly in the variational formulation in the Code 2.8. This adds in the matrix A_h some penalty terms at the indexes related to the boundary elements (see FreeFEM's website for a detailed explanation on how FreeFEM manages Dirichlet boundary conditions).

```
Vh Y,V;
varf stiffness(Y,V) = int2d(Th)(grad(Y)'*grad(V))
                    - on(1,2,3,4,Y=0) //Homogeneous Dirichlet condition
varf mass(Y,V) = int2d(Th)(Y*V);
matrix Mh = mass(Vh,Vh); // n_d × n_d matrix
matrix Ah = stiffness(Vh,Vh);
```

Code 2.8 – Finite element matrices involved in (2.30,2.31)

IpOpt needs to calculate the numerical derivatives of the cost and constraint functions. Keeping in mind Section 2.1.3, either the continuous derivatives of the functions are discretized according to a well-chosen scheme, or the functions are discretized first and their derivatives are computed later. In the first case, the advantage is to keep the structure of the continuous problem as long as possible. In the second case, we are able to return the true numerical derivatives.

2.2.1 Derivatives of discretized functions (*FDTO*)

Here, the problem (2.30,2.31) is first discretized to obtain a usual finite dimensional linear quadratic optimization problem. The mesh, finite element space and matrices were presented in the previous section. Depending on whether option 1 or 2 is chosen, the discretized problem is

Option 1 :
$$\min_{(y_h,u_h)\in\mathbb{R}^{2n_d}} \frac{1}{2}(y_h-y_d^h)^T M_h(y_h-y_d^h)+\frac{\alpha}{2}u_h^T M_h u_h \tag{2.35}$$
$$\text{s.t.} \begin{cases} A_h y_h - M_h u_h = 0 \\ u_a^h \leqslant u_h \leqslant u_b^h, \end{cases} \tag{2.36}$$

Option 2 :
$$\min_{u_h\in\mathbb{R}^{n_d}} \frac{1}{2}(A_h^{-1}M_h u_h-y_d^h)^T M_h(A_h^{-1}M_h u_h-y_d^h)+\frac{\alpha}{2}u_h^T M_h u_h \tag{2.37}$$
$$\text{s.t. } u_a^h \leqslant u_h \leqslant u_b^h. \tag{2.38}$$

The matrices A_h and M_h have been defined in Code 2.8. They depend on the triangulation T_h. Whatever the choice of optimization variables, the numerical problem is written as

$$\min_X J(X)$$
$$\text{s.t.} \begin{cases} C_{lb} \leqslant C(X) \leqslant C_{ub} \\ X_{lb} \leqslant X \leqslant X_{ub}. \end{cases}$$

Option 1 : $X = (y_h, u_h) \in \mathbb{R}^{2n_d}$

$$\begin{aligned} J_h(X) &= \frac{1}{2}(y_h-y_d^h)^T M_h(y_h-y_d^h)+\frac{\alpha}{2}u_h^T M_h u_h \\ C_h(X) &= A_h y_h - M_h u_h \\ \nabla J_h(X) &= \begin{pmatrix} M_h(y_h-y_d^h) & \alpha M_h u_h \end{pmatrix} \\ \nabla C_h(X) &= \begin{pmatrix} A_h & 0_h \\ 0_h & M_h \end{pmatrix} \\ \nabla^2 J_h(X) &= \begin{pmatrix} M_h & 0_h \\ 0_h & \alpha M_h \end{pmatrix} \end{aligned}$$

Option 2 : $X = u_h \in \mathbb{R}^{n_d}$ (no explicit PDE constraints)

$$\begin{aligned} J_h(X) &= \frac{1}{2}(A_h^{-1}M_h u_h-y_d^h)^T M_h(A_h^{-1}M_h u_h-y_d^h)+\frac{\alpha}{2}u_h^T M_h u_h \\ \nabla J_h(X) &= M_h A_h^{-1} M_h(A_h^{-1}M_h u_h-y_d^h)+\alpha M_h u_h \\ \nabla^2 J_h(X) &= M_h A_h^{-1} M_h A_h^{-1} M_h + \alpha M_h. \end{aligned}$$

The functions required by `IpOpt` are expressed above and must be written in `FreeFEM` following the model given in Section 2.1.4.

Experience shows that option 1 is, in this case, almost always the best solution. Since PDE constrained optimization uses many variables, the treatment of sparse matrices is a crucial point and is usually the best option in terms of computational speed and memory allocation. Although the number of optimization variables is larger than in option 2, in option 1 we do not have to compute the inverse of the matrix A_h, which would be a heavy task in high dimension. The matrices A_h and M_h are sparse while their inverse is not. Although the Hessian is constant in both situations, it is more memory greedy in the second case and involves more computation. Finally, for more complicated equations (such as nonlinear PDEs or time-dependent

problems), it is not trivial that treating both state and control variables simultaneously is more efficient, except when state constraints are added to the problem. The adjoint method fits precisely in an approach where the previous one would not have worked.

2.2.2 Discretization of continuous derivatives (*FOTD*)

Unlike the previous strategy, here the continuous derivatives of the functions are first computed and then discretized. In this section, the problem (2.30,2.31) is still linear quadratic. Note that when state constraints are added, the adjoint equation can be much more complicated. We focus on the option where only control is the optimization variable and write the continuous problem as follows

$$\min_{u\in\mathscr{U}_{ad}} J(y(u),u) = \widehat{J}(u).$$

As explained in Section 2.1.2, the adjoint approach facilitates theoretically and numerically the computation of the derivatives of the reduced cost function. The Lagrangian is

$$L : (y,u,p) \in Y\times U\times Z' \mapsto \frac{1}{2}\left(B(y-y_d), y-y_d\right)_U + \frac{\alpha}{2}(Bu,u)_U + (p, Ay-Bu)_U,$$

and we follow the steps (**S.1**, **S.2**, **S.3**, **S.4**). Let $u\in\mathscr{U}_{ad}$ be an optimal solution.

- **S.1** The partial derivatives of J and e have been computed in (2.34). The operators A and B are selfadjoint ($A^* = A$ and $B^* = B$). Corollary 2.5 implies that if (y,u) is solution of (2.30,2.31) then there exists an adjoint $p\in Z' = H_0^1(\Omega)$ such that

$$\begin{aligned} Ay &= Bu,\\ Ap &= -B(y-y_d),\\ \left(\alpha Bu - Bp, u\right)_U &\leqslant \left(\alpha Bu - Bp, v\right)_U \qquad \forall v\in\mathscr{U}_{ad} \end{aligned}$$

  which can be rewritten in the continuous form for $y\in H_0^1(\Omega)$ and $p\in H_0^1(\Omega)$

$$\begin{aligned} -\nabla\cdot(a\nabla y) &= u \quad \text{in } \Omega\\ y &= 0 \quad \text{in } \partial\Omega,\\ \nabla\cdot(a\nabla p) &= y-y_d \quad \text{in } \Omega\\ p &= 0 \quad \text{in } \partial\Omega,\\ \int_\Omega(\alpha u - p)u\,dx \leqslant \int_\Omega(\alpha u-p)v\,dx &\quad \forall v\in\mathscr{U}_{ad}. \end{aligned}$$

- **S.2** We seek $y\in H_0^1(\Omega)$ solution of the state equation $e(y,u)=0$

$$\begin{aligned} -\nabla\cdot(a\nabla y) &= u \qquad \text{in } \Omega\\ y &= 0 \qquad \text{in } \partial\Omega, \end{aligned}$$

```
macro state() {
    solve State(Y,V) = int2d(Th)(a*(grad(Y)'*grad(V)))
                     - int2d(Th)(U*V)
                     + on(1,2,3,4,Y=0); } //
```

Code 2.9 – LQ state equation

which is equivalent to

$$y_h = A_h^{-1} M_h u_h.$$

- **S.3** We seek $p \in H_0^1(\Omega)$ solution of the adjoint equation

$$\begin{aligned} \nabla \cdot (a \nabla p) &= y - y_d \quad \text{in } \Omega \\ p &= 0 \quad \text{in } \partial\Omega, \end{aligned}$$

```
macro adjoint() {
    solve Adjoint(P,Q)= int2d(Th)(a*(grad(P)'*grad(Q)))
                      + int2d(Th)((Y-Yd)*Q)
                      + on(1,2,3,4,P=0); } //
```

Code 2.10 – LQ adjoint equation

which is equivalent to

$$p_h = A_h^{-1} M_h (y_d^h - y_h).$$

- **S.4** The first derivative of the reduced cost function is given by

$$\left(\nabla \widehat{J}(u), q\right)_U = \int_\Omega (\alpha u - p) q \, dx,$$

```
macro interpgrad() {
    solve L2grad(theta,V) = int2d(Th)(theta*V) - int2d(Th)((alpha*U-P)*V)
        ;
    real[int] dJ = theta[]; } //
```

Code 2.11 – LQ gradient's interpolation

which finally returns $\alpha u_h - p_h$ for the $L^2(\Omega)$-inner product. The previous steps **S.2**, **S.3**, **S.4** performed in succession return almost the same gradient as option 2 of the previous *FDTO* approach by performing the same operations but without the final multiplication by M_h.

We give a brief explanation of the numerical step **S.4**. The real number $\langle D\widehat{J}(u), q\rangle_{U',U}$ is expressed using the dual pairing $\langle \cdot, \cdot \rangle_{U',U}$ which here corresponds to the $L^2(\Omega)$-inner product and thus brings out the gradient $\nabla \widehat{J}(u)$. As seen in Section 2.1.1, the routine **varf** gives the matrix of a given variational formulation. Thus writing the lines

```
varf derive(V,Q)= int2d(Th)((alpha*U-P)*Q)
```

is numerically similar to recovering the continuous linear form

$$q \mapsto \int_\Omega (\alpha u - p) q \, dx$$

in a well-chosen finite element space basis. Thus `dJ` denotes the interpolation of this linear form in this basis of the finite element space. One could of course choose another inner product, whose matrix in the finite element space is P_h, which would not return

$$\alpha u_h - p_h$$

but rather

$$P_h^{-1} M_h(\alpha u_h - p_h).$$

This is a crucial point to understand because many problems show the derivative of the reduced cost function in a linear form which must then be interpolated through a good discretization of the space U. Usually, the additional constraints included in $\mathscr{U}_{ad}$ are specified in another function C, the so-called constraint function (e.g. $u \mapsto C(u) = \int_\Omega u(x)\,dx$). The Jacobian of C is then required by `IpOpt`. The only remaining task is to express the cost function and its gradient following the steps (**S.2**, **S.3**, **S.4**) on the triangulation T_h respectively in the codes 2.12 and 2.13.

```
func real J(real &X)
{
    U[] = X; // control u
    state; // returns y solution of (2.9)
    return int2d(Th)((Y-Yd)^2) + alpha*int2d(Th)(U^2);
}
```

Code 2.12 – LQ cost function

```
func real[int] dJ(real &X)
{
    U[] = X;
    state;    // state equation (see Code 2.9)
    adjoint; // adjoint equation (see Code 2.10)
    interpgrad; // interpolation of the gradient (see Code 2.11)
    return dJ;
}
```

Code 2.13 – Derivative of the LQ cost function

In the first method, the calculation of the Hessian is rather easy but it requires a matrix inversion. We advise instead to compute the theoretical first order derivatives and to use the BFGS approximation of the Hessian provided by `IpOpt`.

Remark 2.12. *The optimality conditions produce a state-adjoint system called* extremal system. *In finite-dimensional optimal control, the solution of the first-order optimality system can usually be performed by implementing an indirect shooting method (see [89, Chapter 9]), where, if for example the initial and final states are fixed, one has to properly adjust the initial adjoint vector so that, when integrating the extremal with the corresponding initial state and adjoint, the final state matches the desired value. Numerically, the shooting method consists of integrating a differential equation and applying a Newton method. When the optimal control problem is solved in finite dimension and this dimension is not too large, the method is feasible and, when properly initialized to ensure convergence, it provides a very fast and accurate solution. But, in high dimension, it can be extremely difficult to initialize the method correctly. This is particularly the case in PDE optimization where the size of the adjoint variable is related to the size of the mesh. Therefore, most of the time, it is not realistic to ensure the convergence of a shooting method for optimal PDE control problems.*

2.2.3 Inhomogeneous Dirichlet boundary conditions

The previous example treated the case of homogeneous Dirichlet conditions, which makes easier the numerical study because the variational formulation can be written on well-identified functional spaces (this is also the case when dealing with Neumann or Robin's boundary conditions). We now generalize the above numerical approaches to inhomogeneous Dirichlet conditions. Instead of (2.31), we now consider

$$\begin{cases} -\nabla\cdot(a\nabla y) = u & \text{in } \Omega \\ y = g & \text{in } \partial\Omega, \end{cases} \tag{2.39}$$

where g is assumed to be smooth enough. The way to numerically manage the boundary condition depends on the choice of either Option 1 or Option 2.

In Option 2, the boundary condition can be immediately reported in the variational formulation so that the numerical state equation becomes :

```
macro state() {
    solve State(Y,V) = int2d(Th)(a*(grad(Y)'*grad(V)))
                     - int2d(Th)(U*V)
                     + on(1,2,3,4,Y=g); //y=g in ∂Ω
} //
```

Code 2.14 – State equation

The adjoint equation remains unchanged but the adjoint is well modified because it still depends on the state.

In contrast, Option 1 requires more work. For homogeneous Dirichlet conditions, the stiffness matrix A_h introduced in Code 2.8 forces y to be equal zero on the boundary $\partial\Omega$ and can only be applied to finite element functions in $H_0^1(\Omega)$. Since y now belongs to an affine subset of $H^1(\Omega)$ and not to $H_0^1(\Omega)$, the state equation cannot be written as before. The variational formulation of (2.39) is now :

$$\text{find } y\in H^1(\Omega), \int_\Omega a\nabla y\cdot\nabla v\,dx - \int_\Omega uv\,dx = 0 \qquad \forall v\in H_0^1(\Omega). \tag{2.40}$$

A numerical trick consists of introducing, using the Dirichlet map $\mathbf{D}$ (see [95, Section 10.6]), the solution $\mathbf{D}g$ of

$$-\Delta(\mathbf{D}g) = 0 \text{ on } \Omega \text{ and } \mathbf{D}g = g \text{ on } \partial\Omega$$

and its numerical version :

```
Vh Dg;
solve dirmap(u,v) = int2d(Th)(grad(Y)'*grad(V))
                  + on(1,2,3,4,Y=g);
Dg[] = Y[];
```

Code 2.15 – Dirichlet map

So we look for the solution of (2.39) in the affine space $H_0^1(\Omega)+\mathbf{D}g$ under the form $y = z+\mathbf{D}g$. Therefore, finding a solution $y\in H^1(\Omega)$ of (2.39) is equivalent to finding a solution $z\in H_0^1(\Omega)$ of

(2.31). The optimization variables are (z,u) and the discretized problem is the following

$$\min_{(z_h,u_h)\in\mathbb{R}^{2n_d}} J_h(z_h,u_h)=\frac{1}{2}\left(z_h+(\mathbf{D}g)_h-y_d^h\right)^T M_h\left(z_h+(\mathbf{D}g)_h-y_d^h\right)+\frac{\alpha}{2}u_h^T M_h u_h$$
$$\text{s.t.}\begin{cases}A_h z_h-M_h u_h=0\\ u_a^h\leqslant u_h\leqslant u_b^h.\end{cases}$$

The state constraint is unchanged (as well as its Jacobian) while the cost function now involves $y_h=z_h+(\mathbf{D}g)_h$. Most of the following examples will specify homogeneous Dirichlet conditions but a generalization to inhomogeneous Dirichlet conditions can be made along the lines above.

2.2.4 Automatic differentiation alternative

Unless the derivatives are really easy to calculate, it is very advantageous to use automatic differentiation, which allows the numerical derivatives to be calculated at the computer accuracy. As automatic differentiation is not available in FreeFEM (especially when one wants to make a derivation with respect to the mesh points), it is necessary for the moment to export the problem data collected with FreeFEM (mesh data, matrices of the variational forms involved, etc.) into another language which benefits from an automatic differentiation program. In order to keep a user-friendly interface, we propose two practical solutions : either to use the modeling language AMPL, or the Python package CasADi (with free license and callable from Matlab). Finally, for those who are quite familiar with the language C++, we advise to combine directly IpOpt with an automatic differentiation tool C++ (such as CppAD or Adept, see [61]) for more efficiency. In the following, we provide numerical examples that illustrate how to combine FreeFEMwith AMPL in the linear quadratic case (2.30,2.31). The combination with the Python package CasADi is presented in Appendix 2.C. For a given triangulation T_h and generated with FreeFEM, we store the matrices A_h and M_h constructed in the Code 2.8 of Section 2.2 in sparse matrices via the COO (Coordinate) format. The files "A.txt" and "B.txt" are generated via FreeFEM with the command :

```
{ofstream fout("Ah.txt");
fout << Ah << endl;
}
```

AMPL ("A Mathematical Programming Language", see [1; 44]) is a highly developed software for modeling and solving large-scale optimization problems. Like CasADi, the unknown variables must be declared and the objective and constraints must be defined as a nonlinear programming problem. The transcription of AMPL is based on classical logical operators, aggregation functions and sets, while the transcription of CasADi is done more in the framework of matrix calculus. Both handle the sparsity feature of matrices very well. As far as AMPL is concerned, it is necessary to write the problem using appropriate sets (set of indices of the non-zero elements of a matrix for example). A significant advantage of AMPL is the possibility to call several recognized optimization solvers, like Knitro, CPLEX, IpOpt, etc, which can be used free of charge on the server NEOS. The program is usually divided into three files : "file.mod" contains the model, while "file.dat" collects the parameter allocations and "file.run" the successive commands.

The Code 2.16 includes the introduction of variables (lines 12 and 13), parameters (line 1 and lines 6 to 10), minimization of the objective function (line 15) and constraint functions

(lines 19 to 21). Unlike CasADi, we do not use explicit matrix calculus in AMPL, the **sum** operator on appropriate sets is preferred instead. The parameters A and M represent the sparse matrices A_h and M_h introduced in the Code 2.8. To preserve sparsity, the matrices in AMPL are downloaded from a text file in triplet format (see sparse matrices on the Wikipedia web page for multiple ways to store them). The indices for which A has non-zero values are declared in a two-dimensional set indexA. The parameter A{indexA} depending on this set thus gathers the non-zero values of the matrix A_h (idem for M_h). The quadratic cost function is obtained on line 15 of the Code 2.16 by summing over all the indices (i,j) **in** indexM and we therefore do not take into account in the sum the indices where M has a null value.

```
1  param m integer >=0;
2  set index = 0..m;
3  set indexA dimen 2;
4  set indexM dimen 2;
5
6  param alpha =0.1;
7  param A{indexA}; # stiffness matrix
8  param M{indexM}; # mass matrix
9  param L{index};
10 param yd; # target
11
12 var Y{index}; # state
13 var U{index}; # control
14
15 minimize quad: sum{(i,j) in indexM}( 0.5*(Y[i]-yd)*M[i,j]*(Y[j]-yd) +
       0.5*alpha*(U[i]*M[i,j]*U[j]));
16
17 subject to PDE{i in index}: sum{(i,j) in indexA}(A[i,j]*Y[j]) - sum{(i,j)
       in indexM}(M[i,j]*U[j]) = 0;
18
19 subject to lowbound{i in index}: U[i] >= 0;
20 subject to upbound{i in index}: U[i] <=1;
21
22 subject to volume: sum{i in index} U[i]*L[i] == 0.25; # ∫Ω u(x) dx = 0.25
```

Code 2.16 – AMPL: "file.mod"

This induces a significant gain in computation time. Like all the parameters, the data are assigned in Code 2.17 :

```
1  data;
2
3  param m:= 2600;
4  param: indexA: A:= include A.txt;
5
6  param: indexM: B:= include B.txt;
7
8  param yd:= 0.1;
9  read{i in index}(L[i]) < L.txt;
```

Code 2.17 – AMPL: "file.dat"

AMPL is then called by typing in the terminal the command : `ampl file.run`

```
model file.mod;
data file.dat;

option solver ipopt;
option ipopt_options"max_iter=1000 tol=1.e-12 linear_solver=mumps";

solve;
```

Code 2.18 – AMPL: "file.run"

Contingent upon an easy importation of data as in Code 2.39, AMPL stands out for its ability to manage many efficient optimization solvers such as Knitro, CPLEX and so on, which, as already said, can be used for free on NEOS, on which we have to upload the three above files (in order to be uploaded to NEOS, data file must contain parameters allocations stated in file *"A.txt"*, *"B.txt"* etc, since they cannot be uploaded aside). Python (and Matlab) alternatives are presented in Appendix 2.C.

To conclude this part, we plot in Figure 2.3 the convergence curves of the scaled error respectively associated to the several methods illustrated above. In the case of the Linear-Quadratic problem (2.30,2.31), we notice that the *FDTO* method is ten times faster than the *FOTD* one. AMPL endowed with its own automatic differentation tool is an entirely acceptable alternative.

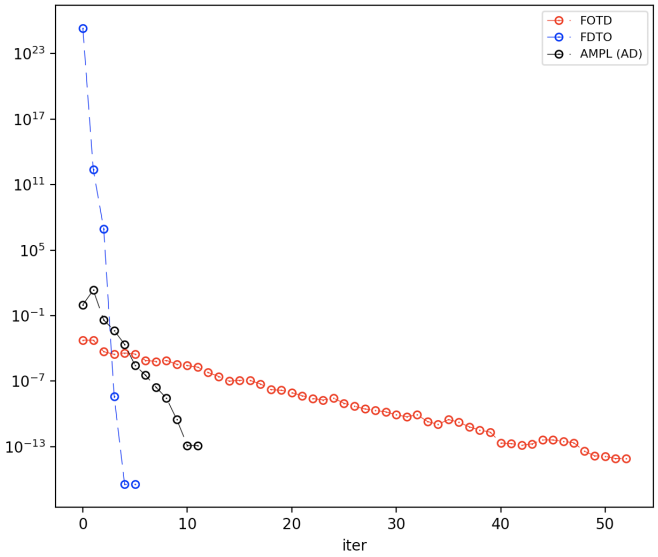


FIGURE 2.3 – Convergence curves of several methods

2.3 Extension to time-dependent problems

In this section, we show how to deal with a time-dependent optimal control problem, which this time consists in determining $u \in L^2((0,T);L^2(\Omega))$ solution of

$$\min J(y,u) = \frac{1}{2}\int_0^T \int_\Omega \left(y(x,t)-y_d(x)\right)^2 dxdt + \frac{\alpha}{2}\int_0^T \int_\Omega u(x,t)^2 dxdt \tag{2.41}$$

$$\text{subject to } \begin{cases} y_t - \nabla\cdot(a\nabla y) = u & \text{in } (0,T)\times\Omega \\ y = 0 & \text{in } (0,T)\times\partial\Omega \\ y(0) = y^0 & \\ u_a \leqslant u \leqslant u_b & \end{cases} \tag{2.42}$$

with $y^0 \in L^2(\Omega)$. For the existence of solutions, we do not treat the question here and refer the reader to [60; 69]. Here we focus on the numerical implementation of such a problem. Several time discretization strategies can be implemented, which can then be combined with the optimization strategies described above. We focus here on option 1, but the other possibilities can be adapted in a similar way to time-dependent problems. We introduce a mesh of the Ω domain where the PDE is written, as in the previous stationary example. The main question is therefore how to deal simultaneously with the temporal and spatial discretizations. A first classical discretization in time is to use an implicit Euler scheme combined (we will prefer most of the time implicit schemes since there is no CFL condition to satisfy) with finite elements $\mathbb{P}_1$ in space. A second possibility is to use the ability of the `FreeFEM` to handle 3D problems by introducing the 3D time-space mesh and to discretize simultaneously in time and space with finite elements $\mathbb{P}_1$.

2.3.1 Implicit Euler scheme.

Although an implicit scheme is a bit harder to implement numerically because it requires a matrix inversion, its advantage over explicit schemes is that it does not require any LFC condition. We consider the mesh T_h introduced in Section 2.2 and the matrices A_h and M_h. Let n_t be an integer and consider a subdivision

$$t_0 = 0 < t_1 < \cdots < t_{n_t} = T$$

of the interval $[0,T]$. We introduce the discrete variables

$$\tilde{Y} = (y_0,\cdots,y_{n_t}) \in \mathbb{R}^{n_d(n_t+1)} \text{ and } \tilde{U} = (u_0,\cdots,u_{n_t}) \in \mathbb{R}^{n_d(n_t+1)}$$

(with $y_i, u_i \in \mathbb{R}^{n_d}$ finite element approximations of $y(t_i,\cdot)$ and $u(t_i,\cdot)$) and we discretize the problem (2.41,2.42) as

$$\min J(\tilde{Y},\tilde{U}) = \frac{1}{2}\sum_{k=1}^{n_t}(y_k - y_d^h)^T M_h (y_k - y_d^h) + \frac{\alpha}{2}\sum_{k=1}^{n_t} u_k^T M_h u_k \tag{2.43}$$

$$\text{subject to } \begin{cases} M_h \frac{y_{k+1}-y_k}{dt} + A_h y_{k+1} = M_h u_{k+1}, & \forall k \in (0...n_t-1) \\ y_0^i = (y^0,\phi_i)_U, & \forall i \in \{1...n_d\} \\ \sum_{i=1}^{n_d} u_a^i \phi_i \leqslant \sum_{i=1}^{n_d} u_k^i \phi_i \leqslant \sum_{i=1}^{n_d} u_b^i \phi_i, & \forall (k,i) \in \{0...n_t\}\times\{1...n_d\}. \end{cases} \tag{2.44}$$

We simplify the notations by introducing the matrices

$$A_t = A_h + \frac{1}{\delta t} M_h, \quad M_t = \frac{1}{\delta t} M_h$$

for $\delta t = \frac{T}{n_t}$ and the sparse matrices

$$\tilde{A} = \underbrace{\begin{pmatrix} I_{n_d} & 0_{n_d} & \cdots & 0_{n_d} \\ M_t & A_t & \cdots & \vdots \\ \vdots & \ddots & \ddots & 0_{n_d} \\ 0_{n_d} & \cdots & M_t & A_t \end{pmatrix}}_{n_d(n_t+1) \text{ columns and rows}} \quad \tilde{M} = \begin{pmatrix} 0_{n_d} & 0_{n_d} & \cdots & 0_{n_d} \\ 0_{n_d} & M_t & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0_{n_d} \\ 0_{n_d} & \cdots & 0_{n_d} & M_t \end{pmatrix} \quad \tilde{D} = \begin{pmatrix} M_t & 0_{n_d} & \cdots & 0_{n_d} \\ 0_{n_d} & M_t & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0_{n_d} \\ 0_{n_d} & \cdots & 0_{n_d} & M_t \end{pmatrix} \tag{2.45}$$

to reformulate the optimization problem (2.43,2.44) as the problem of determining $(\tilde{Y}, \tilde{U}) \in \mathbb{R}^{2n_d(n_t+1)}$ solution of

$$\min J(\tilde{Y}, \tilde{U}) = \frac{1}{2}(\tilde{Y} - Y_d)^T \tilde{D}(\tilde{Y} - Y_d) + \frac{\alpha}{2} \tilde{U}^T \tilde{D} \tilde{U}$$

$$\text{subject to } \begin{cases} \tilde{A}\tilde{Y} - \tilde{M}\tilde{U} = \begin{pmatrix} y_0 \\ 0_{n_d n_t} \end{pmatrix}, \\ U_a \leqslant \tilde{U}_k \leqslant U_b \quad \forall k \in \{0..n_t\} \end{cases}$$

with

$$U_a = \left((u_a, \phi_i)_U\right)_{i \in \{1..n_d\}}, \quad U_b = \left((u_b, \phi_i)_U\right)_{i \in \{1..n_d\}}.$$

We thus obtain a finite-dimensional linear quadratic problem

$$\nabla J(\tilde{Y}, \tilde{U}) = \begin{pmatrix} \tilde{D}(\tilde{Y} - Y_d) \\ \alpha \tilde{D} \tilde{U} \end{pmatrix}, \quad \nabla^2 J(\tilde{Y}, \tilde{U}) = \begin{pmatrix} \tilde{D} & O \\ 0 & \alpha \tilde{D} \end{pmatrix}, \quad \nabla C(\tilde{Y}, \tilde{U}) = \begin{pmatrix} \tilde{A} & 0 \\ 0 & -\tilde{M} \end{pmatrix}.$$

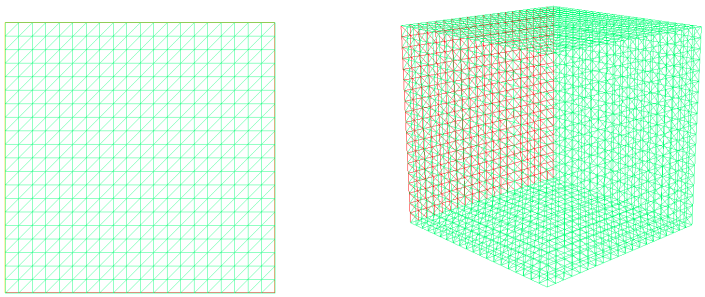
for which the numerical implementation in FreeFEM now follows the one presented in Section 2.2.1 based on the template introduced in Section 2.1.4.

2.3.2 Time discretization with FreeFEM.

Another possibility is to exploit the ability of FreeFEM to easily handle multidimensional problems, by considering the time variable as a third variable of the z space and transforming a 2D (or 1D) problem into a 3D (or 2D) problem. In the example (2.41,2.42), the variational formulation reads as follows for all v in $H^1((0,T) \times \Omega))$ such that $v_{|(0,T)\times\partial\Omega} = 0$:

$$\int_{(0,T)\times\Omega} \left(y_t v + a \nabla y \cdot \nabla v\right) dx dt = \int_{(0,T)\times\Omega} uv \, dx dt \tag{2.46}$$

So we build a new mesh of the 3D domain $(0,T) \times \Omega$ on which the PDE evolves. In order to manage the 3D meshes, we use at the beginning of the file the command : `load "msh3"`. Several ways to build a 3D mesh with FreeFEM are possible. We prefer to start from an initial mesh of the Ω domain subset of$\mathbb{R}^2$ which we transform into a mesh of the cylinder $(0,T) \times \Omega$ thanks to the command `buildlayers` which will extend the given mesh along the axis z :



(a) (b)

FIGURE 2.4 – (a) initial square mesh ; (b) 3D mesh with `buildlayers`

```
mesh Th2 = square(10,10,region=10);
int[int] rup=[1,10],rdown=[6,10],rmid=[2,10,3,10,4,10,5,10]; // labels
mesh3 Th = buildlayers(Th2,n,zbound=[0,1],labelmid=rmid,reffaceup=rup,
    reffacelow=rdown);
```

Code 2.19 – 3D mesh cylinder with 2D mesh basis

and a new finite element space

$$V_h = \left\{ v \in H^1((0,T)\times\Omega), \quad \forall K \in T_h \;\; v_{|K} \in \mathbb{P}_1 \right\}$$

(in this case, the temporal discretization performed with a finite element discretization is symmetric). As in the static case, we define the matrix of the variational formulation (2.46) :

```
varf vA(Y,V) = int3d(Th)(dz(Y)*V+a*(grad(Y)'*grad(V))) //dz(Y)=∂_t y
             +on(2,3,4,5,Y=0) // Dirichlet boundary condition
             +on(6,Y=Y0); // initial condition y^0
matrix Ah = vA(Vh,Vh,solver=sparsesolver);
```

We define the matrix of the cost function :

```
varf vCost(Y,V) = int3d(Th)(Y*V);
matrix Mh = vCost(Vh,Vh,solver=sparsesolver);
```

Given a mesh of the domain $(0,T)\times\Omega$, we now write the discretized optimal problem as

$$\min_{(\tilde{Y},\tilde{U})\in\mathbb{R}^{2N}} \frac{1}{2}(\tilde{Y}-Y_d)^T M_h(\tilde{Y}-Y_d) + \frac{\alpha}{2}\tilde{U}^T M_h \tilde{U}$$
$$\text{s.t.} \begin{cases} A_h\tilde{Y} = M_h\tilde{U} \\ U_a \leqslant \tilde{U} \leqslant U_b \end{cases}$$

where N is the number of degrees of freedom of the chosen finite element space (taking into account the time dimension), U_a and U_b are interpolations on the finite element spaces of functions u_a and u_b. Here, N is of the same order as the quantity $n_d(n_t+1)$ from the previous section. The gradient and Hessian of the cost function and the Jacobian of the constraint function are thus now easy to compute and the numerical implementation again follows that presented in Section 2.2.1.

When considering linear quadratic optimal control problems, the problems resulting from the different discretization strategies remain general high dimensional linear quadratic optimization problems. Nevertheless, the previous examples can be used as models for other examples, more complicated, but which require upstream an important work on the chosen discretization. When the PDE concerned is linear, we recommend to use option 1. Indeed, keeping the PDE as a linear constraint rather than solving it directly, which requires at least a matrix inversion (the matrix inversion will always take place but the linear solver chosen in `IpOpt` will do it), will most of the time bring two advantages. First, it is easier to compute the Jacobian of this constraint and it is almost always easier to compute the derivatives of the cost function with respect to the state and control than the derivatives of the reduced cost function, whose Hessian is even harder to compute. Secondly, we keep the sparse feature of the matrices involved in the discrete writing of the problem. Option 2 (use of an adjoint representation) is more suitable when the dependence on the control is more complex and cannot be expressed easily, as in the case of optimal design of a shape, when the control is a shape (see [7; 59]) or for nonlinear problems. Option 1 or Option 2 differ in the choice of the parameterization of the optimal shape to be found. Furthermore, regardless of the approach chosen, time-dependent problems are much more computationally and memory intensive and require careful attention to potential simplifications.

2.4 Optimization under semilinear PDE constraints

Let us now add a nonlinear term to the previous problem. Given $y_d \in L^2(\Omega)$ and $u_a, u_b \in L^\infty(\Omega)$, we are still looking for an optimal solution $u \in L^2(\Omega)$ for the minimization of a quadratic criterion constrained by a semi-linear elliptic equation

$$\min J(y,u) = \frac{1}{2}\int_\Omega \big(y(x)-y_d(x)\big)^2\,dx + \frac{\alpha}{2}\int_\Omega u(x)^2\,dx \tag{2.47}$$

$$\text{subject to } \begin{cases} -\nabla\cdot(a\nabla y)+\phi(y) = u & \text{in } \Omega \\ y = 0 & \text{in } \partial\Omega \\ u_a \leqslant u \leqslant u_b. \end{cases} \tag{2.48}$$

Following the framework stated in [94, Assumptions 4.2 and 4.3], the function

$$\phi : \mathbb{R} \mapsto \mathbb{R}$$

is a continuous, monotone increasing globally bounded and twice differentiable function. The global boundedness assumption is only used to ensure that $\phi(y) \in L^2(\Omega)$. In our numerical tests, we choose $\phi(y) = y^3$, which is not globally bounded, but Sobolev injections guarantee

$$H^1(\Omega) \hookrightarrow L^6(\Omega)$$

for Ω bounded Lipschitz domain of $\mathbb{R}^2$ or $\mathbb{R}^3$. Therefore, given any $u \in L^2(\Omega)$, the PDE (2.48) admits a unique weak solution $y \in H_0^1(\Omega)$ (see [Remarks on Theorem 4.4]MR2583281). Compared to the equation of state (2.31), the semi-linear PDE is more difficult to solve with the finite element method. We do not have a direct linear variational formulation and therefore there is no way to solve it with a single matrix inversion.

We therefore propose to solve the PDE numerically with iterative fixed point or Newton methods and to use an adjoint representation to compute the derivatives. Let us note here that the adjoint will only need a matrix inversion because the adjoint equation is always linear. Let T_h be the same triangulation of the domain Ω as before and V_h still denote the finite element space $\mathbb{P}_1$. The variational formulation is

$$\int_\Omega a\nabla y\cdot\nabla v\,dx+\int_\Omega \phi(y)v\,dx=\int_\Omega uv\,dx \qquad \forall v\in H_0^1(\Omega).$$

A first possibility is to use a fixed point algorithm 2 to compute a numerical solution of this variational formulation, whose translation in `FreeFEM` macros is given in Code 2.20.

Algorithm 2 Fixed point method for semilinear PDEs

set $err=1$
while $(err>10^{-10})$ **do**
  solve : $\forall v\in H_0^1(\Omega)$, $\displaystyle\int_\Omega a\nabla z\cdot\nabla v\,dx+\int_\Omega \phi(y^{p-1})v\,dx=\int_\Omega uv\,dx$
  compute $err=\|z-y^{p-1}\|_{L^2(\Omega)}$
  set $y^p=z$
  set $p=p+1$.
end while

```
macro semilinearstateFP(){
  real err=1;
  while (err>tol) {
    Vh uold;
    solve semilinear(Y,V) = int2d(Th)(a*grad(Y)'*grad(V))
                          + int2d(Th)(Yold^3*V)  // φ(y) = y^3
                          - int2d(Th)(U*V)
                          + on(1,2,3,4,Y=0);
    real[int] taberr = Y[]-Yold[];
    err = l2norm(taberr); // ||y^{k+1} - y^k||_{L^2(Ω)}
    // l2norm defined in appendix 2.A
    }
} //
```

Code 2.20 – Semilinear state equation with a fixed point method

Another possibility is to implement a Newton-Raphson strategy, consisting in determining $y\in V$ such that $F(y)=0$ with $F:V\mapsto V$, as done in Algorithm 3. In the context of the problem (2.47,2.48), F is the variational formulation of the state equation (2.48) and we have

$$F(y)=\int_\Omega (a\nabla y\cdot\nabla v+\phi(y)v-uv)\,dx$$
$$DF(y)w=\int_\Omega (a\nabla w\cdot\nabla v+\phi'(y)wv)\,dx$$

for some $v\in H_0^1(\Omega)$, and (2.48) is solved in Code 2.21.

Algorithm 3 Newton method

set $err = 1$
set y_0 and w initialized
while $(err > 10^{-10})$ **do**
  $y^p = y^{p-1} - w$
  solve : $DF(y^p)w = F(y^p)$
  $err = \|v\|$
  set $p = p + 1$.
end while

```
macro semilinearstateNewton(){
  real err=1;
  Vh W=0;
  while (err>tol) {
    Y[] -= W[]; // y^{p+1} = y^p - w
    solve semilinear(W,V) = int2d(Th)(a*grad(W)'*grad(V)) // DF(y)w
                          + int2d(Th)(3*Y^2*W*V) // φ(y) = y^3
                          - ( // F(y)
                            int2d(Th)(a*grad(Y)*grad(V) + Y^3*V - U*V) )
                          + on(1,2,3,4,W=0);
    err = l2norm(W[]); // ||w||_{L^2(Ω)}
    // l2norm defined in appendix 2.A
    }
} //
```

Code 2.21 – Semilinear state equation with Newton method

One may wonder whether it is better to use a fixed point or a Newton method. Newton's algorithm is well known for its fast and very accurate convergence, but on the condition that its initialization needs to be close enough to the solution. The fixed point algorithm is less sensitive to this initialization constraint but at the cost of a slower convergence. A possible solution is to consider a hybrid method which consists in using a fixed point method in the first iterations and then switching to a Newton method when one is close enough to the solution (this is the idea of some global or damped Newton methods). As in Section 2.2, Y is the set $H_0^1(\Omega)$ and the set of admissible controls $\mathcal{U}_{ad}$ is the subset of $L^2(\Omega)$ given by

$$\mathcal{U}_{ad} = \{u \in L^2(\Omega), u_a \leqslant u \leqslant u_b\} \subset U = L^2(\Omega),$$

Z is $H^{-1}(\Omega)$, the dual of $H_0^1(\Omega)$, so that Assumption 2.3 is satisfied, and

$$J(y,u) = \frac{1}{2}\int_\Omega \big(y(x) - y_d(x)\big)^2\, dx + \frac{\alpha}{2}\int_\Omega u(x)^2\, dx,$$
$$e(y,u) = Ay + B\phi(y) - Bu.$$

Under [94, Assumptions 4.14], existence of an optimal control $\bar{u} \in \mathcal{U}_{ad}$ is guaranteed and the optimality conditions stated in Corollary 2.3 give

$$y \in H_0^1(\Omega) \text{ solution of : } -\nabla\cdot(a\nabla y) + \phi(y) = u \quad \text{in } \Omega, \tag{2.49}$$
$$p \in H_0^1(\Omega) \text{ solution of : } \nabla\cdot(a\nabla p) - \phi'(y)p = y - y_d \quad \text{in } \Omega, \tag{2.50}$$
$$u \in \mathcal{U}_{ad} \text{ such that : } \big(\alpha u - p, v - u\big)_U \geqslant 0 \quad \forall v \in \mathcal{U}_{ad}. \tag{2.51}$$

Again, like in the example (2.30,2.31), the dual pairing $\langle\cdot,\cdot\rangle_{U',U}$ is compatible with the $L^2(\Omega)$-inner product and hence the adjoint representation yields the gradient of the reduced cost function

$$\left(\nabla\widehat{J}(u),v\right)_U = \int_\Omega (\alpha u - p)\, v\, dx \quad \forall v \in L^2(\Omega).$$

Although the state equation is nonlinear, the adjoint equation is linear and can thus easily be solved with `FreeFEM`:

```
macro semilinearadjoint() {
    solve SLAdjoint(P,Q)= int2d(Th)(a*(grad(P)'*grad(Q)))
                        + int2d(Th)(3*Y^2*P*Q)  // φ'(y) = 3y^2
                        + int2d(Th)((Y-Yd)*Q)
                        + on(1,2,3,4,P=0);
} //
```

Code 2.22 – Semilinear adjoint equation

At this stage, it is sufficient to rewrite the cost function and its derivative as in Codes 2.12 and 2.13 by replacing the macros of the previous state and adjoint equations.

In the case of semi-linear equations, it is possible to work with sparse matrices that do not depend on the state and control, provided that we are careful about how to discretize the non-linear term. Therefore, we can override the adjoint calculation by using automatic differentiation by the software `AMPL` as in Section 2.2.4. In that aim, we still reduce the cost function to the quadratic discretized cost

$$J_h(y_h,u_h) = \frac{1}{2}(y_h - y_d^h)^T M_h (y_h - y_d^h) + \frac{\alpha}{2} u_h^T M_h u_h.$$

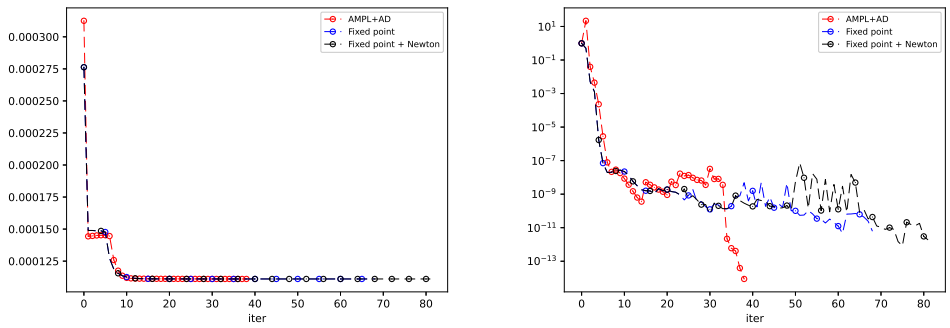
At that point, we decide to approximate the state equation with the following nonlinear equation

$$e_h(y_h,u_h) = A_h y_h + M_h (y_h)^3 - M_h u_h.$$

Here, we decide to approximate both y and y^3 with finite elements $\mathbb{P}_1$. This arbitrary choice is not without consequences since a small approximation error is made when we interpolate y^3 with affine functions when it should be approximated with polynomials of degree 3. Nevertheless, since the quadrature formula in the cost function only requires the values of y on the vertices of the mesh, the approximation error is controllable and will be smaller for a finer mesh. This is an arbitrary choice that allows the use of automatic differentiation and greatly simplifies the solution of the equation of state, compared to the iterative methods used above, but with less accuracy. The file "file.mod" is thus modified accordingly :

```
param m integer >=0;
set index = 0..m;
set indexA dimen 2;
set indexM dimen 2;

param alpha =0.1;
param A{indexA}; # stiffness matrix
param M{indexM}; # mass matrix
param L{index};
param yd; # target

var Y{index}; # state
var U{index}; # control

minimize quad: sum{(i,j) in indexM}( 0.5*(Y[i]-yd)*M[i,j]*(Y[j]-yd) +
    0.5*alpha*(U[i]*M[i,j]*U[j]));

subject to PDE{i in index}: sum{(i,j) in indexA}(A[i,j]*Y[j])
             + sum{(i,j) in indexM}(M[i,j]*Y[j]**3)
             - sum{(i,j) in indexM}(M[i,j]*U[j]) =0;

subject to lowbound{i in index}: U[i] >= 0;
subject to upbound{i in index}: U[i] <=1;

subject to volume: sum{i in index} U[i]*L[i] == 0.25; # ∫_Ω u(x) dx = 0.25
```

Code 2.23 – AMPL: "file.mod" - semilinear case



(a) (b)

FIGURE 2.5 – Convergence curves for semilinear case : (a) objective function ; (b) convergence criterion

The convergence curves of the different methods used are shown in Figure 2.5. The different methods converge more or less quickly to the same solution. Here we emphasize the advantages of automatic differentiation over adjoint methods that bypass solving the state's equation by iteration methods. The convergence of the algorithm is consequently faster and its numerical writing is much easier. Nevertheless, we must pay some attention to the numerical discretization of the cost and state constraint functions.

2.5 Optimal shape design problems

FreeFEM is very well adapted to solve numerically the problems of optimal shape design. It is indeed very user-friendly for solving PDEs, but also for building and modifying meshes. Shape optimization is a vast field and there are many ways to find optimal solutions numerically. It is possible to implement geometric and topological optimization as well as homogenization methods. For the *level-set* method in FreeFEM, we refer the reader to [8; 9] and to the dedicated web site. This method is related to the solution of an advection equation combined with a gradient descent algorithm (see [43] for an example of an efficient gradient descent algorithm applied to shape optimization). We mention two methods that can be combined with IpOpt.

- Shape deformation methods : we look for a way to modify an initial shape to a so-called optimal shape and thus compute the derivatives of the cost and constraint functions using classical shape optimization analysis tools. The command movemesh provided by FreeFEM allows us to write an optimal shape design problem in the context of Hadamard boundary variations. The derivatives with respect to the domain and the appropriate deformation vector fields are thus computed to find the next iteration using the movemesh. We recommend [59] for an introduction to shape variation strategies.
- Relaxation methods : we look for an optimal solution in a larger admissible set containing the classical design sets and expect the solution to be in the starting set of shapes. One of the best known methods is homogenization (see [6; 10]). The optimal solution often presents a gray level instead of being black or white, which underlines the appearance of a relaxation phenomenon.

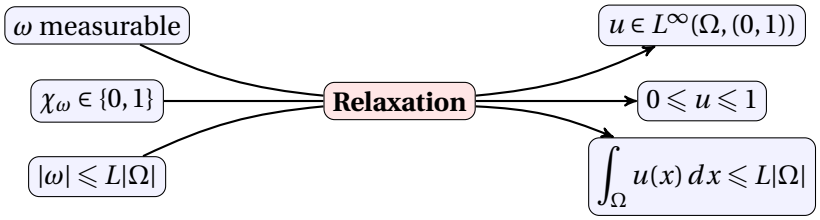
The example of Section 2.6 below is devoted to the illustration of the first method while we quickly describe the second one since it can be written in the framework of the numerical methods of Section 2.2 by considering a variant of the linear quadratic example (2.30,2.31). Let Ω be a subset of $\mathbb{R}^2$ and let $y_d \in L^2(\Omega)$ be a target function. The set of admissible controls is

$$\mathcal{U}_{ad} = \{\omega \subset \Omega \text{ measurable }, \quad |\omega| \leqslant \omega_0\}.$$

We seek a measurable domain ω solution of

$$\min_{\omega \in \mathcal{U}_{ad}} J(y,\omega) = \frac{1}{2}\int_\Omega \big(y(x) - y_d(x)\big)^2\, dx$$
$$\text{s.t.} \begin{cases} -\Delta y = \chi_\omega & \text{in } \Omega \\ y = 0 & \text{in } \partial\Omega. \end{cases}$$

We refer the reader to [63] for sufficient conditions ensuring existence of solutions. For the numerical solving, we relax the indicator function χ_ω of ω :



which leads to the problem

$$\min_{u\in\mathscr{U}_{ad}} J(y,u)=\frac{1}{2}\int_\Omega \big(y(x)-y_d(x)\big)^2\,dx$$
$$\text{s.t.}\begin{cases}-\Delta y = u & \text{in } \Omega\\ y=0 & \text{in } \partial\Omega\\ 0\leqslant u\leqslant 1\end{cases}$$

with

$$\mathscr{U}_{ad}=\left\{u\in L^2(\Omega),\quad 0\leqslant u\leqslant 1 \text{ and } \int_\Omega u(x)\,dx = L|\Omega|\right\}.$$

We now adapt the example (2.30,2.31). Shape optimization is thus performed by stipulating upper and lower bound constraints on the control u in the hope that they will be saturated for the optimal solution (if not, this means that there is relaxation). The generalization of such methods can be found in the *homogenization* method studied in [6]. The optimization solver `IpOpt` can be used.

2.6 Boundary shape optimization

The examples presented in the previous sections are rather classical and aimed at illustrating several numerical methods and providing numerical templates. In this section, our objective is to highlight other features of `FreeFEM` on a more difficult example, related to the problem of designing the best possible shape for micro-swimmers in a fluid, in order to direct them in a given direction. At the interface between fluid-structure interaction and control theory, this problem has already been studied in various forms (see [11; 98]). Here, as illustrated in Figure 2.6, we assume that the micro-swimmers have a membrane Γ that oscillates periodically at some fixed frequency to generate fluid motion. To account for the periodicity of the membrane motion, the fluid domain is assumed to be a torus obtained by bending the square along the y direction so that Σ_2 merges with Σ_4 (torus); the lower boundary Γ is assumed to move at speed v_D so that viscous forces in Γ will imply fluid motion (see Figure 2.6(b)). For ease of study, we further assume that Γ is the graph of a C^2 function (no overlap), furthermore satisfying physical constraints on the second derivative. In a more complicated model, overlap could be allowed by modifying the boundary with two-dimensional deformation vector fields. We perform our study in the reference frame of the moving boundary Γ (velocity v_D), which will induce the vertical motion of the fluid observed in Figure 2.8.

2.6.1 Boundary and domain parametrization

Let Ω_0 denote the initial torus. Given a function

$$f:[0,1]\mapsto[0,1]$$

such that $f(0)=f(1)$, we parametrize the bottom boundary as

$$\Gamma=\{(x,f(x)),x\in[0,1]\},$$

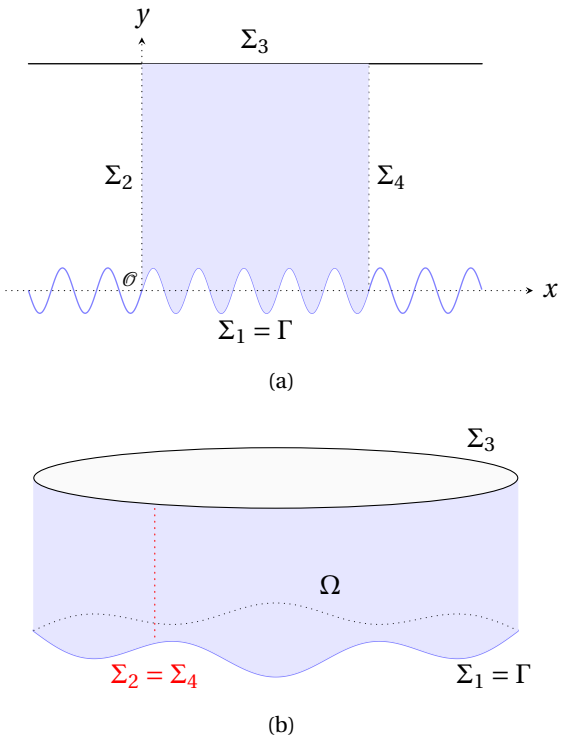


FIGURE 2.6 – Fluid domain : (a) initial square ; (b) torus after periodization.

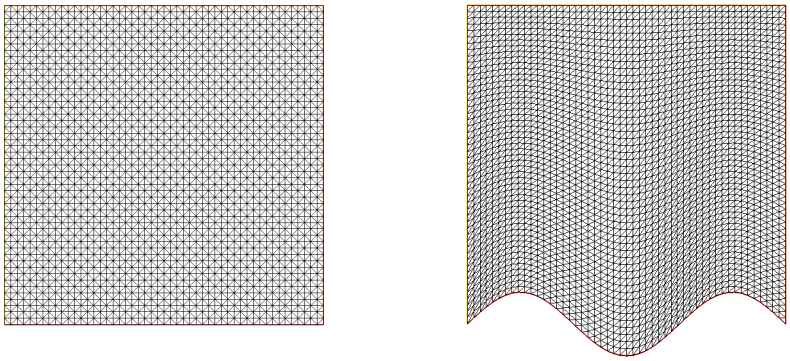
so that Ω_0 is modified to obtain the fluid domain Ω under the action of the vector field

$$\theta(x,y) = (x, y + (1-y)f(x)).$$

As for the numerical approach, although the flow problem introduced below is expressed on the modified cylinder Ω, we discretize the domain with a square mesh such that periodicity will be ensured by considering periodic finite elements. We then introduce a triangulation of Ω (see Figure 2.7(b)) by modifying the previous square mesh (see Figure 2.7(a)) under the action of the vector field θ thanks to the command `movemesh`. As the control f are on the 1D boundary, we introduce a one-dimensional mesh of the boundary which will be necessary for the computation of the gradient. Each iteration of the optimization process will involve a modification of both the square mesh and the one-dimensional mesh along *Gamma*. Numerically, in the Code 2.24, we introduce several meshes in addition to the initial square mesh of Ω_0 thanks to the command `extract`

```
int[int] ll=[1];
mesh Th0 = square(NX,NX*H,[x,y*H]); // initial square mesh
func fclab = (x>0.999)*2 + (x<0.001)*1; // Borders label
ThL = extract(Th0,refedge=ll);
ThL = change(ThL, flabel = fclab); // (0,1) straight mesh with label 1, 2

mesh Th = Th0; // mesh to be modified at each function call to get Ω
meshL ThC = ThL; // mesh to be modified at each function call to get Γ
```

Code 2.24 – 1D finite element space and movemesh command



(a) (b)

FIGURE 2.7 – Fluid domain mesh : (a) initial ; (b) modified for $f(x) = 0.1\sin(3\pi x)$.

Figure 2.7 shows the initial square mesh of Ω_0 and its modification to obtain the mesh of Ω for a particular function. The reference frame of the moving boundary follows a rectilinear uniform motion with speed v_D in the x-direction. Let $u = (u_1, u_2)$ be the velocity of the fluid and let p be the pressure. We consider the Stokes flow equations on the modified domain Ω :

$$-\mu\nabla\cdot\varepsilon(u) + \nabla p = 0 \quad \text{in } \Omega \tag{2.52}$$

$$\operatorname{div}(u) = 0 \quad \text{in } \Omega \tag{2.53}$$

$$\sigma(u,p)\vec{n} = 0 \quad \text{in } \Sigma_3 \tag{2.54}$$

$$u = \begin{pmatrix} 0 \\ f' \end{pmatrix} \quad \text{in } \Gamma \tag{2.55}$$

where $\vec{n}$ is the outer normal vector, μ is the kinematic viscosity,

$$\varepsilon(u) = \frac{1}{2}\left(\nabla u + \nabla u^T\right)$$

is the symmetric gradient of u and

$$\sigma(u,p) = 2\mu\varepsilon(u) - p\,\mathrm{Id}$$

is the Cauchy stress tensor. Assuming that

$$f \in H^2(0,1) \cap H_0^1(0,1),$$

Ω has at least a $C^{1,1}$ boundary by Sobolev injections. Existence of solutions

$$(u,p) \in H^1(\Omega) \times L^2(\Omega)$$

of the Stokes system (2.52,2.53,2.54,2.55) follows from [21, Theorem IV.5.2]. More details on existence and regularity of solutions in less regular domains can be found in [12; 72].

Remark 2.13. *The equation* (2.52) *stands for the balance law between viscous and pressure forces, while the equation* (2.53) *stands for the incompressibility of the fluid. We usually add an integral constraint* $\int_\Omega p(x)\,dx = 0$ *on the pressure in order to guarantee the uniqueness of the*

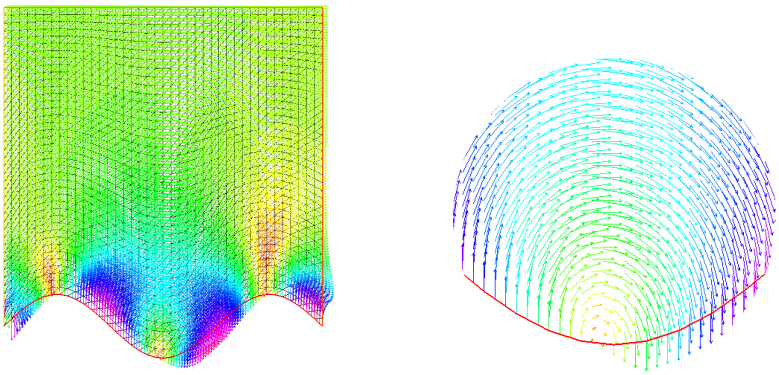


FIGURE 2.8 – Velocity of Stokes fluid u for $f(x) = 0.1\sin(3\pi x)$ and $v_D = -1$

solution (u,p)*, what is not needed here since the condition* (2.54) *yields the pressure on* Σ_3*. The equations* (2.54) *and* (2.55) *respectively imply that the fluid is free of surface forces at the top boundary* Σ_3 *and a no-slip boundary condition on* Γ*, namely the fluid and the solid have the same speed at the interface. For micro-swimmers, inertia effects play no role and the motion is entirely determined by the friction forces encompassed in* (2.55) *and therefore by the shape of* Γ*.*

Given a given boundary shape $f(x) = 0.1\sin(3\pi x)$ moving to the right ($v_D = -1$), we plot the corresponding velocity in Figure 2.8. Near the minimum of f, we observe that the slope of the Γ boundary generates a semblance of a vortex and a fluid velocity that increases with the variation of the slope. The mixed boundary conditions (2.54) and (2.55) are treated in the variational formulation by introducing the *state* finite element space

$$V = \{(u,p) \in H^1(\Omega)^2 \times L^2(\Omega), \quad u = \begin{pmatrix} 0 \\ f' \end{pmatrix} \text{ in } \Gamma\},$$

so that $(u,p) \in V$ is a weak solution of the Stokes system if and only if

$$\int_\Omega \left(\mu\varepsilon(u) : \varepsilon(v) - q\,\mathrm{div}(u) - p\,\mathrm{div}(v)\right) dx = 0 \qquad \forall (v,q) \in V_0 \tag{2.56}$$

with the *trial* functions space

$$V_0 = \{(v,q) \in H^1(\Omega)^2 \times L^2(\Omega), \quad v = 0 \text{ in } \Gamma\}.$$

As in Section 2.1.2, the solution (u,p) belongs to an affine space. The finite element space is chosen such that the discretized linear system corresponding to (2.52,2.53) is invertible. For example, one can choose the Lagrangian elements $\mathbb{P}_2$ for the velocity u and the Lagrangian elements $\mathbb{P}_1$ for the pressure p, which are known to imply an LBB condition (see [45, Chapter 2]) that guarantees this invertibility property. The variational formulation (2.56), whose state and test functions are defined on the modified domain Ω, is solved numerically in a finite element space with periodic boundary conditions via the command

```
fespace Wh(Th,[P2,P2,P1],periodic=[[2,y],[4,y]]); // [u1,u2,p], Σ2 = Σ4 .
Wh [u1,u2,p], [v1,v2,q];
```

with the following macros related to the variational formulations

```
macro SGrad(u,v) [[dx(u),0.5*(dx(v)+dy(u))],[0.5*(dx(v)+dy(u)),dy(v)]]
                 // ε(u)
macro div(u1,u2) (dx(u1)+dy(u2)) // div(u)
```

to solve (2.52,2.53,2.54,2.55) with the state equation macro stated in Code 2.25.

```
macro stokes() { // State equation's macro (2.52,2.53,2.54,2.55)
  solve Stokes( [u1,u2,p],[v1,v2,q] ) =
      int2d(Th)( 2*mu*(SGrad(u1,u2):SGrad(v1,v2)) - div(u1,u2)*q )
    - int2d(Th)( div(v1,v2)*p )
    + on(1,u1=0,u2=vdent*gm)} //
```

Code 2.25 – Stokes state equation

The 1D command $f \in H^2(0,1) \cap H^1(0,1)$ is discretized with one-dimensional finite elements $\mathbb{P}_1$ thanks to the command `fm`. Therefore, its exact numerical derivative **dx**`(fm)` is discretized with finite elements $\mathbb{P}_0$, i.e. it is piecewise constant. However, the boundary condition **on**`(1, u1=0,u2=vd*gm)` of the Code 2.25 implies that `u2` which is continuous, must be equal to `gm` which must therefore be continuous. In an effort to minimize approximation errors, we approximate in the Code 2.26 the initial $\mathbb{P}_0$ discretization of f' using the projection on $L^2(0,1)$ with $\mathbb{P}_1$ finite elements.

```
func real[int] L2regul(real[int] &X)
{
    WhL f,g,cdf;
    f[] = X;
    solve l2regul(cdf,g) = int1d(ThL)(cdf*g) // L^2(Ω) projection of dx(f) (P0)
                         - int1d(ThL)(dx(f)*g); // with P1 elements
    return cdf[];
}
```

Code 2.26 – Regularization of f' with L^2 projection with $\mathbb{P}_1$ finite elements

2.6.2 Shape optimization problem

In Section 2.6, the problem consists in maximizing the velocity of the fluid in the direction $\vec{t} = \binom{1}{0}$ with respect to the boundary Γ, by considering the cost functional

$$\widehat{J}(f) = -\int_{\Sigma_3} u \cdot \vec{t}\, ds \tag{2.57}$$

where the pair (u,p) satisfies (2.52,2.53,2.54,2.55). Some additional constraints on f are added to ensure existence of optimal solutions and also a better convergence of the optimization algorithm. The set $\mathscr{U}_{ad}$ entails *physical* constraints :

- The volume constraint
$$\int_0^1 f(x)\, dx = 0$$
implies that the domain Ω keeps a constant volume in time. It is formulated numerically by introducing the vector `Cvol` that represents the linear form
$$g \mapsto \int_0^1 g(x)\, dx$$

in the $\mathbb{P}_1$ finite element basis :

```
varf varCvol(f,g) = int1d(ThL)(1*g); // ∫_0^1 g(x) dx
real[int] Cvol = varCvol(0,WhL);
```

Code 2.27 – Volume constraint

- Bound constraints on the first derivative of f

$$|f'(x)| \leqslant M_1 \quad \forall x \in (0,1)$$

ensure existence of optimal solutions. This constraint is numerically carried out by using the interpolation matrix of the derivative operator from $\mathbb{P}_1$ finite elements to $\mathbb{P}_0$ finite elements defined in Code 2.29.

- To ensure existence of a solution $(u,p) \in H^1(\Omega) \cap L^2(\Omega)$ of the Stokes system, we assume that $f \in H^2(\Omega)$ so that Ω has a $C^{1,1}$ boundary and $f' \in H^1(0,1)$, and we assume in addition that there is a bound constraint on the second derivative of f

$$|f''(x)| \leqslant M_2 \quad \forall x \in (0,1).$$

This curvature constraint, which is physical, is important to ensure well-posedness and we observe that it induces a better convergence of the optimization algorithm.

The optimization problem (2.52,2.53,2.54,2.55,2.57) is formulated as

$$\min_{f \in \mathscr{U}_{ad}} \widehat{J}(f)$$

where $\mathscr{U}_{ad}$ is the subset of $U = H^2(0,1)$ defined by

$$\mathscr{U}_{ad} = \Big\{ f \in H^2(0,1) \cap H_0^1(0,1), \quad \int_0^1 f(x)\,dx = 0, \\ |f'(x)| \leqslant M_1,\ |f''(x)| \leqslant M_2 \ \text{ for a.e. } x \in (0,1) \Big\}$$

2.6.3 Sensitivity analysis

The function f acts on both the shape of the Ω domain (f' is involved in the boundary condition (2.55)) and the solution (u,p) of the Stokes system (2.52,2.53,2.54,2.55). If the Ω domain were fixed, we could compute the derivative of the reduced cost function directly using the adjoint representation introduced in Section 2.1.2, but here the shape deformations of Γ must also be taken into account. To this end, we revert to a calculus of variations approach and thus write a sensitivity analysis to express the derivative.

Let $f \in \mathscr{U}_{ad}$, we compute the shape derivative of the reduced cost function in the direction $g \in U$ by taking t small enough so that $f + tg \in \mathscr{U}_{ad}$. We define

$$\Omega_t = (\mathrm{id} + t\phi)(\Omega)$$

where

$$\phi(x,y) = \begin{pmatrix} 0 \\ \frac{1-y}{1-f(x)} g(x) \end{pmatrix}$$

is the vector field deforming Ω to Ω_t with respect to the small perturbation g. We define the real-valued function

$$F : t \mapsto F(t) = \widehat{J}(f + tg)$$

and compute its Fréchet derivative $F'(0)$ which gives the reduced cost function derivative

$$F'(0) = \langle D\widehat{J}(f), g\rangle_{U',U}.$$

According to usual methods for the computation of derivatives of solution of a PDE with respect to the domain (see [7] and [59, Chapter 5]) we introduce the material derivative $(\tilde{u}, \tilde{p})$ solution of the linearized system

$$-\mu\nabla\cdot\varepsilon(\tilde{u}) + \nabla\tilde{p} = 0 \quad \text{in } \Omega \tag{2.58}$$

$$\operatorname{div}(\tilde{u}) = 0 \quad \text{in } \Omega \tag{2.59}$$

$$\sigma(\tilde{u}, \tilde{p})\vec{n} = 0 \quad \text{in } \Sigma_3 \tag{2.60}$$

$$\tilde{u} = v_D\begin{pmatrix}0\\ g'\end{pmatrix} - \frac{\partial}{\partial n}\left(u - v_D\begin{pmatrix}0\\ f'\end{pmatrix}\right)\times\phi\cdot\vec{n} \quad \text{in } \Gamma. \tag{2.61}$$

The derivative of the reduced cost function $\widehat{J}$ is

$$\langle D\widehat{J}(f), g\rangle_{U',U} = \int_{\Sigma_3} \tilde{u}\cdot\vec{t}\, ds.$$

Two terms appear in (2.61), respectively coming from the boundary condition $v_D\left(\begin{smallmatrix}0\\ f'\end{smallmatrix}\right)$ and from the domain variation vector field ϕ in (2.61). In other words, the first term stands for the variation of u with respect to the boundary condition (2.55), assuming Ω fixed, while the second one stands for the mesh deformation variation. By (2.61), we thus get $\tilde{u}$ along Γ, under the influence of both terms, while the knowledge of $\tilde{u}$ along Σ_3 is required to compute $\langle D\widehat{J}(f), g\rangle_{U',U}$. The adjoint vector $(v, q) \in H^1(\Omega)^2 \times L^2(\Omega)$ is defined as the solution of

$$-\mu\nabla\cdot\varepsilon(v) + \nabla q = 0 \quad \text{in } \Omega \tag{2.62}$$

$$\operatorname{div}(v) = 0 \quad \text{in } \Omega \tag{2.63}$$

$$\sigma(v, q)\vec{n} = \vec{t} \quad \text{in } \Sigma_3 \tag{2.64}$$

$$v = 0 \quad \text{in } \Gamma. \tag{2.65}$$

Remark 2.14. *The adjoint system (2.62,2.63,2.64,2.65) is similar to the one we would get from the Stokes problem (2.52,2.53,2.54,2.55) considered on a fixed domain Ω with a Dirichlet boundary control.*

Following the variational formulation (2.56) for the numerical solving of the state equation, we solve in Code 2.28 the adjoint system (2.62, 2.63, 2.64, 2.65).

```
macro adjoint() {
    solve StokesAdjoint( [v1,v2,q],[w1,w2,g] ) =
          int2d(Th)( 2*mu*(SGrad(v1,v2):SGrad(w1,w2)) - div(w1,w2)*q )
        - int2d(Th)( div(v1,v2)*g )
        - int1d(Th,3)(w1) // Neumann condition (2.64)
        + on(1,v1=0,v2=0); // Dirichlet condition (2.65).
} //
```

Code 2.28 – Stokes adjoint system

Using the boundary condition (2.64), we have

$$\langle D\widehat{J}(f), g\rangle_{U',U} = \int_{\Sigma_3} \tilde{u}\cdot\vec{\imath}\, ds = \int_{\Sigma_3} \sigma(v,q)\vec{n}\cdot\tilde{u}\, ds.$$

The function $\tilde{u}$ is known along Γ thanks to (2.61), involving the effects of f on the derivative. To express the derivative with respect to f, we use the adjoint system (2.62, 2.63, 2.64, 2.65), in relationship with the expression of $\sigma(v,q)\vec{n}\cdot\tilde{u}$ on Γ, and on Σ_3 by making an integration by parts :

$$\int_\Omega \left(-\mu\nabla\cdot\varepsilon(u)+\nabla p\right)\cdot w\, dx = \int_\Omega \left(\mu\varepsilon(u):\varepsilon(w) - p\,\mathrm{div}(w)\right) dx - \int_{\Gamma\cup\Sigma_3} \sigma(u,p)\vec{n}\cdot v\, ds. \tag{2.66}$$

Applying (2.66) to the solution (u,p) of Stokes system, to the solution $(\tilde{u},\tilde{p})$ of the linearized system and to the solution (v,q) of the adjoint system, we finally compute the reduced cost function derivative as

$$\langle D\widehat{J}(f), g\rangle_{U',U} = \int_\Gamma \sigma(v,q)\vec{n}\cdot\left(v_D\begin{pmatrix}0\\ g'\end{pmatrix} - \frac{\partial}{\partial n}\left(u - v_D\begin{pmatrix}0\\ f'\end{pmatrix}\right)\phi\cdot\vec{n}\right) ds \tag{2.67}$$

where

$$\vec{n} = \frac{1}{(1+f'^2)^{\frac{1}{2}}}\begin{pmatrix}f'\\ -1\end{pmatrix}.$$

We identify the linear form (2.67) with a gradient expressed in $U = H^2(0,1)$ by finding, for a $H^2(0,1)$-inner product to be defined in accordance with the dual pairing $\langle\cdot,\cdot\rangle_{U',U}$, the solution $\rho\in H^2(0,1)$ of

$$(\rho, g)_{H^2(0,1)} = \int_\Gamma (\phi_1(x,y)g'(x) + \phi_2(x,y)g(x))\, ds \qquad \forall g\in H^2(0,1), \tag{2.68}$$

with

$$\phi_1 = v_D\sigma(v,q)\vec{n}\cdot\begin{pmatrix}0\\1\end{pmatrix}, \qquad \phi_2 = \sigma(v,q)\vec{n}\cdot\frac{\partial u}{\partial n}\times\phi\cdot\vec{n} - v_D\sigma(v,q)\vec{n}\cdot\begin{pmatrix}0\\1\end{pmatrix}\times f''\vec{n}_1\phi\cdot\vec{n},$$

where $\vec{n}_1$ is the x-axis component of $\vec{n}$. The above quantities are stored numerically in data types **`func`** The state (u,p) and the adjoint (v,q) are discretized with finite elements $\mathbb{P}_2\times\mathbb{P}_1$ and the control f is discretized with 1D finite elements $\mathbb{P}_1$. Since $f\in H^2(0,1)$, we do not consider here conformal transformations for U but rather nonconformal transformations, in agreement with the general mathematical framework given in the remark 2.11. Moreover, ϕ_1 and ϕ_2 are defined on the modified mesh in (2.68) and their finite element approximation in the Code 2.30 is discontinuous since they involve derivatives of f. Therefore, we introduce below several 1D finite element spaces defined respectively on the square and modified meshes `ThL` and `ThC` to interpolate ϕ_1 and ϕ_2 with finite elements $\mathbb{P}_1$ on $(0,1)$ (see Code 2.30).

```
fespace WhL(ThL,P1); // F.E. functions on straight mesh: f
fespace PhL(ThL,P0); // F.E. functions on straight mesh: f'
fespace WhC(ThC,P1); // F.E. functions on curve mesh
```

Let us give some details : `WhL` is a finite element space used to discretize f and to write the matrix of the scalar product on $H^2(0,1)$ which is required for gradient interpolation; `PhL` is

a finite element space through which to compute the first and second derivatives of f which appear in the constraints and are required to compute the resulting matrix of the second order terms involved in the $H^2(0,1)$-inner product. The matrix corresponding to the first derivative operator is constructed using the operator **dx** which thus allows to express the derivative of a finite element function $\mathbb{P}_1$ with finite elements $\mathbb{P}_0$. For the second derivative, we construct by hand the matrix of jumps of the first order derivatives of the finite element basis $\mathbb{P}_1$ `WhL` (see the Code 2.37 later).

```
matrix mDx  = interpolate(PhL,WhL,t=0,op=1); // ∂x operator: P1 to P0
matrix mDxx;
MatJumpofDx(WhL,ThL,mDxx); // returns the jumps of mDx (see Code 2.37)
```

Code 2.29 – Matrices of first and second derivatives of f

Finally, `WhC` allows us to numerically compute the integral along the curve boundary Γ involved in (2.68). In Code 2.30, we compute a regularization of ϕ_1 and ϕ_2 by performing a projection onto $L^2(\Omega)$ with $\mathbb{P}_1$ finite elements on $(0,1)$.

```
func real[int] L2regulphi(int indexphi)
{
    WhC vC,phiC;
    WhL phiL;
    func phi = (1.0-y)/(1.0-fm);
    func nx = dx(fm)*(1.0+dx(fm)^2)^(-0.5);
    func ny = -1.0*(1.0+dx(fm)^2)^(-0.5);
    if (indexphi == 1){ // for ϕ1 non continuous
        func phi1 = vdent*sigman(v1,v2,q,nx,ny)'*[0.0,1.0];
        solve l2regulphi(phiC,vC) = int1d(ThC)(phiC*vC) - int1d(ThC)(phi1*vC);
        }
    if (indexphi == 2){ // for ϕ2 non continuous
        func phi2 = -1.0*phi*ny*sigman(v1,v2,q,nx,ny)'*[dx(u1)*nx+dy(u1)*ny,dx(
            u2)*nx+dy(u2)*ny]
                +vdent*c2gm*nx*phi*ny*sigman(v1,v2,q,nx,ny)'*[0,1];
        solve l2regulphi(phiC,vC) = int1d(ThC)(phiC*vC) - int1d(ThC)(phi2*vC);
        }
    phiL = phiC;
    return phiL[];
}
```

Code 2.30 – L^2 regularization for ϕ_1 and ϕ_2

The **macro** `sigman(v1,v2,q,nx,ny)` stands for the vector $\sigma(v,q)\vec{n}$. This interpolation is necessary because we cannot accurately interpolate the discontinuous approximations of ϕ_1 and ϕ_2 defined on Γ (given by `phi1` and `phi2` in Code 2.30) with continuous $\mathbb{P}_1$ finite elements on $(0,1)$. Once this regularization is done, the interpolation from Γ to $(0,1)$ is straightforward in `FreeFEM` by typing `phi1L = phi1C`. We next compute the gradient based on the $H^2(0,1)$-inner product chosen with the macro in Code 2.31.

```
macro gradInterp(){
    varf linearform(u,v) = int1d(ThL)(-dx(phi1L)*v + phi2L*v); // (2.67)
    real bdJ = linearform(0,WhL);
    real[int] dJ = MH2^-1*bdJ; // Gradient in H^2(0,1)
} //
```

Code 2.31 – Stokes gradient's interpolation

Remark 2.15. *Numerically, we observe that the algorithm is converging in a much better way when we compute the gradient by interpolating the linear form (Code 2.31)*

$$(\rho,g)_{H^2(0,1)} = \int_0^1 (-\phi_1'^L(x)+\phi_2^L(x))g(x)\,dx \qquad \forall g\in H^2(0,1),$$
$$\textit{versus } (\rho,g)_{H^2(0,1)} = \int_0^1 (\phi_1^L(x)g'(x)+\phi_2^L(x)g(x))\,dx \qquad \forall g\in H^2(0,1),$$

where ϕ_1^L and ϕ_2^L are respectively the interpolations of ϕ_1 and ϕ_2 with $\mathbb{P}_1$ finite elements on $(0,1)$, computed in Code 2.30.

The matrix representing the inner product of $H^2(0,1)$ is constructed below with the matrix representing the inner product of $H^1(0,1)$ and the one giving the jumps of the first order derivatives of the finite element basis (see the Code 2.29)

```
// H^2(0,1)-inner product
varf scalarH1(u,v) = int1d(ThL)(u*v+dx(u)*dx(v));
matrix MH1 = scalarH1(WhL,WhL); // (ρ,g)_{H^1(0,1)}
matrix mDxxL = mDxx'*mDxx;
matrix MH2 = MH1 + mDxxL; // (ρ,g)_{H^2(0,1)}
set(MH2,solver=sparsolver);
```

Code 2.32 – $H^2(0,1)$-inner product's construction

Remark 2.16. *As stated in [27; 73], we may consider a weighted version of the usual $H^2(0,1)$-inner product*

$$(f,g)_{H^2(0,1)} = \int_0^1 (\alpha^2 f''g'' + f'g' + fg)\,dx,$$

with $\alpha > 0$ to be tuned depending on the mesh.

2.6.4 Codes and results

We finally write the complete algorithm with the several macros and functions defined above and we first initialize the numerical framework based on Section 2.6 to solve (2.52, 2.53, 2.54, 2.55).

```
load "msh3"
load "gsl"
load "ff-Ipopt"

bool hotrestart = 1;
verbosity=0;
real HUB = 10.0; // |f''(x)| ⩽ M2
real CUB = 0.5; // |f'(x)| ⩽ M1
real XUB = 0.2;
real H = 1; //
int NX = 25; // boundary mesh size
real vdent = -1.0; // boundary's speed

int[int] ll=[1];
mesh Th0 = square(NX,NX*H,[x,y*H]);
func fclab = (x>0.999)*2 + (x<0.001)*1; // Borders label
meshL ThL = extract(Th0,refedge=ll);
ThL = change(ThL, flabel = fclab); // (0,1) Straight Mesh with label 1, 2

mesh Th = Th0;
meshL ThC = ThL; // Curve Mesh for Γ

fespace WhL(ThL,P1);
fespace PhL(ThL,P0);
fespace WhC(ThC,P1);
fespace Wh(Th,[P2,P2,P1],periodic=[[2,y],[4,y]]);

WhL fm,phi1L,phi2L,cgm,gm;
WhL c2gm;
```

With the macros and functions introduced in previous sections, we are now able to compute a numerical version of the cost function. We first solve the PDE stated in Code 2.25 and then use the velocity u to compute the cost function (2.57) according to a quadrature formula on the boundary Σ_3 :

```
func real J(real[int] &X)
{
    fm[] = X;
    Th = movemesh(Th0,[x,fm+y*(H-fm)/H]);

    cgm[] = L2regul(fm[]); // Code 2.26

    stokes(); // Code 2.25
    return = -int1d(Th,3)(u1);
}
```

Code 2.33 – Cost function for Stokes problem

Besides, we follow the sensitivity analysis made before to compute the derivative of the previous cost function. We saw that it involves both state and adjoint macros defined in Codes 2.25 and 2.28. Thus, we beforehand solve state equation stated in Code 2.25 that returns both the pressure p and the velocity u that are needed for the adjoint's computations via Code 2.28. Once we solved the adjoint's equation, we proceed to a regularization of the functions involved in the equation (2.67) and finally express the gradient according to the chosen inner-product :

```
func real[int] dJ(real[int] &X)
{
    fm[] = X;

    Th = movemesh(Th0,[x,fm+y*(H-fm)/H]); // Ω0 ↦ Ω
    ThC = movemesh(ThL,[x,fm+y*(H-fm)/H]); // (0,1) ↦ Γ

    cgm[] = L2regul(fm[]); // Code 2.26

    stokes(); // Code 2.25
    adjoint(); // Code 2.28

    phi1L[] = L2phiregul(1); // Code 2.30
    phi2L[] = L2phiregul(2); // Code 2.30

    gradInterp(); // Code 2.31
    return dJ;
}
```

Code 2.34 – Derivative of the cost function for Stokes problem

Before using the gradient interpolation macro written in Code 2.31, we perform a regularization of the functions ϕ_1 and ϕ_2 (Code 2.30). The constraint introduced in Section 2.6.2 and its Jacobian are computed with the matrices introduced in Code 2.29 and read :

```
func real[int] C(real[int] &X)
{
    real[int] cont(1+PhL.ndof+WhL.ndof);
    cont[0] = Cvol'*X; // ∫_0^1 f(x) dx = 0
    cont(1:PhL.ndof) = mDx*X; // |f'(x)| ⩽ M1
    cont(PhL.ndof+1:PhL.ndof+WhL.ndof) = mDxx*X; // |f''(x)| ⩽ M2
    return cont;
}

matrix dc;
func matrix jacC(real[int] &X)
{
    real[int,int] dcc(1,WhL.ndof); dcc = 0.0;
    dcc(0,:) = Cvol;
    dc = dcc;
    dc = [[dc],[mDx]];
    dc = [[dc],[mDxx]]; // [volume, |f'| ⩽ M1, |f''| ⩽ M2 ]
    return dc;
}
```

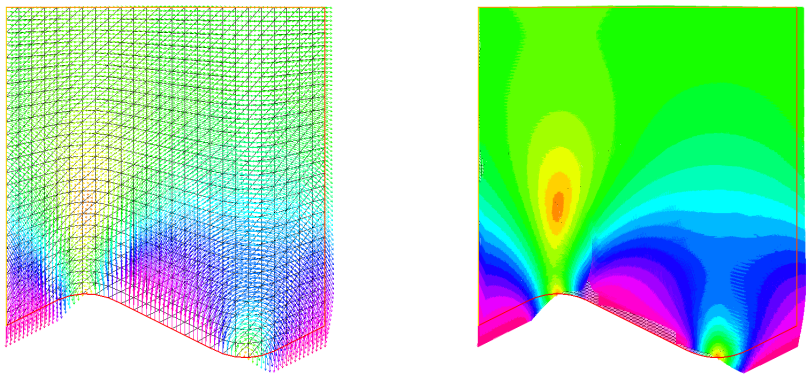
Code 2.35 – Length and curvature constraints of the boundary

In `FreeFEM`, the possibility use matrices and arrays instead of solving the variational formulation usually guarantees a more efficient algorithm in terms of execution speed and required memory. We wrote most of macros and functions with the explicit formulation for the sake of clarity, but execution is quicker when dealing with matrices. Finally, it remains to call the optimization routine `IpOpt` :

```
real[int] start(WhL.ndof);
real[int] xub(WhL.ndof);
real[int] xlb(WhL.ndof);
real[int] cub(1+PhL.ndof+WhL.ndof);
real[int] clb(1+PhL.ndof+WhL.ndof);

// Unknowns bounds
xub=  XUB;
xlb= -XUB;
cub(1:PhL.ndof)=  CUB;
clb(1:PhL.ndof)= -CUB;
cub(PhL.ndof+1:PhL.ndof+WhL.ndof)=  HUB;
clb(PhL.ndof+1:PhL.ndof+WhL.ndof)= -HUB;

xub[0] = 0.0; // f(0) = 0
xlb[0] = 0.0; // f(0) = 0
xub[WhL.ndof-1] = 0; // f(1) = 0
xlb[WhL.ndof-1] = 0; // f(1) = 0
clb[0] = 0.0; // ∫_0^1 f(x) dx = 0
cub[0] = 0.0; // ∫_0^1 f(x) dx = 0

clb[PhL.ndof+WhL.ndof] = -HUB/alpha;
cub[PhL.ndof+WhL.ndof] = HUB/alpha;
clb[PhL.ndof+1] = -HUB/alpha;
cub[PhL.ndof+1] = HUB/alpha;

// intialization
WhL X0=0.0125/2*sin(x*pi*2*2);
if (hotrestart){
start = HOTRESTART(hotrestart); }

IPOPT(J,dJ,C,jacC,start,lb=xlb,ub=xub,clb=clb,cub=cub,tol=1.e-8);
```

Code 2.36 – Calling `IpOpt` in the Stokes problem

The optimal solution returned by `IpOpt` is plotted in Figure 2.9 for a rough and for a fine mesh.



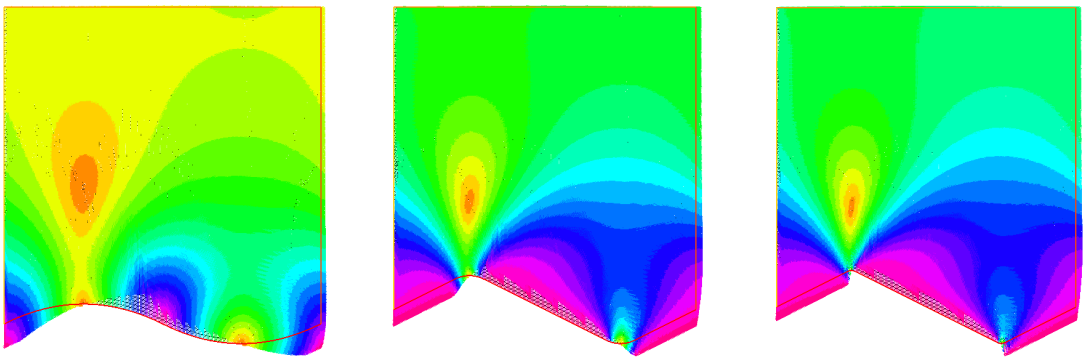
(a) (b)

FIGURE 2.9 – Optimal solution of (2.52,2.53,2.54,2.55,2.57) for $M_1 = 0.4$ and $M_2 = 5.0$ on : (a) rough mesh ; (b) fine mesh.

As mentioned above, the constraints on the first and second derivatives of f have been introduced to guarantee well-posedness and a good numerical convergence of the optimization process. As expected, the optimal solution saturates these constraints. Since PDE optimization problems involve a large number of variables, iterations of the algorithm require a larger computational time as the mesh is finer. Having a good initial guess is also important. Hot-restart loops turn out to be effective by providing a better initialization that we refine by first running the algorithm on a rougher mesh and then on a finer one. An interpolation on the finer mesh of the solution obtained on the rough mesh is taken as a new initialization, in order to make the algorithm converge in a better way on the fine mesh. The hot-restart procedure is described in Appendix 2.A.

2.6.5 Further comments

We plot in Figure 2.10 the optimal solutions for various values of M_2. We observe that the value of the cost functional at the optimal solution increases when M_2 is taken larger and that, as $M_2 \to +\infty$, the sequence of optimal solutions seems to converge to a triangular-shaped function, which may be the optimal solution of the problem when, formally, $M_2 = +\infty$, i.e., f varies in $H^1(0,1)$ instead of $H^2(0,1)$ without any constraint on f''. However, although this limit problem seems to be tractable from the numerical point of view, treating it rigorously from the theoretical point of view is much more difficult because existence of solutions of the Stokes problem (2.52, 2.53, 2.54, 2.55) is not guaranteed in $H^1(\Omega) \cap L^2(\Omega)$: it is required to consider other functional spaces (see [12; 72]) and the framework becomes much more complicated. We leave this issue as an open, interesting problem.



(a) (b) (c)

FIGURE 2.10 – Optimal solution of (2.52,2.53,2.54,2.55,2.57) for $M_1 = 0.4$ and : (a) $M_2 = 2$ and $J \approx 0.07$; (b) $M_2 = 10.0$ and $J \approx 0.16$; (c) $M_2 = 50.0$ and $J \approx 0.19$;

Numerically, we can proceed as follows. Ignoring the second derivatives of f involved in the constraint function `C` and in Codes 2.31 and 2.32, the control f now varies in the subset of $U = H^1(0,1)$

$$\mathcal{U}_{ad} = \left\{ f \in H_0^1(0,1), \quad \int_0^1 f(x)\,dx = 0 \text{ and } |f'(x)| \leqslant M_1, \text{ for a.e. } x \in (0,1) \right\}.$$

The optimal solution returned by `IpOpt` is plotted in Figure 2.11.

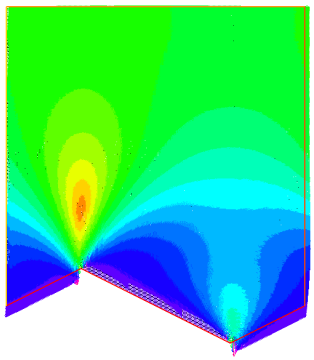


FIGURE 2.11 – Solution with no curvature constraint ($M_2 = +\infty$) and $J \approx 0.196$

Appendix

2.A Some FreeFEM functions

The $L^2(\Omega)$ norm for functions defined on a finite element space V_h is coded as follows :

```
func l2norm(real[int] &X)
{
    // Th,Vh already constructed
    Vh u,v;
    varf varl2norm(u,v) = int2d(Th)(u*v);
    matrix M = varl2norm(Vh,Vh); // better to be constructed outside
    real[int] uu = M*u[];
    l2error = sqrt(X'*uu); // = (∫_Ω u(x)^2 dx)^1/2
    return l2error;
}
```

The macro to build the matrix of jumps of the derivatives of the finite element basis functions is :

```
macro MatJumpofDx(Vh,Th,A)
{
    if (A.n) A.clear;
    matrix Adx(Vh.ndof,Th.nt);
    fespace Ph(Th,P0);
    matrix Dx = interpolate(Ph,Vh,op=1);
    assert(Vh.ndofK==2);
    int nt = Th.nt;
    for(int k=0; k< nt;++k)
    {
     Adx(Vh(k,0),k)=+1;
     Adx(Vh(k,1),k)=-1;
    }
    A = Adx*Dx;
} //
```

Code 2.37 – Matrix of derivative's jumps

Hot-restart requires first to use the option `warm_start_init_point yes` in the `IpOpt` option file *"optfile.opt"*. It is assumed that the code has already been run so that the files

"Th0old.msh" and "fsol.txt" already exist. Then, it suffices to add the following lines at end of the code :

```
if (hotrestart == 0){
    savemesh(Th0,"Th0old.msh");
    {
    ofstream file("fsol.txt");
    file << X0[];
    }
}
```

The interpolation of a solution or of an initialization function only requires to provide a new mesh `ThL` and a finite element space `WhL` with the following routine :

```
func real[int] HOTRESTART(bool &hot)
{
    if (hot){
        WhL Xinit;
        mesh Th00 = readmesh("Th0old.msh"); // Initial mesh
        meshL ThL0 = extract(Th00,refedge=11);
        ThL0 = change(ThL0, flabel = fclab); // Straight Mesh for hotrestart
        fespace WhL0(ThL0,P1);
        WhL0 X00;
        {
        ifstream file("fsol.txt");
        file >> X00[];
        }
        Xinit = X00; // interpolation on the new mesh
        return Xinit[];
    }
}
```

Code 2.38 – Hot restart routine

Ho-restart is very easy to perform in `FreeFEM` thanks to the facility of interpolating from one given mesh to another one by just typing `ustart=uh1` with `ustart` and `uh1` respectively defined on the meshes `Th2` and `Th1`.

2.B Semi-automatic differentiation and adjoint method

In this section, we show that the adjoint code of the reduced cost functional of the problem (2.29,2.28) implicitly makes appear a discretization of the adjoint equation obtained by the application of the Pontryagin maximum principle (see [89, Chapter 7]) :

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} + \begin{pmatrix}u\\ v\end{pmatrix} \qquad \begin{pmatrix}x(0)\\ y(0)\end{pmatrix} = \begin{pmatrix}1\\ 1\end{pmatrix}, \tag{2.69}$$

$$\begin{pmatrix}\dot{p}\\ \dot{q}\end{pmatrix} = \begin{pmatrix}-1 & -1\\ -1 & 1\end{pmatrix} + \begin{pmatrix}x-1\\ 0\end{pmatrix} \qquad \begin{pmatrix}p(T)\\ q(T)\end{pmatrix} = \begin{pmatrix}0\\ 0\end{pmatrix}, \tag{2.70}$$

$$\int_0^T \big(p(u-\phi)+q(v-\psi)\big)\,dt \leqslant 0 \quad \forall (\phi,\psi)\in \mathscr{U}_{ad}. \tag{2.71}$$

The variational inequality (2.71) comes from the maximization condition of the Hamiltonian of the optimal control problem and makes appear the duality pairing as the L^2-inner product

$$\left(D\widehat{J}\binom{u}{v}, \binom{\phi}{\psi}\right)_{L^2(0,T)} = -\int_0^T \left(p\phi + q\psi\right) dt$$

involving the derivative of the reduced cost functional, which thus yields its gradient $\nabla\widehat{J}\binom{u}{v} = -\binom{p}{q}$. The reduced cost functional $\widehat{J}$ obtained by solving (2.69) is computed thanks to an implicit Euler discretization (with n time steps) with the matrices

$$A = \begin{pmatrix} 1-\delta t & -\delta t \\ -\delta t & \delta t + 1 \end{pmatrix}, \quad \text{with } \delta t = \frac{1}{n-1}$$

and

$$A^{-1} = \frac{1}{1-2\delta t^2}\begin{pmatrix} \delta t + 1 & 1 \\ 1 & 1-\delta t \end{pmatrix} = \mu\begin{pmatrix} a & 1 \\ 1 & b \end{pmatrix}$$

so that (2.69) and (2.70) are respectively discretized according to implicit and explicit schemes as

$$\forall k \in \{0...n-2\}, \quad \binom{x}{y}^{k+1} = A^{-1}\binom{x}{y}^{k} + \delta t A^{-1}\binom{u}{v}^{k+1} \qquad \binom{x}{y}^{0} = \binom{1}{1}, \tag{2.72}$$

$$\forall k \in \{0...n-2\}, \quad \binom{p}{q}^{k+1} = A\binom{p}{q}^{k} + \delta t\begin{pmatrix} x_k - 1 \\ 0 \end{pmatrix} \qquad \binom{p}{y}^{n-1} = \binom{0}{0}. \tag{2.73}$$

The cost (2.28) is then computed by discretization according to a rectangle rule :

```
1  def J(u,v):
2      x[0]=0,y[0]=0,cost=0 # initialization
3      a = 1+dt, b = 1-dt
4      mu = 1/(1-2*dt**2)
5
6      for k in range(1,n-1): # dynamic loop
7          x[k] = mu*(a*x[k-1]+y[k-1]) + dt*mu*(a*u[k]+v[k])
8          y[k] = mu*(x[k-1]+b*y[k-1]) + dt*mu*(u[k]+b*v[k])
9          cost+= 0.5*(x[k]-1)**2
10
11     return cost
```

Following the successive steps for adjoint code generation presented in Section 2.1.5, we write the derivative of the previous function obtained by automatic differentiation in reverse mode. To do so, some adjoint variables l_x, l_y, l_u, l_v, l_c are introduced, respectively related to the variables $x, y, u, v, cost$. The key is to range from the final step back to the first steps and to derive the computation line with respect to the variable involved and to finally update the corresponding adjoint variable. For example, the line 8

```
(y[n-k]=mu*(x[n-k-1]+b*y[n-k-1])+dt*mu*(u[n-k]+b*v[n-k]))
```

will generate lines 23 to 26 for updating the variables

```
(lx[n-k-1],ly[n-k-1],lu[n-k-1],lv[n-k-1])
```

```
def gradJ(u,v):
    x[0]=0,y[0]=0,cost=0 # initialization
    a = 1+dt, b = 1-dt
    mu = 1/(1-2*dt**2)

    for k in range(1,n-1): # dynamic loop
        x[k] = mu*(a*x[k-1]+y[k-1]) + dt*mu*(a*u[k]+v[k])
        y[k] = mu*(x[k-1]+b*y[k-1]) + dt*mu*(u[k]+b*v[k])
        cost+= 0.5*(x[k]-1)**2

    lx = 0, ly = 0, lc = 0 # arrays of size n
    lc[n-1] = 1 # reverse mode initialization

    for k in range(1,n-1):
        lc[n-k-1] += lc[n-k]             # line 9: cost[n-k]
        lx[n-k] += lc[n-k]*(x[n-k] - 1)

        lx[n-k-1] += mu*ly[n-k]       # line 8: y[n-k]
        ly[n-k-1] += b*mu*ly[n-k]
        lu[n-k] += dt*mu*ly[n-k]
        lv[n-k] += dt*b*mu*ly[n-k]

        lx[n-k-1] += a*mu*lx[n-k]     # line 7: x[n-k]
        ly[n-k-1] += mu*lx[n-k]
        lu[n-k] += dt*a*mu*lx[n-k]
        lv[n-k] += dt*mu*lx[n-k]

    return lu,lv # Gradient
```

Having updated the adjoint variables, we have, for every $k \in \{0..n-2\}$,

$$\begin{pmatrix} l_x \\ l_y \end{pmatrix}^k = A^{-1} \begin{pmatrix} l_x \\ l_y \end{pmatrix}^{k+1} + \begin{pmatrix} x_k - 1 \\ 0 \end{pmatrix}$$
$$\begin{pmatrix} l_u \\ l_v \end{pmatrix}^k = \delta t A^{-1} \begin{pmatrix} l_x \\ l_y \end{pmatrix}^k$$

which gives, considering the variables l_u, l_v,

$$\begin{pmatrix} l_u \\ l_v \end{pmatrix}^k = A^{-1} \begin{pmatrix} l_u \\ l_v \end{pmatrix}^{k+1} + \delta t A^{-1} \begin{pmatrix} x_k - 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} l_u \\ l_v \end{pmatrix}^{n-1} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{2.74}$$

Since $\begin{pmatrix} p \\ q \end{pmatrix}^k = -\begin{pmatrix} l_u \\ l_v \end{pmatrix}^k$, we recognize an implicit Euler time discretization of (2.70), like for the state equation (2.69) in the backward direction which is to be compared with the explicit discretization (2.73) in forward direction. The derivative of the reduced cost function is therefore expressed as $-(p_0, ..., p_{n-1}, q_0, ..., q_{n-1})$ according to the chosen discretization of the integral in (2.28,2.71).

A Crank-Nicolson scheme could have been chosen in (2.69). This would give a matrix A such that $A^{-1} = A$ and a Crank-Nicolson scheme for (2.70) too. We refer to [87] for a discussion on the relationship of automatic differentiation in reverse mode with derivative computation using adjoint variables, and resulting appropriate choices of discretization schemes.

2.C PDE Optimization with Python or Matlab

Section 2.1.5 was devoted to automatic differentiation within AMPL for PDE constrained problems. Possible alternatives are either to write our own adjoint in C++ code by calling routines like CppAD and Adept or to export the problem in Python or Matlab by using the CasADi package. Loading data into Python is done by :

```
ndof = int(numpy.loadtxt('ndof.txt'))
iA, jA, AM = numpy.loadtxt('A.txt', unpack=1, skiprows=3);
iM, jM, MM = numpy.loadtxt('B.txt', unpack=1, skiprows=3);
ydArray = numpy.loadtxt('target.txt',unpack=1,skiprows=1);

Ac = scipy.sparse.coo_matrix((AM, (iA, jA)), shape=(ndof,ndof));
Mc = scipy.sparse.coo_matrix((MM, (iM, jM)), shape=(ndof,ndof));
```

Code 2.39 – Sparse matrices importation in Python

The three first lines stand for the characteristics of the matrices and are skipped by using the option skiprows=3 in the numpy.loadtxt command. The optimization package CasADi (see [13]) is then called and the problem is written as a nonlinear programming problem via the toolbox

```
import casadi as ca
opti = ca.Opti()
```

Declaring the variables is done by using the MX symbolics (assuming that most of operations between multiple sparse-matrix valued objects are allowed) whose description is available on the CasADi website.

```
A = ca.MX(Ac); # convert sparse matrices above in MX object
M = ca.MX(Mc);
yd = ca.MX(ydArray) # target y_d in (2.30)

y = opti.variable(ndof) #Unknowns declaration
u = opti.variable(ndof)
```

It remains to express the cost and constraint functions. Matrix multiplication Ax is done either by mtimes(A,x) or A @ x commands. Most of possible operations are described in [2, Docs tab].

```
opti.minimize( 0.5*(y.T-yd.T) @ M @ (y-yd) + 0.5*alpha*u.T @ M @ u ) # (2.30)
opti.subject_to( A @ y - M @ u == 0.0 ) # PDE constraint function (2.36)
opti.subject_to( opti.bounded(0,Mu,1) ) # Bounds on control 0 ⩽ u ⩽ 1

opti.solver('ipopt',{'ipopt':{'max_iter':50, 'tol':1.e-11,
            'Hessian_approximation':'limited-memory'}})

sol = opti.solve()
```

Code 2.40 – CasADi template for LQ PDE Optimization

Computing the Hessian by automatic differentiation is in general too much memory greedy and too long in the execution, and BFGS thus appears as an alternative by using the IpOpt option 'Hessian_approximation':'limited-memory'. The previous code is written in the setting of **Option 1**. But **Option 2** can also be considered :

```
u = opti.variable(ndof) # only control as variable
us = M @ u
y = solve(A,us) # Solve state equation

cost = 0.5*(y.T-yd.T) @ M @ (y-yd) + 0.5*alpha*u.T @ M @ u
F = Function('F',[u],[cost]) # way to declare u↦Ĵ(u)

opti.minimize( F(u) )
opti.subject_to( opti.bounded(0,Mu,1) )
```

Chapter 3

Numerical solving of time-varying shape design problems

Table of contents

3.1 Numerical shape design for parabolic PDE models 108
3.1.1 Particular case of dimension 1 108
3.1.2 Level-set method in dimension $d \geqslant 2$ 110
3.1.3 Convexification method 113
3.1.4 Discussion : comparison of implemented methods 115
3.2 Numerical examples illustrating the turnpike phenomenon 121
3.3 Further prospects 125
3.3.1 Symmetries of solutions 125
3.3.2 Numerical extension to semilinear PDEs 125

Abstract In this chapter, we illustrate how we perform the numerical solution of the two problems $(\boldsymbol{DSD})_{\mathbf{T}}$ et $\boldsymbol{SSD}$. We first focus on the one-dimensional case using the software `AMPL` combined with `CPLEX`. Then, in a larger dimension, we first focus on the stationary case and run both *level-set* method and a *convexification* method based on the relaxed $(\boldsymbol{OCP})_{\mathbf{T}}$ problem using the framework introduced in the 2 chapter. We then compare the two algorithms and choose the most efficient one to solve the time dependent problems. Finally, we illustrate the turnpike phenomenon with several examples.

3.1 Numerical shape design for parabolic PDE models

In the Chapter 1, we have been interested in linear parabolic PDEs with distributed control seen as the indicator function of a certain time-varying shape to be optimized in order to minimize a quadratic criterion. In this chapter, we are interested in how to solve it numerically. We recall the introduced dynamic optimal shape design problem which always reads for $y_0, y_d \in L^2(\Omega)$ arbitrary

$$\begin{aligned} \min J_T(\omega(\cdot)) &= \frac{1}{2T}\int_0^T \|y(t)-y_d\|^2_{L^2(\Omega)}\,dt \\ \text{subject to : } \partial_t y + Ay = \chi_{\omega(\cdot)}, &\qquad y_{|\partial\Omega}=0, \qquad y(0)=y_0. \end{aligned}$$

As mentioned earlier, A always represents a uniformly elliptic second order differential operator with Dirichlet boundary conditions. Homogeneous conditions were considered in the Chapter 1 and inhomogeneous or Robin conditions can be taken instead by following the alternatives outlined in the Chapter 2, in the 2D and 3D cases. We further write its associated stationary optimal shape design problem (***SSD***)

$$\begin{aligned} \min J(\omega) &= \frac{1}{2}\|y-y_d\|^2_{L^2(\Omega)} \\ \text{subject to : } Ay = \chi_\omega, &\qquad y_{|\partial\Omega}=0. \end{aligned}$$

The computation of the cost function is possible provided that we solve the PDE (1.3). We discuss the possibilities of numerical solution of such problems which will then lead us to choose one among those studied. When the dimension is strictly greater than one, we will prefer to use the variational formulation and the finite element approach highlighted in the Chapter `FreeFEM`chap :PDEffem. Using `FreeFEM`, we will consider a mesh of the domain Ω, a finite element space with Lagrangian elements $\mathbb{P}_1$ and discretize (***SSD***) according to this triangulation. However, we first present the particular case of dimension 1 where the optimization treatment is performed with the software `AMPL`. Then, we will present chronologically the different ways that we have privileged concerning the stationary shape optimization problem and their more or less good extensibility to the time dependent problem.

3.1.1 Particular case of dimension 1

We first regard a variant to $(\boldsymbol{DSD})_{\mathbf{T}}$ by taking the Laplacian in dimension 1 which is no other than to derive twice in space on the segment $(0,L)$

$$\min\ J_T(\omega(\cdot)) = \frac{1}{2T}\int_0^T\int_0^L \big(y(t,x)-y_d(x)\big)^2\,dxdt \tag{3.1}$$

$$\text{subject to : } \begin{cases} \partial_t y - \partial_{xx} y = \chi_{\omega(\cdot)} & \forall (t,x)\in(0,T)\times(0,L) \\ y(t,0)=y(t,L)=0, & \forall t\in(0,T) \\ y(0,x)=y_0, & \forall x\in(0,L). \end{cases} \tag{3.2}$$

From the numerical point of view, we choose implicit and centered finite differences scheme on uniform subdivision of both segments $(0,L)$ and $(0,T)$ with respectively n_x+1 and n_t+1

points such that following approximation of derivatives read

$$\partial_t y(t_i,x_j) = \frac{1}{\delta_t}(y(t_{i+1},x_j) - y(t_i,x_j))$$
$$\partial_{xx} y(t_i,x_j) = \frac{1}{\delta_x^2}\left(y(t_i,x_{j+1}) - 2y(t_i,x_j) + y(t_i,x_{j-1})\right)$$

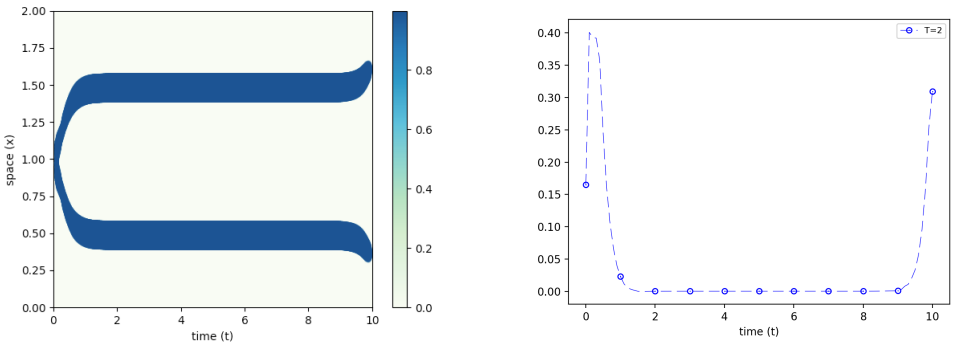
with $\delta_t = \frac{T}{n_t}$ and $\delta_x = \frac{L}{n_x}$ and lead to finally discretize (3.2) as

$$\frac{1}{\delta_t}(y_j^{i+1} - y_j^i) = \frac{1}{\delta_x^2}\left(y_{j+1}^{i+1} - 2y_j^{i+1} + y_{j-1}^{i+1}\right) + \chi_{\omega,j}^{i+1}, \quad \forall (i,j) \in \{0..n_t-1\} \times \{1..n_x-1\},$$

with $y_j^i = y(t_i,x_j)$ and $\chi_{\omega,j}^i = \chi_\omega(t_i,x_j)$. Since $\chi_{\omega(\cdot)}$ takes values into $\{0,1\}$ for all $t \in (0,T)$, we aim to specify that we deal with integer lower than 1 an upper to 0 for the quantities $\chi_{\omega,j}^{i+1}$. In addition, we discretize the cost function by using trapezoidal formula

$$\sum_{(i,j)\in\{0..n_t-1\}\times\{1..n_x-1\}} (y_j^i - (y_d)_j)^2 + (y_{j+1}^i - (y_d)_{j+1})^2 + (y_j^{i+1} - (y_d)_j)^2 + (y_{j+1}^{i+1} - (y_d)_{j+1})^2).$$

The optimization process is made through `AMPL` software endowed with the `CPLEX` solver (see [23]). `CPLEX`, which is originally built to solve linear programming problems with either integer or real variables, is well-performing for mixed-integer quadratic programming so that the variables $\chi_{\omega,j}^i$ are searched for being either 0 or 1. Moreover, we solve the stationary problem using the same discretization. An alternative could be to deal with the solver `IpOpt`, which does not handle integer variables but a continuous version of them. We therefore specify to `AMPL` that the variable takes values in $(0,1)$ instead, which amounts to reformulating the problem in its relaxed form presented in Chapter 1. We ran both methods and observed the same solution in output, which we expected since the assumptions of the Theorem 1.1 are satisfied. We plot in Figure 3.1 the time-varying optimal shape and the error between the static and optimal triples in time. The final time has been set to $T = 10$, which is large enough to observe that the time-varying shape is mostly stationary. The error between the two optimal triples leads us to conjecture that an exponential turnpike property should be found theoretically. The ability



(a) (b)

FIGURE 3.1 – (a) optimal time-varying shape; (b) $t \mapsto \|y_T(t) - \bar{y}\| + \|p_T(t) - \bar{p}\| + \|\chi_{\omega_T(t)} - \chi_{\bar{\omega}}\|$

to deal with integer variables in AMPL has greatly facilitated the handling of the χ_{omega} indicator function. When dealing with higher dimensions, we will prefer the finite element methods for PDE constrained optimization introduced in the Chapter 2. We will first treat the stationary problem to illustrate the different numerical approaches used and we will conclude on their possible generalization to the dynamic case. To solve numerically (***SSD***), two methods can be distinguished : the level-set method combined with a gradient descent algorithm and the convexification method combined with the interior point optimization method IpOpt. While the level-set method deals directly with deformation fields of shapes, the convexification introduces a more general problem whose solution is supposed to be a shape under the assumptions of the Theorem 1.1.

3.1.2 Level-set method in dimension $d \geqslant 2$

Let the stationary problem (***SSD***) read

$$\min_{\omega \in \mathscr{U}_L} J(\omega), \qquad C(\omega) \leqslant 0.$$

The optimization procedure here relies on a gradient descent method whose descent direction θ is sought in the set $W^{1,\infty}(\mathbb{R}^2,\mathbb{R}^2)$. A shape sensitivity analysis is first performed by moving an initial shape ω_0 by the vector field $I+\theta$ which is a diffeomorphism of $\mathbb{R}^2$ into $\mathbb{R}^2$ under the assumption that θ is sufficiently "small". The shape derivatives of the cost and constraint functions are thus written in the Fréchet sense by considering the derivative with respect to a real variable of the function $J(I+t\theta(\omega))$ when $t=0$ (see [59, Chapter 5]). So we have

$$J((I+\theta)(\omega)) = J(\omega) + DJ(\omega)(\theta) + o(\theta) \text{ with } \lim \frac{o(\theta)}{\|\theta\|_{W^{1,\infty}(\mathbb{R}^2)}} = 0$$

with $DJ(\omega)$ denoting a continuous linear form on $W^{1,\infty}(\mathbb{R}^2)$. The above shape sensitivity analysis allows us to find a relevant vector field θ to decrease the objective function by moving the initial shape ω to the new $(I+\theta)\omega$. In the case of dimensions greater than two, the finite element method is more suitable, both for managing the construction and modification of meshes and for solving PDEs. Thanks to the tools presented in the Chapter 2, we introduce a mesh of the considered domain Ω in (***SSD***) and a finite element space $\mathbb{P}_1$. A very intuitive but costly procedure would be to directly process the modifications of the mesh with the vector field θ found with the help of the previous shape sensitivity analysis. At each iteration, this vector field plays the role of a descent direction and will modify the mesh in such a way that a remeshing of the initial mesh is performed. This can be very expensive, especially in dimension 3. Moreover, we have not put any constraints on the topology of the admissible domains since we are looking for optimal solutions in the domain of measurable sets with a volume constraint. These topology changes are not well handled by deformation methods based on Hadamard boundary variations. Thus a first idea to solve the stationary problem was to proceed to shape deformations thanks to the level-set method. For this purpose, the domain of study Ω is a subset of R^d, for $d \in \{2,3\}$ and meshed only once (it can however happen to remesh at some isolated iterations). According to [9], the shape to be optimized and its boundary are respectively considered as the

upper level-set and the level-set of a function defined in Ω.

$$\begin{cases} \phi(x) < 0 & \iff x \in \omega \\ \phi(x) = 0 & \iff x \in \partial\omega \cap \Omega \\ \phi(x) > 0 & \iff x \in \Omega \backslash \bar{\omega}. \end{cases} \tag{3.3}$$

Along the optimization procedure, the shape is going to evolve at the same time as ϕ which is governed by a Hamilton-Jacobi equation. At a given iteration and for a vector field θ beforehand determined, the motion of the shape ω is determined over an interval $(0,s)$ by the level-set of a function $\phi(t,x)$ defined on $(0,s)$ and solution of

$$\begin{aligned} \frac{\partial \phi}{\partial t} + \theta \cdot \nabla \phi = 0, \quad \forall t \in (0,s), x \in \Omega \\ \phi(0,\cdot) = \phi^0 \end{aligned} \tag{3.4}$$

where ϕ^0 is the level function at the previous iteration and s is an adjustable parameter for the step size depending on the chosen descent direction. The main advantages are the ease of computing the normal n to the shape ω (which is usually involved in computing the shape derivative) which is rewritten as a function of ϕ as $n = \nabla\phi/|\nabla\phi|$ and the ability to keep the mesh constant throughout the optimization procedure. A less intuitive advantage is the easy but still partial treatment of topological changes as mentioned in [96]. The cost and constraint functions denote respectively the distance L^2 to the target function y_d and the volume constraint

$$J(\omega) = \frac{1}{2}\int_\Omega (y_\omega - y_d)^2 \, dx \qquad C(\omega) = \int_\omega 1 \, dx - L|\Omega|,$$

for some $\omega \in \mathcal{U}_L = \{\omega \subset \Omega \text{ measurable}, |\omega| \leqslant L|\Omega|\}$ and y_ω the solution of

$$Ay = \chi_\omega \text{ in } \in \Omega, \qquad y = 0 \text{ in } \partial\Omega. \tag{3.5}$$

Following [59], we compute derivatives of cost and constraint functions. Derivative of the volume constraint reads thus

$$DC(\omega)(\theta) = \int_{\partial\omega} \theta \cdot n \, ds.$$

When it comes to the derivative of J, we use the chain rule and firstly compute the derivative of the application $S : \omega \mapsto y_\omega$ solution of (3.5) by derivating the variational formulation associated so that $DS(\omega)$ returns for some θ the solution of the variational formulation

$$\text{find } z \in H_0^1(\Omega) \text{ such that} : (Az, v)_{L^2(\Omega)} = \int_{\partial\omega} v\theta \cdot n \, ds \quad \forall v \in H_0^1(\Omega).$$

$DS(\omega)$ is a continuous linear map from $W^{1,\infty}(\mathbb{R}^2,\mathbb{R}^2)$ to $H_0^1(\Omega)$ (it suffices to write the energy inequalities for z as a solution of a PDE on one side on ω and on the other side on $\Omega\backslash\omega$ and to use the continuity of the trace operator from $W^{1,p}(\mathbb{R}^2)$ to $L^p(\partial\omega)$). The introduction of the adjoint state $p \in H_0^1(\Omega)$ solution of

$$\begin{aligned} -A^* p = y - y_d, \quad \text{in } \Omega \\ p = 0, \quad \text{in } \partial\Omega, \end{aligned} \tag{3.6}$$

allows to finally reword the derivative of the cost function as

$$DJ(\omega)(\theta) = \int_{\partial\omega} p\theta \cdot n \, ds. \tag{3.7}$$

We first employed an augmented Lagrangian algorithm from [76, Chapter 17] to take into account the volume constraint, combined with a conjugate gradient algorithm. In the first iterations, we observed that the cost function decreases while keeping the volume constraint verified. But, the convergence of the algorithm was mostly unattainable. We even noticed that the cost function started to increase again after a large number of iterations and started to oscillate. We did not pursue this method because we could not make it perform well. There were two explanations : first, the chosen algorithm was too simple and showed some limitations. Secondly, we took as gradient descent $\theta = (p+\lambda)n$ where λ was the penalty constant based on the augmented Lagrangian. However, it is more convenient in shape optimization to proceed with the regularization of the guessed direction of descent before the update of the level-set function ϕ with (3.4). Indeed, the derivative recovered in the equation (3.7) gives values to θ only on the $\partial\omega$ boundary while its knowledge on the whole Ω domain is required to move the whole ω domain. It is indeed common to introduce a Hilbert H such that H subset of $W^{1,\infty}(\mathbb{R}^2,\mathbb{R}^2)$ and to identify the linear form in (3.7) with a well-chosen inner product on H. The Sobolev injection implies that H is generally assigned to $H^1(\Omega)$ with the weighted inner product

$$(\theta,\theta')_{H^1(\Omega)} = \int_\Omega (\alpha^2 \nabla\theta : \nabla\theta' + \theta\cdot\theta') \, dx.$$

As detailed in the Chapter 2 and following [27; 73] the introduction of such identification problems allows us to define θ in the whole domain with certain regularity properties which are numerically relevant insofar as complications during mesh modifications can be avoided. Moreover we obtain a hilbertian structure which allows us to work with the gradient algorithm (as we did with `IpOpt` in the Chapter 2). The interpolation of $DJ(\omega)$ and $DC(\omega)$ is thus done by solving

$$\begin{aligned} \forall\theta \in H^1(\Omega,\mathbb{R}^2) \qquad \big(\nabla J(\omega),\theta\big)_{H^1(\Omega,\mathbb{R}^d)} &= DJ(\omega)(\theta) \\ \forall\theta \in H^1(\Omega,\mathbb{R}^2) \qquad \big(\nabla C(\omega),\theta\big)_{H^1(\Omega,\mathbb{R}^d)} &= DC(\omega)(\theta). \end{aligned}$$

At the same time, an advanced and efficient algorithm for general optimization problems with equality constraints and particularly well suited in the context of shape optimization has been developed in [43]. An example of its application to our problem is highlighted in the Algorithm 4.

Having in mind the results illustrated in the Chapter 1, we are not able to answer the existence question when the target function does not verify the required assumptions of the Theorem 1.1. Replacing the volume constraint with a perimeter constraint should allow direct existence of the shape solution of (***SSD***) and the above algorithm would be particularly well suited to solve it when ω lies in the set

$$\mathscr{U}_{PL} = \{\omega \subset \Omega \text{ measurable }, \mathrm{Per}(\omega) \leqslant M\}.$$

Conversely, when the hypotheses of the Theorem 1.1 are true, it seems relevant to write numerically (***SSD***) in its convexified formulation (***SOP***) and to use the tools developed in the Chapter 2 for the general problems of optimization under PDE constraints.

Algorithm 4 Optimal shape design of (***SSD***) via null space gradient flow method

for k = 1...maxiter **do**
1. Solve (3.5) and compute $J_k = J(\omega_k)$ and $C_k = C(\omega_k)$
2. Solve (3.6) to get $p_k = p(\omega_k)$
2. Solve identifications problem :
find ∇J_k such that $\int_\Omega (\alpha^2 \nabla\theta : \nabla\theta' + \theta\cdot\theta')\,dx = \int_{\partial\omega_k} (p_k\theta\cdot n)\,ds \quad \forall\theta \in H^1(\Omega,\mathbb{R}^2)$ and
find ∇C_k such that $\int_\Omega (\alpha^2 \nabla\theta : \nabla\theta' + \theta\cdot\theta')\,dx = \int_{\partial\omega_k} (\theta\cdot n)\,ds \quad \forall\theta \in H^1(\Omega,\mathbb{R}^2)$
3. Put $\lambda_k = \arg\min_{\lambda\in\mathbb{R}} \|\nabla J_k + \lambda \nabla C_k\|$
4. Compute descent direction :
$\xi_{J_k} = \nabla J_k + \lambda_k \nabla C_k$
$\xi_{C_k} = \frac{C_k}{|\nabla C_k|^2}\nabla C_k$
$V_k = \alpha_J \xi_{J_k} + \alpha_C \xi_{C_k}$
5. Compute the new iterate :
for i = 0...maxtrial **do**
Solve : $\phi_t + V_k\,|\nabla\phi| = 0, \quad \phi(0) = \phi_k$
$\phi_{k+1} = \phi(dt)$
$\omega_{k+1} = \{\phi_{k+1} < 0\}$
if $\text{merit}_k(\omega_{k+1}) \leqslant \text{merit}_k(\omega_k)$, with merit function $\text{merit}_k(\omega) = \alpha_J\big(J(\omega) + \lambda_k C(\omega)\big) + \frac{\alpha_C}{2}\left(\frac{C(\omega)^2}{|\nabla C_k|^2}\right)$ **then**
exit line search loop and come back to 1.
else
$dt = \frac{dt}{2}$
end if
end for
end for

3.1.3 Convexification method

In shape optimization, the existence of a solution is most of the time not easily guaranteed and we usually have to compromise. Either the solution shape is sought in a smaller and more compact set (by adding a volume or perimeter constraint, for example), or, conversely, the solution is allowed to be in a larger set by using a relaxation method. While the level-set method deals directly with the deformation of the shape by the ϕ function, the relaxation (or convexification) method converts the initial problem into a convexified optimal control problem whose solution lies in a larger set containing $\mathscr{U}_L$, which should make it easier to be studied by classical PDE optimal control tools. This is a well-known method in shape optimization which has the double advantage of providing strategies for establishing theoretical results (see Chapter 1 and see [82]) and a mathematical framework for finding numerical solution (see Chapter 2). On our side, we pursue the same convexification as in the Chapter 1. Given any measurable subset ω, we identify ω with its indicator function $\chi_\omega \in L^\infty(\Omega;\{0,1\})$ and, following [10; 82; 83], we identify $\mathscr{U}_L$ with a subset of $L^\infty(\Omega)$. Then, the optimal solutions are searched in the convex closure of $\mathscr{U}_L$ in the weak star topology of L^∞.

$$\overline{\mathscr{U}}_L = \left\{ a \in L^\infty\big(\Omega;[0,1]\big) \;\middle|\; \int_\Omega a(x)\,dx \leqslant L|\Omega| \right\} \subset U = L^2(\Omega). \tag{3.8}$$

More generally, we refer to [6] for details on convexification and homogenization methods. We remind the *convexified* (or *relaxed*) optimal control problem $(\boldsymbol{OCP})_{\mathbf{T}}$ as the problem of determining a control $t \mapsto a(t) \in \overline{\mathscr{U}}_L$ minimizing the cost

$$J_T(a) = \frac{1}{2T}\int_0^T \|y(t) - y_d\|^2_{L^2(\Omega)}\, dt$$

under the dynamical constraints

$$\partial_t y + Ay = a, \qquad y_{|\partial\Omega} = 0, \qquad y(0) = y_0. \tag{3.9}$$

The corresponding convexified static optimization $(\boldsymbol{SOP})$ problem still reads

$$\min_{a\in\overline{\mathscr{U}}_L} \frac{1}{2}\|y - y_d\|^2_{L^2(\Omega)}, \qquad Ay = a, \qquad y_{|\partial\Omega} = 0. \tag{$\boldsymbol{SOP}$}$$

We have seen in the Chapter 1 that, under the assumptions of the Theorem 1.1, the solving of $(\boldsymbol{OCP})_{\mathbf{T}}$ and of $(\boldsymbol{SOP})$ implies that the optimal solutions of these convexified problems are upper level-sets of the adjoint state and are thus shapes which are the optimal solution to within a zero measure domain, respectively of $(\boldsymbol{DSD})_{\mathbf{T}}$ and of $(\boldsymbol{SSD})$. When the assumptions of the Theorem 1.1 are not satisfied, a relaxation phenomenon can be observed, namely that it can happen that the optimal solution of $(\boldsymbol{OCP})_{\mathbf{T}}$ or of $(\boldsymbol{SOP})$ is not the characteristic function of a certain subset, insofar as it takes values in $(0,1)$ on a subset of positive measure. In the Section 3.1.4, we give an example of a target function y_d such that the optimal solution $\bar{a}$ of $(\boldsymbol{SOP})$ is not 0 or 1 everywhere.

We then generate a mesh T_h of Ω, that as with the level-set method will not be modified during the optimization process. Control a and state y are discretized according to $\mathbb{P}_1$ finite elements and matrices of the variational formulation are built such that numerical problems are reworded according to the Option 1 of *FDTO* method stated in Chapter 2. Let denote $(\phi_i)_{1\leqslant i\leqslant n_d}$ the $\mathbb{P}_1$ finite elements basis where n_d is the size of the finite element space (for $\mathbb{P}_1$-Lagrange elements, n_d is the number of vertices) such that both discretization of state and control variables respectively lie in $Y_h, \mathscr{U}^h_{ad} \subset U_h$ where

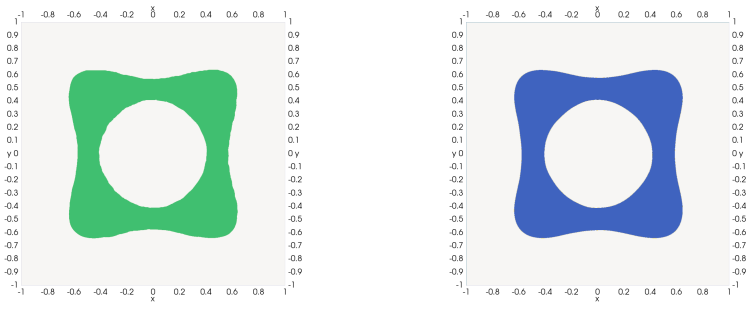
$$\begin{aligned} V_h &= \left\{ v \in H^1(\Omega), \quad \forall K \in T_h \;\; v_{|K} \in \mathbb{P}_1 \right\} = \mathrm{Vect}(\phi_i)_{i\in\{1..n_d\}} \\ Y_h &= \{v \in V_h, v_{|\partial\Omega} = 0\} = \mathrm{Vect}(\phi_i)_{i\in\{1..n\}} \quad \text{for } 0 < n < n_d, \\ \mathscr{U}^h_{ad} &= \{u \in V_h, u^h_a \leqslant u \leqslant u^h_b\} \subset U_h = V_h. \end{aligned}$$

We introduce the matrices involved in the cost computation and the PDE constraint discretization

$$A_{(i,j)\in\{1..n_d\}^2} = (A\phi_i, \phi_j)_{L^2(\Omega)}, \quad M_{(i,j)\in\{1..n\}^2} = (\phi_i, \phi_j)_{L^2(\Omega)}, \quad L_{v,i\in\{1..n_d\}} = (\phi_i, 1)_{L^2(\Omega)}. \tag{3.10}$$

The stiffness matrix is a $n\times n$ matrix that is filled in a $n_d \times n_d$ matrix by stipulating a penalization term on boundary vertices indexes to force homogeneous Dirichlet boundary conditions. The generalization to the case of non homogeneous case is treated in Chapter 2. We finally write $(\boldsymbol{SOP})$ as

$$\min_{(Y,U)\in\mathbb{R}^{2n_d}} (Y - Y_d)^T M (Y - Y_d), \quad AY = MU, \quad L_v^T U \leqslant L|\Omega| \quad 0 \leqslant U \leqslant 1 \tag{3.11}$$



(a) (b)

FIGURE 3.2 – Optimal static shape : (a) Convexification method ; (b) Level-set method.

When it comes for the time case $(\boldsymbol{OCP})_{\mathbf{T}}$, we use an implicit Euler scheme with n_t steps and build the same matrices (2.45) as in Chapter 2 to rewrite $(\boldsymbol{OCP})_{\mathbf{T}}$ as

$$\min_{(Y,U)\in\mathbb{R}^{2n_d n_t}} (Y-Y_d)^T \tilde{D}(Y-Y_d), \quad \tilde{A}Y = \tilde{M}U, \quad L_v^T U_i \leqslant L|\Omega| \quad 0 \leqslant U_i \leqslant 1 \qquad \forall i \in \{1\cdots N_t\}. \tag{3.12}$$

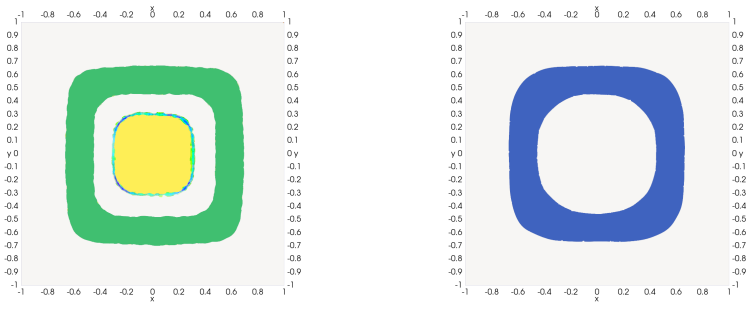
The state variable Y is seen here as an optimization variable (referring to Option 1 in Chapter 2) and avoids us to explicitly solve the state equations involved in (3.9) and $(\boldsymbol{SOP})$ and associated adjoint equations mentioned in Chapter 2. The discretized convexified problems are linear quadratic optimization problems with inequality and equality constraints. It is then easy to compute the derivatives of the various functions involved. We implement them as well as their derivatives and we call the optimization routine `IpOpt` via `FreeFEM`following the template illustrated in Chapter 2. We thus plot in Figure 3.2 optimal shapes obtained with the both level-set and convexification methods for the stationary case on a mesh including 28800 triangles and 14641 vertices. Stationary solutions are very similar unlike the effectiveness of both procedures.

3.1.4 Discussion : comparison of implemented methods

With the aim in mind to solve the optimal time-varying shape design $(\boldsymbol{DSD})_{\mathbf{T}}$, we proceed to a comparison of level-set and convexification methods on several stationary examples of the following form

$$\min_{\omega} \frac{1}{2}\|y-y_d\|^2, \quad Ay = \chi_\omega, \quad y_{|\partial\Omega} = 0, \quad |\omega| \leqslant L|\Omega|$$

for various elliptic operators A defined in the Chapter 1, spaces Ω and target functions y_d. This will allow us to put forward an efficient procedure for the dynamic case, which is more delicate. Moreover, this comparison brings us some knowledge about the numerical behavior of the two procedures when the limiting cases are treated. Indeed, we want to observe what happens both when the target function does not verify the assumptions of the Theorem 1.1 and also when the operator A does not have a strictly positive ellipticity constant θ. Finally, the 3D case is also of interest since the computation time and the memory allocation will increase significantly with the size of the optimization variables.



(a) (b)

FIGURE 3.3 – Optimal static solution : (a) Convexification method ; (b) Level-set method.

Example 1 : $\Omega = [-1,1]^2$**,** $y_d = 0.1$ **and** $A = -\Delta$

Represented on Figure 3.2, the function y_d satisfies well the assumptions of the Theorem 1.1. The numerical solutions obtained with the level-set and convexification algorithms are very similar. Although the level-set method guarantees that the solution is a shape in case of convergence, the convexification method seems here to be faster and requires less iterations. This may be due to two things : first, the convexification method is known to be a very efficient and robust optimization method and we have furthermore implemented the Hessian of the cost function, while in the level-set method we only perform a gradient descent algorithm without Hessian or approximation of it.

Example 2 : $\Omega = [-1,1]^2$**,** $y_d = \frac{2-x^2+y^2}{20}$ **and** $A = -\Delta$

The target function y_d is here set so that the assumptions of the Theorem 1.1 are not satisfied. As already observed in the numerical simulations of Chapter 1, a relaxation phenomenon occurs. On the Figure 3.3(a) we can see that the optimal solution $\bar{a}$ takes values in $(0,1)$. On the contrary, the level-set method gives back to a form. This leads us to think that the assumptions made in the Theorem 1.1 can potentially be weakened. The search for an optimal solution by means of a level-set function does not allow the solution to take values in $(0,1)$. Thus, the existence of an optimal shape of the problems $(\boldsymbol{SSD})$ et $(\boldsymbol{DSD})_{\mathbf{T}}$ may be possible under weaker assumptions than those described in the Chapter 1. Nevertheless, the two numerical solutions remain quite similar, in that the sets where the convexified solution and the level-set solution take the value 1 coincide.

Example 3 : $\Omega = [-1,1]^2$**,** $y_d = 0.1$ **and** A **a second-order operator**

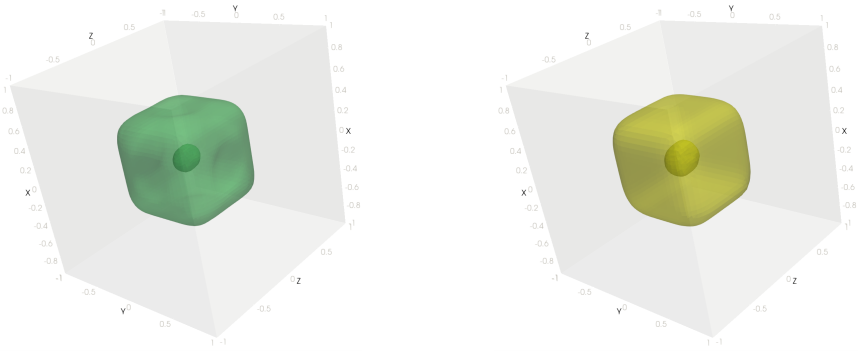
The target function is again in the lines of Theorem 1.1 and several second-order operators A are this time examined. We expect to observe various behavior according to whether the ellipticity constant is $\theta > 0$, $\theta = 0$ or $\theta < 0$. When it comes to the turnpike property, if the operator A is no more uniformly elliptic, we won't be able to write energy inequality stated in Appendix 1.4 that are necessary for the turnpike results stated in Chapter 1. On Figure 3.4, we take, successively :

(a) (b)

(c) (d)

(e) (f)

FIGURE 3.4 – Optimal static solution : (a) $\theta > 0$: Convexification method; (b) $\theta > 0$: Level-set method; (c) $\theta = 0$: Convexification method; (d) $\theta = 0$: Level-set method; (e) $\theta < 0$: Convexification method; (f) $\theta < 0$: Level-set method.



(a) (b)

FIGURE 3.5 – Optimal static shape 3D : (a) Convexification method ; (b) Level-set method.

- $Ay = -\partial_{11}y - \partial_{22}y - 0.5\partial_{12}y - 0.5\partial_{21}y$ and $\theta > 0$
- $Ay = -\partial_{11}y - \partial_{22}y - \partial_{12}y - \partial_{21}y$ and $\theta = 0$
- $Ay = -\partial_{11}y - \partial_{22}y - 1.5\partial_{12}y - 1.5\partial_{21}y$ and $\theta < 0$

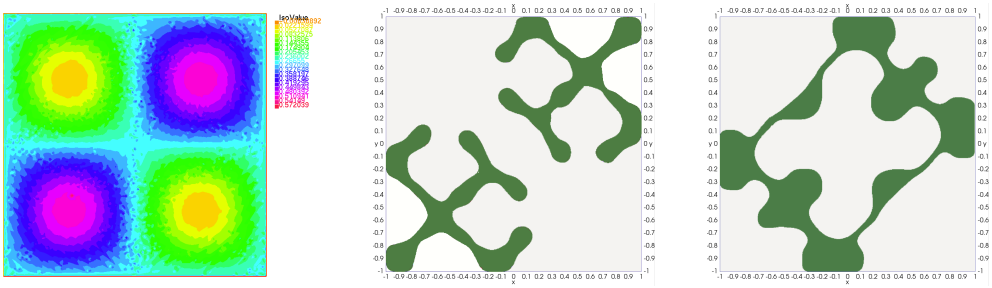
When $\theta > 0$, both methods are giving back the same solution. When $\theta = 0$, the level-set solution is similar to the convexified one but is slightly more regular. Indeed, the convexification method allows a wider variety panel of solutions since (***SOP***) is solved in $L^\infty(\Omega;(0,1))$ while the level-set method forces the solution to keep some regularity properties insofar we modify step by step an initial regular shape through successive regular enough vector fields. Finally, when $\theta < 0$, the level-set algorithm does not even converge and convexification gives back a shape solution without discernible structure.

Example 4 : $\Omega = [-1,1]^3$**,** $y_d = 0.025$ **and** $A = -\Delta$

`FreeFEM` is also well-indicated to solve PDEs in 3D. Tools developed in Chapter 2 are easily generalizable by loading the appropriate environment. Mesh generation is a bit more different but stays user-friendly. It is interesting to observe what can happen in 3D where topology modifications are more varied than in 2D. Except for mesh generation, the previously described methods are unchanged. Of course, in 3D the time required to see the algorithm converge is much larger insofar it is directly due to the time required to solve a PDE in 3D. Moreover, since the time computation is going to be larger, it will be decisive in the choice of the more effective method for the time-varying shape design problem. A last one example was the one of taking a target function solution of

$$-\Delta y_d = a_d \text{ in } \Omega, \quad y_d = 0 \text{ in } \partial\Omega,$$

with $a_d \in L^\infty(\Omega;(0,1))$ such that a_d satisfies the volume constraint $\int_\Omega a_d(x)\,dx \leqslant L|\Omega|$. Convexification should return a_d as the optimal solution and we are interested in the one returned by the level-set method. Let $\Omega = [-1,1]^2$ and $a_d = \frac{1}{4}(xy+1)$. Figure 3.6(a) describes the optimal solution returned by means of relaxation and which coincides well with the graph of a_d. Moreover, the shapes returned by the level-set with several initialization level-set functions are



(a) (b) (c)

FIGURE 3.6 – Inverse problem - optimal solution : (a) Convexification method ; (b) Level-set method with $\phi^0 = a_d$; (c) Level-set method with $\phi^0 = (x^2 + y^2 - 0.1)(x^2 + y^2 - 0.4)$.

again concentrated where the values of a_d are the largest. We stopped the algorithm when the convergence criterion was around 10^{-4} and the objective function around 10^{-5}. Convergence may not be attainable since we normally expect to find almost $J = 0$.

Moreover, this example highlights a drawback of the level-set method which admits a strong dependence of the topology of the returned optimal shape on the one resulting from the initialization level-set function. We store in the Table 3.1 some comparative data of the two procedures in order to justify our choice for the next simulations. We compare the total computation time, the processing time, the number of iterations, the violation constraints, and the convergence criterion. We first deduce that under the assumptions of the Theorem 1.1 both methods obtain the same solution with less efficiency from the level-set method. Outside this scenario, the convexification remains unchanged in terms of its performance while the other method decreases in terms of convergence speed. Moreover, in order to deal with a time-varying shape and thus to add at least thirty times more degrees of freedom, the computation time of the level-set method is prohibitive compared to the convexified method. This is explained by the use of a Newton algorithm and the power of the interior point method combined with the PDEffem method already highlighted in the Chapter 2.

Problem	Level-set	Convexification
2D no relaxation	iter = 90 $J = 0.0074$ $C \approx 10^{-4}$ $\|\nabla J\| = 10^{-5}$ $t_{cpu} \approx 90$s	iter = 21 $J = 0.0071$ $C \approx 10^{-31}$ $\|\nabla J\| = 10^{-8}$ $t_{cpu} \approx 6$s
2D relaxation	iter = 135 $J = 0.00098$ $C \approx 10^{-4}$ $\|\nabla J\| = 10^{-5}$ $t_{cpu} \approx 115$s	iter = 21 $J = 0.00097$ $C \approx 10^{-17}$ $\|\nabla J\| = 10^{-8}$ $t_{cpu} \approx 6$s
Elliptic case	iter = 300 $J = 0.00697$ $C \approx 10^{-6}$ $\|\nabla J\| = 10^{-5}$ $t_{cpu} \approx 250$s	iter = 23 $J = 0.00697$ $C \approx 10^{-17}$ $\|\nabla J\| = 10^{-8}$ $t_{cpu} \approx 3$s
3D	iter = 90 $J = 0.0262$ $C \approx 10^{-5}$ $\|\nabla J\| = 10^{-4}$ $t_{cpu} \approx 6000$s	iter = 21 $J = 0.026$ $C \approx 10^{-31}$ $\|\nabla J\| = 10^{-8}$ $t_{cpu} \approx 2850$s

TABLE 3.1 – Level-set and convexification methods comparison

3.2 Numerical examples illustrating the turnpike phenomenon

In accordance with the comparative section above, we will focus in the following on the combination of `FreeFEM` and `IpOpt` based on the tools described in the Chapter 2 to solve time-varying optimal shape design problems. Our main objectives are on the one hand to highlight the turnpike phenomenon on several examples and on the other hand to deduce some perspectives for further research on the theoretical results elaborated in the Chapter 1. We first give numerical simulations for different domains, target functions y_d and operators A in the framework of the Theorem 1.1. The operator A is discretized according to the associated variational formulation and its matrix representation on a mesh of about 15000 vertices. We carry out the time discretization by means of an implicit Euler scheme with 35 time steps such that the convexification problem $(\boldsymbol{OCP})_{\mathbf{T}}$ of $(\boldsymbol{DSD})_{\mathbf{T}}$ is numerically equivalent to the solution of the quadratic problem (3.12) with $\tilde{A}, \tilde{M}$ and $\tilde{D}$ defined as in (2.45) based on the matrices A, B and L defined in (3.10) and this with nearly 1000000 variables. The computation time is about half a day to obtain a solution with a small enough tolerance of convergence. Indeed, if the termination criterion is too large, we observe that the output is a relaxed function. We force the algorithm to go further in the optimization process by decreasing the convergence criterion of the tolerance in order to observe the optimal solution as a shape evolving in time. However, this results in a higher computation time. Once the solutions are found, they are exported in vtk format to be visualized with the free software Paraview. Illustrative examples are

- Example 1 : $\Omega = [-1,1]^2$, $y_d = 0.1$ and $Au = -\Delta u$ on Figure 3.7
- Example 2 : $\Omega = [-1,1]^2$, $y_d = \frac{1}{20}(xy+1)$ and $Au = -\Delta u$ on Figure 3.9
- Example 3 : $\Omega =$ half-stadium, $y_d = \frac{1}{20}(xy+1)$ and $Au = -\Delta u$ on Figure 3.10
- Example 4 : $\Omega = [-1,1]^3$, $y_d = 0.025$ and $Au = -\Delta$ on Figure 3.11
- Example 5 : $\Omega = [-1,1]^2$, $y_d = \frac{1}{20}(2\sin(2(x^2+y^2))+1)$ and $Au = -\partial_x((x-y)^2\partial_x u) - \partial_y((x+y)^2\partial_y u)$ on Figure 3.12.

We plot on all the examples the cylinder of the time evolution of the shape and the behavior of the shape at certain times. In addition, we also plot the optimal stationary shape of the associated stationary problem (***SOP***) to compare it to the shape in the middle of the time interval. To observe the phenomenon of exponential turnpike, we plot the error between both optimal triples $t \mapsto \|y_T(t) - \bar{y}\| + \|p_T(t) - \bar{p}\| + \|\chi_{\omega_T(t)} - \chi_{\bar{\omega}}\|$ for different final times T on the Figure 3.8. The larger the final time T is, the more often the residual between the two optimal triples is close to 0. The behavior of the residual function is typical of the exponential turnpike and is in line with our conjecture that this phenomenon could be shown theoretically. Moreover, we observe that most of the time, the dynamic optimal shape remains very close to the static shape. These numerical observations seem to hold systematically under the assumptions made in the Theorem 1.1. The turnpike phenomenon occurs even when we observe a relaxation as long as $\theta > 0$. On the contrary, if $\theta \leqslant 0$ we have no guarantee to write energy inequalities with a constant independent of the final time (see Appendix 1.A), and the turnpike phenomenon is not guaranteed. In conclusion, these numerical simulations make two key points : first, the numerical turnpike results lead us to believe that an exponential turnpike property for state, adjoint, and control should appear. Nevertheless, we must ask ourselves if the behavior of the error $t \mapsto \|y_T(t) - \bar{y}\| + \|p_T(t) - \bar{p}\| + \|\chi_{\omega_T(t)} - \chi_{\bar{\omega}}\|$ is a numerical effect that should occur only for the discretized problem or if it can be generalized to the continuous problem. The second

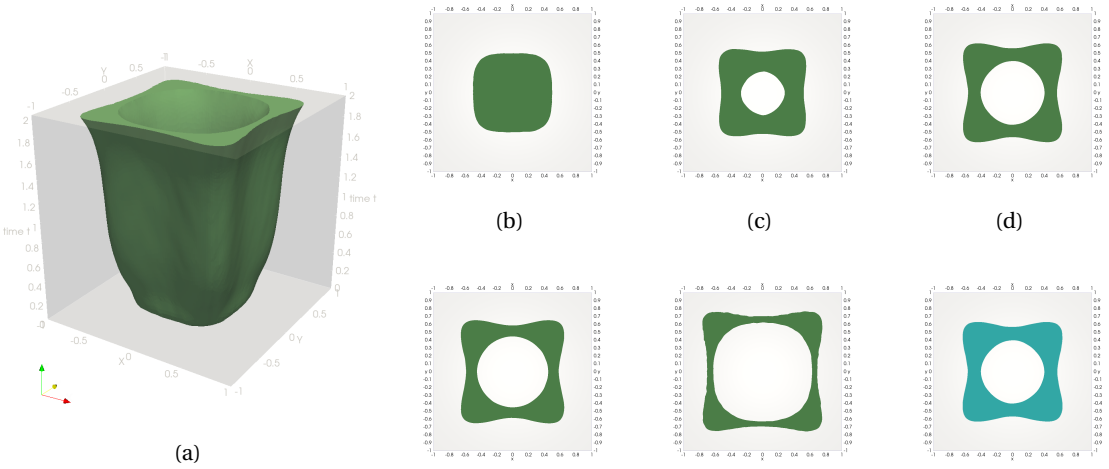


FIGURE 3.7 – Time-varying optimal shape : (a) Time shape; (b) $t = 0$; (c) $t = 0.5$; (d) $t \in (0.5, 1.5)$; (e) $t = 1.5$; (f) $t = T$; (g) Static shape.

point concerns the existence of a shape solution and more particularly of a time-varying shape. The Theorem 1.1 establishes the existence of a stationary shape as a solution of (***SOP***) when assumptions are made on the target function. For the time-varying shape obtained for our several examples, we expect the same assumption to imply the existence of a time-varying shape as a solution of $(\boldsymbol{DSD})_{\mathbf{T}}$. So far, we have tried to adapt the proof of the stationary case without success. We have found a similar strategy in the literature for stationary problems in [57] but the case of time-dependent solutions seems a more difficult issue to tackle.

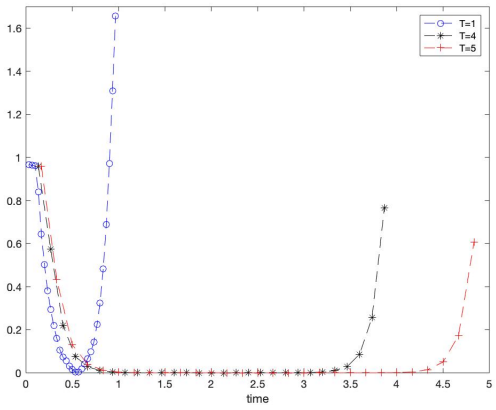


FIGURE 3.8 – $t \mapsto \|y_T(t) - \bar{y}\| + \|p_T(t) - \bar{p}\| + \|\chi_{\omega_T(t)} - \chi_{\bar{\omega}}\|$ for $T \in \{1, 3, 5\}$

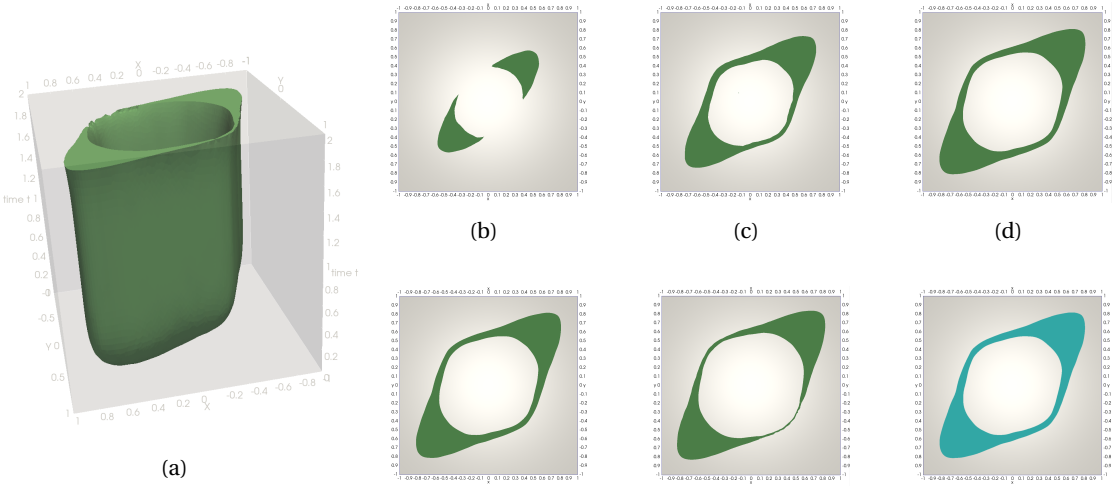


FIGURE 3.9 – (a) Time shape; (b) $t = 0$; (c) $t = 0.5$; (d) $t \in (0.5, 1.5)$; (e) $t = 1.5$; (f) $t = T$; (g) Static shape.

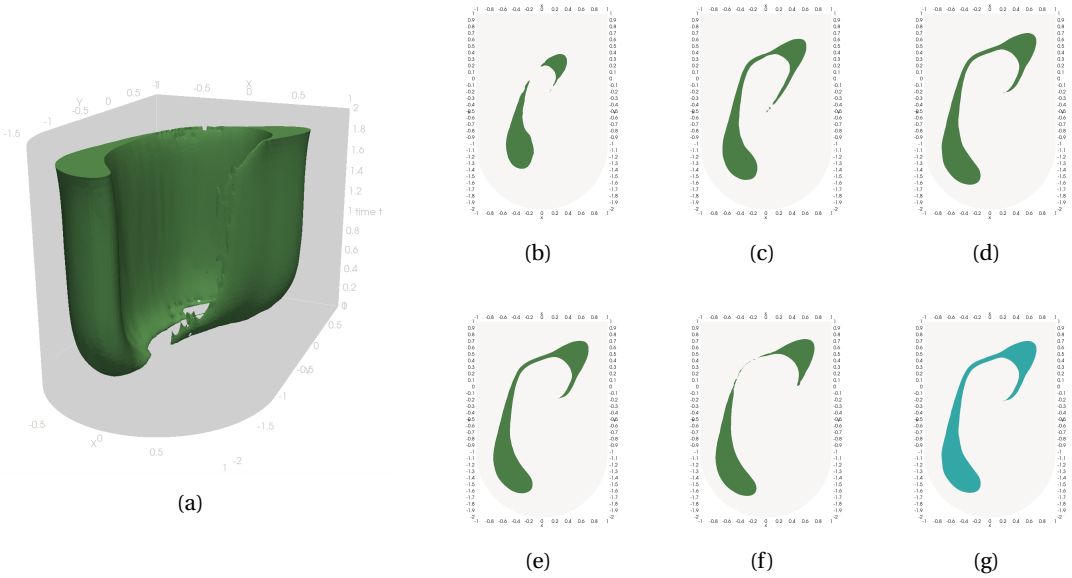
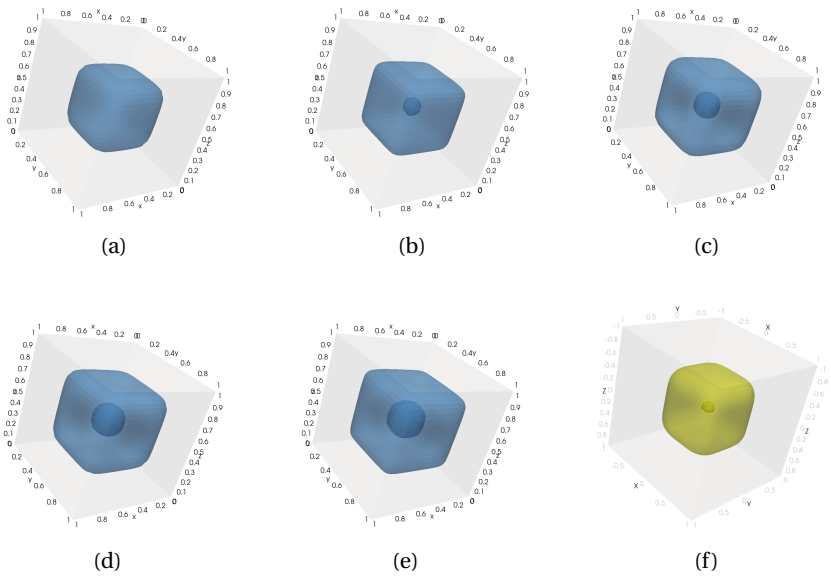
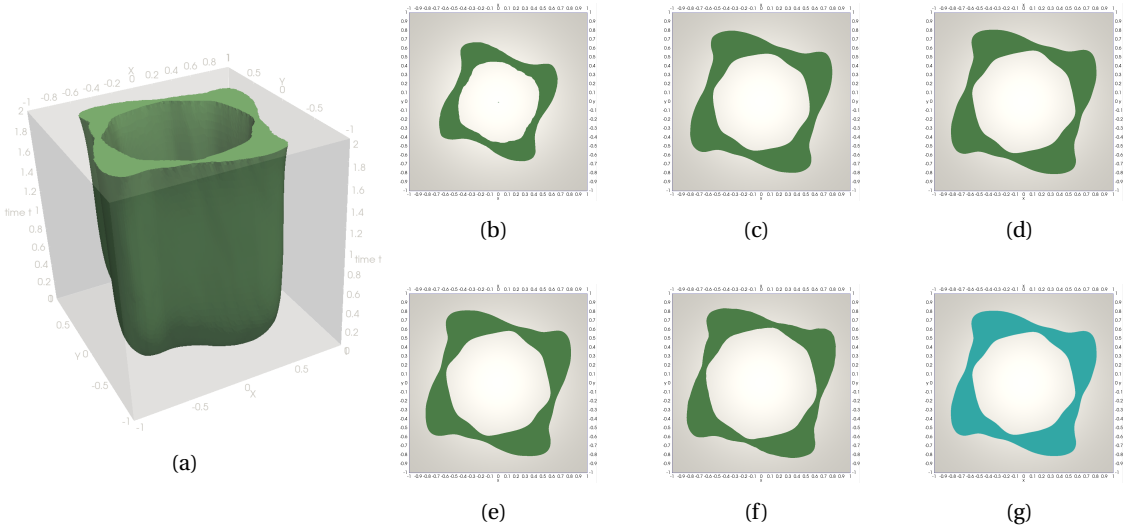


FIGURE 3.10 – (a) Time shape; (b) $t = 0$; (c) $t = 0.5$; (d) $t \in (0.5, 1.5)$; (e) $t = 1.5$; (f) $t = T$; (g) Static shape.



(a) (b) (c) (d) (e) (f)

FIGURE 3.11 – Time shape : (a) $t = 0$; (b) $t = 0.1$; (c) $t \in]0.1, 0.9[$; (d) $t = 0.9$; (e) $t = T = 1$; (f) Static shape.



(a) (b) (c) (d) (e) (f) (g)

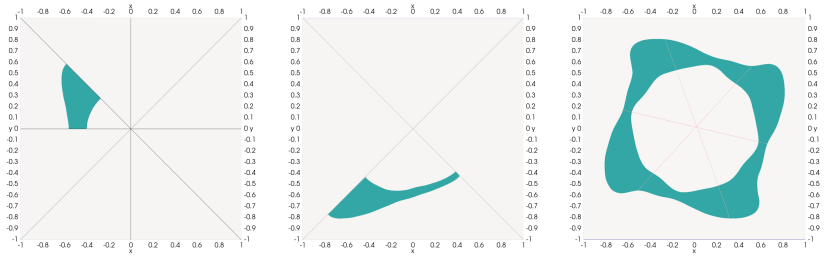
FIGURE 3.12 – (a) Time shape ; (b) $t = 0$; (c) $t = 0.5$; (d) $t \in (0.5, 1.5)$; (e) $t = 1.5$; (f) $t = T$; (g) Static shape.

3.3 Further prospects

As we have said, the question of the existence of a time-varying shape and the manifestation of an exponential turnpike property remains open. Other properties could be established such as the geometric characteristics of the optimal solutions with respect to the initial data.

3.3.1 Symmetries of solutions

On the basis of our numerical examples, we conjecture that if Ω, the operator A and the target function y_d share the same symmetry properties, then optimal solutions $\bar{\omega}$ and $\omega_T(\cdot)$ share as well the same symmetry properties. For instance, Figure 3.13(a) highlights four axes of symmetry ($x=0, y=0, y=x, y=-x$), Figure 3.13(b) two axes ($y=x, y=-x$) and Figure 3.13(c) shows up a central symmetry property (of center $(0,0)$ and with angle π).



(a) (b) (c)

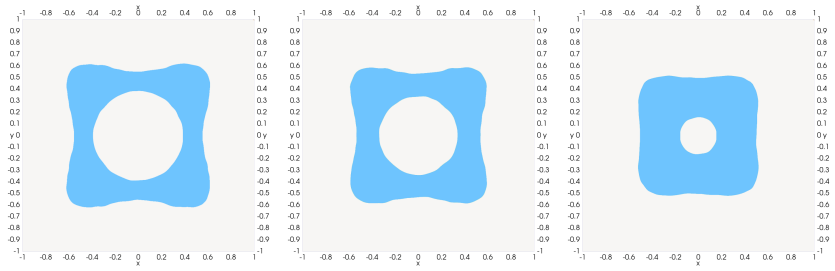
FIGURE 3.13 – Symmetries of static optimal shapes : (a) $A=-\Delta, y_d=0.1$; (b) $A=-\Delta, y_d=\frac{xy+1}{20}$; (c) $Au=-\partial_x((x-y)^2\partial_x u)-\partial_y((x+y)^2\partial_y u), y_d=\frac{1}{20}(2\sin(2(x^2+y^2))+1)$.

3.3.2 Numerical extension to semilinear PDEs

An extension of both numerical and theoretical results is possible when dealing with semi-linear PDEs such as

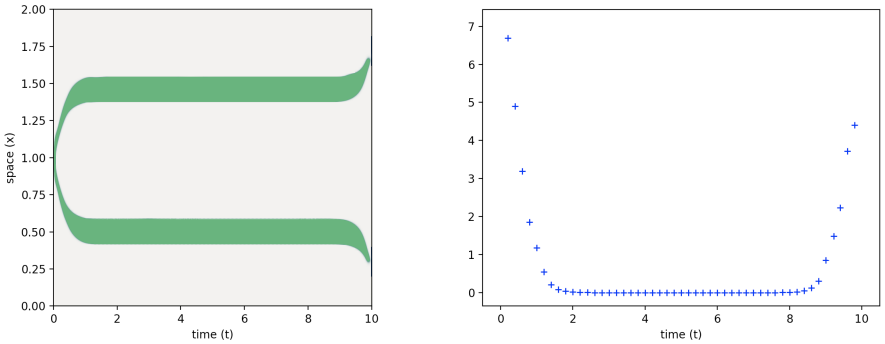
$$\partial_t y + Ay + f(y) = \chi_\omega, \quad y_{|\partial\Omega} = 0, \quad y(0) = y_0. \tag{3.13}$$

Some examples already exist in the literature, for instance in [58] where the authors are interested in semilinear shape optimization problems and whose existence proofs involve shape optimization problems close to the one we treat in (3.13). The assumptions of our theorem must be modified accordingly and some constraints on the function f must be introduced following the general framework for the existence of optimal control problems governed by a semilinear equation stated in [60; 94]. The results already stated on the turnpike phenomenon for the semi-linear equations [55; 79; 80] go in the direction that its research in the context of shape optimization is promising. For our part, we provide below several numerical simulations as a possible guarantee that the topic is worth addressing. We always relax the function χ_{omega} with a taking values in $(0,1)$ and use the convexification approach combining `IpOpt` and `FreeFEM` for the numerical solution. We can no longer discretize it with a linear quadratic



(a) (b) (c)

FIGURE 3.14 – Semilinear static optimal designs : (a) $f(y) = y(1-y^2)\exp(y)\sin(y)$; (b) $f(y) = (y+0.4)^3$; (c) $f(y) = \frac{1}{y+5}$.



(a) (b)

FIGURE 3.15 – Semilinear optimal designs : (a) Dynamical optimal shape; (b) $t \mapsto \|y_T(t) - \bar{y}\| + \|p_T(t) - \bar{p}\| + \|\chi_{\omega_T)(t)} - \chi_{\bar{\omega}}\|$.

optimization problem such as (3.11) and (3.12) since the PDE to be solved is not linear anymore. Instead, we use a fixed point method as described in the Chapter 2. We present on Figure 3.14 some examples of stationary solutions for several f-functions and for an operator $A = -\Delta$ and a target function y_d verifying the assumptions of the Theorem 1.1. We observe that no relaxation occurs and that the optimal solutions are quite similar to the solutions obtained in the linear case. Some additional examples with a less classical function f should be considered to be more convincing. Moreover, we still observe the turnpike phenomenon for the optimal design of the shape in 1D. We take $\Omega = [0,2]$, A the Dirichlet Laplacian, $y_d = 0.1$ and $T = 10$. We plot on the Figure 3.15 the dynamic optimal shape and we observe that it remains stationary most of the time (which corresponds to the solution of the corresponding static optimal shape problem). Moreover, we plot the error between the two optimal triples and we always observe a semblance of exponential turnpike phenomenon.

Conclusion and perspectives

In summary, in the Chapter 1 we address the issue of the turnpike phenomenon in shape optimization which has never been studied before and which we have called "shape turnpike". We focus more specifically on linear parabolic models controlled by domains evolving with time and acting as a source term seen as the characteristic function of this domain, whose optimum is sought to minimize a quadratic criterion with functions *Mayer* and *Lagrange*. Based on a relaxation of the initial problem, we manage to determine under certain assumptions the existence and uniqueness of the optimal shapes. Moreover, when we treat only a terminal cost, we show that the Hausdorff distance between the time-varying optimal shape and a stationary shape to be specified verifies an exponential turnpike property, i.e. the time-varying shape is, most of the time, exponentially close to that stationary shape in the sense of the Hausdorff distance. Moreover, when we consider an integral functional, we show that the state and adjoint variables resulting from the application of the Pontryagin maximum principle verify both the measure and the integral turnpike properties in the sense of the L^2 norm. Moreover, we perform some numerical experiments in the Chapter 3 which strongly urge us to pursue the research of this phenomenon in two main directions :

- in the *Lagrange* case and under the assumptions of the 1.1 theorem, we are convinced that the existence of an optimal time-varying shape can be proved. The stationary case shows some required properties of the target function and we prove the existence through the regularity properties of the state and adjoint variables. For the time-dependent case with an integral cost, the existence could be proved thanks to arguments coming from the maximum regularity properties and Hölder estimates for the solutions of parabolic equations. Indeed, we need regularities on the solutions in time to be able to apply the same arguments as in the stationary case.
- Moreover, the same numerical simulations provide strong motivation to continue the search for an exponential turnpike property for the two optimal triples of $(\boldsymbol{DSD})_{\mathbf{T}}$ and $(\boldsymbol{SSD})$ under the assumptions of the Theorem 1.1. In the Mayer case and following [47], if we can transpose the exponential turnpike property of the Hausdorff distance to the symmetric difference sense, the state and adjoint turnpike properties follow directly from the energy inequalities in the Appendix 1.A. More broadly, based on what we observe numerically, we expect the state and the adjoint to remain exponentially close to the stationary state and adjoint, and the time-varying shape to approach exponentially the stationary shape in the sense of symmetric difference.

As an open question, we draw attention to the fact that the existence of solutions of $(\boldsymbol{DSD})_{\mathbf{T}}$ and $(\boldsymbol{SSD})$ is much easier if we replace the volume constraint by a perimeter constraint since this adds some compactness properties on the minimizing sequences that one takes when one

wants to show the existence of solutions. But in this case, we have to find alternatives to writing the optimality conditions to characterize the solutions in order to exhibit turnpike properties. Indeed, with a relaxation method in mind, the bathtub principle no longer seems appropriate to characterize optimal control as upper level-sets and according to the tools illustrated in the Chapter 2. The mathematical framework might require the introduction of BV functions to characterize the controls. This is therefore a more challenging but still attractive issue. Moreover, we have performed numerical simulations for semi-linear parabolic models which are in line with the generalization of our results stated in the Chapter 1. The assumptions of the Theorem 1.1 must be adapted accordingly and the existence of the optimal shape seems to be a more difficult problem. We can take inspiration from [58] where the authors look for an optimal shape with a measurement constraint that minimizes a Dirichlet type energy involving the solution of a semi-linear elliptic PDE. By means of a relaxation method, they manage to prove the existence of optimal shapes.

Even more generally, the turnpike phenomenon could be studied in the context of shape optimization for other PDE models. Still in the case of parabolic models, we could move to the case of boundary control (see [54] for a turnpike result with parabolic models), i.e., finding an optimal shape of the boundary of a domain such that a parabolic equation is verified inside in order to minimize a quadratic criterion. In the following, we present a possible industrial application. According to the results already stated in previous works, the case of PDE models similar to the wave equation also seems promising.

In Chapter 2, we highlighted the power of the partial differential solver `FreeFEM` when connected to the interior point method `IpOpt` to solve difficult optimal control problems. We focus on several examples that can serve as models. On a linear quadratic example, we present four solution methods that are divided into direct and indirect methods. We emphasize the greater efficiency of direct approaches for linear quadratic problems in that a Newton algorithm can be used. Nevertheless, indirect methods, which require the solution of the adjoint equation, are most often more suitable for nonlinear and shape design problems. Moreover, we carry out automatic differentiation methods via the optimization software `AMPL` having previously discretized the problem with `FreeFEM` (let us digress here; it would be convenient to connect `FreeFEM` with a renamed automatic differentiation software such as `Adept` or `CppAD` in order to automate the computation of the derivatives directly in `FreeFEM`). Finally, we underline the good adaptability of `FreeFEM` to deal with shape optimization and mesh modifications on the last, non-trivial problem of determining the optimal shapes of microswimmers in a fluid. For this purpose, we model the fluid domain by a torus and we search for the optimal shape of the lower boundary so that the fluid velocity is maximal in a given direction. We add some additional constraints on the first and second order derivatives of the admissible shape boundary in order to easily obtain an appropriate mathematical framework for the numerical discretization. We finally plot the optimal solutions on several examples and notice that the optimal solution "tends" to a triangular shaped solution when the bounds on the second order derivatives constraint are larger. We therefore suggest as an open question to approach the limit problem by removing the constraints on the second derivatives, which is very well done numerically but nevertheless requires a much more complicated mathematical framework from the theoretical point of view. In addition, it might be interesting to allow the boundary shape to overlap via general 2D deformation vector fields with perimeter constraints on the boundary. Finally, regarding the special case of shape optimization and the class of ho-

mogenization methods, it would be very interesting to observe the effectiveness of `FreeFEM` combined with `IpOpt` on the classical example of minimizing the compliance of a cantilever and on other structural optimization problems.

To continue with the micro-swimmer example, we assume that the boundary moves at a fixed velocity so that we are dealing with a stationary Stokes PDE. It might be interesting to make the velocity time dependent and introduce a time dependent Stokes equation (or even Navier-Stokes). Depending on what we observed in the stationary case and with the periodicity properties of the problem, we could expect to observe a periodic turnpike phenomenon, i.e. the time-optimal boundary of the micro-swimmers would give rise to a recurrent pattern which could be the optimal shape of the stationary Stokes (or Navier-Stokes) problem. The velocity can also denote a control variable so that in the middle of the trajectory we expect it to remain close to an optimal stationary velocity. Such a theoretical result seems quite complicated but the numerical part, based on the tools illustrated in the Chapter 2, seems more accessible at first.

We have other examples of very challenging problems for numerical simulations such as solving dynamic shape design problems for partial differential equations that play a role in fluid mechanics. We draw on recent literature and some papers on wave-maker's optimization (see [24; 75]). We introduce a shallow water equation into a channel so that the shape of the bottom changes the free surface and the fluid velocity. It is natural to ask what happens if we consider a wave-maker whose shape can evolve in time. Our study is based on a kinetic interpretation of the shallow water equations (see [16; 78]), which brings up a new variable verifying a linear kinetic equation with additional nonlinear constraints on the state. Having in mind the numerical solution of optimal control problems governed by a nonlinear PDE, the possibility of switching to a linear PDE by adding constraints on the state seems promising to us and would deserve to be further studied. As observed in the Chapter 2, the resulting adjoint equation implies multipliers associated with the additional constraints and direct methods would then be more suitable. Second, industrial applications to this time-varying shape optimization problem of a wave generator arising from it would be straightforward.

Throughout the Chapter 3, we discuss more precisely the numerical solution of $(\boldsymbol{DSD})_{\mathbf{T}}$and $\boldsymbol{SSD}$. We first focus on the question in 1D so that the problem can be discretized with finite differences in space and an implicit Euler scheme in time. We point out the possibility of doing optimal shape design with the software `AMPL` which handles binary variables very well using the solver `CPLEX`. It could be interesting to generalize the call to `CPLEX` in higher dimensions. We then compare the *convexification* and *level-set* methods on stationary problems so that the convexified method seems more relevant in our case. Nevertheless, the level-set method can converge even if we are not in the lines of the Theorem 1.1. This leads us to think that the existence of optimal shapes can occur for weaker assumptions than those made in the Chapter 1. We then propose several numerical simulations and focus mainly on those in dimension 2 since the combination of 3D finite elements with a temporal discretization brought an already large number of variables and thus limited computation time and memory allocation. We could facilitate the study of the turnpike phenomenon in the optimization of 3D shapes by using parallel methods, based on the symmetry properties of the solutions and the domain decomposition methods.

We note that, from a numerical point of view, there is a significant advantage to observing the turnpike phenomenon since we can design stationary rather than time-dependent solu-

tions and thus solve stationary optimization problems instead of time-varying problems which are much more expensive in terms of computation time and memory allocation. Stationary solutions remain a good approximation. Indeed, the idea of replacing the time-dependent optimal control problem by the stationary one is often used in practical applications without really checking the occurrence of a turnpike phenomenon. This is even more the case in the design of shapes where optimization techniques can be quite limited when dealing with time dependent shapes. Indeed, we present a more applied question of the applications of the turnpike based on the results observed on the heat equation with distributive control. Let us imagine for example the practical example of a room heated by an underfloor heating system divided into several cells that can be activated or not independently of the others. Let us further imagine that it is a given temperature outside and that Dirichlet (or Robin) boundary conditions are set on the walls so that a room temperature of about twenty degrees is desired throughout the day. One possible strategy might be to turn on all the cells until the ideal temperature is reached, then turn them off, bringing the temperature down far from the desired one and iterate. Of course, it is expected that the more you move from an active to an inactive position, the more energy you consume. In other words, this strategy is not at all optimal from an energy consumption point of view. Assuming that we observe the turnpike property, we expect that most of the time during the day only a few cells are activated in order to stay around twenty degrees for example. This means that the other cells, which can be expensive and energy intensive, are less useful than expected. Finding the best location for the cells within the limits of the available pieces can be seen as a problem of designing optimal shapes, whose turnpike behavior would allow us to locate them only where they would have the maximum efficiency.

Bibliography

[1] AMPL : https://ampl.com/products/ampl/. 75

[2] CasADi : https://web.casadi.org/docs/. 105

[3] FreeFem++ : https://doc.freefem.org/documentation/index.html/. 47

[4] IpOpt : https://coin-or.github.io/Ipopt/. 61

[5] F. Alabau-Boussouira, Y. Privat, and E. Trélat. Nonlinear damped partial differential equations and their uniform discretizations. *J. Funct. Anal.*, 273(1) :352–403, 2017. 60

[6] G. Allaire. *Shape optimization by the homogenization method*, volume 146 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2002. 86, 87, 114

[7] G. Allaire. *Conception optimale de structures*, volume 58 of *Mathématiques & Applications (Berlin) [Mathematics & Applications]*. Springer-Verlag, Berlin, 2007. With the collaboration of Marc Schoenauer (INRIA) in the writing of Chapter 8. 13, 81, 93

[8] G. Allaire, F. de Gournay, F. Jouve, and A.-M. Toader. Structural optimization using topological and shape sensitivity via a level set method. *Control Cybernet.*, 34(1) :59–80, 2005. 86

[9] G. Allaire, F. Jouve, and A.-M. Toader. Structural optimization using sensitivity analysis and a level-set method. *J. Comput. Phys.*, 194(1) :363–393, 2004. 86, 110

[10] G. Allaire, A. Münch, and F. Periago. Long time behavior of a two-phase optimal design for the heat equation. *SIAM J. Control Optim.*, 48(8) :5333–5356, 2010. 23, 86, 113

[11] F. Alouges, A. Desimone, and A. Lefebvre-Lepot. Optimal strokes for low reynolds number swimmers : An example. *J. Nonlinear Science*, 18 :277–302, 06 2008. 18, 87

[12] C. Amrouche and V. Girault. On the existence and regularity of the solution of Stokes problem in arbitrary dimension. *Proc. Japan Acad. Ser. A Math. Sci.*, 67(5) :171–175, 1991. 89, 100

[13] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1) :1–36, 2019. 105

[14] S. Arguillère, E. Trélat, A. Trouvé, and L. Younes. Shape deformation analysis from the optimal control viewpoint. *J. Math. Pures Appl. (9)*, 104(1) :139–178, 2015. 13

[15] J. Aubin. Un théorème de compacité. *C. R. Acad. Sci. Paris*, 256 :5042 – 5044, 1963. 28

[16] E. Audusse, F. Bouchut, M.-O. Bristeau, R. Klein, and B. Perthame. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. *SIAM J. Sci. Comput.*, 25(6) :2050–2065, 2004. 129

[17] S. Auliac. *Développement d'outils d'optimisation pour freefem++*. Theses, Université Pierre et Marie Curie - Paris VI, Mar. 2014. :tel-01001631

[18] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning : A survey. *J. Mach. Learn. Res.*, 18(1) :5595–5637, Jan. 2017. 65

[19] J. Berberich, J. Köhler, F. Allgöwer, and M. A. Müller. Indefinite linear quadratic optimal control : strict dissipativity and turnpike properties. *IEEE Control Syst. Lett.*, 2(3) :399–404, 2018. 12

[20] J. Bolte, A. Daniilidis, O. Ley, and L. Mazet. Characterizations of łojasiewicz inequalities : subgradient flows, talweg, convexity. *Trans. Amer. Math. Soc.*, 362(6) :3319–3363, 2010. 33

[21] F. Boyer and P. Fabrie. *Mathematical tools for the study of the incompressible Navier-Stokes equations and related models*, volume 183 of *Applied Mathematical Sciences*. Springer, New York, 2013. 89

[22] D. A. Carlson, A. B. Haurie, and A. Leizarowitz. *Infinite horizon optimal control*. Springer-Verlag, Berlin, 1991. Deterministic and stochastic systems, Second revised and enlarged edition of the 1987 original [MR1117222]. 26

[23] I. I. Cplex. V12. 1 : User's manual for cplex. *International Business Machines Corporation*, 46(53) :157, 2009. 19, 109

[24] J. Dalphin and R. Barros. Shape optimization of a moving bottom underwater generating solitary waves ruled by a forced KdV equation. working paper or preprint, 2017. 129

[25] M. Dambrine and B. Puig. Oriented distance point of view on random sets. *ESAIM Control Optim. Calc. Var.*, 26 :Paper No. 84, 24, 2020. 15, 31, 33

[26] T. Damm, L. Grüne, M. Stieler, and K. Worthmann. An exponential turnpike theorem for dissipative discrete time optimal control problems. *SIAM J. Control Optim.*, 52(3) :1935–1957, 2014. 12

[27] F. De Gournay. Velocity extension for the level-set method and multiple eigenvalues in shape optimization. *SIAM J. Control Optim.*, 45(1) :343–367, 2006. 96, 112

[28] M. C. Delfour and J.-P. Zolésio. *Shapes and geometries*, volume 4 of *Advances in Design and Control*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001. Analysis, differential calculus, and optimization. 34

[29] R. Dorfman, P. A. Samuelson, and R. M. Solow. *Linear programming and economic analysis*. A Rand Corporation Research Study. McGraw-Hill Book Co., Inc., New York-Toronto-London, 1958. 10

[30] C. Esteve-Yagüe, B. Geshkovski, D. Pighin, and E. Zuazua. Large-time asymptotics in deep learning. working paper or preprint, Mar. 2021. :hal-02912516

[31] L. C. Evans. *Partial differential equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, second edition, 2010. 22, 28, 33, 41, 42, 53, 56

[32] L. C. Evans and R. F. Gariepy. *Measure theory and fine properties of functions*. Textbooks in Mathematics. CRC Press, Boca Raton, FL, revised edition, 2015. 28, 29

[33] T. Faulwasser and D. Bonvin. On the design of economic nmpc based on an exact turnpike property. *IFAC-PapersOnLine*, 48(8) :525–530, 2015. 9th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2015. 12

[34] T. Faulwasser and D. Bonvin. On the design of economic nmpc based on approximate turnpike properties. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 4964–4970, 2015.

[35] T. Faulwasser and D. Bonvin. Exact turnpike properties and economic nmpc. *European Journal of Control*, 35 :34–41, 2017. 12

[36] T. Faulwasser, K. Flaßkamp, S. Ober-Blöbaum, and K. Worthmann. Towards velocity turnpikes in optimal control of mechanical systems. *IFAC-PapersOnLine*, 52(16) :490–495, 2019. 11th IFAC Symposium on Nonlinear Control Systems NOLCOS 2019. 12

[37] T. Faulwasser, K. Flaßkamp, S. Ober-Blöbaum, and K. Worthmann. A dissipativity characterization of velocity turnpikes in optimal control problems for mechanical systems. *IFAC-PapersOnLine*, 54(9) :624–629, 2021. 24th International Symposium on Mathematical Theory of Networks and Systems MTNS 2020. 12

[38] T. Faulwasser, L. Grüne, J.-P. Humaloja, and M. Schaller. The interval turnpike property for adjoints. *arXiv : Optimization and Control*, 2020. 12

[39] T. Faulwasser, A. Hempel, and S. Streif. On the turnpike to design of deep neural nets : Explicit depth bounds. *CoRR*, abs/2101.03000, 2021. :journals/corr/abs-2101-03000

[40] T. Faulwasser, M. Korda, C. N. Jones, and D. Bonvin. Turnpike and dissipativity properties in dynamic real-time optimization and economic mpc. In *53rd IEEE Conference on Decision and Control*, pages 2734–2739, 2014. 12

[41] T. Faulwasser, M. Korda, C. N. Jones, and D. Bonvin. On turnpike and dissipativity properties of continuous-time optimal control problems. *Automatica J. IFAC*, 81 :297–304, 2017. 12, 26

[42] T. Faulwasser and A. Murray. Turnpike properties in discrete-time mixed-integer optimal control. *IEEE Control Systems Letters*, 4(3) :704–709, 2020. 12

[43] F. Feppon, G. Allaire, and C. Dapogny. Null space gradient flows for constrained optimization with applications to shape optimization. *ESAIM Control Optim. Calc. Var.*, 26 :Paper No. 90, 45, 2020. 19, 86, 112

[44] R. Fourer, D. M. Gay, and B. Kernighan. Algorithms and model formulations in mathematical programming. chapter AMPL : A Mathematical Programming Language, pages 150–151. Springer-Verlag, Berlin, Heidelberg, 1989. 17, 75

[45] V. Girault and P.-A. Raviart. *Finite element methods for Navier-Stokes equations*, volume 5 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1986. Theory and algorithms. 90

[46] A. Griewank and A. Walther. *Evaluating derivatives*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2008. Principles and techniques of algorithmic differentiation. 64

[47] H. Groemer. On the symmetric difference metric for convex bodies. *Beiträge Algebra Geom.*, 41(1) :107–114, 2000. 34, 127

[48] L. Grüne and R. Guglielmi. Turnpike properties and strict dissipativity for discrete time linear quadratic optimal control problems. *SIAM J. Control Optim.*, 56(2) :1282–1302, 2018. 12

[49] L. Grüne and R. Guglielmi. On the relation between turnpike properties and dissipativity for continuous time linear quadratic optimal control problems. *Math. Control Relat. Fields*, 11(1) :169–188, 2021. 12

[50] L. Grüne, C. M. Kellett, and S. R. Weller. On the relation between turnpike properties for finite and infinite horizon optimal control problems. *J. Optim. Theory Appl.*, 173(3) :727–745, 2017. 12

[51] L. Grüne and M. A. Müller. On the relation between strict dissipativity and turnpike properties. *Systems Control Lett.*, 90 :45–53, 2016. 12

[52] L. Grüne, S. Pirkelmann, and M. Stieler. Strict dissipativity implies turnpike behavior for time-varying discrete time optimal control problems. In *Control systems and mathematical methods in economics*, volume 687 of *Lecture Notes in Econom. and Math. Systems*, pages 195–218. Springer, Cham, 2018. 12

[53] L. Grüne, M. Schaller, and A. Schiela. Exponential sensitivity and turnpike analysis for linear quadratic optimal control of general evolution equations, Dezember 2018. 12

[54] L. Grüne, M. Schaller, and A. Schiela. Sensitivity analysis of optimal control for a class of parabolic PDEs motivated by model predictive control. *SIAM J. Control Optim.*, 57(4) :2753–2774, 2019. 12, 128

[55] L. Grüne, M. Schaller, and A. Schiela. Abstract nonlinear sensitivity and turnpike analysis and an application to semilinear parabolic PDEs. *ESAIM Control Optim. Calc. Var.*, 27 :Paper No. 56, 28, 2021. 125

[56] F. Hecht. New development in freefem++. *J. Numer. Math.*, 20(3-4) :251–265, 2012. 16, 36, 47

[57] A. Henrot and H. Maillot. Optimization of the shape and the location of the actuators in an internal control problem. *Boll. Unione Mat. Ital. Sez. B Artic. Ric. Mat. (8)*, 4(3) :737–757, 2001. 122

[58] A. Henrot, I. Mazari, and Y. Privat. Shape optimization of a Dirichlet type energy for semilinear elliptic partial differential equations. *ESAIM Control Optim. Calc. Var.*, 27(suppl.) :Paper No. S6, 32, 2021. 125, 128

[59] A. Henrot and M. Pierre. *Shape variation and optimization*, volume 28 of *EMS Tracts in Mathematics*. European Mathematical Society (EMS), Zürich, 2018. A geometrical analysis, English version of the French publication [MR2512810] with additions and updates. 25, 81, 86, 93, 110, 111

[60] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. *Optimization with PDE constraints*, volume 23 of *Mathematical Modelling : Theory and Applications*. Springer, New York, 2009. 16, 56, 68, 78, 125

[61] R. J. Hogan. Fast reverse-mode automatic differentiation using expression templates in $C++$. *ACM Trans. Math. Software*, 40(4) :Art. 26, 16, 2014. 75

[62] T. Hytönen, J. van Neerven, M. Veraar, and L. Weis. *Analysis in Banach spaces. Vol. I. Martingales and Littlewood-Paley theory*, volume 63 of *Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge. A Series of Modern Surveys in Mathematics [Results in Mathematics and Related Areas. 3rd Series. A Series of Modern Surveys in Mathematics]*. Springer, Cham, 2016. 27

[63] G. Lance, E. Trélat, and E. Zuazua. Shape turnpike for linear parabolic PDE models. *Systems Control Lett.*, 142 :104733, 9, 2020. 86

[64] I. Lasiecka and R. Triggiani. *Control theory for partial differential equations : continuous and approximation theories. I*, volume 74 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2000. Abstract parabolic systems. 60

[65] I. Lasiecka and R. Triggiani. *Control theory for partial differential equations : continuous and approximation theories. II*, volume 75 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2000. Abstract hyperbolic-like systems over a finite time horizon. 60

[66] H. Le Dret. *Nonlinear elliptic partial differential equations*. Universitext. Springer, Cham, 2018. An introduction, Translated from the 2013 French edition [MR3235838]. 28

[67] X. J. Li and J. M. Yong. *Optimal control theory for infinite-dimensional systems*. Systems & Control : Foundations & Applications. Birkhäuser Boston, Inc., Boston, MA, 1995. 24

[68] E. H. Lieb and M. Loss. *Analysis*, volume 14 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, second edition, 2001. 24, 32, 43

[69] J.-L. Lions. *Optimal control of systems governed by partial differential equations*. Translated from the French by S. K. Mitter. Die Grundlehren der mathematischen Wissenschaften, Band 170. Springer-Verlag, New York-Berlin, 1971. 16, 24, 78

[70] J. Lohéac and A. Munnier. Controllability of 3D low Reynolds number swimmers. *ESAIM Control Optim. Calc. Var.*, 20(1) :236–268, 2014. 18

[71] L. W. McKenzie. Turnpike theorems for a generalized leontief model. *Econometrica*, 31(1/2) :165–180, 1963. 10

[72] M. Mitrea and M. Wright. Boundary value problems for the Stokes system in arbitrary Lipschitz domains. *Astérisque*, (344) :viii+241, 2012. 89, 100

[73] B. Mohammadi and O. Pironneau. *Applied shape optimization for fluids*. Numerical Mathematics and Scientific Computation. The Clarendon Press, Oxford University Press, New York, 2001. Oxford Science Publications. 13, 96, 112

[74] E. Nelson. Analytic vectors. *Ann. of Math. (2)*, 70 :572–615, 1959. 25

[75] H. Nersisyan, D. Dutykh, and E. Zuazua. Generation of 2d water waves by moving bottom disturbances. *IMA Journal of Applied Mathematics*, 80(4) :1235–1253, 2015. :10.1093/imamat/hxu051

[76] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006. 112

[77] R. Ou, M. Baumann, L. Grüne, and T. Faulwasser. A simulation study on turnpikes in stochastic lq optimal control. *ArXiv*, abs/2010.12201, 2020. 12

[78] B. Perthame and C. Simeoni. A kinetic scheme for the Saint-Venant system with a source term. *Calcolo*, 38(4) :201–231, 2001. 129

[79] D. Pighin. The turnpike property in semilinear control. *ESAIM Control Optim. Calc. Var.*, 27 :Paper No. 48, 48, 2021. 125

[80] A. Porretta and E. Zuazua. Long time versus steady state optimal control. *SIAM J. Control Optim.*, 51(6) :4242–4273, 2013. 12, 14, 26, 125

[81] A. Porretta and E. Zuazua. Remarks on long time versus steady state optimal control. In *Mathematical paradigms of climate science*, volume 15 of *Springer INdAM Ser.*, pages 67–89. Springer, [Cham], 2016. 12, 14, 26, 40

[82] Y. Privat, E. Trélat, and E. Zuazua. Optimal shape and location of sensors for parabolic equations with random initial data. *Arch. Ration. Mech. Anal.*, 216(3) :921–981, 2015. 23, 113

[83] Y. Privat, E. Trélat, and E. Zuazua. Optimal observability of the multi-dimensional wave and Schrödinger equations in quantum ergodic domains. *J. Eur. Math. Soc. (JEMS)*, 18(5) :1043–1111, 2016. 23, 113

[84] A. Rapaport and P. Cartigny. Turnpike theorems by a value function approach. *ESAIM Control Optim. Calc. Var.*, 10(1) :123–141, 2004. 12

[85] P.-A. Raviart and J.-M. Thomas. *Introduction à l'analyse numérique des équations aux dérivées partielles*. Collection Mathématiques Appliquées pour la Maîtrise. [Collection of Applied Mathematics for the Master's Degree]. Masson, Paris, 1983. 16, 47, 69

[86] P. A. Samuelson. The periodic turnpike theorem. *Nonlinear Anal.*, 1(1) :3–13, 1976. 12

[87] J. M. Sanz-Serna. Symplectic Runge-Kutta schemes for adjoint equations, automatic differentiation, optimal control, and more. *SIAM Rev.*, 58(1) :3–33, 2016. 17, 59, 66, 104

[88] C. D. Sogge. *Hangzhou lectures on eigenfunctions of the Laplacian*, volume 188 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, NJ, 2014. 32, 40

[89] E. Trélat. *Contrôle optimal*. Mathématiques Concrètes. [Concrete Mathematics]. Vuibert, Paris, 2005. Théorie & applications. [Theory and applications]. 10, 16, 66, 73, 102

[90] E. Trélat. Linear turnpike theorem. preprint, Oct. 2020. :hal-02978505

[91] E. Trélat and C. Zhang. Integral and measure-turnpike properties for infinite-dimensional optimal control systems. *Math. Control Signals Systems*, 30(1) :Art. 3, 34, 2018. 12, 14, 26, 31, 40

[92] E. Trélat, C. Zhang, and E. Zuazua. Steady-state and periodic exponential turnpike property for optimal control problems in Hilbert spaces. *SIAM J. Control Optim.*, 56(2) :1222–1252, 2018. 12

[93] E. Trélat and E. Zuazua. The turnpike property in finite-dimensional nonlinear optimal control. *J. Differential Equations*, 258(1) :81–114, 2015. 11, 13

[94] F. Tröltzsch. *Optimal control of partial differential equations*, volume 112 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2010. Theory, methods and applications, Translated from the 2005 German original by Jürgen Sprekels. 16, 81, 83, 125

[95] M. Tucsnak and G. Weiss. *Observation and control for operator semigroups*. Birkhäuser Advanced Texts : Basler Lehrbücher. [Birkhäuser Advanced Texts : Basel Textbooks]. Birkhäuser Verlag, Basel, 2009. 57, 58, 74

[96] N. P. van Dijk, K. Maute, M. Langelaar, and F. van Keulen. Level-set methods for structural topology optimization : a review. *Struct. Multidiscip. Optim.*, 48(3) :437–472, 2013. 111

[97] A. Wächter and L.-T. Biegler. On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming 106(1), pp. 25-57*, 2006. 17, 36, 61

[98] S. Walker and E. Keaveny. Analysis of shape optimization for magnetic micro-swimmers. *SIAM Journal on Control and Optimization*, 51, 01 2013. 87

[99] J. C. Willems. Dissipative dynamical systems. I. General theory. *Arch. Rational Mech. Anal.*, 45 :321–351, 1972. 12, 26

[100] M. Zanon, L. Grüne, and M. Diehl. Periodic optimal control, dissipativity and MPC. *IEEE Trans. Automat. Control*, 62(6) :2943–2949, 2017. 12

[101] A. J. Zaslavski. Existence and structure of optimal solutions of infinite-dimensional control problems. *Appl. Math. Optim.*, 42(3) :291–313, 2000.

[102] A. J. Zaslavski. *Turnpike theory of continuous-time linear optimal control problems*, volume 104 of *Springer Optimization and Its Applications*. Springer, Cham, 2015. 12

[103] E. Zuazua. Large time control and turnpike properties for wave equations. *Annual Reviews in Control*, 44 :199–210, 2017. 14