ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

MATEMATICA

Ciclo 37

Settore Concorsuale: 01/A5 - ANALISI NUMERICA

Settore Scientifico Disciplinare: MAT/08 - ANALISI NUMERICA

NOZZLE AND MULSTREG: NUMERICAL OPTIMIZATION TOOLS FOR ENERGY INDUSTRY BLACK-BOX OPTIMIZATION FOR CLEAN ENERGY TECHNOLOGIES

Presentata da: Filippo Marini

Coordinatore Dottorato

Giovanni Mongardi

Supervisore

Margherita Porcelli

Esame finale anno 2025

Alma Mater Studiorum — Università di Bologna

DOTTORATO DI RICERCA IN
MATEMATICA

Ciclo XXXVII

Settore Concorsuale: 01/A5 - ANALISI NUMERICA

Settore Scientifico Disciplinare: MAT/08 - ANALISI NUMERICA

NOZZLE and MU$^{\ell}$STREG: Numerical Optimization Tools for Energy Industry

Black-Box Optimization for clean energy technologies

Presentata da: Filippo Marini

Coordinatore Dottorato:
Prof. Giovanni Mongardi

Supervisore:
Prof.ssa Margherita Porcelli
Prof.ssa Elisa Riccietti

Esame finale anno 2025

Borsa di dottorato del Programma Operativo Nazionale Ricerca e Innovazione 2014-2020 (CCI 2014IT16M2OP005), risorse FSE REACT-EU, Azione IV.4 “Dottorati e contratti di ricerca su tematiche dell’innovazione” e Azione IV.5 “Dottorati su tematiche Green”, CUP J35F21003200006.

Contents

Abstract

In this work, we study how to exploit Derivative-Free Optimization (DFO) and Black-Box Optimization (BBO) in the design and validation phases of a cooling system in a gas turbine. For the first phase, we define NOZZLE, a numerical model of a section of the cooling system, and we use an optimization method to obtain an efficient design of the section; for the second one, we develop MU$^\ell$STREG: an optimization method to enhance the validation procedures of an entire cooling system.

NOZZLE is a Black-Box function that simulates an impingement cooling system for a turbine nozzle starting from a model well-known in the literature that correlates the design features of the cooling system with efficiency parameters. The optimization model is defined as a mixed-variable constrained BBO problem and we numerically illustrate how to use DFO algorithms to find a reference solution that is useful for practitioners.

MU$^\ell$STREG is a new multilevel stochastic framework for the solution of optimization problems where the value of the objective function is affected by random noise. In this work, we focus on data-fitting problems with random uncertainty that arise in the validation phase of a complete cooling system in a gas turbine. The proposed approach uses random regularized first-order models that exploit an available hierarchical description of the problem, being either in the classical variable space or in the function space, meaning that different levels of accuracy for the objective function are available. The convergence analysis of the method is conducted and its numerical behavior is tested on the solution of finite-sum minimization problems. Indeed, the multilevel framework is tailored to the solution of such problems resulting in fact in a nontrivial variance reduction technique with adaptive step-size that outperforms standard approaches when solving nonconvex problems. Differently from classical deterministic multilevel methods, our stochastic method does not require the finest approximation to coincide with the original objective function. This allows us to avoid the evaluation of the full sum in finite-sum minimization problems, opening to the solution of classification problems with large data sets.

Keywords Cooling systems, gas turbine, Black-Box Optimization, Derivative-Free Opti-

mization, direct search algorithm, multilevel methods, stochastic optimization, adaptive regularization, variance reduction methods.

Introduction

One of the most important challenges of the present and future is meeting the growing demand for energy from all countries around the world. Moreover, this demand must be met in a way that has the lowest environmental impact possible. From this point of view, the last few decades have witnessed an extraordinary development of power generation technologies from renewable sources, such as solar power and wind power, which are in addition to the well-known hydropower. Some of the main advantages of these renewable energy sources are that they are more evenly distributed over the planet and emit far fewer greenhouse gases than fossil energy sources. However, renewable energy sources are also intermittent, since they depend mainly on atmospheric (solar power and wind power) or hydrogeological (hydroelectricity) phenomena, which are unpredictable and uncontrollable; this difficulty will be overcome in the long run with the increase and diversification of renewable energy installations and the redesign of energy distribution grids, but in the short and medium term it is essential to integrate these new energy sources with existing ones.

In this sense in the current context of power generation technologies, gas turbines

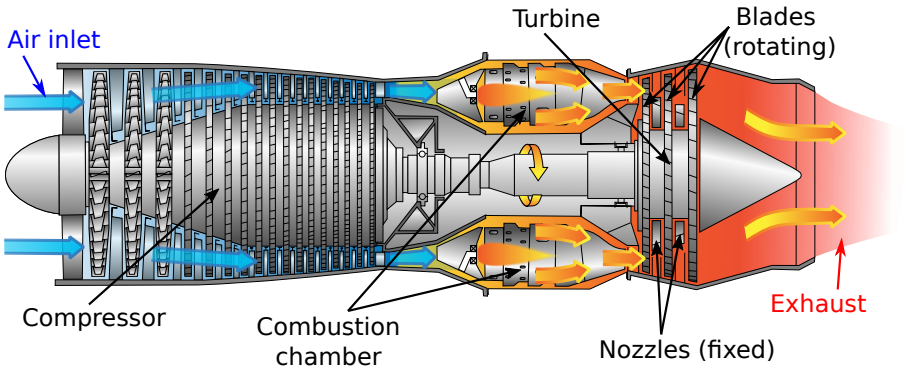

Figure 1: Simple representation of a gas turbine

play a key role. Indeed, it is a technology that has been successfully used for more than fifty years, during which it has been studied, developed, and spread enormously, with an ability to provide efficient, flexible power generation with lower greenhouse gas emissions than technologies using other fossil fuels. These characteristics make gas turbines an ideal technology to complement renewable energy sources during the energy transition. For this reason, optimizing the performance of gas turbines has become a priority. Another important reason for optimizing the performance of turbines is the possibility of using them with non-fossil fuels, such as hydrogen, whose combustion does not produce greenhouse gases.

This thesis, supported by the program "Programma Operativo Nazionale Ricerca e Innovazione 2014-2020" - Azione IV.5 "Dottorati e contratti di ricerca su tematiche green"[1], takes place in this context: developing, implementing and validating algorithms and numerical tools that helps increasing the efficiency of a gas turbine through optimization methods.

A gas turbine (schematically shown in Figure 1) is a system that converts thermal energy from gas combustion into mechanical energy through the Brayton cycle. Its operation consists of three stages. In the first stage, a compressor takes external air and channels it at a certain pressure into the combustion chamber, in the so-called primary flow. Then, in the second stage, the air is heated in the combustion chamber by burning gas, typically methane, increasing the temperature and specific volume of the air-gas mixture. Finally, the expanded mixture flows through the turbine, which is composed of fixed (the nozzles) and rotating (the blades) elements; during this phase, the movement of the blades generates kinetic energy that can be used for various purposes, such as, for instance, run an electric power generator [42].

To achieve higher efficiency at the same pressures, it is necessary for the gas combustion temperature to be as high as possible, thus raising it above the melting point of the materials used to craft the turbine nozzles and blades. Therefore, it is essential to design an efficient cooling system for all the turbine components.

This thesis aims to develop a suitable optimization framework to improve the design of a turbine cooling system.

[1]"Programma Operativo Nazionale Ricerca e Innovazione 2014-2020 (CCI2014IT16M2OP005)" - Azione IV.5 "Dottorati e contratti di ricerca su tematiche green" XXXVII ciclo, code DOT1303154-4, CUP J35F21003200006.

Depending on the size of a turbine, the components that need to be cooled can number in the hundreds or thousands, making the cooling system of a gas turbine extremely complex. Therefore, its design and fine-tuning involve several stages in which numerical models and optimization methods can be used. In this thesis, we focus on two of these stages. The first one is the numerical modeling of a section of the cooling system involving a single type of component and using optimization methods to get the design of that section to improve its performance. The second one is the development of numerical methods that improve the final validation procedures of the design. In the following two chapters, we study two possible applications of optimization to these two stages. Indeed, in Chapter 1 we study and implement a numerical tool called NOZZLE that coupled with a suitable optimization method can be used to design a cooling system for a specific type of turbine component. In Chapter 2 we develop a general numerical method called MU$^{\ell}$STREG that can be used for data-matching during the validation phase of the design of an entire cooling system.

For both applications, we consider Derivative-Free Optimization (DFO) [8, 26], a field of nonlinear optimization that addresses the optimization of functions whose derivatives are not available. Indeed, the gradient of a function may not be computable for many reasons; for example, the function might be defined as the output of a simulation software or other numerical procedures, so the analytical expression of the function might be unknown or not available for licensing reasons, and consequently the same happens for its gradient. In another situation the value of the objective function is obtained by solving a particularly complex system of equations thus the definition of a closed formulation for the gradient is even more complex or impossible. The use of DFO methods is also suitable in cases where the analytical formulation of the derivatives of the objective function is known: in fact, in some cases, the size and complexity of the problem make the computational cost of evaluating or estimating the derivatives too high; in other cases, the objective function is noisy (and thus its derivatives) so that the evaluation or approximation of the derivatives is unreliable and therefore useless. When we have access only to the input and the output of the objective function and its derivative is neither available nor efficiently approximable we deal with *black-box functions*, and when DFO methods are applied to such functions we refer to Black-Box Optimization (BBO) [9, 26].

NOZZLE: a Black-Box Optimization tool

As we mentioned earlier, one of the main topics of this work is the development of a numerical tool for optimizing the design of a specific device for cooling one type of turbine component: a fixed nozzle.

Before getting more specific, let us provide an outline of how the cooling of turbine components occurs.

Cooling in gas turbines. Primarily, the cooling system is supplied by diverting a fraction of the primary flow of air into a so-called secondary flow within a network of ducts, which distributes the air to all the turbine components that need to be cooled. Finally, the air is expelled outside the turbine along with the exhaust gases.

Component cooling is essentially accomplished by heat transfer. In particular, the air in the secondary flow exchanges heat with the metal surface of the component which is heated by the hot flow. There are three main ways in which heat exchange occurs: conduction, convection, and radiation. We are only interested in convection heat transfer, which is the basis of the cooling system discussed in this thesis.

Between two systems there is heat exchange by convection when in addition to having a transfer of energy due to an interaction between elementary volumes with more energy (i.e., warmer) with elementary volumes with less energy (i.e., colder), as in conduction, there is also a transfer of internal energy from one point to another in the system, due to the relative motion that the volumes constituting a continuous medium have with respect to each other. As in gas turbines, this phenomenon involves the cooling air and the solid parts of the components, thus convection affects only the fluid. There are several ways to implement a cooling system based on convective heat transfer, but now we focus on one of the most used ones: the impingement cooling system.

Impingement cooling. For a nozzle, an impingement cooling device is an internal cooling system implemented by creating near the inner wall of the nozzle a series of jets allowing the cooling air to hit directly against the wall itself. This increases the turbulence of the internal flow, causing heat exchange through convection. An example of an impingement system is shown in Figure 2, and it is easy to realize that the improvement of the design of the impingement insert is the main issue. So far, there is no generalized and rigorous way to approach this issue and, in most cases, this phase depends on the experience of the engineers that are working at the moment on that particular machine.

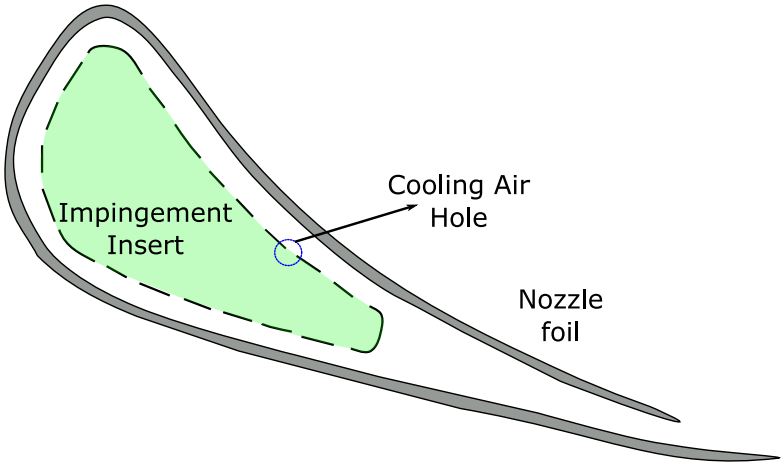

Figure 2: Section of an impingement cooling system of a nozzle. The green area is the section of the impingement insert with holes on its boundary.

Moreover, whenever a new design for the impingement insert is proposed, due to the hydro- and thermodynamics involved, it must be tested in a Computational Fluid Dynamics and Thermodynamics simulation to validate the expected performance and to check that there is no violation of any engineering constraint. This kind of simulation often requires a huge computational effort. Therefore, we propose the definition of the black-box function NOZZLE which together with a derivative-free technique constitutes a fast and automatic method to improve the design of the impingement cooling system. We want NOZZLE to be a (computationally) cheap simulator of an impingement cooling system with outputs sufficiently close to reality and that takes into account the engineering constraints that are present in such a real-life application. Defining NOZZLE as a suitable black-box function to be used in a DFO framework allows us great freedom in defining all its components, since the only information used by the DFO framework are the inputs and outputs. Indeed, DFO is the most suitable approach for this kind of industrial application, see e.g. [9, 45, 63].

As we explain in more detail in Chapter 1, the analysis and development of the Black-Box function and DFO framework to solve this particular industrial optimization problem lead us to formulate a constrained BBO problem (see (1.35)) with real and categorical variables, i.e. non-numeric, unconstrained and implicitly unordered variables. There are two main issues in this formulation that influence the choice of the DFO approach: the presence of black-box constraints and the use of mixed continuous and categorical

variables (see Section 1.1.2.3).

DFO literature overview. Focusing on Derivative-Free Optimization methods for constrained BBO, three different approaches can be found in the literature: filter approaches, model-based approaches, and penalty approaches. For a more complete overview refer to [47].

Filter methods aim to address black-box constraints and algebraic constraints by concurrently minimizing both objective and constraint violation. Audet and Dennis introduced in [6] a pattern-search technique for general constrained optimization that accepts steps that improve either the objective or the violation of black-box constraints. Further approaches can be found in [30, 64].

Model-based approaches define a surrogate problem by building models to replace the simulation-based functions (objective and constraints). Powell in [65] develops a direct search method for constrained optimization which approximates the objective and constraint functions using linear models defined over a simplex. Bürmen *et al.* in [19] presented a version of Mesh Adaptive Direct Search (MADS) applied to a surrogate problem defined using strongly convex quadratic model for the objective function and linear models for the constraints. In [59], a trust-region method employing fully linear models of both constraint and objective functions was developed. An alternative method employs interpolating radial basis function surrogates of the objective and constraint functions (CONORBIT) [67]. Finally, in [27] a simplex-gradient-based approach is considered to approximate normal cones when black-box constraints are quantifiable.

Regarding penalty methods, Audet and Dennis [7] propose a progressive-barrier method within MADS method with quadratic penalty for relaxable black-box constraints and an extreme-barrier penalty for unrelaxable ones; in [40], an extreme-barrier penalty is used again to handle unrelaxable constraints, while an exact penalty is used for relaxable ones, everything within a Directional Direct Search (DDS) framework; the paper [31] proposes a line-search method with a sequence of quadratic penalty functions to address non-differentiable constraint and objective functions; Sampaio and Toint propose a derivative-free variant of trust-funnel method to deal equality constraints without using neither merit functions nor filters, see [70]. Finally, two augmented Lagrangian frameworks, one developed in [60] where the merit function is defined using Gaussian process models of the objective and constraint functions, and the other presented in [52] where the linear constraints are treated outside the augmented Lagrangian merit function.

The literature comprising DFO algorithms that handle mixed variables is not very extensive. Papers by Audet and Dennis [5], by Lucidi *et al.* [54] and by Abramson [2] consider the presence of categorical variables. In particular, the work [2] extends the MADS algorithms for solving constrained mixed variable optimization problems. These algorithms have been successfully applied to relevant engineering applications, see e.g. [1, 45]. Finally, we mention the recent work [62] where the pattern search method Brute Force Optimizer (BFO) proposed in [61] for solving problems with continuous and discrete variables, has been extended to handle categorical variables. More details on BFO are provided in Section 1.3.2.

To summarize, in Chapter 1 we develop a possible model for optimizing the efficiency of an impingement cooling system in a nozzle as a constrained BBO problem. The study of the problem leads to the development and implementation of NOZZLE, a black-box function that simulates impingement cooling in a turbine nozzle. We also provide a description of a DFO approach to couple with NOZZLE to solve the constrained BBO problem and we validate it with numerical tests derived from real-world scenarios.

MU$^\ell$STREG: a Multilevel Stochastic Gradient method

The second topic covered in this work concerns one of the final stages in the design of a cooling system. Specifically, once the design phase of a complete cooling system has been concluded, it is necessary to build and test it on the turbine. During the testing phase, a certain number N of measurements of characteristic quantities of the cooling system (e.g., flow rates or pressures) distributed evenly over the entire cooling system network are taken, and the agreement between the measurements and the quantities predicted by the numerical model is checked. If there is a discrepancy between the measurements and the numerical results, a correction of the model is made, which consists of tuning a certain number n of parameters, for instance, the discharge coefficients at certain nodes in the network of ducts that distribute the secondary flow of cooling air to all the cooling systems of the individual components. This operation is called *data-matching* (or *data-fitting*) and can be interpreted as a minimization problem of the form

$$\min_{x \in \mathbb{R}^n} f(x) \tag{1}$$

where $x \in \mathbb{R}^n$ are the parameter to be tuned and f is a function that models the error between the N measurements and the numerical data. This type of problem is suitable to be addressed with DFO approaches because, within the definition of f, we have a numerical model (or a simulator) of the cooling system, making it extremely difficult to obtain an analytical expression for f and its derivative. This problem poses further issues. The first concerns its scale: indeed, the number n of parameters and the number N of measurements influence the computational cost of any minimization method, and if n and/or N increase too much the computational effort and the time needed to obtain a solution will be very high. In an industrial context, the complexity of the machinery being tested often results in a high number n of parameters (i.e. variables) to tune, and the reliability of the testing process requires a large number N of measurements. Thus, our data-fitting problem is on a very large scale. Another issue arises from the fact that the definition of f involves measurements that may be affected by random uncertainty or, more generally, by random noise that cannot be neglected. These challenges make our data-fitting problem a large-scale stochastic problem. Many other modern applications require the solution of large-scale stochastic optimization problems, i.e., the minimization of functions whose value can only be computed with some noise [3]. This can happen, for instance, in medicine in the design of laboratory experiments to collect data on the efficacy of a new drug, in traffic engineering to set the timing of traffic lights in a traffic network, or in business to make short- and long-term investments decisions [71].

Moreover, being N the number of measurements taken, f could be defined as an average over the number of measurements, so the problem may have the following formulation:

$$\min_{x \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^{N} f_i(x) \tag{2}$$

where f_i defines the error on the i-th measurement. The problem (2) is called *finite-sum* minimization problem.

Stochastic variance reduced gradient methods. In the particular case of problem (2), where the objective function is defined by finite sums, many strategies have been proposed to handle cases where the number of elements N is very large. Many

of these strategies are based on (random) subsampling and are mainly variations of the Stochastic Gradient (SG) method. The main problem with such methods is the tuning of the step size, which is a difficult task that requires trial and error. Moreover, to ensure convergence of the methods it is often necessary to employ a decreasing step size, which leads to really slow convergence. In order to avoid this issue, variance reduction methods have been proposed in the literature [16], i.e., techniques to reduce the variance of the stochastic gradient estimates. Among them, we focus on gradient aggregation methods, which improve the quality of the search directions by storing gradient estimates corresponding to samples employed in previous iterations, updating one (or some) of these estimates in each iteration, and defining the search direction as a weighted average of these estimates. Among these we mention SVRG and SAGA [28, 44, 66]. SVRG was originally proposed in [44] with a convergence analysis for smooth and strongly convex objective function. Since then the practical behavior of the method and strategies to fix the hyperparameters have been studied in [4] and [66] for both the convex and nonconvex cases.

Multilevel methods. In classical scientific computing a powerful class of methods has been developed to cope with structured optimization problems where the limiting factor is the size n of the variable: *multilevel methods*. When the structure of the problem at hand allows for a hierarchical description of the problem itself, these methods reduce the cost of the problem's solution by considering a hierarchy of surrogate functions defined on subspaces of progressively smaller dimensions. Thanks to this, they achieve not only a considerable reduction in computational effort but also an improvement in the solution quality in various applications, spanning from the solution of partial differential equations to image reconstruction [38, 48, 49, 57].

As a natural extension of multigrid methods [18] to a nonlinear context, multilevel methods were first proposed by Nash through the MG/OPT framework [57] and later extended to trust-region schemes [38]. Recently these methods have been extended to other contexts: high-order models [20], non-smooth optimization [49], machine learning [36, 37, 46]. A multilevel method that exploits hierarchies in the function space has been explored in [17], where a multilevel variance reduction method is proposed for deterministic convex problems of the form (2.2) leveraging the multilevel scheme of MG/OPT developed in [57]. Recent research [36] proposes a (deterministic) multilevel version of the Objective Function Free Optimization (OFFO) method that does not require function

evaluations and that is based on the classical multilevel scheme constructed on the variable space. Existing multilevel methods are however limited to a deterministic context and are thus unsuitable to address stochastic optimization problems. Moreover, most of them have always been used on problems whose structure allows for the construction of a hierarchy in the variables space, such as problems arising from the discretization of infinite dimensional ones on selected grids. However, in many modern applications, the limiting factor can be the accuracy of the function estimates rather than the size of the model. Indeed, in this thesis we focus on this case: the objective function is the outcome of a simulation or arises from a data-fitting application over a large dataset.

Derivative free optimization. Since the objective function of our problem is noisy, the same occurs to its derivatives making them unreliable. Thus, as we said before, the problem we are considering is suitable to be addressed with DFO approaches. In order to address large-scale problems (as ours), in the last years there have been some contributions to DFO literature that carry an idea close to that of multilevel methods to alternate between accurate steps and cheap steps using more or less information. One of them can be found in full-low evaluation derivative-free optimization for direct search methods [13, 69]. Another technique that has been considered to reduce the cost of the problems is random subset selection [22]. In [14] the authors propose a Levenberg-Marquardt adaptation of the Stochastic Optimization with Random Models (STORM) framework (see [24]) for stochastic derivative-free least squares problems. As in our work, the step size in this context is updated through a regularization parameter. We inherited from this work the dependence of the regularization parameter from the norm of the gradient (cf. (2.3) below) and the definition of accurate models (cf. Definition 1). The recent literature on variants of the standard trust-region method based on the use of random models is very extensive, we refer to [10–12, 41, 68] to name a few and references therein.

In Chapter 2 we present the MU$^\ell$STREG method to address the general problem (1). The proposed approach is an extension of multilevel methods to a stochastic setting and uses random regularized first-order models that exploit an available hierarchical description of the problem, being either in the classical variable space or in the function space, meaning that different levels of accuracy for the objective function are available. We provide a convergence analysis and we perform some numerical tests for an adaptation of MU$^\ell$STREG for binary classification problems of the form (2). Indeed, the

multilevel framework is tailored to the solution of such problems resulting in fact in a nontrivial variance reduction technique with adaptive step-size that outperforms standard approaches when solving nonconvex problems. Remarkably, our method allows us to avoid the full evaluation of the objective sum opening at the solution of classification problems with large data sets.

Thesis contributions

The main contributions in Chapter 1, devoted to the design of an impingement cooling system and the development of NOZZLE, are the following.

- The new BBO model for the optimization of the design of an impingement cooling system for the nozzle of a gas turbine that results in a new example derived from a real-world application.
- A simple but still quite accurate numerical simulator that has been implemented and validated to be used by the scientific community as a test case for any kind of BBO method.
- A Derivative-Free Optimization approach that, coupled with the use of our black-box function, defines an automatic and reliable procedure for the optimization of the efficiency of a cooling system in a gas turbine.
- A standalone version of NOZZLE is available in the S2PMJ [39] format on `GitHub` page `https://github.com/GrattonToint/S2MPJ/blob/main/matlab_problems/NOZZLEfp.m` in Matlab, Python and Julia.

These contributions are also presented in the paper

- L. Cocchi, F. Marini, M. Porcelli, and E. Riccietti, "Black-box optimization for the design of a jet plate for impingement cooling," *Optim. Eng.*, 2025. DOI: `10.1007/s11081-025-09981-0`.

Chapter 2, dedicated to the study and development of MU$^\ell$STREG, brings the following contributions.

- MU$^\ell$STREG is the first stochastic framework for multilevel methods, that are currently limited to the deterministic case.
- The proposed multilevel framework allows for hierarchies in the function space, i.e., building by considering function approximation with variable accuracy.
- The developed stochastic multilevel framework allows us to overcome the limiting factor of classical deterministic multilevel methods whose convergence theory requires the fine level function to coincide with the original target function, so that such methods cannot be used in cases where the original problem has a too large size.
- MU$^\ell$STREG is the first stochastic analysis of *first-order* adaptive regularization methods (our multilevel framework also covers the classical one-level case).
- Our method can be specialized for finite-sum problems and offers a variance reduction technique with an adaptive step size that outperforms mini-batch SVRG on nonconvex problems.

These contributions are also presented in the preprint:

■ F. Marini, M. Porcelli, and E. Riccietti, *A multilevel stochastic regularized first-order method with application to training*, 2024. arXiv: 2412.11630 [math.OC]. [Online]. Available: https://arxiv.org/abs/2412.11630.

Chapter 1

NOZZLE

In this chapter, we define the NOZZLE black-box function that simulates the operation of an impingement cooling system for a fixed nozzle, and we will use it to set up a derivative-free framework to optimize the cooling of the nozzle. Starting from the well-known model by Florschuetz *et al.* [32, 33] we develop a numerical model for a simulation that includes the estimation of temperature distribution both on the internal and external wall of the nozzle, and the estimation of the outlet pressure. Once the simulator is defined, we embed it in a Black-Box Optimization framework in order to optimize the design of the impingement system so that the highest possible cooling efficiency is achieved.

In more detail, we look for an insert design that maximizes the Heat Transfer Coefficient (HTC) h_c of the coolant in the feasible set $V \subset \mathbb{R}^n$ defined by the engineering constraints. If we identify the main geometric variables that characterize the design of the impingement insert with the vector $\mathbf{v} \in \mathbb{R}^n$, we are interested in the solution of the optimization problem:

$$\begin{aligned} \max_{\mathbf{v}\in\mathbb{R}^n} \quad & H(\mathbf{v}) \\ \text{s.t.} \quad & \mathbf{v} \in V, \end{aligned} \tag{1.1}$$

where the function $H : \mathbb{R}^n \to \mathbb{R}$ is a scalar-valued function that models the correlation between the geometric variables $\mathbf{v}$ and the value of the HTC h_c. Specifically, since the HTC is a nonconstant distribution within the cooling system, the function H has to return a scalar value that is representative of the overall heat transfer as, for example,

the mean, the quadratic mean, or the root mean square (RMS) of the HTC distribution.

The numerical solution of (1.1) poses several challenges that are due to the fact that the overall black-box problem is a mixed variable problem: some geometric variables are continuous and one is categorical, that is non-numeric, unconstrained and implicitly unordered. In addition, the feasible set V is determined by black-box constraints. We, therefore, propose to use a new flexible and robust penalized DFO approach that handles the constraints using an ℓ_1-penalty function and the Brute Force Optimizer (BFO) [61, 62], which is able to handle the mentioned above problem peculiarities, for the solution of the resulting penalized problem.

Chapter 1 is organized as follows. In Section 1.1 we illustrate the modeling for problem (1.1) that we employ by defining the geometric variables $\mathbf{v}$, the function H and the inequalities characterizing the feasible set V. In Section 1.2 we describe the construction of the function H and of the constraints as black-box functions and in Section 1.3 we describe the proposed strategy to solve the arising optimization problem through a DFO approach. Finally, we numerically illustrate in Section 1.4 that our strategy allows us to automatically find an improved design for the cooling system taking into account the main engineering requirements.

1.1 Optimization model for the impingement cooling system

The cooling system of a gas turbine nozzle is broadly structured as shown in Figure 1.1. The nozzle is surrounded by hot gas, characterized by a temperature T_g and an HTC h_g, coming from the combustion chamber. Inside the nozzle we have a duct called the impingement insert where the coolant fluid flows at pressure p_c^{in} and temperature T_c, the fluid exits from the insert through orifices, it hits the inner wall of the nozzle plate and finally exits the nozzle through an opening at the tail of the nozzle with pressure p_c^{out}.

When the cool fluid impinges on the inner wall of the nozzle, there is a heat exchange between the surface and the fluid, whereby the cool air subtracts heat from the nozzle wall that has been heated by the hot gas on the outside. Because of the thermal conductivity of the wall, by subtracting heat from inside the nozzle we are then able to cool the external wall of the nozzle. In this way, we reduce the damage caused to the nozzle by

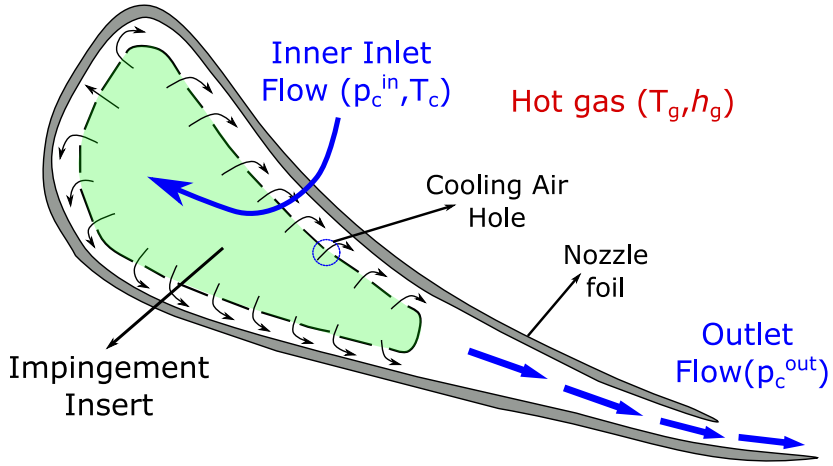

Figure 1.1: Section of an impingement cooling system of a nozzle

the high temperature of the surrounding external gas.

For an impingement cooling system, the main component is the impingement insert; in particular, its efficiency depends on the position of the insert inside the nozzle and on the size and disposition of the orifices on its surface.

A fluid that is often used in turbine cooling systems is air, which is drawn in from the surrounding environment. Most of this flow is used in the primary flow for the fuel combustion process, while a portion is diverted as secondary flow into the cooling system. Throughout our discussion, we assume that the cooling fluid used is air.

The air employed in the cooling system does not actively contribute to work generation by the gas turbine engine. Moreover, coolant ejection in the main flow can generate secondary flows and mixing losses which may reduce the aerodynamic efficiency of the airfoil [42]. These evidences justify the need to maximize the efficiency of cooling systems, i.e., obtain the desired cooling effect using a (minimum) fixed coolant mass flow rate.

1.1.1 The objective function

A major parameter for evaluating heat transfer coefficients is the nondimensional Nusselt number Nu, which is the ratio of the heat flux exchanged by convection to that exchanged by conductivity, in this way the measurement of the HTC is related only to the properties of the cooling air. The number Nu is related to the coolant HTC h_c by the following

relation:

$$\mathrm{Nu} = \frac{h_c d}{k_c}; \tag{1.2}$$

where the constant k_c is the thermal conductivity of the cooling air and d is the diameter of the orifices through which the coolant flow occurs. Therefore, once we get the distribution of Nu, we are able to get h_c inverting (1.2), giving

$$h_c = \frac{k_c \mathrm{Nu}}{d}. \tag{1.3}$$

Hence our objective function H has as its core a model for a correlation between geometric parameters and (the distribution of) the Nusselt number within the cooling system.

Given the wide use of impingement cooling systems, many mathematical models have been developed over time to describe their functioning and study the correlations between design features and performance. An extensive collection of impingement heat transfer correlations can be found in the work by Zuckerman and Lior [72].

The mathematical model we choose to build the objective function H is the experimental correlation developed by L. W. Florschuetz *et al.* [32–34]. This model was chosen for several reasons. First, it is a relatively simple model, since starting from the characteristics of the cooling system it returns a one-dimensional distribution of the Nusselt number; therefore, it is a model with a very low computational cost and the returned results are simple to interpret.

Another reason why it was chosen is that it was developed for an array of orifices placed on a single plate, which is the configuration closest to that of our interest for the design of the impingement insert; in fact, the insert is made from a metal plate that is drilled following the desired layout and then it is bent to obtain the final shape (see Figure 1.1).

Thus, the correlation by Florschuetz represents a good trade-off between low computational costs and meaningful modeling of the impingement cooling system.

1.1.1.1 Problem geometry and variables

Let us now introduce the geometry of the cooling system defined in Florschuetz's work to which we refer for further details [32, 33].

The geometry of the impingement cooling system studied by Florschuetz is schematically depicted in Figure 1.2 and consists of a plate of jets of size $L_x \times L_y$ placed at a distance z_n from the target surface. We have set up the reference system in Figure 1.2 such that the cooling air, once it leaves the jets, flows out of the duct made by the impingement plate and the target surface in the direction of the x-axis in our reference system. Because of this, the x-direction is called stream-wise, while the $y-$direction is called span-wise.

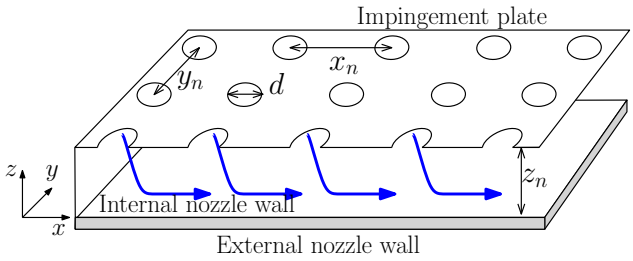

Figure 1.2: Reference geometry of the impingement cooling system

On the plate, round orifices of diameter d are arranged to have distance between centers x_n along the direction of the abscissa and y_n along the direction of the ordinate. It is also imposed that the distances to the edges of the first row are $\frac{x_n}{2}$ in the x direction and $\frac{y_n}{2}$ in the y direction (see Figure 1.3).

Concerning the direction stream-wise, given two points $A(x_A, y_A)$ and $B(x_B, y_B)$ on the plate, A is said to be 'upstream of' B if $x_A < x_B$ and at the same time B is said to be 'downstream of' A.

Holes could be arranged in two different ways on the plate: inline or staggered. In both cases we have $N_x := \lfloor \frac{L_x}{x_n} \rfloor$ span-wise rows each containing $N_y := \lfloor \frac{L_y}{y_n} \rfloor$ orifices. In the inline layout the centers of the orifices on the same span-wise row have the same $x-$coordinate and the ones on the same stream-wise row have the same $y-$coordinate (see Figure 1.3, left). Staggered layout derives from the inline layout by shifting by $\frac{y_n}{2}$ the span-wise rows of even position, counting from upstream (see Figure 1.3, right).

After this initial explanation, it is already possible to define the design variables that are the components of the input vector $\mathbf{v}$ of our objective function H. The variables are

- d: the diameter of the impingement holes;
- x_n: stream-wise distance between the centers jet holes;

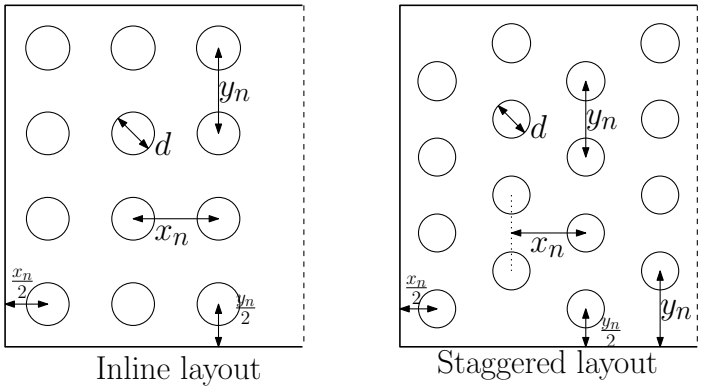

Figure 1.3: The two possible layouts of the holes on the jet plate: inline (left) and staggered (right).

- y_n: span-wise distance between the centers jet holes;
- z_n: distance of the impingement plate from the target surface (meatus width);
- *layout*: specifies the hole pattern.

We must notice that, while d, x_n, y_n, and z_n are positive real continuous variables, *layout* is a non-ordinal categorical variable that can take two values: `inline` and `staggered` [1].

1.1.1.2 Florschuetz correlation

The correlation between the design variables $\mathbf{v} = (x_n, y_n, z_n, d, layout)$ of the impingement plate introduced in the previous chapter and the stream-wise distribution of the Nusselt number Nu is defined by the following equation:

$$\begin{aligned} \mathrm{Nu}(x_i) = A\mathrm{Re}_j^{\alpha}(x_i)\mathrm{Pr}_c^{\frac{1}{3}} \left(1 - B\left(\frac{z_n}{d}\left[\frac{G_c(x_i)}{G_j(x_i)}\right]\right)^{\beta}\right), \\ \text{for } x_i = x_n\left(i - \frac{1}{2}\right), \text{ with } i = 1, ..., N_x; \end{aligned} \tag{1.4}$$

where x_i is the x-coordinate of the centers of the holes of the i-th stream-wise row and Pr_c denotes the Prandtl number of the coolant; the coefficients A, α, B e β depend on

[1] We remark that in the current problem formulation, the *layout* variable can take two values and therefore it could be treated as a binary variable. For the sake of generality, we prefer to treat it as a categorical variable as it is of engineering interest to investigate models that admit orifice arrangements other than inline and staggered (see Chapter 2.4.4).

the geometric parameters x_n, y_n, z_n, d and *layout* according to the following relationship:

$$\star(x_n, y_n, z_n, d) = C_\star \left(\frac{x_n}{d}\right)^{\gamma_{\star x}} \left(\frac{y_n}{d}\right)^{\gamma_{\star y}} \left(\frac{z_n}{d}\right)^{\gamma_{\star z}}, \quad \text{with } \star \in \{A, \alpha, B, \beta\}; \tag{1.5}$$

where the constants $C_\star$, $\gamma_{\star x}$, $\gamma_{\star y}$ e $\gamma_{\star z}$ were estimated empirically and are displayed in Table 1.1 (see [33]). Note that the values of the constants presented in the table differ depending on the value taken by the categorical variable *layout* (i.e. `inline` or `staggered`).

	Inline pattern				Staggered pattern			
$\star$	$C_\star$	$\gamma_{\star x}$	$\gamma_{\star y}$	$\gamma_{\star z}$	$C_\star$	$\gamma_{\star x}$	$\gamma_{\star y}$	$\gamma_{\star z}$
A	1.18	-0.944	-0.642	0.169	1.87	-0.771	-0.999	-0.257
α	0.612	0.059	0.032	-0.022	0.571	0.028	0.092	0.039
B	0.437	-0.095	-0.219	0.275	1.03	-0.243	-0.307	0.059
β	0.092	-0.005	0.599	1.04	0.442	0.098	-0.003	0.304

Table 1.1: Coefficients A, α, B e β for (1.5)

The *layout* variable is also important in defining the feasible set V, but this topic will be covered in Subsection 1.1.2.

In (1.4) we also have the quantities $G_j(x_i)$, $G_c(x_i)$ and $\mathrm{Re}_j(x_i)$ distributed along the x-coordinate and dependent on the geometric variables.

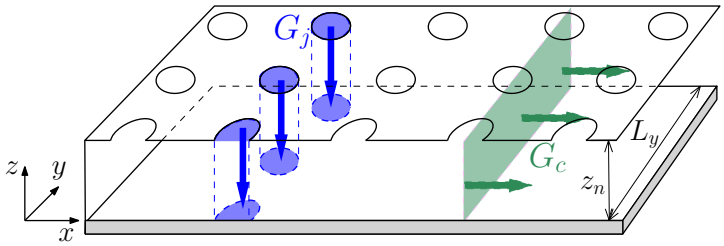

Figure 1.4: Representation of jet mass velocity G_j (blue) and crossflow mass velocity G_c (green).

$G_j(x_i)$ is the mass velocity (unit: $kg \cdot m^{-2} \cdot s^{-1}$) of the flow of cooling air passing through a single jet of abscissa x_i referred to the area of the jet hole (in blue in Figure 1.4). In particular, let us consider that our cooling system receives a certain flow rate $\dot{m}_{tot}$ (unit: $kg \cdot s^{-1}$) of cooling air that is distributed among the span-wise rows of jets.

If a row having abscissa x_i and N_y jets of area A_j has a mass flow rate $\dot{m}_j$, then we have that

$$\dot{m}_j(x_i) = G_j(x_i) A_j N_y \;\Rightarrow\; G_j(x_i) = \frac{\dot{m}_j(x_i)}{A_j N_y}, \quad x_i = x_n \left(i - \frac{1}{2}\right), \text{ for } i = 1, ..., N_x; \tag{1.6}$$

moreover, G_j is considered constant along the dimension y, so each span-wise row is characterized by a single value of G_j.

G_c is the crossflow mass velocity and it is thus related to the area of the cross-section of the duct, given by the product $z_n L_y$ (in green in Figure 1.4). Therefore, if we have a transverse flow rate $\dot{m}_c$ at $x-$coordinate x_i of a certain row of jets we have

$$\dot{m}_c(x_i) = G_c(x_i) L_y z_n \;\Rightarrow\; G_c(x_i) = \frac{\dot{m}_c(x_i)}{L_y z_n}, \quad x_i = x_n \left(i - \frac{1}{2}\right), \text{ for } i = 1, ..., N_x. \tag{1.7}$$

We assume that G_c is constant along $y-$direction. Particularly, in (1.4) we can notice that the distribution of the Nusselt number depends on the ratio

$$\left[\frac{G_c}{G_j}\right](x_i) := \frac{G_c(x_i)}{G_j(x_i)}, \quad x_i = x_n \left(i - \frac{1}{2}\right), \text{ for } i = 1, ..., N_x. \tag{1.8}$$

Finally, in (1.4) we have the contribution of the distribution of the Reynolds number of the cooling air flowing through an orifice in position x_i. The Reynolds number is the nondimensional ratio between inertia forces and internal viscous forces of fluid, and it is related to the jet mass velocity $G_j(x_i)$ by the relation

$$\mathrm{Re}_j(x_i) = \frac{G_j(x_i) d}{\mu_c}, \quad x_i = x_n \left(i - \frac{1}{2}\right), \text{ for } i = 1, ..., N_x; \tag{1.9}$$

where μ_c is the dynamic viscosity coefficient of the cooling air (unit: $kg \cdot m^{-1} \cdot s^{-1}$) and d the diameter of the jet hole. The coefficient μ_c depends on the coolant's characteristics and mainly on its temperature T_c and it can be estimated via interpolation since the cooling fluid is air and it is known the behavior of μ_c with respect to temperature.

So, in order to determine the Nusselt number distribution with (1.4) it is necessary to estimate the distributions of G_j, G_c and Re_j.

The distribution of G_j can be derived directly from a mathematical model developed by Florschuetz [33] for the stream-wise distribution of the ratio of the jet mass velocity

G_j to the average jet mass velocity $\overline{G}_j$ (see [33])

$$\left[\frac{G_j}{\overline{G}_j}\right](x_i) = \frac{\delta N_x \cosh\left(\delta \frac{x_i}{x_n}\right)}{\sinh\left(\delta N_x\right)}, \quad x_i = x_n\left(i - \frac{1}{2}\right), \text{ for } i = 1, ..., N_x; \tag{1.10}$$

where δ is a constant defined by

$$\delta = \frac{C_D\sqrt{2}}{y_n z_n}\frac{\pi d^2}{4} = \frac{C_D\sqrt{2}A_j}{y_n z_n};$$

with C_D being the discharge coefficient for every hole.

Note that $\overline{G_j}$ is constant and depends only on the total flow rate of cooling air $\dot{m}_{tot}$ supplied to the system and the sum of the areas of all jets on the plate A_j^{tot} according to the relationship:

$$\overline{G}_j = \frac{\dot{m}_{tot}}{A_j^{tot}}; \tag{1.11}$$

where $A_j^{tot} = N_x N_y A_j$.

By substituting (1.11) in (1.10) we obtain the following analytical model for the distribution of the jet mass velocity:

$$G_j(x_i) = \frac{\dot{m}_{tot}}{A_j^{tot}}\frac{\delta N_x \cosh\left(\delta \frac{x_i}{x_n}\right)}{\sinh\left(\delta N_x\right)}, \quad x_i = x_n\left(i - \frac{1}{2}\right), \text{ for } i = 1, ..., N_x. \tag{1.12}$$

To determine the crossflow mass velocity distribution, one must refer to the geometry in Figure 1.4. In this representation, the duct between the impingement plate and the target surface is closed at one end; this implies that the crossflow has only one direction. Thus it is reasonable to assume that the crossflow at a point of abscissa x_i is due to the contribution of the flows passing through all the jets upstream of x_i; in particular, we can assume that the crossflow mass velocity at x_i is due to the sum of all the jet mass

velocities coming from upstream, so for G_c we have

$$\begin{aligned} &G_c(x_1) = 0, \\ &G_c(x_i) = \frac{\dot{m}_c(x_i)}{L_y z_n} = \frac{1}{L_y z_n} \sum_{k=1}^{i-1} \dot{m}_j(x_k) = \frac{A_j N_y}{L_y z_n} \sum_{k=1}^{i-1} G_j(x_k), \\ &\quad \text{for } i = 2, ..., N_x. \end{aligned} \tag{1.13}$$

Thanks to the model (1.12) and the assumption (1.13), we can estimate the distribution of the ratio G_c/G_j following the simple procedure shown in Algorithm 1.

Algorithm 1 Estimation of the distribution of G_c/G_j

Initialization

Take the variables $\mathbf{v} = (x_n, y_n, z_n, d, layout)$ and the parameters L_x, L_y, $\dot{m}_{tot}$, C_D.

Compute $N_x = \lfloor \frac{L_x}{x_n} \rfloor$, $N_y = \lfloor \frac{L_y}{y_n} \rfloor$ and $A_j = \frac{\pi d^2}{4}$.

Define the vector $x_j = \frac{1}{2}x_n : x_n : (N_x - \frac{1}{2})x_n$ of the x-coordinate of the centers of the span-wise rows.

Jet mass velocity distribution

Obtain the distribution $G_j(x_i)$ for $i = 1, ..., N_x$ with (1.12).

Crossflow mass velocity distribution

Set $G_c(x_1) = 0$, because there is no flow coming from upstream.

for $i = 2, ..., N_x$ **do**

Using (1.13) obtain crossflow mass velocity

$$G_c(x_i) = \frac{1}{L_y z_n} \sum_{k=1}^{i-1} \dot{m}_j(x_k).$$

end for

Evaluation of the ratio $\frac{G_c}{G_j}$

for $i = 1, ..., N_x$ **do**

Set

$$\left[\frac{G_c}{G_j}\right](x_i) = \frac{G_c(x_i)}{G_j(x_i)}$$

end for

To finally determine the distribution of Nu we still need to derive the Prandtl number Pr_c and the dynamic viscosity μ_c of the refrigerant fluid. Moreover, in order to use (1.3) to get the distribution of h_c we must estimate the thermal conductivity k_c of the cooling air. These three coefficients depend mainly on temperature T_c and fluid composition and can be easily estimated by nonlinear interpolation.

The procedure for determining the stream-wise distribution of HTC h_c is outlined in

Algorithm 2.

Algorithm 2 HTC stream-wise distribution

Initialization

1: Take the variables $\mathbf{v} = (x_n, y_n, z_n, d, layout)$ and the parameters L_x, L_y, T_c, T_g, h_g, $\dot{m}_{tot}$, C_D.
2: Compute $N_x = \lfloor \frac{L_x}{x_n} \rfloor$, $N_y = \lfloor \frac{L_y}{y_n} \rfloor$ e $A_j = \frac{\pi d^2}{4}$.
3: Define the vector $x_j = \frac{1}{2}x_n : x_n : (N_x - \frac{1}{2})x_n$ of the x-coordinate of the centers of the span-wise rows.
4: Calculation by nonlinear interpolation of $\mu_c := \mu_c(T_c)$, $\mathrm{Pr}_c := \mathrm{Pr}_c(T_c)$ and $k_c := k_c(T_c)$ of the coolant fluid.

Mass velocity distributions

5: Estimation of the distributions of G_j and $\frac{G_c}{G_j}$ using Algorithm 1.

Jet Reynolds number distribution

6: Computation of the distributions of jet Reynolds number $Re_j(x_i)$ using (1.9).

HTC h_c distribution

7: Using (1.4) and (1.5) obtain the stream-wise distribution of the Nusselt number Nu.
8: Compute the distribution of h_c with (1.3):

$$h_c(x_i) = \frac{\mathrm{Nu}(x_i)k_c}{d}, \quad i = 1, ..., N_x.$$

Algorithm 2 is straightforward; after receiving the input $\mathbf{v} = (x_n, y_n, z_n, d, layout)$ and the parameters derived from the boundary conditions it computes the number N_x of span-wise rows, the number N_y of holes for every span-wise row and the area of a single hole A_j; then defines the vector containing the $x-$coordinate of the centers of the holes on the span-wise rows (Step 1 - Step 3). In Step 4 it uses the inlet temperature T_c of the cooling air as a query point to interpolate the dynamic viscosity coefficient μ_c, the Prandtl number Pr_c and the thermal conductivity k_c. Step 5 computes the distribution of the jet mass velocity G_j and of the ratio $\frac{G_c}{G_j}$ via Algorithm 1. Step 6 computes the distribution of the jet Reynolds number $Re_j(x_i)$ using (1.9). Finally, in Step 7 and Step 8 the algorithm uses Florschuetz's model (1.4) to get the distribution of the Nusselt number Nu and uses (1.3) to estimate the distribution h_c of the HTC of the cooling air.

Let us remark that Algorithm 2 returns a one-dimensional distribution of h_c, so $h_c(\mathbf{v}) \in \mathbb{R}^{N_x}$. In order to define the scalar objective function $H(\mathbf{v})$ we use the root mean square (RMS) of $h_c(\mathbf{v})$, then

$$H(\mathbf{v}) := (h_c(\mathbf{v}))_{RMS} = \frac{\|h_c(\mathbf{v})\|_2}{\sqrt{N_x}}. \tag{1.14}$$

1.1.2 The constraints

The design of a cooling system, like many other industrial applications, is subject to constraints that arise from the need to have solutions that are actually applicable in a real-world context or at least retain a minimum of relevance to the physics of the problem we are solving.

In our particular case, the constraints arise from several requirements. First, the cooling system must be efficient enough to ensure a minimal durability of the nozzle, this means that the design of the impingement insert must avoid configurations that allow the external heat to cause excessive damage to the nozzle walls. Secondly, it is necessary that the system is actually manufacturable so, for example, the solution to the problem cannot lead to an impingement plate with holes that are too small or too close. Finally, since our objective function is derived from the empirically developed mathematical model (1.4), the variables must be constrained in a space in which the model has been validated; this is because outside that space the validity of the model is not guaranteed. All these requirements are represented by a set of constraint functions that involve the design variables $\mathbf{v} = (x_n, y_n, z_n, d, layout)$, the distributions of the temperatures on the internal and external nozzle walls and the (outlet) pressure of the cooling air.

In this section, the constraints necessary for the final formulation of the problem are defined and explained.

1.1.2.1 Temperature constraints

To get an idea of the efficiency of the cooling system, we need to quantify how much it can cool the inner and outer nozzle wall, so we need to estimate the stream-wise distributions of the internal and external nozzle wall temperature, which we denote as T_{wi} and T_{we} respectively. All temperatures are in kelvin (unit:K).

To do this, we can assume that heat transfer occurs from the external of the nozzle, where we have the hot gas with temperature T_g and HTC h_g, to the inside of the nozzle where we have the cool air with temperature T_c and HTC h_c. We can further assume that the heat transfer from the external to the internal of the nozzle consists of three phases. In the first stage, heat from the external gas is transferred by convection to the external wall of the nozzle, then heat is transferred to the inside of the nozzle by thermal conductivity from the external wall to the inner one (not considering the thermal

conductivity that occurs perpendicularly to this direction), and finally, heat is transferred from the inside wall to the cooling fluid again by convection. This process is represented in Figure 1.5.

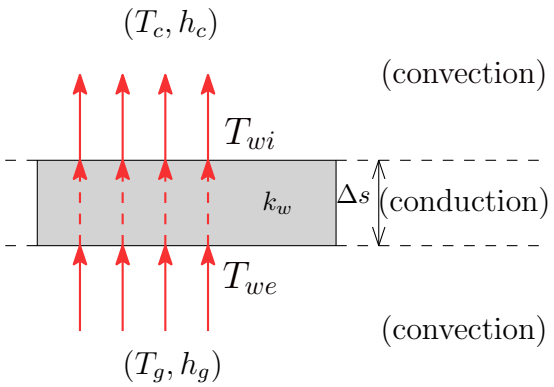

Figure 1.5: Scheme of the heat transfer through the nozzle wall.

Given these assumptions, and having derived the heat transfer coefficient of the cooling air, we can calculate the distributions of T_{wi} and T_{we} by solving for every $i = 1, ..., N_x$ the following linear system

$$\begin{cases} h_c(x_i)\left(T_{wi}(x_i) - T_c\right) = \frac{k_w}{\Delta s}\left(T_{we}(x_i) - T_{wi}(x_i)\right) \\ \frac{k_w}{\Delta s}\left(T_{we}(x_i) - T_{wi}(x_i)\right) = h_g\left(T_g - T_{we}(x_i)\right) \end{cases} ; \tag{1.15}$$

where k_w and Δs are respectively the thermal conductivity (unit: $kg \cdot m \cdot s^{-3} \cdot K^{-1}$) and the thickness of the nozzle wall. This formulation comes from the time-independent heat equation

$$\begin{cases} \nabla^2 T_{in} = 0 & \text{on } \Omega, \\ \frac{\partial T_{in}}{\partial \mathbf{n}} = q_g & \text{on } \Gamma_g, \\ \frac{\partial T_{in}}{\partial \mathbf{n}} = q_c & \text{on } \Gamma_c, \\ \frac{\partial T_{in}}{\partial \mathbf{n}} = 0 & \text{on } \partial\Omega \setminus (\Gamma_g \cup \Gamma_c)\,. \end{cases} \tag{1.16}$$

Here T_{in} is the temperature distribution inside the nozzle wall, the domain Ω is the rectangular section of the nozzle wall with size $L_x \times \Delta s$, the boundaries Γ_g and Γ_c are the surfaces subject to convective heat flow of the hot gas and cooling air respectively. The remaining boundary of Ω is supposed to be adiabatic [43]. If we discretize Ω in N_x rectangular elements of size $x_n \times \Delta s$ and we use the Finite Differences (FD) method

assuming that there is no heat flow between two contiguous elements we obtain the system (1.15).

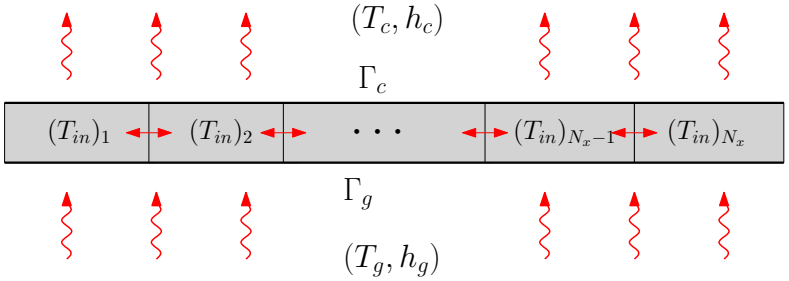

Figure 1.6: Discretization of the heat transfer on the nozzle wall.

A more accurate, but computationally more expensive, estimation of T_{wi} and T_{we} can be obtained in two steps. First, we solve (1.16) with FD taking into account the heat transfer between adjacent elements in order to obtain the one-dimensional distribution of T_{in} which is the stream-wise temperature distributions inside the nozzle wall (see Figure 1.6); then we substitute the distribution of T_{in} in system (1.15), in particular, we put T_{in} in place of T_{we} in the first equation and in place of T_{wi} in the second equation, obtaining two separate equations:

$$h_c(x_i)\left(T_{wi}(x_i) - T_c\right) = \frac{k_w}{\Delta s}\left(T_{in}(x_i) - T_{wi}(x_i)\right) \tag{1.17}$$

$$\frac{k_w}{\Delta s}\left(T_{we}(x_i) - T_{in}(x_i)\right) = h_g\left(T_g - T_{we}(x_i)\right). \tag{1.18}$$

We use the distributions T_{wi} and T_{we} to define two constraint functions. The first one bounds the external wall temperature T_{we} and it is defined as

$$\left(T_{we}(\mathbf{v})\right)_{RMS} \leq T_{we}^{\max}. \tag{1.19}$$

By setting this constraint we guarantee that the mechanical properties of the material are sufficient to allow the component to reach the expected life span.

The second constraint function is defined to bound the temperature gradient between the external and internal walls; in our case, it means to set a bound for the distribution of the difference between T_{we} and T_{wi}, thus

$$\left(\Delta T(\mathbf{v})\right)_{RMS} \leq \Delta T^{\max}; \tag{1.20}$$

where $\Delta T(\mathbf{v}) = T_{we}(\mathbf{v}) - T_{wi}(\mathbf{v})$. This constraint is necessary to prevent structural damage to the nozzle wall caused by thermal deformation due to an excessive difference between the external and internal temperatures on the wall.

1.1.2.2 Pressure constraints

Another factor that affects the performance of an impingement cooling system is the pressure of the cooling air. In particular, in our case, we consider the ratio of the outlet pressure p_c^{out} to the inlet pressure p_c^{in} of the cooling air

$$r_p := \frac{p_c^{out}}{p_c^{in}}. \tag{1.21}$$

In our case $r_p \in (0, 1]$. This is because, since fluids move in the opposite direction of the pressure gradient, to have air flow there must be a pressure difference. If $p_c^{in} = p_c^{out}$, that is, if $r_p = 1$, we have no flow of cooling air through the system. It is not possible to have the case $p_c^{out} > p_c^{in}$, which means $r_p > 1$, because if so the flow would be in the opposite direction, and this is ruled out by the way we have defined the model that simulates the impingement cooling. On the other hand, $r_p > 0$ since pressure is strictly positive by definition.

The ratio r_p is related to the flow rate of air through the cooling system and thus its velocity. The lower r_p, the higher the mass flow rate and the jet velocity. This is true until the velocity approaches the speed of sound. In particular, if the flow becomes sonic in the nozzle the so-called choked condition is reached, corresponding to the maximum flow rate: further reducing the discharge pressure does not lead to an increase in coolant flow rate but results in an underexpanded supersonic jet. In this regime, complex shock patterns and a recirculation pattern at the stagnation point occur, resulting in a degradation of heat transfer performance [35].

In order to keep the ratio r_p away from zero we set the following constraint

$$r_p \geq \tilde{r}; \tag{1.22}$$

with $\tilde{r} \in (0, 1)$.

Recall that in the case of our interest the inlet pressure p_c^{in} of the cooling air is constant and is given as a boundary condition, while the outlet pressure p_c^{out} is unknown, hence to

check constraint (1.22) it is necessary to estimate p_c^{out} with respect to the design variables vector $\mathbf{v}$.

We assume that $p_c^{out}(\mathbf{v})$ is equal to the outlet pressure at the most downstream row of orifices, i.e. the row with centers of abscissa x_{N_x}. Focusing on this last row, from the theory of isentropic flow the pressure ratio r_p is related to the mass flow rate of cooling air through the last row $\dot{m}_j\left(x_{N_x}\right)$ by the equation

$$\dot{m}_j\left(x_{N_x}\right) = N_y A_j C_D \left(\frac{p_c^{out}}{p_c^{in}}\right)^{\frac{1}{\gamma}} \sqrt{\frac{2\gamma}{\gamma-1}\frac{p_c^{in}}{\rho_c^{in}}\left[1-\left(\frac{p_c^{out}}{p_c^{in}}\right)^{\frac{\gamma-1}{\gamma}}\right]}, \tag{1.23}$$

where $\gamma = \frac{c_P}{c_V}$ is ratio of specific heats of the air, C_D is the jet discharge coefficient, $A_j = \pi d^2/4$ is the surface of a single orifice, N_y is the number of holes in every row and $\rho_c^{in} = \frac{p_c^{in}}{RT_c}$ is the density of inlet cooling air, with R being the ideal gas constant (unit: $kg \cdot m^2 \cdot s^{-2} \cdot K^{-1} \cdot mol^{-1}$) [35].

On the other hand, we know from (1.6) that

$$\dot{m}_j\left(x_{N_x}\right) = A_j N_y G_j(x_{N_x}); \tag{1.24}$$

where the jet flow mass velocity $G_j(x_{N_x})$ for the last row can be evaluated using (1.12).

Thus, substituting (1.24) in (1.23) and simplifying we obtain the equation

$$G_j(x_{N_x}) = C_D \left(\frac{p_c^{out}}{p_c^{in}}\right)^{\frac{1}{\gamma}} \sqrt{\frac{2\gamma}{\gamma-1}\frac{p_c^{in}}{\rho_c^{in}}\left[1-\left(\frac{p_c^{out}}{p_c^{in}}\right)^{\frac{\gamma-1}{\gamma}}\right]}. \tag{1.25}$$

It is possible to estimate the value of $p_c^{out}(\mathbf{v})$ for a given design vector $\mathbf{v}$ as the solution of a scalar nonlinear equation

$$f(p) := p^{\frac{1}{\gamma}} \sqrt{(p_c^{in})^{\frac{\gamma-1}{\gamma}} - p^{\frac{\gamma-1}{\gamma}}} (p_c^{in})^{\frac{\gamma-1}{2\gamma}} - \frac{G_j(x_{N_x})}{C_D}\sqrt{\frac{\gamma-1}{2\gamma}RT_c} = 0, \tag{1.26}$$

where the function $f : [0, p_c^{in}] \to \mathbb{R}$ is derived from (1.25). It can be shown that problem (1.26) admits two solutions in $(0, p_c^{in})$ with one in the open sub-interval (p^*, p_c^{in}) (see e.g.

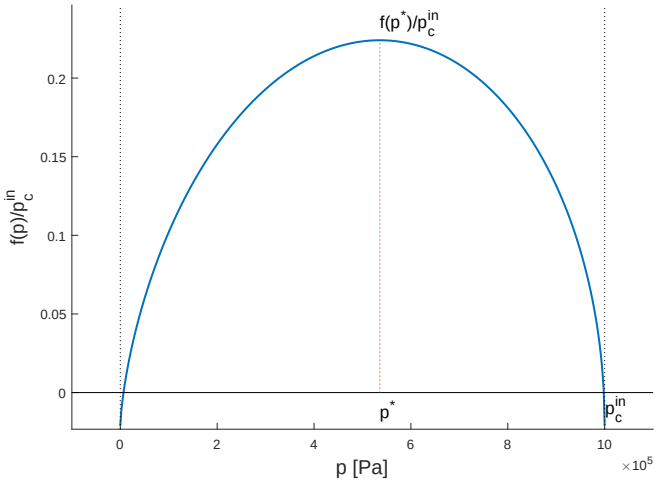

Figure 1.7: Graphical representation of the function $f(p)/p_c^{in}$, with $f(p)$ defined in (1.26) in the interval $\left[0, p_c^{in}\right]$, for $p_c^{in} = 2 \cdot 10^6$ Pa.

Figure 1.7), where p^* is the critic pressure defined as

$$p^* = \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma}{\gamma-1}} p_c^{in}. \tag{1.27}$$

We look for the solution in (p^*, p_c^{in}), since for $p_c^{out} \leq p^*$ supersonic flow surely occurs somewhere in the cooling system. Once we have estimated p_c^{out} we can check the constraint (1.22), which can be rearranged as a constraint on pressure difference in the following way

$$\Delta p_c\left(\mathbf{v}\right) := p_c^{in} - p_c^{out}(\mathbf{v}) \leq \left(1 - \tilde{r}\right) p_c^{in} =: \Delta p_c^{\max}. \tag{1.28}$$

1.1.2.3 Feasibility linear constraints

Feasibility linear constraints are meant to ensure a meaningful and applicable solution. We do not want to get an uncraftable or physically meaningless design for an impingement plate. Some unacceptable designs are, for example, ones with holes too small, ones with jet rows too close to each other, or ones with overlapping holes. Most of these unwanted results can be avoided by setting suitable box constraints for the continuous variables (x_n, y_n, z_n, d), while the variable *layout* is "unconstrained" since it is a non-ordinal categorical variable which admits only two values.

Moreover, the validity of Florschuetz's model (1.4) has to be ensured and, it is thus

important to keep the variables in a subspace where (1.4) has been validated. To this end, the following linear constraints

$$1 \leq \frac{z_n}{d} \leq 3, \tag{1.29}$$

$$4 \leq \frac{y_n}{d} \leq 8, \tag{1.30}$$

$$6.25 \cdot 10^{-1} \leq \frac{x_n}{y_n} \leq 3.75, \tag{1.31}$$

$$5 \leq \frac{x_n}{d} \leq \begin{cases} 15 & \text{if } \textit{layout} = \texttt{inline} \\ 10 & \text{if } \textit{layout} = \texttt{staggered} \end{cases}. \tag{1.32}$$

have to be satisfied. The coefficients and the form of the inequalities above were empirically defined in [32, 33]. Note that depending on the value of the variable *layout*, the constraint (1.32) changes. It is easy to show that when constraints (1.29)-(1.32) are fulfilled, overlapping holes are avoided in the design of the impingement plate, both for inline and staggered layout.

1.2 Black-box definition

In this section, we merge all the ingredients defined in Section 1.1 to present the black-box formulation of NOZZLE for the modeling of the impingement cooling system for a nozzle in a gas turbine. We remind that by black-box we mean a set of computational models for the optimization problem (objective and constraints) that can be evaluated to simulate the cooling system under consideration.

The basic structure of NOZZLE is represented in the flow chart in Figure 1.8 and described with more detail in Algorithm 3. NOZZLE takes as input the variable vector $\mathbf{v} = (x_n, y_n, z_n, d, \textit{layout})$ that defines the design of the impingement plate, and the fixed parameters given by assumptions and boundary conditions: the inlet temperature T_c and pressure p_c^{in} of the cooling air, the total mass flow $\dot{m}_{tot}$ of cooling air coming from upstream of the cooling system, the temperature T_g and HTC h_g of the hot gas that surrounds the nozzle, the discharge coefficient C_D of the holes and the sizes L_x and L_y of the rectangular impingement plate (Step 1).

The inlet temperature T_c is used to interpolate the values of the dynamic viscosity

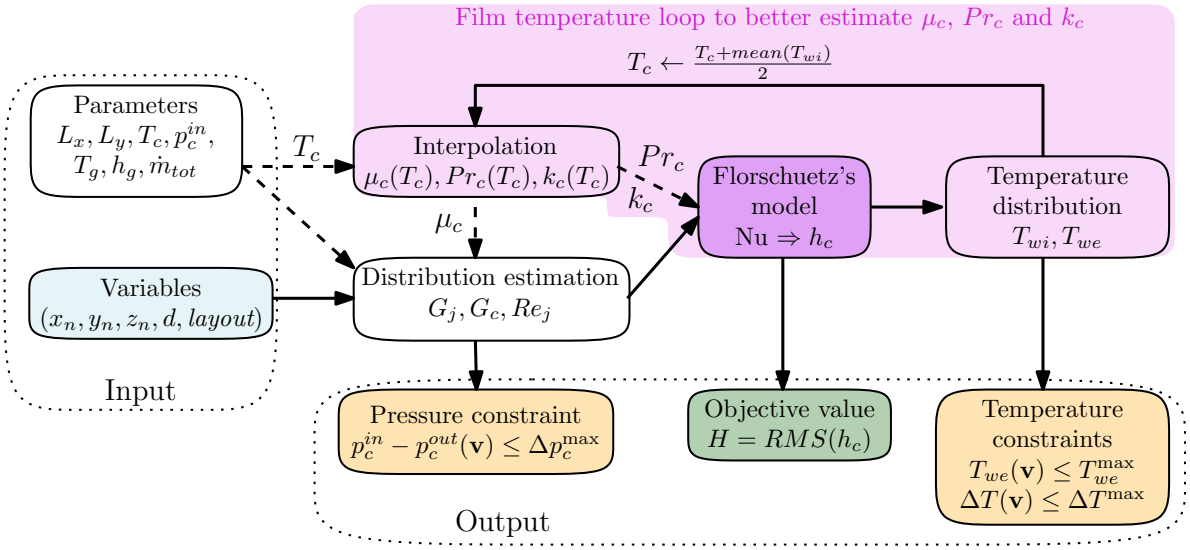

Figure 1.8: NOZZLE's flowchart.

μ_c, the thermal conductivity k_c, and Prandtl number Pr_c for the cooling air (Step 4).

The design variables $\mathbf{v}$, together with the dynamic viscosity μ_c and the parameters L_x, L_y, $\dot{m}_{tot}$ and C_D are given as inputs to Algorithm 1 to estimate the stream-wise distributions at every row of holes of the flow mass velocities to the jets G_j and to the cross-section G_c and of the jet Reynolds number Re_j.

Distributions of G_j and G_c are then used to estimate the outlet pressure p_c^{out} of the cooling air by solving the nonlinear equation (1.26) (Steps 5-7).

The HTC distribution $h_c(\mathbf{v})$ of the cooling air is estimated using the variables $\mathbf{v}$, the distributions G_j, G_c and Re_j and the coefficients k_c and Pr_c with Florschuetz's model (1.4) and (1.3) (Steps 8-10).

The HTC distribution $h_c(\mathbf{v})$ is then used to calculate the objective value $H(\mathbf{v}) = (h_c(\mathbf{v}))_{RMS}$ and to estimate the wall temperature distributions $T_{wi}(\mathbf{v})$ and $T_{we}(\mathbf{v})$ using one of the two approaches explained in Subsection 1.1.2.1 (Step 24).

The black-box returns as outputs the HTC distribution h_c to obtain the objective value $H(\mathbf{v})$ and the temperature distributions T_{wi} and T_{we} and the value of p_c^{out} to verify the constraints (1.19)-(1.20) and (1.28) respectively.

We observe that the wall temperature distributions are not only used to define constraints. Indeed, the distribution of T_{wi} is also used for better estimation of the thermal conductivity k_c of the cooling air. As discussed in Subsection 1.1.1.2, the value of this

coefficient is obtained by interpolation using T_c as query value; however, assuming T_c as the temperature of the cooling air in the boundary layer near the inner nozzle wall is quite inaccurate because in that region the cooling air is affected by the temperature of the wall, heated by conduction. So, as query value, we use the film temperature T_f which is an approximation of the temperature of a fluid inside a convection boundary layer [32, 33], and it is defined as

$$T_f = \frac{\text{mean}\,(T_{wi}) + T_c}{2}. \tag{1.33}$$

After a first interpolation of $k_c := k_c(T_c)$ it is necessary to estimate again the distributions of h_c, T_{wi} and T_{we}. We recall that k_c is involved directly in (1.3) for the evaluation of the distribution h_c and consequently in (1.15) for the estimation of T_{wi} and T_{we}. We include all these steps into a loop that in every iterate generates new estimations h_c, T_{wi}, and T_{we} and it ends when the relative error between two subsequent estimations of h_c is below a certain tolerance tol_h, i.e.

$$\frac{\|h_c - (h_c)_{old}\|_2}{\|(h_c)_{old}\|_2} \leq \text{tol}_h. \tag{1.34}$$

This loop is described in Steps 9-23 of Algorithm 3. Note that this loop does not involve the other coefficients μ_c and Pr_c, that is because in that case assuming T_c as the query for the interpolation is acceptable. Furthermore, the loop does not involve directly the distribution of the external wall temperature T_{we}.

We note that the evaluation of the feasibility constraints (1.29)-(1.32) is not included in Algorithm 3 as they can be easily treated outside the black-box.

Finally, we note that the NOZZLE implementation of the black-box results in a computational cost for the evaluation of the objective function and of the constraints that is rather cheap. This is due to the small number of variables (four continuous and one categorical) and to the small size of the distributions handled by the function (e.g. T_{we}, T_{wi}, h_c). Moreover, most of the auxiliary quantities and distributions needed are obtained straightforwardly, with the only exception of the solution of the nonlinear equation (1.26) to get the outlet pressure p_c^{out}, which uses an iterative method. Thus, a single evaluation of the black-box requires a small amount of computational time and memory.

Algorithm 3 NOZZLE

Initialization

Take the variables $\mathbf{v} = (x_n, y_n, z_n, d, layout)$ and the parameters L_x, L_y, T_c, T_g, h_g, $\dot{m}_{tot}$, C_D.

Compute $N_x = \lfloor \frac{L_x}{x_n} \rfloor$, $N_y = \lfloor \frac{L_y}{y_n} \rfloor$ e $A_j = \frac{\pi d^2}{4}$.

Define the vector $x_j = \frac{1}{2}x_n : x_n : (N_x - \frac{1}{2})x_n$ of the x-coordinate of the centers of the span-wise rows.

Calculation by nonlinear interpolation of $\mu_c := \mu_c(T_c)$, $\mathrm{Pr}_c := \mathrm{Pr}_c(T_c)$.

Mass velocity distributions

Estimation of the distributions of G_j and $\frac{G_c}{G_j}$ using Algorithm 1.

Jet Reynolds number distribution

Computation of the distributions of jet Reynolds number $Re_j(x_i)$ using (1.9).

Outlet pressure estimation

Estimate p_c^{out} as a solution of (1.26).

Nu distribution

Using (1.4) and (1.5) obtain the stream-wise distribution of the Nusselt number Nu.

First interpolation of k_c and first evaluation of h_c, T_{wi} and T_{we}

Calculation by nonlinear interpolation of $(k_c)_{old} := k_c(T_c)$ of the coolant fluid.

Compute the distribution of $(h_c)_{old}$ using (1.3):

$$(h_c)_{old}(x_i) = \frac{\mathrm{Nu}(x_i)\,(k_c)_{old}}{d}, \quad i = 1, ..., N_x.$$

Estimate of the distribution of $(T_{wi})_{old}$ and $(T_{we})_{old}$ via solving (1.15).

Film temperature loop

for $it = 1, 2, ...$ **do**

Set

$$T_f = \frac{\overline{(T_{wi})_{old}} + T_c}{2}.$$

Interpolate $k_c := k_c(T_f)$.

Evaluate

$$h_c(x_i) = \frac{\mathrm{Nu}(x_i)k_c}{d}, \quad i = 1, ..., N_x.$$

Estimate new distribution T_{wi} and T_{we} via solving (1.15) using h_c.

Compute the relative error ε_{rel} as

$$\varepsilon_{rel} = \frac{\|h_c - (h_c)_{old}\|_2}{\|(h_c)_{old}\|_2}.$$

if $\varepsilon_{rel} \leq \mathrm{tol}_h$ **then**

Break.

else

Set

$$(h_c)_{old} = h_c; \quad (T_{wi})_{old} = T_{wi}; \quad (T_{we})_{old} = T_{we}.$$

end if

end for

Objective value

Set $H = (h_c)_{RMS}$

Outputs

Return H, T_{wi}, T_{we} and p_c^{out}.

1.3 DFO for the solution of the black-box model

In this section, we embed the NOZZLE simulator described in the previous sections in the DFO framework. We therefore describe the main model features and propose a DFO-based procedure for its minimization.

1.3.1 The overall constrained BBO formulation

In Subsection 1.1.1 we have described the formulation of the objective function $H(\mathbf{v})$ while in Subsection 1.1.2 we have defined and motivated the constraint functions on wall temperature distributions $T_{wi}(\mathbf{v})$, $T_{we}(\mathbf{v})$, on the outlet pressure $p_c^{out}(\mathbf{v})$ and on design variables $\mathbf{v} = (x_n, y_n, z_n, d, layout)$. We can gather all the functions defined so far to formulate our problem as a standard minimization problem as follows.

$$
\begin{aligned}
\min_{\mathbf{v}} \quad & -H(\mathbf{v}) \\
\text{s.t.} \quad & c_1(\mathbf{v}) := (T_{we}(\mathbf{v}))_{RMS} - T_{we}^{\max} \leq 0; \\
& c_2(\mathbf{v}) := (\Delta T(\mathbf{v}))_{RMS} - \Delta T^{\max} \leq 0; \\
& c_3(\mathbf{v}) := \Delta p_c(\mathbf{v}) - \Delta p_c^{\max} \leq 0; \\
& c_4(\mathbf{v}) := 1 - \frac{z_n}{d} \leq 0; \\
& c_5(\mathbf{v}) := \frac{z_n}{d} - 3 \leq 0; \\
& c_6(\mathbf{v}) := 4 - \frac{y_n}{d} \leq 0; \\
& c_7(\mathbf{v}) := \frac{y_n}{d} - 8 \leq 0; \\
& c_8(\mathbf{v}) := \frac{x_n}{y_n} - 3.75 \leq 0; \\
& c_9(\mathbf{v}) := 6.25 \cdot 10^{-1} - \frac{x_n}{y_n} \leq 0; \\
& c_{10}(\mathbf{v}) := 5 - \frac{x_n}{d} \leq 0; \\
& c_{11}(\mathbf{v}) := \begin{cases} \frac{x_n}{d} - 15 \leq 0 & \text{if } layout = \texttt{inline} \\ \frac{x_n}{d} - 10 \leq 0 & \text{if } layout = \texttt{staggered} \end{cases}; \\
& (x_n, y_n, z_n, d) \in \mathcal{B} \subset \mathbb{R}^4_{>0}; \quad layout \in \{\texttt{inline}, \texttt{staggered}\}.
\end{aligned}
\tag{1.35}
$$

Formulation (1.35) represents a constrained Black-Box Optimization problem, where the objective $-H(\mathbf{v})$ is the negative RMS of the HTC distribution h_c defined in (1.14) and is returned as one of the NOZZLE outputs defined in Section 1.2. The other outputs of NOZZLE are used to define the constraint functions $c_1(\mathbf{v})$, $c_2(\mathbf{v})$, $c_2(\mathbf{v})$ which are derived, respectively, from (1.19), (1.20) and (1.28). The constraint functions $c_i(\mathbf{v})$ with $i = 4, ..., 11$ are derived directly from the feasibility constraints (1.29)-(1.32). We note that in (1.35) the first three constraint functions are not as dimensionless as the others, and this could be a source of poor scaling for the problem. To overcome this issue we simply divide c_1, c_2 and c_3 respectively by $T_{we}^{\max}$, $\Delta T^{\max}$ and $\Delta p_c^{\max}$.

Referring to the Black-Box Optimization constraint taxonomy presented in [50] we can identify two kinds of constraint functions in (1.35). Functions c_1, c_2, and c_3 are black-box simulation-based, thus any kind of (sub-)gradient is unavailable, and they are also relaxable since an impingement plate design that violates these constraints is still meaningful and can be post-processed. The remaining functions, from c_4 to c_{11}, are algebraic since they are expressed in an explicit form but they are unrelaxable because, as we explained in Subsection 1.1.2.3, they describe the validity space of Florschuetz's model used to define the black-box.

1.3.2 Our DFO proposal: the ℓ_1-penalty BFO method

The structure of the optimization problem (1.35) clearly calls for DFO tools. We propose a new DFO penalty method that uses the general ℓ_1- penalty method for derivative-based optimization, see e.g. [58]. Indeed we consider problem (1.35), and define the penalty function $\phi_1(\mathbf{v}, \varepsilon)$ as

$$\phi_1(\mathbf{v}, \varepsilon) = -H(\mathbf{v}) + \varepsilon C(\mathbf{v}), \tag{1.36}$$

where $\varepsilon > 0$ is the penalty parameter and the constraint violation function gathers all the constraint functions $C(\mathbf{v})$ as follows

$$C(\mathbf{v}) := \sum_{j=1}^{11} \max\{0, c_j(\mathbf{v})\}. \tag{1.37}$$

By choosing an increasing sequence of penalty parameters $\{\varepsilon_k\}_{k\in\mathbb{N}}$ such that $\varepsilon_k \to \infty$ we define a sequence of unconstrained minimization problems of the form

$$\begin{aligned} \min_{\mathbf{v}} \quad & \phi_1(\mathbf{v}, \varepsilon_k) = -H(\mathbf{v}) + \varepsilon_k C(\mathbf{v}) \\ \text{s.t.} \quad & (x_n, y_n, z_n, d) \in \mathcal{B} \subset \mathbb{R}^4_{>0}, \quad \textit{layout} \in \{\texttt{inline}, \texttt{staggered}\}, \end{aligned} \tag{1.38}$$

where we penalize constraint violations more severely, thereby forcing the minimizer of the penalty function closer to the feasible region for problem (1.35). Then, we use a derivative-free algorithm for solving (1.38) for every value of ε_k. In particular, our choice for the inner solver is the Brute Force Optimizer (BFO) [61, 62].

BFO is a simple random pattern search algorithm specifically designed for Black-Box Optimization since it can deal with unconstrained (it only handles simple bounds on the variables) optimization problems without any regularity or convexity assumption on the objective function. In particular, BFO is suitable for the minimization of the nonsmooth black-box function (1.36).

As a pattern search method, for every iterate $\overline{v}$, BFO creates a polling set of directions $\mathcal{P}$ that defines a finite local mesh around $\overline{v}$, BFO searches for any improvement of the objective function on this mesh by evaluating all the points on the mesh and if it succeeds in finding a better value for the objective function at a new point $\hat{v}$ the iterate is updated; otherwise if BFO fails in finding an improvement on the local mesh, the mesh is refined (i.e. a new mesh is defined closer to $\overline{v}$) and a new search is performed.

BFO can handle different types of variables like continuous, integer, discrete, mixed, or categorical. If the optimization problem has mixed variables, e.g. continuous and categorical variables, the search phase is more articulated and is called tree-search strategy, see further details in [61, 62]. More precisely, the search phase is firstly performed involving only the continuous components of the iterate $\overline{v}$ while the discrete ones are kept fixed, then, if there are no improvements on the continuous mesh, instead of refining the mesh a further search is performed by exploring the meshes defined around an iterate defined by fixing successively each of the non-continuous variables to a value neighboring that present in $\overline{v}$. As an example let us consider the vector of the design variables $\overline{\mathbf{v}} = \left(\overline{x_n}, \overline{y_n}, \overline{z_n}, \overline{d}, \texttt{staggered}\right)$, at first the mesh is built around the continuous part $\left(\overline{x_n}, \overline{y_n}, \overline{z_n}, \overline{d}\right)$ while keeping fixed $\overline{\textit{layout}} = \texttt{staggered}$; if BFO fails in finding an improvement another search is done on the same mesh for the continuous part but setting

$\overline{layout}$ = `inline`.

The overall proposed algorithm is detailed in Algorithm 4.

Algorithm 4 The ℓ_1-penalty BFO scheme

1: Given $\mathbf{v}_0^s$, $\varepsilon_0 > 0$, $\nu > 1$, $\tau > 0$, $k_{\max} > 0$
2: **for** $k = 0, 1, ..., k_{\max}$ **do**
3: Use BFO to find a minimizer $\mathbf{v}_k$ of $\phi_1(\mathbf{v}, \varepsilon_k)$, starting from $\mathbf{v}_k^s$.
4: **if** $C(\mathbf{v}_k) \leq \tau$ **then**
5: Stop and return $\mathbf{v}_k$.
6: **end if**
7: Set $\varepsilon_{k+1} = \nu\varepsilon_k$.
8: Set $\mathbf{v}_{k+1}^s = \mathbf{v}_k$.
9: **end for**

In the beginning, we choose an initial guess $\mathbf{v}_0^s$ and an initial value $\varepsilon_0 > 0$ for the penalty parameter. We also fix the update coefficient $\nu > 1$ to increase the penalty parameter, we set a tolerance $\tau > 0$ for the constraint violation and a maximum number of iterations $k_{\max}$. At every k-th iteration BFO is called to find a minimizer of the penalty function $\phi_1(\mathbf{v}, \varepsilon_k)$, where the penalty parameter is kept fixed, returning a point $\mathbf{v}_k$. Then the value of the constraint violation function $C(\mathbf{v}_k)$ is checked and if $C(\mathbf{v}_k) \leq \tau$ then $\mathbf{v}_k$ is accepted as an acceptable solution and the procedure stops. Otherwise, the penalty parameter ε_{k+1} is increased by a factor ν, and the new starting point $\mathbf{v}_{k+1}^s$ is set equal to the minimizer just found $\mathbf{v}_k$ and a new iteration starts. The update strategy for the new starting point at Step 8 is motivated by the fact that with a good choice of the initial penalty parameter ε_0, it is possible to obtain from the first iteration an approximate minimizer that does not excessively violate the constraints, so it will be sufficient to search for an admissible solution in a neighborhood of the last minimizer found, having chosen an appropriate parameter ν for the penalty update.

1.4 Experimental results

In this section, we numerically solve problem (1.35) using NOZZLE with the ℓ_1-penalty BFO algorithm.

Recalling that we aim to find "better" geometric variables x_n, y_n, z_n, d, and *layout* that define the design for an impingement plate for a fixed nozzle of a gas turbine, we consider two different problem settings obtained considering two different sets of

boundary conditions. The first, called here the "laboratory case", represents a situation that is encountered on a laboratory test bench, that is conditions similar to those under which Florschuetz and collaborators carried out the experiments to derive the model (1.4) (see [32]) are reproduced. The second has boundary conditions that reflect the typical values of an actual gas turbine and, for this reason, it will be referred to as the "industrial case". The boundary conditions and the upper bounds for both experimental cases are gathered in Table 1.2. At the moment we emphasize that for both situations we use the same value for the discharge coefficient C_D and mass flow rate $\dot{m}_{tot}$. In addition, the upper limit for the pressure difference $\Delta p_c^{\max}$ is also somewhat the same, specifically the two values for $\Delta p_c^{\max}$ are chosen such that the ratio of the pressure difference to the inlet pressure has as an upper limit equal to 0.04, i.e. we look for a configuration that allows a pressure difference lower than the 4% of p_c^{in}.

For the evaluation of the distributions of the wall temperature we solve the general problem defined in (1.16) using finite differences as described in Subsection 1.1.2.1.

All the experiments have been carried out using Matlab R2023a on an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz machine with 16 GB RAM and the new release 2.0 of the Matlab BFO package available at `https://github.com/m01marpor/BFO`. Default parameters have been set for BFO while $\varepsilon_0 = 1.5$, $\nu = 10$, $\tau = 10^{-3}$ and $k_{\max} = 15$ have been set in Algorithm 4.

Parameter	Description [Unit]	Laboratory value	Industrial value
L_x	Plate stream-wise length [m]	$1.27 \cdot 10^{-1}$	$5 \cdot 10^{-2}$
L_y	Plate span-wise length [m]	$1.22 \cdot 10^{-1}$	$5 \cdot 10^{-2}$
T_c	Cooling air inlet temperature [K]	$2.93 \cdot 10^{2}$	$7.73 \cdot 10^{2}$
p_c^{in}	Cooling air inlet pressure [Pa]	$2.03 \cdot 10^{5}$	$1.01 \cdot 10^{6}$
T_g	External hot gas temperature [K]	$3.73 \cdot 10^{2}$	$1.27 \cdot 10^{3}$
h_g	External hot gas HTC [W m^{-2}K^{-1}]	$1 \cdot 10^{2}$	$1 \cdot 10^{3}$
$\dot{m}_{tot}$	Cooling air mass flow rate [kg s^{-1}]	$1.00 \cdot 10^{-2}$	$1.00 \cdot 10^{-2}$
C_D	Jet discharge coefficient [-]	0.85	0.85
k_w	Wall thermal conductivity [W m^{-1}K^{-1}]	$1 \cdot 10^{2}$	$2 \cdot 10^{1}$
Δs	Wall thickness [m]	$1.00 \cdot 10^{-2}$	$3.00 \cdot 10^{-3}$
$\Delta T^{\max}$	Upper bound for $T_{we} - T_{wi}$ [K]	$3.00 \cdot 10^{1}$	$6.00 \cdot 10^{1}$
$T_{we}^{\max}$	Upper bound for T_{we} [K]	$3.43 \cdot 10^{2}$	$1.07 \cdot 10^{3}$
$\Delta p_c^{\max}$	Upper bound for $p_c^{in} - p_c^{out}$ [Pa]	$8.11 \cdot 10^{3}$	$4.04 \cdot 10^{4}$

Table 1.2: Parameters for boundary conditions and bounds for black-box constraints for laboratory (3rd column) and industrial (4th column) cases.

1.4.1 Laboratory case

Referring to the third column of Table 1.2, the temperature T_g of the hot gas is around 100°C with low HTC h_g while the inlet temperature of the cooling air T_c is around 50°C and inlet pressure p_c^{in} is about twice the atmospheric pressure. The impingement plate is nearly square with approximately 12 cm per side, and the target surface is 1 cm thick with good thermal conductivity. In this experiment, we start from an initial guess $\mathbf{v}_0^s$ chosen with $x_n = 1.75 \cdot 10^{-2}$ m, $y_n = 8.40 \cdot 10^{-3}$ m, $z_n = 6.30 \cdot 10^{-3}$ m, $d = 2.10 \cdot 10^{-3}$ m and *layout* = `staggered` as it shown in Figure 1.9. The initial guess has a value for the objective value $H(\mathbf{v}_0^s) = 2.03 \cdot 10^2 \text{W m}^{-2}\text{K}^{-1}$. In this way $\mathbf{v}_0^s$ satisfies the algebraic constraints (1.29)-(1.32) and the box constraints defined by the set $\mathcal{B} \subset \mathbb{R}^4_{>0}$

$$\mathcal{B} := \left\{ (x_n, y_n, z_n, d) \in \left[\frac{L_x}{30}, \frac{L_x}{2}\right] \times \left[\frac{L_y}{30}, \frac{L_y}{5}\right] \times \left[10^{-3}\text{m}, 10^{-2}\text{m}\right] \times \left[2 \cdot 10^{-3}\text{m}, \frac{L_y}{2}\right] \right\}. \tag{1.39}$$

This definition of $\mathcal{B}$ allows a very simple design for the impingement plate. After only

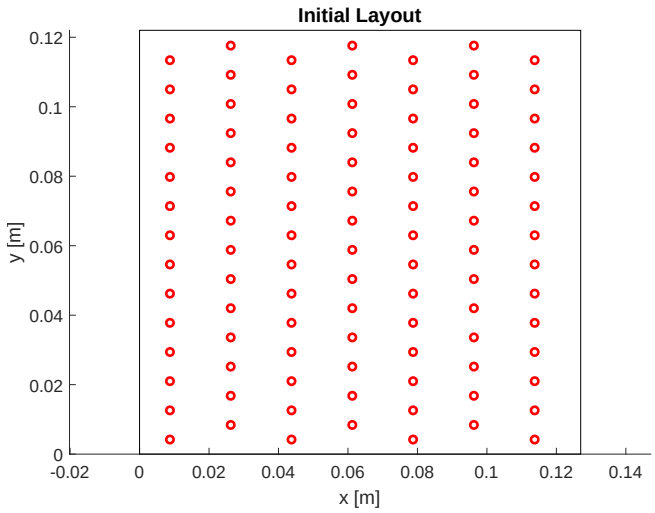

Figure 1.9: 2-D representation of the initial guess for the laboratory case.

one iteration of ℓ_1-penalty BFO method and 346 evaluations of NOZZLE the procedure converges to a solution $\tilde{\mathbf{v}}$ such that $C(\tilde{\mathbf{v}}) \leq \tau$. In particular the geometric variables of this solution are $x_n = 2.54 \cdot 10^{-2}$ m, $y_n = 1.53 \cdot 10^{-2}$ m, $z_n = 4.60 \cdot 10^{-3}$ m, $d = 2.00 \cdot 10^{-3}$ m and *layout* = `inline` with a corresponding objective value $H(\tilde{\mathbf{v}}) = 4.16 \cdot 10^2 \text{W m}^{-2}\text{K}^{-1}$; in Figure 1.11 (right) we show in 2-D the resulting layout.

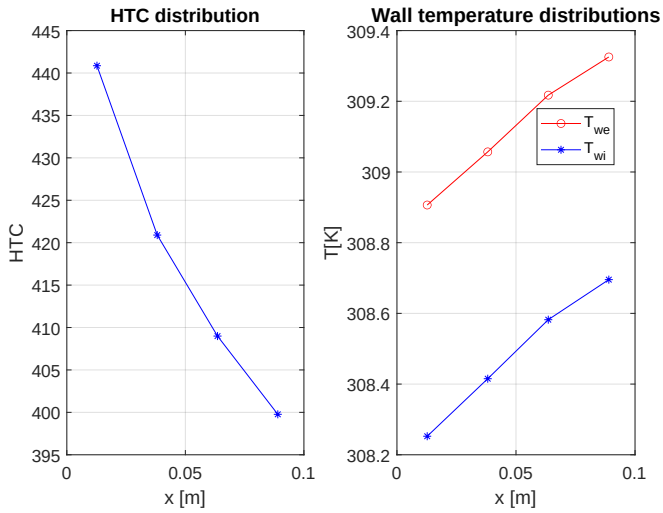

Figure 1.10: **Laboratory case:** On the left is the 1-D distribution of the HTC h_c of the cooling air. On the right is the 1-D distribution of the internal and external wall temperature distributions.

In Figures 1.10 and 1.11 there are some plots to show the distribution of the HTC of the cooling air h_c (Figure 1.10, left), the distributions of the wall temperatures (Figure 1.10, right) and the distribution of the wall temperature difference (Figure 1.11, left). This last plot shows very low values for the difference between external and internal temperature, and this is due to the thickness of the wall combined with its thermal conductivity. From the other plots, it is possible to see that there are no violations in temperature constraints.

1.4.2 Industrial case

In this case, the temperatures are significantly higher. In fact the temperature T_g of the external hot gas is 1000°C and the inlet temperature T_c is 500°C. From the 4th column of Table 1.2, we can see that also the HTC of the hot gas h_g is higher and that the inlet pressure p_c^{in} of the cooling air is ten times the atmospheric pressure. On the other hand, we are considering a smaller impingement plate (a square with with 5cm-long side) and a thinner target surface (only 3mm thick) with a lower thermal conductivity. Since the boundary conditions change, the upper bounds for the temperature constraints must be increased (see Table 1.2, 4th column). The box constraints $\mathcal{B}$ are identical to the ones defined previously in the "laboratory case" with only one change for the constraint on the hole diameter d. In particular, since the plate is smaller, we allow d to be smaller,

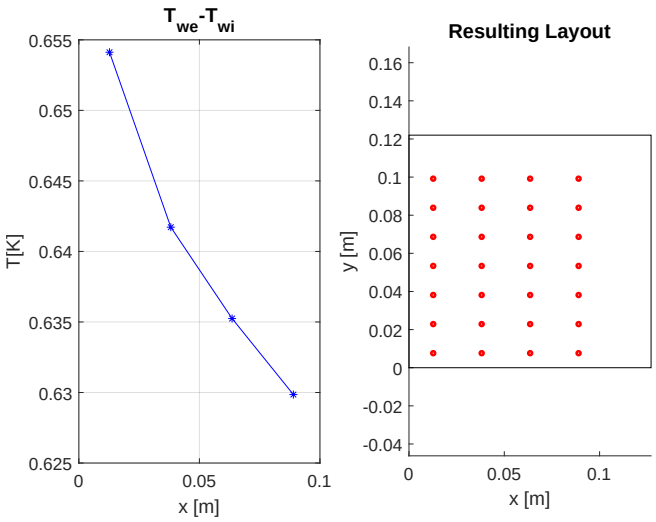

Figure 1.11: **Laboratory case:** On the left is the 1-D distribution of the difference between external and internal wall temperatures. On the right is a 2-D representation of the final layout of the impingement plate.

thus the constraint on d becomes $d \in \left[5 \cdot 10^{-4}\text{m}, \frac{L_y}{2}\right]$.

The initial guess $\mathbf{v}_0^s$ has values $x_n = 5.00 \cdot 10^{-3}$ m, $y_n = 4.00 \cdot 10^{-3}$ m, $z_n = 3.00 \cdot 10^{-3}$ m, $d = 1.00 \cdot 10^{-3}$ m and *layout* = `staggered` and has an objective value of $H(\mathbf{v}_0^s) = 1.01 \cdot 10^3 \text{W m}^{-2}\text{K}^{-1}$. The initial layout is shown in Figure 1.12.

Again, with only one iteration of ℓ_1-penalty BFO method and 252 NOZZLE evaluations the procedure converges to a solution $\tilde{\mathbf{v}}$ with $H(\tilde{\mathbf{v}}) = 2.71 \cdot 10^3 \text{W m}^{-2}\text{K}^{-1}$. The geometric variables have the following values $x_n = 6.28 \cdot 10^{-3}$ m, $y_n = 3.87 \cdot 10^{-3}$ m, $z_n = 1.46 \cdot 10^{-3}$ m, $d = 5.00 \cdot 10^{-4}$ m and *layout* = `inline`. The resulting layout is shown in Figure 1.14 (right).

In Figure 1.13 are plotted the distribution of HTC h_c (left) and the distributions of the wall temperatures (right); in Figure 1.14, on the left, we have the distribution of the temperature difference on the target wall. Also in this case there is no violation of the temperature constraints.

1.4.3 Comments on the numerical results

Summarizing, in both the problem settings, our DFO approach allows us to compute solutions that improve the performance of the cooling system with low computational effort (only a few hundred function evaluations). In particular, we improve the RMS

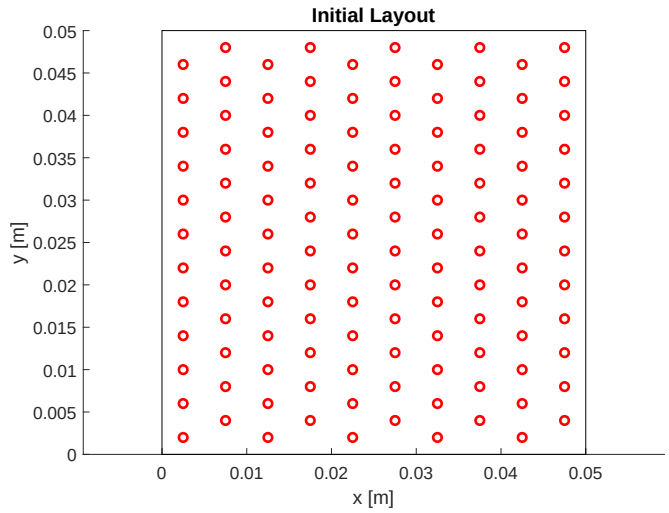

Figure 1.12: 2-D representation of the initial guess for the industrial case.

heat transfer h_c from $2.03 \cdot 10^2 \text{W m}^{-2}\text{K}^{-1}$ to $4.16 \cdot 10^2 \text{W m}^{-2}\text{K}^{-1}$ for the "laboratory case" and from $1.01 \cdot 10^3 \text{W m}^{-2}\text{K}^{-1}$ to $2.71 \cdot 10^3 \text{W m}^{-2}\text{K}^{-1}$ for the "industrial case". In addition, the behavior of the temperature distributions shown in Figures 1.10, 1.11, 1.13 and 1.14 are consistent with the studies previously done (see [32, 33]). Let us notice that both the computed solutions are not "far" from the initial guesses, that is because the procedure described in Algorithm 4 is a local optimization strategy, i.e. the algorithm strongly depends on the choice of the initial guess. On the other hand, we remark the switch in the value of the categorical *layout* from `staggered` to `inline`.

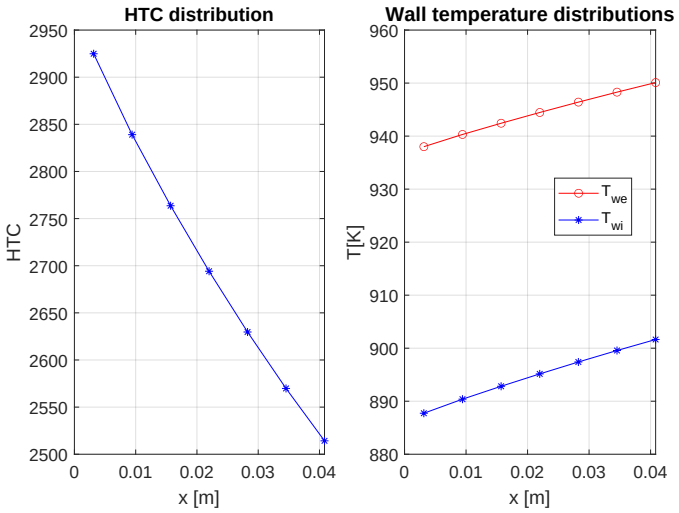

Figure 1.13: **Industrial case:** On the left is the 1-D distribution of the HTC h_c of the cooling air. On the right is the 1-D distribution of the internal and external wall temperature distributions.

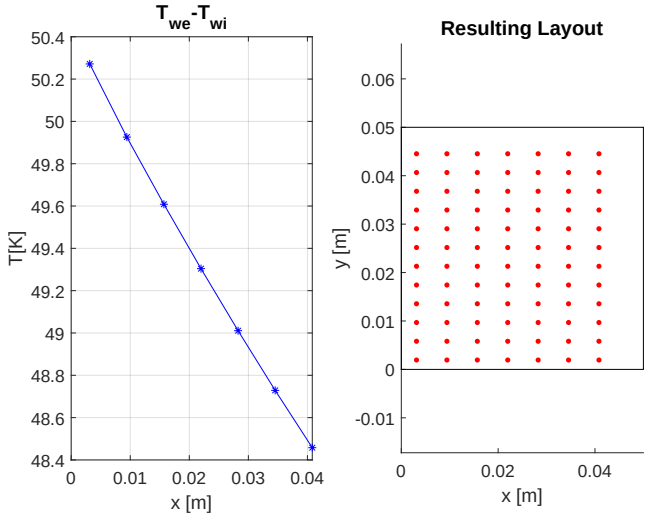

Figure 1.14: **Laboratory case:** On the left is the 1-D distribution of the difference between external and internal wall temperatures. On the right is a 2-D representation of the final layout of the impingement plate.

Chapter 2

MU$^\ell$STREG

In this chapter, we propose a new multilevel framework for the solution of stochastic optimization problems.

More specifically, we consider the solution of

$$\min_{x \in \mathbb{R}^n} f(x) \tag{2.1}$$

where f is a function that is assumed to be smooth and bounded from below, whose value can only be computed with some noise. When considering problem (2.1), it is usually assumed that realizations of f of the form $f(x, \varepsilon)$ are available, with ε a random variable [14, 24]. In this work, we allow for more flexibility by assuming that we have access to a hierarchy of noisy representations of f, built either by reducing the dimension in the variables space or by reducing the noise of the function approximation, or both. A level ℓ thus corresponds to a subset of variables and to a noise level in the function approximation. As in the classical case, a multilevel method in this context alternates "fine steps", i.e., steps computed considering large subsets of variables and accurate function approximations, and "coarse steps" computed taking into account just small subsets of variables and inaccurate function approximations. However, differently from the classical setting, the steps at each level are stochastic.

A strong motivation for the interest in this setting is given by the following approximation of (2.1)

$$\min_{x \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^{N} f_i(x) \tag{2.2}$$

with $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \ldots, N$ smooth and bounded from below. Usually, either n or N (or both) are really large. This problem has indeed its origin in large-scale data analysis applications where models depending on a large number of parameters n are fitted to a large set of N training samples.

Stochastic optimization problems, in both forms (2.1) and (2.2), can arise from data-matching problems such as those arising from validating an engineering design. In the particular context of this work, as we anticipated in the Introduction, we focused on validating the design of a complete cooling system for a gas turbine. The validation is done by identifying a setting of n parameters to minimize the error with respect to N measurements defined by the objective function f. Since it is a data-matching problem between measured values and values predicted by a simulator of the cooling system defined by the design to be validated, random noise sources can come from a random uncertainty either on the N measurements or they can be intrinsic to the simulator, both possibilities can occur at the same time.

Several methods have been developed to cope with the large sizes of the datasets (N) in problem (2.2). In particular, optimization techniques based on subsampling techniques have been proposed, among them the numerous variations of the classical Stochastic Gradient Descent (SGD) method.

When considering problem (2.2), there is a natural way of building a hierarchy in the "function space" through the definition of nested subsample sets $\mathcal{S}^l \subseteq \{1, .., N\}$ such that $\emptyset \neq \mathcal{S}^1 \subset ... \subset \mathcal{S}^l \subset ... \subset \mathcal{S}^{l_{\max}-1} \subset \mathcal{S}^{l_{\max}} \subseteq \{1, ..., N\}$ and by considering a hierarchy of subsampled functions obtained by averaging the functions f_i in $\mathcal{S}^l$. As in the classical case, a multilevel method in this context can alternate "fine steps", i.e., steps computed considering large subsets of data and "coarse steps" computed taking into account just small subsets of data. The coarse steps are computed by minimizing a model that is built from the coarse level approximations by adding a correction term, usually known as "first-order coherence" in the multilevel literature, which (in this context) accounts for the discrepancy between the full gradient and the subsampled gradient. This is similar to the same term that is added in the reduced variance gradient estimate of the mini-batch version of SVRG [44] (cf. equations (2.60) and (2.57) below). Multilevel methods can thus also be interpreted as variance reduction methods, cf. [17]. Their advantage is that they allow for an automatic choice of the step size, either in the form of a line-search [57] or in the form of a trust-region-like strategy [38]. Indeed, even if this usually requires a

function evaluation per iteration, by keeping the number of steps taken at the finest level limited and by leveraging the coarse steps, updating the step-size remains feasible even when evaluating the objective function for large datasets, thus resulting in a variance reduction method with automatic step-size selection.

We propose a stochastic multilevel first-order Adaptive Regularization (AR1) technique named MU$^\ell$STREG for MUltilevel STochastic REegularized Gradient[1]. Adaptive Regularization methods are globally convergent deterministic optimization techniques that builds a sequence of points $\{x_k\}$ by minimizing local models of the objective function. Every model is defined starting from the information at the current iterate and the model minimizer is used to define the next iterate. In particular, ARq methods work with regularized models which are the combination of a Taylor polynomial of the objective function of order q (for a suitable $q \in \mathbb{N}$) and a regularization term of order $q+1$. Thus in the specific case of AR1 methods, we have a first-order Taylor polynomial and a quadratic regularization term. For example, let us assume we want to minimize a smooth function $g : \mathbb{R}^n \to \mathbb{R}$, and let us consider a generic iterate x_k. The model m_k of g around x_k is defined, for every $s \in \mathbb{R}^n$, as:

$$m_k(s) := g(x_k) + \nabla_x g(x_k)^T s + \frac{1}{2}\lambda_k \|s\|^2 := T_k\left[g\right](s) + \frac{1}{2}\lambda_k \|s\|^2,$$

where $\lambda_k > 0$ is the regularization parameter. To move from x_k to x_{k+1} we define a step s_k as the minimizer of $m_k(s)$, which means that s_k takes the form

$$s_k = -\frac{\nabla_x g(x_k)}{\lambda_k}.$$

As in classical Levenberg-Marquardt and trust region methods, once we have s_k we would like to use it to define $x_{k+1} = x_k + s_k$, but before doing so, we have to check for the decrease $g(x_k) - g(x_k + s_k)$ and compare it with the decrease for the Taylor polynomial $T_k\left[g\right](0) - T_k\left[g\right](s_k)$. This check is performed by computing the ratio ρ_k as

$$\rho_k = \frac{g(x_k) - g(x_k + s_k)}{T_k\left[g\right](0) - T_k\left[g\right](s_k)} = \lambda_k \frac{g(x_k) - g(x_k + s_k)}{\|\nabla_x g(x_k)\|^2}.$$

If $\rho_k \approx 1$, we have a sufficient decrease of g, thus we can accept the step s_k, set $x_{k+1} =$

[1] The ℓ denotes the number of levels in the hierarchical problem description.

$x_k + s_k$ and reduce the regularization parameter λ_k. Otherwise, we keep the iterate unchanged, i.e. $x_{k+1} = x_k$, and we increase λ_k. The regularization term changes at each iteration based on the outcome of the previous iteration, and this also affects the size of s_k. If in fact at iteration $k-1$ we had sufficient decrement then λ_k is smaller than λ_{k-1} and this allows a longer s_k step along the direction of the antigradient $-\nabla_x g(x_k)$. If, on the other hand, we have had failure λ_k is larger than λ_{k-1}, hence the step s_k will be shorter. We choose to focus on AR1 since it is easier to adapt to a multilevel context than the Trust Region [20] and it has the same evaluation complexity bounds as first-order trust-region method [21, Theorem 2.4.4].

In this chapter, we present the MU$^\ell$STREG method to address the general problem (2.1). In particular, we focus on a version with a two-level hierarchy that we use to prove the convergence properties of the method. After that, we show in detail a version of MU$^\ell$STREG specific for problem (2.2) that exploits a hierarchy in the function approximations only. This gives us the chance to investigate from a practical point of view the behavior of multilevel methods as variance reduction methods with adaptive regularization. We test the resulting method on both convex and nonconvex problems and we compare it to a mini-batch SVRG, due to the close relation of our method to variance reduction methods. We show that while achieving comparable performance of non-fined tuned versions of mini-batch SVRG on convex problems, our method greatly outperforms SVRG on nonconvex ones. Moreover, we investigate the theoretical and practical advantages of the stochastic multilevel framework. Notably, differently from deterministic multilevel schemes, the stochastic framework does not require the fine-level objective function to coincide with the original objective. Thus in the context of problem (2.2), considering the full sample set is not necessary, while it is required by the convergence theory of classical variance reduction methods. We show in practice that the method remains robust without dropping accuracy when considering fine levels with smaller sample sets.

The chapter is organized as follows. In Section 2.1 we introduce our MU$^\ell$STREG method both in its general formulation and in a two-level version (MU^{2}STREG); we propose the convergence analysis using MU^{2}STREG, in Section 2.2. In Section 2.3 we specialize the MU$^\ell$STREG framework to the finite-sum setting of problem (2.2) and we analyze the numerical performance of the method in Section 2.4.

2.1 The multilevel stochastic regularized gradient method

In this section, we describe our new MUltilevel STochastic REegularized Gradient method (MU$^\ell$STREG) for the solution of problem (2.1).

2.1.1 Hierarchical representation of problem (2.1)

We assume access to a hierarchy of stochastic functions $\{f^\ell\}$ for $\ell = 1, \ldots, \ell_{\max}$, that approximate f. More precisely, our function approximations will take the form $f^\ell := f^h(x^h, \varepsilon^l)$, where $\{\varepsilon^l\}_{l=1}^{l_{\max}}$ are random variables such that, for fixed h, the evaluation of $f^h(x^h, \varepsilon^l)$ is more accurate (less noisy) than the evaluation of $f^h(x^h, \varepsilon^{l-1})$ for each $l = 2, \ldots, l_{\max}$. Moreover, f^h for $h = 1, \ldots, h_{\max}$ are function approximations defined on lower dimensional spaces, i.e., $x^h \in \mathcal{V}^h$, with $\mathcal{V}^1 \subseteq \mathcal{V}^2 \subseteq \cdots \subseteq \mathcal{V}^{h_{\max}}$. This structure defines a *stochastic multilevel problem description* of problem (2.1), where $f^{\ell_{\max}} = f^{h_{\max}}(x^{h_{\max}}, \varepsilon^{l_{\max}})$ corresponds to the *fine* level function and $f^\ell = f^h(x^h, \varepsilon^l)$ are the *coarse* approximations for $\ell = (h, l)$, $\ell = 2, \ldots, \ell_{\max}$. *For each level ℓ, $\phi^\ell(x)$ will denote a computable version of $f^h(x^h, \varepsilon^l)$, where ε^l is a random variable*, and we assume that $\nabla\phi^\ell(x)$ is available as well. If the hierarchy is built both in variable space and function space, the level $\ell = (h, l)$ is identified by a subset of variables and a noise level l, such that $h \leq h + 1$ and $l \leq l + 1$ and at least one of these inequalities is strict. As in classical multilevel methods, we assume to have at disposal some transfer operators R^ℓ (restriction) and P^ℓ (prolongation) to transfer the information (variables and gradients) from level ℓ to level $\ell - 1$ and vice-versa, such that $R^\ell = \nu(P^\ell)^T$ for some $\nu > 0$ [18]. Differently from the classical framework, such operators may be random. If the hierarchy is built just in the functions space all the variables will have the same dimension and the transfer operators will thus just be the identity.

Example 1. *In the case of Problem (2.2), if the hierarchy is built just in the sample space (i.e., $h = h_{\max}$ for all ℓ) the approximations ϕ^ℓ would be the averaged sum of the f_i over nested subsets of this large set, that is $\phi^\ell := f^{\mathcal{S}^\ell}$ where:*

$$f^{\mathcal{S}^\ell}(x) = \frac{1}{|\mathcal{S}^\ell|} \sum_{i \in \mathcal{S}^\ell} f_i(x),$$

for $\mathcal{S}^\ell \subseteq \mathcal{S}^{\ell+1}$ for all ℓ (cf. Section 2.3). If the sampling is done randomly, such function approximations will depend on a random variable ε^l, that defines hierarchy in the function space.

Example 2. *Consider the following problem:*

$$\min_{x\in\mathbb{R}^n} \sum_{j=1}^{n} (Au(x_j) - g(x_j))^2 + \sum_{i\in\mathcal{M}} (u(x_i) - \bar{u}_i)^2$$

arising from the discretization of a partial differential equation on a grid with n points. The first term takes into account the residual of the partial differential equation and the second one is a data-fitting term to a set of available measures $\mathcal{M} = \{\bar{u}_i\}$. For this problem, we can build a hierarchy in both spaces. Let us consider a level $\ell = (h, l)$: h will be associated to a coarser grid, i.e., to a subset of the variables $\mathcal{V}^h \subset \mathbb{R}^n$, while l to a subset of the measurements $\mathcal{S}^l$, drawn randomly from $\mathcal{M}$. In classical multigrid, such subsets are chosen in a deterministic way, in our framework they can be chosen randomly. The function approximation for level ℓ will thus be

$$\phi^\ell(x) = \|Au(x) - g(x)\|^2 + \sum_{i\in\mathcal{S}^l} (u(x_i) - \bar{u}_i)^2, \quad x \in \mathcal{V}^h.$$

Example 3. *Consider the setting proposed in [22]: given $x_k \in \mathbb{R}^n$, assume to randomly choose a p-dimensional affine space $\mathcal{Y}_k \subset \mathbb{R}^n$ with $p < n$ given by the range of $Q_k \in \mathbb{R}^{n\times p}$, i.e.,*

$$\mathcal{Y}_k = \{x_k + Q_k\hat{s} : \hat{s} \in \mathbb{R}^p\}.$$

A random lower-dimensional approximation to f would be given by

$$\phi(x) = f(x_k + Q_k x), \quad \text{for } x \in \mathbb{R}^p.$$

2.1.2 The step computation

For any level ℓ and at each iteration k, our multilevel gradient method can choose between two different types of stochastic steps: a gradient step, which is known as the *fine step*, or a *coarse step* computed by exploiting the approximations of f. Notice that the steps are all stochastic as, differently from classical deterministic multilevel schemes, all the

function approximations (including $\phi^{\ell_{\max}}$, which does not need to be equal to f) are random approximations. In both cases, given the objective function f^ℓ of that level, the step is computed by minimizing a regularized model of the form

$$m_k^{R,\ell}(s) = m_k^\ell(s) + \frac{\lambda_k^\ell \|\nabla_x f^\ell(x_k^\ell)\|}{2}\|s\|^2, \tag{2.3}$$

for some $\lambda_k^\ell > 0$. If $\ell = \ell_{\max}$ then $f^\ell = \phi^{\ell_{\max}}$ is the finest approximation. For the lower levels, f^ℓ is the regularized model from the immediate upper level, as specified below. The definition of m_k^ℓ also depends on the kind of step taken.

- *Fine step.* In this case, we define m_k^ℓ as the first-order Taylor series

  $$T_k[f^\ell](s) := f^\ell(x_k^\ell) + \nabla_x f^\ell(x_k^\ell)^T s,$$

  of f^ℓ in x_k^ℓ, the objective function of that level. Minimizing the regularized model (2.3) thus amounts to choosing the step

  $$s_k^\ell = -\frac{1}{\lambda_k^\ell \|\nabla f^\ell(x_k^\ell)\|}\nabla_x f^\ell(x_k^\ell),$$

  i.e., a classical (stochastic) gradient step, where the step-size depends on the norm of the gradient as in [14], cf. discussion in [14, section 3.1].

- *Coarse step.* The random model m_k^ℓ is in this case built exploiting the stochastic approximations $\{\phi^\ell\}_{\ell=1}^{\ell_{\max}}$ of f and is thus either defined in a lower dimensional space, or employs inaccurate function approximations, or both. The algorithm in this case recursively calls itself to find the coarse step. More precisely, starting at the finest level $\ell_{\max} = (h_{\max}, \ell_{\max})$ and considering the finest approximation $\phi^{\ell_{\max}}$ of f and the immediately coarser approximation $\phi^{\ell_{\max}-1}$, at iteration k we define $m_k^{\ell_{\max}} = \varphi_k^{\ell_{\max}-1}$ where

  $$\begin{aligned}\varphi_k^{\ell_{\max}-1}(s^{\ell_{\max}-1}) = {} & \phi^{\ell_{\max}-1}(R^{\ell_{\max}} x_k^{\ell_{\max}} + s^{\ell_{\max}-1}) \\ & + (R^{\ell_{\max}}\nabla_x \phi^{\ell_{\max}}(x_k^{\ell_{\max}}) - \nabla_x \phi^{\ell_{\max}-1}(R^{\ell_{\max}} x_k^{\ell_{\max}}))^T s_k^{\ell_{\max}-1},\end{aligned}$$

  i.e., $\varphi_k^{\ell_{\max}-1}$ is a modification of the coarse function $\phi^{\ell_{\max}-1}$ through the addition of a correction term. This correction aims to enforce the following relation for

$s^{\ell_{\max}} = P s^{\ell_{\max}-1}$:

$$\nabla_s \varphi_k^{\ell_{\max}-1}(0)^T s^{\ell_{\max}-1} = R^{\ell_{\max}} \nabla_x \phi^{\ell_{\max}}(x_k^{\ell_{\max}}),$$

which ensures that the behaviour of the coarse model is coherent with the fine objective function, up to order one.

The regularized model $m_k^{R,\ell}$ is then (approximately) minimized wrt s, by recursively calling the multilevel procedure, thus taking either a fine step on level $\ell - 1$ or building a coarse model for $m_k^{R,\ell_{\max}}(s)$ involving the approximation $\phi^{\ell_{\max}-2}$ and so on. The recursive call is stopped as soon as a step $s_k^{\ell-1}$ is found that satisfies the following conditions:

$$m_k^{R,\ell}(s_k^{\ell-1}) < m_k^{R,\ell}(0), \quad \left\| \nabla_s m_k^{R,\ell}(s_k^{\ell-1}) \right\| \leq \epsilon^{\ell-1} \|s_k^{\ell-1}\|, \tag{2.4}$$

for some $\epsilon^{\ell-1} > 0$, and we set $s_k^\ell := P^\ell s_k^{\ell-1}$. As we will see, these conditions will ensure the convergence of the multilevel method in the spirit of the Adaptive-Regularization algorithm with a first-order model described e.g., in [21, Sec. 2.4.1]. Note that even if we use a first-order model at the fine level, we could use a higher-order method to minimize the lower level model.

To be meaningful, the coarse steps are restricted to iterations such that

$$\|R^\ell \nabla_x f^\ell(x_k^\ell)\| \geq \kappa^\ell \|\nabla_x f^\ell(x_k^\ell)\|,$$

for $\kappa^\ell \in (0, \min\{1, \|R^\ell\|\})$ [38].

This framework is flexible and encompasses several actual implementations: at each iteration k one needs to choose whether to employ the fine or the coarse step. A sketch of a possible MU$^\ell$STREG cycle of iterations is depicted in Figure 2.1.

2.1.3 The step acceptance

The step s_k^ℓ is used to define a trial point $x_k^\ell + s_k^\ell$ and two estimates of $f^\ell(x_k^\ell)$ and $f^\ell(x_k^\ell + s_k^\ell)$, denoted by $f_k^{\ell,0}$ and $f_k^{\ell,s}$, which involve approximations of $f^\ell(x_k^\ell)$ and $f^\ell(x_k^\ell + s_k^\ell)$. The achieved reduction given by $f_k^{\ell,0} - f_k^{\ell,s}$ over the predicted reduction $m_k^\ell(0) - m_k^\ell(s_k^\ell)$

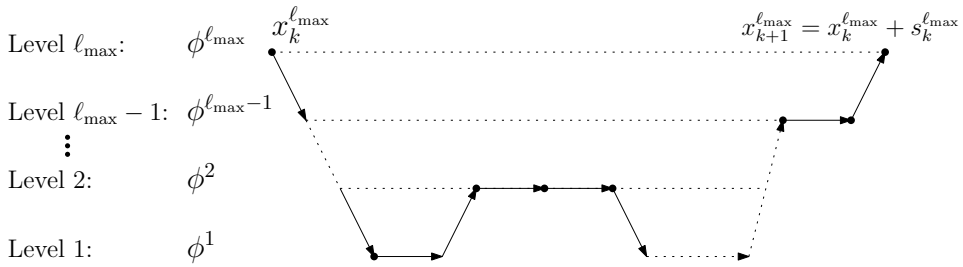

Figure 2.1: Sketch of a possible iteration scheme for MU$^\ell$STREG.

is computed to decide whether to accept the trial point or not. More precisely, the step acceptance is based on the ratio:

$$\rho_k = \frac{f_k^{\ell,0} - f_k^{\ell,s}}{m_k^\ell(0) - m_k^\ell(s_k^\ell)}. \tag{2.5}$$

A *successful* iteration is declared if the model is accurate, i.e., ρ_k is larger than or equal to a chosen threshold $\eta_1 \in (0,1)$ and $\|\nabla f_k^\ell(x_k^\ell)\| \geq \frac{\eta_2}{\lambda_k^\ell}$ for some $\eta_2 > 0$; otherwise the iteration is declared *unsuccessful* and the step is rejected. The test for the step acceptance is combined with the update of the regularization parameter λ_k^ℓ for the next iteration. The update is still based on the ratio (2.5). If the step is successful, the regularization parameter is decreased, otherwise it is increased.

The full multilevel procedure with ℓ levels, specialized for the problem (2.2), is described in Algorithm 6 and will be introduced in Section 2.3. In the following section, for the sake of simplicity, we detail the procedure in the two-level case.

2.1.4 MU^{2}STREG: the two-level case

We assume here that we have just two approximations to our objective at our disposal and therefore we omit the superscript ℓ: we denote by Φ the approximation at the highest level ($\Phi = \phi^{\ell_{\max}} = \varphi^{\ell_{\max}}$ in the previous notation) and by ϕ the other less accurate approximation available. Moreover, let n_1 and n_2 be the dimensions of the fine and coarse spaces, respectively, and let R and P be the grid operators. We sketch the MU$^\ell$STREG procedure in Algorithm 5 where we rename it as MU^{2}STREG. Below we collect the main assumptions on the algorithmic steps that will be used in the convergence analysis in the next section.

Assumption 1. *At each iteration k of Algorithm 5 let*

$$m_k^R(s) = m_k(s) + \frac{\lambda_k \|\nabla_x \Phi(x_k)\|}{2}\|s\|^2, \tag{2.6}$$

and

$$m_k(s) = \begin{cases} T_k[\Phi](s) := \Phi(x_k) + \nabla_x \Phi(x_k)^T s, & \textit{(fine step)}, \\ \varphi_k(s) := \phi(Rx_k + s) + (R\nabla_x \Phi(x_k) - \nabla_x \phi(Rx_k))^T s, & \textit{(coarse step)}. \end{cases} \tag{2.7}$$

The step $s_k \in \mathbb{R}^{n_1}$ is computed so that either:

$$s_k = -\frac{\nabla_x \Phi(x_k)}{\lambda_k \|\nabla_x \Phi(x_k)\|}, \qquad \textit{(fine step) or} \tag{2.8}$$

$$s_k = Ps^*, \ s^* \in \mathbb{R}^{n_2}, \qquad \textit{(coarse step)} \tag{2.9}$$

where

$$m_k^R(s^*) < m_k^R(0) \quad \textit{and} \quad \|\nabla_s m_k^R(s^*)\| = \|\nabla_s \varphi_k(s^*) + \lambda_k \|\nabla_x \Phi(x_k)\| s^*\| \leq \theta \|s^*\| \tag{2.10}$$

for some $\theta > 0$. The definition of the coarse model ensures that

$$\nabla_s \varphi_k(0) = R\nabla_x \Phi(x_k). \tag{2.11}$$

The use of the coarse step is restricted to iterations k such that

$$\|R\nabla_x \Phi(x_k)\| \geq \kappa_H \|\nabla_x \Phi(x_k)\| \tag{2.12}$$

for $\kappa_H \in (0, \min\{1, \|R\|\})$. We assume that $R = \nu P^T$ with $\nu = 1$, without loss of generality, and that $\|R\| = \|P\| \leq \kappa_R$ for $\kappa_R > 0$.

Remark 1. *From (2.6), (2.7) and (2.10), when the coarse model is used, it follows:*

$$\varphi_k(s^*) - \varphi_k(0) < -\frac{1}{2}\lambda_k \|\nabla_x \Phi(x_k)\| \|s^*\|^2. \tag{2.13}$$

Notice that Algorithm 5 is a flexible framework encompassing several actual implementations: at Step 2 one needs to choose whether to employ the fine or the coarse step.

Algorithm 5 MU^{2}STREG$(x_0, \Phi, \phi, \lambda_0, \epsilon)$ two-level stochastic regularized gradient method

1: • **Initialization:** Choose $x_0 \in \mathbb{R}^n$ and $\lambda_0 > \lambda_{\min}$ with $\lambda_{\min} > 0$. Set the constants $\eta_1 \in (0,1)$, $\eta_2 > 0$ and $\gamma \in (0,1)$. Set $k = 0$.

2: • **Model choice:** If (2.12) holds, choose if to use the fine level model and go to Step 3, or the coarse level model and go to Step 4. Otherwise, go to Step 3.

3: • **Fine step computation:** Define $m_k(s) = T_k[\Phi](s) = \Phi(x_k) + \nabla_x \Phi(x_k)^T s$. Set $s_k = -\frac{\nabla_x \Phi(x_k)}{\lambda_k \|\nabla_x \Phi(x_k)\|}$. Go to Step 5.

4: • **Coarse step computation:** Define a lower level model and its regularized version as:

$$\varphi_k(s) = \phi(R\, x_k + s) + [R\, \nabla_x \Phi(x_k) - \nabla_x \phi(R\, x_k)]^T s,$$
$$m_k^R(s) = \varphi_k(s) + \frac{1}{2}\lambda_k \|\nabla_x \Phi(x_k)\| \|s\|^2.$$

Approximately minimize m_k^R, yielding an approximate solution s_k satisfying (2.10). Define $m_k(s) = \varphi_k(s)$.

5: • **Acceptance of the trial point and regularization parameter update:** Obtain estimates f_k^0 and f_k^s of $f(x_k)$ and $f(x_k + s_k)$, respectively and compute $\rho_k = \dfrac{f_k^0 - f_k^s}{m_k(0) - m_k(s_k)}$. **If** $\rho_k \geq \eta_1$ and $\|\nabla_x \Phi(x_k)\| \geq \eta_2/\lambda_k$ **then** set $x_{k+1} = x_k + s_k$ and $\lambda_{k+1} = \gamma \lambda_k$. **Else** set $x_{k+1} = x_k$ and $\lambda_{k+1} = \gamma^{-1} \lambda_k$.

6: • **Check stopping criterion**. If satisfied stop, otherwise set $k = k + 1$ and go to Step 2.

2.2 Convergence theory

In this section, we provide a theoretical analysis of the proposed multilevel method proving the global convergence to first-order critical points. Note that, as the method is recursive, we can restrict the analysis to the two-level case. We thus focus on MU^{2}STREG as described in Subsection 2.1.4. The analysis follows the scheme proposed in [24] and is extended to adaptive regularization methods and adapted to include also the multilevel steps.

Let us now first state our assumptions: we need some regularity assumptions as in [15].

Assumption 2. *Let $f : \mathbb{R}^n \to \mathbb{R}$,$\Phi : \mathbb{R}^{n_1} \to \mathbb{R}$ and $\phi : \mathbb{R}^{n_2} \to \mathbb{R}$ with $n \geq n_1 \geq n_2$, be continuously differentiable and bounded below functions. Let us assume that the gradients of f, Φ and ϕ are Lipschitz continuous, i.e., that there exist constants L_f, L_Φ, L_ϕ such*

that

$$\begin{aligned}
\|\nabla_x f(x) - \nabla_x f(y)\| &\leq L_f \|x - y\| \quad \textit{for all} \quad x, y \in \mathbb{R}^n, \\
\|\nabla_x \Phi(x) - \nabla_x \Phi(y)\| &\leq L_\Phi \|x - y\| \quad \textit{for all} \quad x, y \in \mathbb{R}^{n_1}, \\
\|\nabla_x \phi(x) - \nabla_x \phi(y)\| &\leq L_\phi \|x - y\| \quad \textit{for all} \quad x, y \in \mathbb{R}^{n_2}.
\end{aligned}$$

We assume that the models we are considering are random functions and so is their behavior and influence on the iterations. Hence, M_k will denote a random model in the k-th iteration, while we will use the notation $m_k = M_k(\omega)$ for its realizations. As a consequence of using random models, the iterates X_k, the regularization parameter Λ_k and the steps S_k are also random quantities, and so $x_k = X_k(\omega)$, $\lambda_k = \Lambda_k(\omega)$, $s_k = S_k(\omega)$ will denote their respective realizations. Similarly, let random quantities F_k^0, F_k^s denote the estimates of $f(X_k)$ and $f(X_k + S_k)$, with their realizations denoted by $f_k^0 = F_k^0(\omega)$ and $f_k^s = F_k^s(\omega)$. In other words, Algorithm 5 results in a stochastic process $\{M_k, X_k, S_k, \Lambda_k, F_k^0, F_k^s\}$. Our goal is to show that under certain conditions on the sequences $\{M_k\}$ and $\{F_k^0, F_k^s\}$ the resulting stochastic process has desirable convergence properties with probability one. In particular, we will assume that models M_k and estimates F_k^0, F_k^s are sufficiently accurate with sufficiently high probability, conditioned on the past. To formalize conditioning on the past, let $\mathcal{F}_{k-1}^{M \cdot F}$ denote the σ-algebra generated by $M_0, \ldots, M_{k-1}$ and $F_0, \ldots, F_{k-1}$ and let $\mathcal{F}_{k-1/2}^{M \cdot F}$ denote the σ-algebra generated by $M_0, \ldots, M_k$ and $F_0, \ldots, F_{k-1}$. To formalize sufficient accuracy we use the measure for the accuracy introduced in [14], which adapts to regularized models those originally proposed in [24].

Definition 1. *Suppose that ∇f is Lipschitz continuous. Given $\lambda_k > 0$, a function m is a κ-fully linear model of f around the iterate x_k provided for $\kappa = (\kappa_f, \kappa_g)$, that for all y in a neighborhood of x_k:*

$$\|\nabla_x f(y) - \nabla_x m(y)\| \leq \frac{\kappa_g}{\lambda_k}, \tag{2.14}$$

$$|f(y) - m(y)| \leq \frac{\kappa_f}{\lambda_k^2}. \tag{2.15}$$

Remark 2. *The first-order correction imposed on the coarser levels ensures that (at least locally) the coarse model is fully linear. Thus we will ask for this requirement on the fine*

level model $T_k[\Phi]$. Imposing this condition on the fine level only will be enough to ensure convergence of the method.

Specifically, we will consider probabilistically fully linear models, according to the following definition [24]:

Definition 2. *A sequence of random models $\{M_k\}$ is said to be α-probabilistically κ-fully linear with respect to the corresponding sequence $\{X_k, \Lambda_k\}$ if the events*

$$I_k = \{M_k \text{ is a } \kappa\text{-fully linear model of } f \text{ around } X_k\} \tag{2.16}$$

satisfy the condition

$$\mathbb{P}(I_k|\mathcal{F}_{k-1}^{MF}) \geq \alpha,$$

where $\mathcal{F}_{k-1}^{MF}$ is the σ-algebra generated by $M_0, \ldots, M_{k-1}$ and $F_0, \ldots, F_{k-1}$.

We will also require function estimates to be sufficiently accurate.

Definition 3. *The estimates f_k^0 and f_k^s are said to be ϵ_f-accurate estimates of $f(x_k)$ and $f(x_k + s_k)$ respectively, for a given λ_k if*

$$|f_k^0 - f(x_k)| \leq \frac{\epsilon_f}{\lambda_k^2} \text{ and } |f_k^s - f(x_k + s_k)| \leq \frac{\epsilon_f}{\lambda_k^2}.$$

In particular, we will consider probabilistically accurate estimates as in [24]:

Definition 4. *A sequence of random estimates $\{F_k^0, F_k^s\}$ is said to be β-probabilistically ϵ_f-accurate with respect to the corresponding sequence $\{X_k, \Lambda_k, S_k\}$ if the events*

$$J_k = \left\{F_k^0, F_k^s \text{ are } \epsilon_f\text{-accurate estimates of } f(x_k) \text{ and } f(x_k + s_k), \text{ respectively, for } \Lambda_k\right\} \tag{2.17}$$

satisfy the condition

$$\mathbb{P}(J_k|\mathcal{F}_{k-1/2}^{MF}) \geq \beta,$$

where ϵ_f is a fixed constant and $\mathcal{F}_{k-1/2}^{MF}$ is the σ-algebra generated by $M_0, \ldots, M_k$ and $F_0, \ldots, F_{k-1}$.

Following [24], in our analysis we will require that our method has access to α-probabilistically κ-fully linear models, for some fixed κ and to β-probabilistically ϵ_f

accurate estimates, for some fixed, sufficiently small ϵ_f. Cf. [24, Section 5] for procedures for constructing probabilistically fully linear models, and probabilistically accurate estimates. Basically, when the function approximations come from a subsampling this construction is possible if the model accounts for enough samples. Notice that we will assume this condition only on the finest level, for the coarse ones this is not necessary thanks to (2.11), obtained from the definition of the coarse model.

2.2.1 Convergence analysis

We start by recalling three useful relations, following from Taylor's theorem, see for example [21, Corollary A.8.4].

Lemma 1. *Let $g : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function with Lipschitz continuous gradient, with L the corresponding Lipschitz constant. Given its first order truncated Taylor series in x $T[g](s) := g(x) + \nabla_x g(x)^T s$, it holds:*

$$g(x+s) = T[g](s) + \int_0^1 [\nabla_x g(x+\xi s) - \nabla_x g(x)]^T s \, d\xi, \tag{2.18}$$

$$|g(x+s) - T[g](s)| \leq \frac{L}{2}\|s\|^2, \tag{2.19}$$

$$\|\nabla_x g(x+s) - \nabla_s T[g](s)\| \leq L\|s\|. \tag{2.20}$$

We now propose two technical lemmas on the coarse step.

Lemma 2. *Let Assumptions 1 and 2 hold. Consider a realization of Algorithm 5 where at iteration k the coarse model is used and let $s_k = Ps^*$ be the resulting step. Then it holds:*

$$|\varphi_k(0) - \varphi_k(s^*) - (T_k[\Phi](0) - T_k[\Phi](s_k))| \leq \frac{L_\phi}{2}\|s^*\|^2. \tag{2.21}$$

Proof. Using the first order Taylor expansion of φ_k and (2.18) applied to φ_k, and considering that from (2.11), $\nabla_s \varphi_k(0)^T s^* = \nabla_x \Phi(x_k)^T s_k$, we can write:

$$\varphi_k(0) - \varphi_k(s^*) = -\nabla_x \Phi(x_k)^T s_k - \int_0^1 [\nabla_s \varphi_k(\xi s^*) - \nabla_s \varphi_k(0)]^T s^* \, d\xi.$$

Since $\nabla_x \Phi(x_k)^T s_k = T_k[\Phi](s_k) - T_k[\Phi](0)$, using Assumption 2 and recalling that φ_k

and ϕ differ just by a linear term, we obtain:

$$\begin{aligned}&|\varphi_k(0) - \varphi_k(s^*) - (T_k[\Phi](0) - T_k[\Phi](s_k))| \\ &\leq \int_0^1 |[\nabla_s \varphi_k(\xi s^*) - \nabla_s \varphi(0)]^T s^*| \, d\xi \leq \int_0^1 \|\nabla_s \varphi_k(\xi s^*) - \nabla_s \varphi_k(0)\| \|s^*\| \, d\xi \leq \frac{L_\phi}{2} \|s^*\|^2.\end{aligned}$$

□

Lemma 3. *Under Assumptions 1 and 2, for any realization of Algorithm 5 and for each iteration k where the coarse step is used, it exists a constant $K > 0$ such that:*

$$\|R\nabla_x \Phi(x_k)\| \leq (K + \lambda_k \|\nabla_x \Phi(x_k)\|)\|s^*\|, \quad \text{with } K = 2L_\Phi \kappa_R^2 + L_\phi + \theta. \tag{2.22}$$

Proof. From the Lipschitz continuity of $\nabla_x \Phi(x_k)$, we have:

$$\begin{aligned}\|R\nabla_x \Phi(x_k)\| &\leq \|R(\nabla_x \Phi(x_k) - \nabla_x \Phi(x_k + s_k))\| + \|R\nabla_x \Phi(x_k + s_k)\| \\ &\leq L_\Phi \|R\| \|s_k\| + \|R\nabla_x \Phi(x_k + s_k)\| \leq L_\Phi \kappa_R^2 \|s^*\| + \|R\nabla_x \Phi(x_k + s_k)\|\end{aligned}$$

where the last inequality follows from $s_k = Ps^*$. Moreover,

$$\begin{aligned}\|R\nabla_x \Phi(x_k + s_k)\| \leq &\|R(\nabla_x \Phi(x_k + s_k) - \nabla_s T_k[\Phi](s_k))\| \\ &+ \|R\nabla_s T_k[\Phi](s_k) - \nabla_s \varphi_k(s^*)\| \\ &+ \|-\lambda_k \|\nabla_x \Phi(x_k)\| \|s^*\| + \nabla_s \varphi_k(s^*)\| \\ &+ \lambda_k \|\nabla_x \Phi(x_k)\| \|s^*\|.\end{aligned}$$

Let us consider the first three terms separately.

1. By (2.20),

$$\|R(\nabla_x \Phi(x_k + s_k) - \nabla_s T_k[\Phi](s_k))\| \leq L_\Phi \kappa_R \|s_k\| \leq L_\Phi \kappa_R^2 \|s^*\|.$$

2. For the second term, using the definition of $T_k[\Phi](s)$ and $\varphi_k(s)$ in (2.7) and the

Lipschitz continuity of $\nabla_x \phi(x)$, it holds:

$$\begin{aligned}
&\|R\nabla_s T_k[\Phi](s_k) - \nabla_s \varphi_k(s^*)\| \\
&\quad = \|R\nabla_x \Phi(x_k) - \nabla_x \phi(Rx_k + s^*) - (R\nabla_x \Phi(x_k) - \nabla_x \phi(Rx_k))\| \\
&\quad = \|R\nabla_x \Phi(x_k) - \nabla_x \phi(Rx_k + s^*) - R\nabla_x \Phi(x_k) + \nabla_x \phi(Rx_k)\| \\
&\quad = \|\nabla_x \phi(Rx_k) - \nabla_x \phi(Rx_k + s^*)\| \leq L_\phi \|s^*\|.
\end{aligned}$$

3. The third term from (2.10) is bounded by $\theta\|s^*\|$.

Thus we finally obtain the thesis. □

The following lemma relates the coarse step size and the regularization parameter λ.

Lemma 4. *Let Assumptions 1 and 2 hold. Assume that at iteration k the coarse step is used. Let K be defined as in (2.22) and assume that*

$$\frac{1}{\lambda_k} \leq \min\left\{\frac{1}{K}, \frac{1}{2L_\phi}\right\}\|\nabla_x \Phi(x_k)\|, \tag{2.23}$$

then

$$\frac{\kappa_H}{2\lambda_k} \leq \|s^*\| \leq \frac{4\kappa_R}{\lambda_k}. \tag{2.24}$$

Proof. The first inequality follows from assumption (2.23), (2.22) and (2.12):

$$\|s^*\| \geq \frac{\|R\nabla_x \Phi(x_k)\|}{K + \lambda_k \|\nabla_x \Phi(x_k)\|} \geq \frac{\kappa_H \|\nabla_x \Phi(x_k)\|}{K + \lambda_k \|\nabla_x \Phi(x_k)\|} \geq \frac{\kappa_H}{2\lambda_k}. \tag{2.25}$$

The second inequality follows from (2.19) applied to φ_k:

$$|\varphi_k(s^*) - \varphi_k(0)| - |\nabla_s \varphi_k(0)^T s^*| \leq |\varphi_k(s^*) - \varphi_k(0) - \nabla_s \varphi_k(0)^T s^*| \leq \frac{L_\phi}{2}\|s^*\|^2,$$

where we have used the fact that from Assumption 2 and (2.7) φ_k is L_ϕ smooth. Thus, from (2.11),

$$\begin{aligned}
\varphi_k(0) - \varphi_k(s^*) &\leq |\nabla_s \varphi_k(0)^T s^*| + \frac{L_\phi}{2}\|s^*\|^2 = |\nabla_x \Phi(x_k)^T s_k| + \frac{L_\phi}{2}\|s^*\|^2 \\
&\leq \|\nabla_x \Phi(x_k)\|\|s_k\| + \frac{L_\phi}{2}\|s^*\|^2 \leq \kappa_R \|\nabla_x \Phi(x_k)\|\|s^*\| + \frac{L_\phi}{2}\|s^*\|^2.
\end{aligned}$$

Combining this with (2.13) we have:

$$\frac{1}{2}\lambda_k\|\nabla_x\Phi(x_k)\|\|s^*\|^2 \overset{(2.13)}{\leq} \varphi_k(0) - \varphi_k(s^*) \leq \kappa_R\|\nabla_x\Phi(x_k)\|\|s^*\| + \frac{L_\phi}{2}\|s^*\|^2.$$

Thus

$$\left(\frac{1}{2}\lambda_k\|\nabla_x\Phi(x_k)\| - \frac{L_\phi}{2}\right)\|s^*\|^2 \leq \kappa_R\|\nabla_x\Phi(x_k)\|\|s^*\|.$$

From (2.23) $\frac{1}{2}\lambda_k\|\nabla_x\Phi(x_k)\| - \frac{L_\phi}{2} \geq \frac{1}{4}\lambda_k\|\nabla_x\Phi(x_k)\|$ and thus

$$\frac{1}{4}\lambda_k\|\nabla_x\Phi(x_k)\|\|s^*\| \leq \kappa_R\|\nabla_x\Phi(x_k)\|.$$

□

In the following lemma, we measure the decrease predicted by the model.

Lemma 5. *Let Assumptions 1 and 2 hold. For any realization of Algorithm 5 and for each k it holds:*

$$m_k(s_k) - m_k(0) \leq \begin{cases} -\frac{\|\nabla_x\Phi(x_k)\|}{\lambda_k} & \text{if fine step} \\ -\frac{\lambda_k\|\nabla_x\Phi(x_k)\|}{2}\|s^*\|^2 & \text{if coarse step} \end{cases} \tag{2.26}$$

Proof. If the fine step is used,

$$\begin{aligned} m_k(s_k) - m_k(0) &= T_k[\Phi](s_k) - T_k[\Phi](0) \\ &= \nabla_x\Phi(x_k)^T s_k = -\frac{\|\nabla_x\Phi(x_k)\|^2}{\lambda_k\|\nabla_x\Phi(x_k)\|} = -\frac{\|\nabla_x\Phi(x_k)\|}{\lambda_k}. \end{aligned}$$

If the coarse step is used:

$$m_k(s_k) - m_k(0) = \varphi_k(s^*) - \varphi_k(0) \overset{(2.13)}{\leq} -\frac{\lambda_k\|\nabla_x\Phi(x_k)\|}{2}\|s^*\|^2.$$

□

We now prove some auxiliary lemmas that provide conditions under which the decrease of the true objective function f is guaranteed. The first lemma states that if the regularization parameter is large enough relative to the size of the model gradient and

if the model is fully linear, then the step s_k provides a decrease in f proportional to the size of the model gradient.

Lemma 6. *Under Assumptions 1 and 2, suppose that $T_k[\Phi]$ is a (κ_f, κ_g)-fully linear model of f in a neighborhood of x_k. If*

$$\frac{1}{\lambda_k} \leq \min\left\{\frac{1}{K}, \frac{\kappa_H^2}{64\kappa_f}, \frac{1}{2L_\phi}\right\}\|\nabla_x \Phi(x_k)\|, \tag{2.27}$$

then the trial step s_k leads to an improvement in $f(x_k + s_k)$ such that

$$f(x_k + s_k) - f(x_k) \leq -\frac{\kappa_H^2}{32}\frac{\|\nabla_x \Phi(x_k)\|}{\lambda_k}.$$

Proof. We distinguish two cases depending on the used step.

1. In the fine step case, from (2.26) we get

$$\begin{aligned} f(x_k + s_k) - f(x_k) &= f(x_k + s_k) - T_k[\Phi](s_k) + T_k[\Phi](s_k) - \\ &\quad - T_k[\Phi](0) + T_k[\Phi](0) - f(x_k) \\ &\overset{(2.15)}{\leq} \frac{2\kappa_f}{\lambda_k^2} - \frac{\|\nabla_x \Phi(x_k)\|}{\lambda_k} \\ &\overset{(2.27)}{\leq} -\frac{1}{2}\frac{\|\nabla_x \Phi(x_k)\|}{\lambda_k} \leq -\frac{\kappa_H^2}{32}\frac{\|\nabla_x \Phi(x_k)\|}{\lambda_k}. \end{aligned}$$

   where we have used that, from (2.27), $\frac{1}{\lambda_k} \leq \frac{\kappa_H^2}{64\kappa_f}\|\nabla_x \Phi(x_k)\| \leq \frac{1}{4\kappa_f}\|\nabla_x \Phi(x_k)\|$.

2. When the coarse step is used, we have

$$\begin{aligned} f(x_k + s_k) - f(x_k) &= f(x_k + s_k) - T_k[\Phi](s_k) \\ &\quad + T_k[\Phi](s_k) - T_k[\Phi](0) - \varphi_k(s^*) + \varphi_k(0) \\ &\quad - \varphi_k(0) + \varphi_k(s^*) \\ &\quad + T_k[\Phi](0) - f(x_k). \end{aligned}$$

   The first and the last terms are bounded by $\frac{\kappa_f}{\lambda_k^2}$ from (2.15). The second term from Lemma 2 is bounded by $\frac{L_\phi}{2}\|s^*\|^2$. The third term is bounded by $-\frac{\lambda_k\|\nabla_x \Phi(x_k)\|}{2}\|s^*\|^2$

from (2.13). Thus

$$\begin{aligned} f(x_k + s_k) - f(x_k) &\leq \frac{2\kappa_f}{\lambda_k^2} + \left(\frac{L_\phi}{2} - \frac{\lambda_k \|\nabla_x \Phi(x_k)\|}{2}\right) \|s^*\|^2 \\ &\overset{(2.27)}{\leq} \frac{2\kappa_f}{\lambda_k^2} - \frac{\lambda_k \|\nabla_x \Phi(x_k)\|}{4} \|s^*\|^2 \\ &\overset{(2.24)}{\leq} \frac{2\kappa_f}{\lambda_k^2} - \frac{\|\nabla_x \Phi(x_k)\| \kappa_H^2}{16 \lambda_k} \\ &\overset{(2.27)}{\leq} -\frac{\kappa_H^2}{32} \frac{\|\nabla_x \Phi(x_k)\|}{\lambda_k}. \end{aligned}$$

□

The next lemma shows that for a sufficiently large regularization parameter λ_k relative to the size of the true gradient $\nabla_x f(x_k)$, the guaranteed decrease in the objective function, provided by s_k, is proportional to the size of the true gradient.

Lemma 7. *Let Assumptions 1 and 2 hold and suppose that $T_k[\Phi]$ is a (κ_f, κ_g)-fully linear model of f in a neighborhood of x_k. If*

$$\frac{1}{\lambda_k} \leq \min\left\{\frac{1}{K + \kappa_g}, \frac{1}{(64\kappa_f/\kappa_H^2) + \kappa_g}, \frac{1}{2L_\phi + \kappa_g}\right\} \|\nabla_x f(x_k)\|, \tag{2.28}$$

then the trial step s_k leads to an improvement in $f(x_k + s_k)$ such that

$$f(x_k + s_k) - f(x_k) \leq -C_1 \frac{\|\nabla_x f(x_k)\|}{\lambda_k}, \tag{2.29}$$

with $C_1 := \frac{\kappa_H^2}{32} \max\left\{\frac{K}{K+\kappa_g}, \frac{64\kappa_f}{64\kappa_f + \kappa_g \kappa_H^2}, \frac{2L_\phi}{2L_\phi + \kappa_g}\right\}$.

Proof. We first prove that the assumption of Lemma 6 is satisfied, and we use its result to deduce the decrease of the objective function in terms of $\|\nabla_x f(x_k)\|$ rather than $\|\nabla_x \Phi(x_k)\|$, by linking these two quantities through the assumption of κ-fully linear model, which yields that

$$\|\nabla_x \Phi(x_k)\| \geq \|\nabla_x f(x_k)\| - \frac{\kappa_g}{\lambda_k}. \tag{2.30}$$

From assumption (2.28) it holds

$$\|\nabla_x f(x_k)\| \geq \max\left\{K + \kappa_g, 64\kappa_f/\kappa_H^2 + \kappa_g, 2L_\phi + \kappa_g\right\} \frac{1}{\lambda_k},$$

and thus from (2.30) we have

$$\|\nabla_x \Phi(x_k)\| \geq \|\nabla_x f(x_k)\| - \frac{\kappa_g}{\lambda_k} \geq \max\{K, 64\kappa_f/\kappa_H^2, 2L_\phi\}\frac{1}{\lambda_k}.$$

Thus the assumption of Lemma 6 is satisfied and

$$f(x_k + s_k) - f(x_k) \leq -\frac{\kappa_H^2}{32}\frac{\|\nabla_x \Phi(x_k)\|}{\lambda_k}.$$

In the same way from (2.28) and (2.30) we have

$$\begin{aligned}
\|\nabla_x \Phi(x_k)\| &\geq \|\nabla_x f(x_k)\| - \frac{\kappa_g}{\lambda_k} \\
&\geq \|\nabla_x f(x_k)\| - \kappa_g \min\left\{\frac{1}{K + \kappa_g}, \frac{1}{64\kappa_f/\kappa_H^2 + \kappa_g}, \frac{1}{2L_\phi + \kappa_g}\right\}\|\nabla_x f(x_k)\| \\
&= \max\left\{\frac{K}{K + \kappa_g}, \frac{64\kappa_f}{64\kappa_f/\kappa_H^2 + \kappa_g}, \frac{2L_\phi}{2L_\phi + \kappa_g}\right\}\|\nabla_x f(x_k)\| := \tilde{C}_1\|\nabla_x f(x_k)\|.
\end{aligned}$$

We conclude that

$$f(x_k + s_k) - f(x_k) \leq -\frac{\kappa_H^2}{32}\frac{\|\nabla_x \Phi(x_k)\|}{\lambda_k} \leq -\frac{\kappa_H^2 \tilde{C}_1}{32}\frac{\|\nabla_x f(x_k)\|}{\lambda_k} := -C_1\frac{\|\nabla_x f(x_k)\|}{\lambda_k}.$$

□

We now prove a lemma that states that, if the estimates are sufficiently accurate, the fine model is fully linear and the regularization parameter is large enough with respect to the size of the model gradient, then a successful step is guaranteed.

Lemma 8. *Let Assumptions 1 and 2 hold. Suppose that $T_k[\Phi]$ is a (κ_f, κ_g)-fully linear model in a neighborhood of x_k and that the estimates $\{f_k^0, f_k^s\}$ are ϵ_f-accurate with $\epsilon_f \leq \kappa_f$. If*

$$\frac{1}{\lambda_k} \leq \min\left\{\frac{1}{K}, \frac{1}{\eta_2}, \frac{1 - \eta_1}{32\kappa_f/\kappa_H^2 + L_\phi}\right\}\|\nabla_x \Phi(x_k)\|, \tag{2.31}$$

then the k-th iteration is successful.

Proof. Let us consider ρ_k in Step 5. of Algorithm 5:

$$\begin{aligned}\rho_k &= \frac{f_k^0 - f_k^s}{m_k(0) - m_k(s_k)} \\ &= \frac{f_k^0 - f(x_k)}{m_k(0) - m_k(s_k)} + \frac{f(x_k) - m_k(0)}{m_k(0) - m_k(s_k)} + \frac{m_k(0) - m_k(s_k)}{m_k(0) - m_k(s_k)} \\ &+ \frac{m_k(s_k) - f(x_k + s_k)}{m_k(0) - m_k(s_k)} + \frac{f(x_k + s_k) - f_k^s}{m_k(0) - m_k(s_k)} := \varrho_k + 1. \end{aligned} \tag{2.32}$$

Let us now consider the numerators in this expression. Those of the first and the last terms are bounded from the assumption on the function estimates (cf. Definition 3):

$$|f_k^0 - f(x_k)| \leq \frac{\epsilon_f}{\lambda_k^2} \leq \frac{\kappa_f}{\lambda_k^2}, \qquad |f_k^s - f(x_k + s_k)| \leq \frac{\epsilon_f}{\lambda_k^2} \leq \frac{\kappa_f}{\lambda_k^2}.$$

To bound the other terms, let us now consider two cases. First, when the fine step is used $m_k = T_k[\Phi]$, which is a κ-fully linear model of f by assumption, thus the numerator of the second and fourth terms are bounded by (2.15). Consequently, the numerator of $|\varrho_k| = |\rho_k - 1|$ is bounded by $\frac{4\kappa_f}{\lambda_k^2}$. The denominator is given in (2.26). Thus by the assumption

$$|\varrho_k| = |\rho_k - 1| \leq \frac{4\kappa_f}{\lambda_k \|\nabla_x \Phi(x_k)\|} \leq 1 - \eta_1.$$

If the coarse step is used we have $m_k = \varphi_k$ and we need to further develop the expression of ρ_k:

$$\begin{aligned}\rho_k &= \frac{f_k^0 - f_k^s}{m_k(0) - m_k(s_k)} \\ &= \frac{f_k^0 - f(x_k)}{m_k(0) - m_k(s_k)} + \frac{f(x_k) - T_k[\Phi](0)}{m_k(0) - m_k(s_k)} + \frac{-T_k[\Phi](s_k) - \varphi_k(0) + \varphi_k(s^*) + T_k[\Phi](0)}{m_k(0) - m_k(s_k)} \\ &+ \frac{\varphi_k(0) - \varphi_k(s^*)}{\varphi_k(0) - \varphi_k(s^*)} + \frac{T_k[\Phi](s_k) - f(x_k + s_k)}{m_k(0) - m_k(s_k)} + \frac{f(x_k + s_k) - f_k^s}{m_k(0) - m_k(s_k)} \\ &= \varrho_k + 1 + \frac{-T_k[\Phi](s_k) - \varphi_k(0) + \varphi_k(s^*) + T_k[\Phi](0)}{m_k(0) - m_k(s_k)}. \end{aligned}$$

Concerning the previous development we thus just have an additional term. Let us

bound its absolute value:

$$\left|\frac{-T_k[\Phi](s_k) - \varphi_k(0) + \varphi_k(s^*) + T_k[\Phi](0)}{m_k(0) - m_k(s_k)}\right| \overset{(2.21)+(2.13)}{\leq} \frac{\frac{L_\phi}{2}\|s^*\|^2}{\frac{\lambda_k\|\nabla_x\Phi(x_k)\|}{2}\|s^*\|^2} = \frac{L_\phi}{\lambda_k\|\nabla_x\Phi(x_k)\|}. \tag{2.33}$$

The numerators of the terms in ϱ_k can be bounded as in the first case. We thus have

$$|\varrho_k| \overset{(2.13)}{\leq} \frac{\frac{4\kappa_f}{\lambda_k^2}}{\frac{\lambda_k\|\nabla_x\Phi(x_k)\|}{2}\|s^*\|^2} \overset{(2.24)}{\leq} \frac{\frac{4\kappa_f}{\lambda_k^2}}{\frac{\lambda_k\|\nabla_x\Phi(x_k)\|}{2}\frac{\kappa_H^2}{4\lambda_k^2}} = \frac{32\kappa_f}{\lambda_k\|\nabla_x\Phi(x_k)\|\kappa_H^2}. \tag{2.34}$$

Thus from (2.31)

$$|\rho_k - 1| \leq \frac{32\kappa_f/\kappa_H^2 + L_\phi}{\lambda_k\|\nabla_x\Phi(x_k)\|} \leq 1 - \eta_1.$$

Hence in every case $\rho_k \geq 1$. Moreover, since $\|\nabla_x\Phi(x_k)\| \geq \frac{\eta_2}{\lambda_k}$ from (2.31), the k-th iteration is successful □

Finally, we state and prove the lemma that guarantees an amount of decrease of the objective function on a true successful iteration.

Lemma 9. *Under Assumptions 1 and 2, suppose that the estimates $\{f_k^0, f_k^s\}$ are ϵ_f-accurate with $\epsilon_f < \frac{\eta_1\eta_2\kappa_H^2}{16}$. If a trial step s_k is accepted then the improvement in f is bounded below by:*

$$f(x_{k+1}) - f(x_k) \leq -\frac{C_2}{\lambda_k^2} \tag{2.35}$$

where $C_2 = \frac{\eta_1\eta_2\kappa_H^2}{8} - 2\epsilon_f > 0$.

Proof. If the iteration is successful, this means that $\|\nabla_x\Phi(x_k)\| \geq \frac{\eta_2}{\lambda_k}$ and $\rho_k \geq \eta_1$. Thus, if the fine step is used,

$$f_k^0 - f_k^s \geq \eta_1(m_k(0) - m_k(s_k)) \overset{(2.26)}{\geq} \eta_1\frac{\|\nabla_x\Phi(x_k)\|}{\lambda_k} \geq \frac{\eta_1\eta_2}{\lambda_k^2}.$$

If the coarse step is used

$$f_k^0 - f_k^s \geq \eta_1(m_k(0) - m_k(s_k)) \overset{(2.26)}{\geq} \frac{\eta_1}{2}\lambda_k\|\nabla_x\Phi(x_k)\|\|s^*\|^2$$
$$\overset{(2.24)}{\geq} \frac{\eta_1\kappa_H^2}{8}\lambda_k\|\nabla_x\Phi(x_k)\|\frac{1}{\lambda_k^2} \geq \frac{\eta_1\eta_2\kappa_H^2}{8}\frac{1}{\lambda_k^2}.$$

Then, since the estimates are ϵ_f-accurate, we have that the improvement in f can be bounded as

$$f(x_k + s_k) - f(x_k) = f(x_k + s_k) - f_k^s + f_k^s - f_k^0 + f_k^0 - f(x_k) \leq -\frac{C_2}{\lambda_k^2},$$

where $C_2 = \frac{\eta_1\eta_2\kappa_H^2}{8} - 2\epsilon_f > 0$. □

To prove convergence of Algorithm 5 we need to assume that the models $\{M_k\}$ and the estimates $\{F_k^0, F_k^s\}$ are sufficiently accurate with sufficiently high probability. We recall that in this case, the models M_k are the models defined for the fine step in (2.7).

Assumption 3. *Given values of α, $\beta \in (0,1)$ and $\epsilon_f > 0$, there exist κ_g and κ_f such that the sequence of models $\{M_k\}$ and estimates $\{F_k^0, F_k^s\}$ generated by Algorithm 1 are, respectively, α-probabilistically (κ_f, κ_g)-fully linear and β-probabilistically ϵ_f-accurate.*

The following theorem states that the regularization parameter λ_k converges to $+\infty$ with probability one. Together with its corollary, it gives conditions on the existence of κ_g and κ_f given α, β and ϵ_f. The proof of the theorem follows similar ideas to the proof of [24, Theorem 4.11].

Theorem 1. *Let Assumptions 1, 2 and 3 be satisfied and assume that in Algorithm 5 the following holds.*

- *The step acceptance parameter η_2 is chosen so that*

$$\eta_2 \geq \max\{K, 24\kappa_f\}. \tag{2.36}$$

- *The accuracy parameter of the estimates satisfies*

$$\epsilon_f \leq \min\left\{\kappa_f, \frac{\eta_1\eta_2\kappa_H^2}{32}\right\}. \tag{2.37}$$

Then α and β can be chosen so that, if Assumption 3 holds for these values, then the sequence of regularization parameters $\{\Lambda_k\}$ generated by Algorithm 5 satisfies

$$\sum_{k=0}^{\infty} \frac{1}{\Lambda_k^2} < \infty \tag{2.38}$$

almost surely.

Proof. The scheme of the proof is the same as that of [24, Theorem 4.11]. We denote with x_k, λ_k, s_k and m_k the realizations of random quantities defined by Algorithm 5 X_k, Λ_k, S_k and M_k, respectively.

Let us fix the constant $\nu \in (0, 1)$ such that

$$\frac{\nu}{1-\nu} > \max\left\{\frac{4}{\gamma^2\zeta C_1}, \frac{16}{\gamma^2\eta_1\eta_2\kappa_H^2}, \frac{1}{\gamma^2 3\kappa_f}\right\}; \tag{2.39}$$

where C_1 is defined like in Lemma 7. We then use ν to define the random function

$$\Psi_k = \nu f(X_k) + (1-\nu)\frac{1}{\Lambda_k^2}; \tag{2.40}$$

and we denote with ψ_k the realization of Ψ_k.

In order to prove (2.38) we want to show that there exist $\sigma > 0$ such that

$$\mathbb{E}\left[\Psi_{k+1} - \Psi_k | \mathcal{F}_{k-1}^{M\cdot F}\right] \leq -\sigma\frac{1}{\Lambda_k^2} < 0, \quad \forall k. \tag{2.41}$$

Since f is bounded from below and $\frac{1}{\Lambda_k} > 0$, then Ψ_k is bounded from below for every k; thus if (2.41) holds for every iteration of Algorithm 5, then by summing (2.41) over $k \in \mathbb{N}$ and taking the expected value on both sides we obtain that (2.38) holds with probability 1.

Then we must prove (2.41), and to do this we need to estimate the decrease $\psi_{k+1} - \psi_k$ for any iteration k and any realization of Algorithm 5. To prove that (2.41) holds for every k, we consider two separate cases depending on the value of $\|\nabla_x f(x_k)\|$. Let us choose a constant ζ such that

$$\zeta \geq \kappa_g + \max\left\{\eta_2, \frac{64\kappa_f/\kappa_H^2 + L_\phi}{1-\eta_1}\right\}. \tag{2.42}$$

The first case considers iterations for which $\|\nabla_x f(x_k)\| \geq \frac{\zeta}{\lambda_k}$, while the second case considers iterations for which $\|\nabla_x f(x_k)\| < \frac{\zeta}{\lambda_k}$. We will prove that (2.41) holds in both cases.

Each iteration k is characterized by the occurrence or non-occurrence of the events I_k and J_k, defined in (2.16) and (2.17). Thus, for both cases ($\|\nabla_x f(x_k)\| \geq \frac{\zeta}{\lambda_k}$ and $\|\nabla_x f(x_k)\| < \frac{\zeta}{\lambda_k}$), we will consider the following four combinations defined with the events I_k and J_k. In the first one, we have the occurrence of both I_k and J_k. In the second one, we have the occurrence of I_k and the non-occurrence of J_k. In the third one, we have the non-occurrence of I_k and the occurrence of J_k. Finally, in the fourth one, we have the non-occurrence of both I_k and J_k.

A further distinction that must be made at each iteration k, regardless of events I_k and J_k, is between successful and unsuccessful iterations. In particular, let us consider a realization of Algorithm 5. In all successful iterations we have $x_{k+1} = x_k + s_k$ and $\lambda_{k+1} = \max\{\lambda_{\min}, \gamma\lambda_k\}$, with $\gamma \in (0,1)$, hence

$$\psi_{k+1} - \psi_k \leq \nu\left(f(x_{k+1}) - f(x_k)\right) + (1-\nu)\left(\frac{1}{\gamma^2} - 1\right)\frac{1}{\lambda_k^2}. \tag{2.43}$$

On the other hand, for all unsuccessful iterations, we have $x_{k+1} = x_k$ and $\lambda_{k+1} = \frac{\lambda_k}{\gamma}$, thus

$$\psi_{k+1} - \psi_k = (1-\nu)\left(\gamma^2 - 1\right)\frac{1}{\lambda_k^2} =: b_1 < 0. \tag{2.44}$$

The main idea is to show that if at iteration k we have the occurrence of at least one of the events I_k and J_k, which means that we have either a good model or good estimates of f (or both), we can choose a parameter $\nu \in (0,1)$ close enough to one such that the decrease in ψ_k is greater in the case of successful iteration than in the case unsuccessful of iteration. If, on the other hand, we have no occurrence of neither I_k nor J_k, so we have a bad model and bad estimates, an increase in ψ_k can occur. This increase is bounded by a value proportional to $\frac{1}{\lambda_k^2}$ when $\|\nabla_x f(x_k)\| < \frac{\zeta}{\lambda_k}$. When $\|\nabla_x f(x_k)\| \geq \frac{\zeta}{\lambda_k}$, though, the increase in ψ_k may be proportional to $\frac{\|\nabla_x f(x_k)\|}{\lambda_k}$. However, we will show that iterations in which we have a good model and good estimates, thus the occurrence of both I_k and J_k, also provide a decrease in ψ_k proportional to the $\frac{\|\nabla_x f(x_k)\|}{\lambda_k}$; thus by choosing values of α and β close enough to 1 we can ensure the decrease in expected value of ψ_k, and thus the validity of (2.41) for every iteration.

Case 1: $\|\nabla_x f(x_k)\| \geq \frac{\zeta}{\lambda_k}$.

(a) We have both I_k and J_k, thus good models and good estimates on iteration k. From the definition of ζ we have

$$\|\nabla_x f(x_k)\| \geq \left(\kappa_g + \max\left\{\eta_2, \frac{64\kappa_f/\kappa_H^2 + L_\phi}{1-\eta_1}\right\}\right)\frac{1}{\lambda_k}.$$

Condition (2.28) of Lemma 7 holds thanks to $\kappa-$fully linearity of $T_k[\Phi]$ together with assumptions (2.36) and (2.37). Thus,

$$f(x_k + s_k) - f(x_k) \leq -C_1 \frac{\|\nabla_x f(x_k)\|}{\lambda_k},$$

with $C_1 := \frac{\kappa_H^2}{32}\max\left\{\frac{K}{K+\kappa_g}, \frac{64\kappa_f}{64\kappa_f+\kappa_g\kappa_H^2}, \frac{2L_\phi}{2L_\phi+\kappa_g}\right\}$.
Moreover, since

$$\|\nabla_x \Phi(x_k)\| \geq \|\nabla_x f(x_k)\| - \frac{\kappa_g}{\lambda_k} \geq (\zeta - \kappa_g)\frac{1}{\lambda_k} \geq \max\left\{\eta_2, \frac{64\kappa_f/\kappa_H^2 + L_\phi}{1-\eta_1}\right\}\frac{1}{\lambda_k},$$

and the estimates f_k^0 and f_k^s are ϵ_f-accurate, with $\epsilon_f \leq \kappa_g$, it holds condition (2.31) of Lemma 8, thus iteration k is successful, meaning that $x_{k+1} = x_k + s_k$ and $\lambda_{k+1} = \max\{\lambda_{\min}, \gamma\lambda_k\}$. Combining (2.29) and (2.43) we obtain

$$\psi_{k+1} - \psi_k \leq -\nu C_1 \frac{\|\nabla_x f(x_k)\|}{\lambda_k} + (1-\nu)\left(\frac{1}{\gamma^2} - 1\right)\frac{1}{\lambda_k^2} =: b_2, \tag{2.45}$$

with C_1 defined as in Lemma 7. Due to the fact that $\|\nabla_x f(x_k)\| \geq \frac{\zeta}{\lambda_k}$ we have

$$b_2 \leq \left[-\nu C_1 \zeta + (1-\nu)\left(\frac{1}{\gamma^2} - 1\right)\right]\frac{1}{\lambda_k^2} < (1-\nu)\left(\gamma^2 - 1\right)\frac{1}{\lambda_k^2} = b_1, \tag{2.46}$$

with $\nu \in (0,1)$ satisfying (2.39).

(b) If I_k occurs and J_k does not, we have a good model m_k but bad estimates $\{f_k^0, f_k^s\}$ at iterate k. Lemma 7 always holds and s_k brings sufficient decrease in f, so if iteration k is successful we get again (2.45) and (2.46).

However, because of inaccurate estimates of f, step s_k could be (mistakenly)

rejected. So we will have an unsuccessful iterate k and consequently (2.44) would hold. Since the condition on ν (2.39) holds, we have that

$$b_2 \leq \left[-\nu C_1 \zeta + (1-\nu)\left(\frac{1}{\gamma^2} - 1\right)\right] \frac{1}{\lambda_k^2} < b_1;$$

Thus (2.44) holds whether the iteration is successful or not.

(c) If, on the other hand, I_k does not occur but J_k does, it means that we have a bad model (not κ-fully linear) and good estimates $\{f_k^0, f_k^s\}$ at iterate k. In this case, the iteration may or may not be successful. If it is unsuccessful it holds (2.44). On the other hand, if it is successful since $\{f_k^0, f_k^s\}$ are ϵ_f-accurate and (2.37) holds, then by Lemma 9 it holds (2.35) thus we have

$$\psi_{k+1} - \psi_k \leq \left[-\nu C_2 + (1-\nu)\left(\frac{1}{\gamma^2} - 1\right)\right] \frac{1}{\lambda_k^2} < b_1. \tag{2.47}$$

As in case b, since we choose ν such that (2.39) is satisfied, we have that (2.44) holds both for successful and unsuccessful iterations.

(d) If neither I_k nor J_k occurs, it means we have a bad model and bad estimates at iteration k. The inaccuracy of estimates $\{f_k^0, f_k^s\}$ can make the algorithm accept an increasing step for f. In such case $\psi_{k+1} - \psi_k$ can be positive. However, the increase of f can be bounded from above by combining the Taylor expansion of $f(x_k)$ at $x_k + s_k$ and the Lipschitz continuity of $\nabla_x f$. From Lemma 1 we have that (2.19) holds. Thus dropping the absolute value and rearranging the terms in (2.19) we have

$$f(x_k + s_k) - f(x_k) \leq \nabla_x f(x_k)^T (s_k) + \frac{L_f}{2}\|s_k\|^2. \tag{2.48}$$

Note that if the step s_k is the *fine step* defined in (2.8) then $\|s_k\| = \frac{1}{\lambda_k}$. Otherwise if s_k is defined as the *coarse step* (2.9) we have that Lemma 4 holds, then using (2.24) and the fact that the prolongation operator P has norm $\|P\| = \kappa_R$ we have that

$$\|s_k\| = \|P s^*\| \leq \|P\|\|s^*\| \leq \frac{4\kappa_R^2}{\lambda_k}.$$

Combining these observations and recalling that $\|\nabla_x f(x_k)\| \geq \frac{\zeta}{\lambda_k}$ we can bound the increase of $f(x_k)$ with

$$f(x_k + s_k) - f(x_k) \leq C_3 \|\nabla_x f(x_k)\| \frac{1}{\lambda_k},$$

with $C_3 := \frac{40 L_f}{\zeta} + 4$. Hence we can bound the difference in ψ_k in the following way

$$\psi_{k+1} - \psi_k \leq \nu C_3 \|\nabla_x f(x_k)\| \frac{1}{\lambda_k} + (1-\nu)\left(\frac{1}{\gamma^2} - 1\right)\frac{1}{\lambda_k^2} =: b_3. \tag{2.49}$$

Now we take the conditioned expected value for $\Psi_{k+1} - \Psi_k$ when $\|\nabla_x f(x_k)\| \geq \frac{\zeta}{\lambda_k}$.

Due to Assumption 3 we know that case (a) occurs with probability (conditioned on the past) $\alpha\beta$ and in that case $\psi_{k+1} - \psi_k \leq b_2 < 0$ with b_2 defined in (2.45).

Case (d) occurs with probability $(1-\alpha)(1-\beta)$ and in that case $\psi_{k+1} - \psi_k \leq b_3$, $b_3 > 0$ defined in (2.49).

Case (b) and (c) occur respectively with probability $\alpha(1-\beta)$ and $(1-\alpha)\beta$ and in these two cases $\psi_{k+1} - \psi_k \leq b_1 < 0$, with b_1 defined in (2.44). Note that $b_1 > b_2$ because we choose ν in order to satisfy (2.39).

Finally we can combine (2.44), (2.45),(2.47) and (2.49) and denote with B_1, B_2 and B_3 the random variables that have respectively b_1, b_2 and b_3 as their realizations in order to get the following bound

$$\begin{aligned}
&\mathbb{E}\left[\Psi_{k+1} - \Psi_k \middle| \mathcal{F}_{k-1}^{M\cdot F}, \left\{\|\nabla_x f(X_k)\| \geq \frac{\zeta}{\Lambda_k}\right\}\right] \\
&\leq \alpha\beta B_2 + \left[\alpha(1-\beta) + (1-\alpha)\beta\right] B_1 + (1-\alpha)(1-\beta) B_3 \\
&= \alpha\beta\left[-\nu C_1 \frac{\|\nabla_x f(X_k)\|}{\Lambda_k} + (1-\nu)\left(\frac{1}{\gamma^2} - 1\right)\frac{1}{\Lambda_k^2}\right] + \\
&+ \left[\alpha(1-\beta) + (1-\alpha)\beta\right](1-\nu)\left(\gamma^2 - 1\right)\frac{1}{\Lambda_k^2} + \\
&+ (1-\alpha)(1-\beta)\left[\nu C_3 \|\nabla_x f(X_k)\| \frac{1}{\Lambda_k} + (1-\nu)\left(\frac{1}{\gamma^2} - 1\right)\frac{1}{\Lambda_k^2}\right].
\end{aligned}$$

Rearranging the terms we get

$$\begin{aligned}
&\mathbb{E}\left[\Psi_{k+1}-\Psi_k\middle|\mathcal{F}_{k-1}^{M\cdot F},\left\{\|\nabla_x f(X_k)\|\geq\frac{\zeta}{\Lambda_k}\right\}\right] \qquad (2.50)\\
&\leq\left[-\nu C_1\alpha\beta+(1-\alpha)(1-\beta)\nu C_3\right]\frac{\|\nabla_x f(X_k)\|}{\Lambda_k}+\\
&+\left[\alpha\beta-\gamma^2\left(\alpha(1-\beta)+(1-\alpha)\beta\right)+(1-\alpha)(1-\beta)\right](1-\nu)\left(\frac{1}{\gamma^2}-1\right)\frac{1}{\Lambda_k^2}\\
&\leq\left[-C_1\alpha\beta+(1-\alpha)(1-\beta)C_3\right]\nu\frac{\|\nabla_x f(X_k)\|}{\Lambda_k}+(1-\nu)\left(\frac{1}{\gamma^2}-1\right)\frac{1}{\Lambda_k^2},
\end{aligned}$$

where the last inequality is true because

$$\alpha\beta-\gamma^2\left(\alpha(1-\beta)+(1-\alpha)\beta\right)+(1-\alpha)(1-\beta)\leq\left[\alpha+(1-\alpha)\right]\left[\beta+(1-\beta)\right]=1.$$

Let us choose α and β in $(0,1)$ such that

$$\frac{\left(\alpha\beta-\frac{1}{2}\right)}{(1-\alpha)(1-\beta)}\geq\frac{C_3}{C_1},$$

which implies

$$\left[C_1\alpha\beta-(1-\alpha)(1-\beta)C_3\right]\geq\frac{C_1}{2}\geq 2\frac{(1-\nu)\left(\frac{1}{\gamma^2}-1\right)}{\nu\zeta},$$

where the last inequality holds because of (2.39).

Recalling that $\|\nabla_x f(x_k)\|<\frac{\zeta}{\lambda_k}$ we can continue bounding (2.50), thus

$$\begin{aligned}
&\left[-C_1\alpha\beta+(1-\alpha)(1-\beta)C_3\right]\nu\frac{\|\nabla_x f(X_k)\|}{\Lambda_k}+(1-\nu)\left(\frac{1}{\gamma^2}-1\right)\frac{1}{\Lambda_k^2}\\
&\leq\left[-C_1\alpha\beta+(1-\alpha)(1-\beta)C_3\right]\nu\frac{\|\nabla_x f(X_k)\|}{\Lambda_k}\leq-\frac{1}{4}C_1\nu\frac{\|\nabla_x f(X_k)\|}{\Lambda_k}.
\end{aligned}$$

In this way we have

$$\mathbb{E}\left[\Psi_{k+1}-\Psi_k\middle|\mathcal{F}_{k-1}^{M\cdot F},\left\{\|\nabla_x f(X_k)\|\geq\frac{\zeta}{\Lambda_k}\right\}\right]\leq-\frac{1}{4}C_1\nu\frac{\|\nabla_x f(X_k)\|}{\Lambda_k} \qquad (2.51)$$

and

$$\mathbb{E}\left[\Psi_{k+1}-\Psi_k\middle|\mathcal{F}_{k-1}^{M\cdot F},\left\{\|\nabla_x f(X_k)\|\geq\frac{\zeta}{\Lambda_k}\right\}\right]\leq-\frac{1}{2}(1-\nu)\left(\frac{1}{\gamma^2}-1\right)\frac{1}{\Lambda_k^2}. \quad (2.52)$$

For the proof of this theorem (and also for Theorem 2) bound (2.52) is enough. Bound (2.51) is used to prove Theorem 3.

Case 2: $\|\nabla_x f(x_k)\|<\frac{\zeta}{\lambda_k}$.

First, let us notice that if $\|\nabla_x\Phi(x_k)\|<\frac{\eta_2}{\lambda_k}$ iteration k is unsuccessful and (2.44) holds. Let us then assume that $\|\nabla_x\Phi(x_k)\|\geq\frac{\eta_2}{\lambda_k}$ and we are going to consider the four combinations of events as we did in the previous case.

(a) We have the occurrence of both I_k and J_k, thus good model and good estimates on iteration k. If k is successful the $\kappa-$fully linearity of the model ensures a decrease for f. We can use the same arguments of Case 1c to conclude that (2.44) holds both in case of a successful iteration or case of an unsuccessful one.

(b) We have the occurrence of I_k and no occurrence of J_k which means good model and bad estimates at iteration k. If it is unsuccessful, (2.44) holds. Otherwise from Lemma 5 we have that

$$m_k(s_k)-m_k(0)\leq\begin{cases}-\frac{\|\nabla_x\Phi(x_k)\|}{\lambda_k} & \text{if } \textit{fine step}\\ -\frac{\lambda_k\|\nabla_x\Phi(x_k)\|}{2}\|s^*\|^2 & \text{if } \textit{coarse step}.\end{cases}$$

Let us consider the case of the coarse step. Using also Lemma 4 and the fact that $\|\nabla_x f(x_k)\|<\frac{\zeta}{\lambda_k}$ we have

$$m_k(0)-m_k(s_k)\geq\frac{\kappa_H^2\eta_2}{8\lambda_k^2}.$$

Since I_k occurs, the model m_k is $\kappa-$fully linear, considering (2.36), we can

write

$$\begin{aligned} & f(x_k) - f(x_k + s_k) \\ & = f(x_k) - m_k(0) + m_k(0) - m_k(s_k) + m_k(s_k) - f(x_k + s_k) \\ & \geq \left(\frac{\kappa_H^2 \eta_2}{8} - 2\kappa_f \right) \frac{1}{\lambda_k^2} \geq -\frac{3\kappa_f}{\lambda_k^2}. \end{aligned}$$

As a consequence, if the $k-$th iterate is successful, we have

$$\psi_{k+1} - \psi_k \leq \left[-3\nu\kappa_f + (1-\nu) \left(\frac{1}{\gamma^2} \right) \right] \frac{1}{\lambda_k^2}. \tag{2.53}$$

Since we choose $\nu \in (0,1)$ in order to satisfy (2.39), we have that the right-hand side of (2.53) is strictly smaller than b_1 defined in (2.44). Thus (2.44) holds in any case, successful or not. If we use the fine step, i.e.

$$m_k(0) - m_k(s_k) \geq \frac{\eta_2}{\lambda_k^2}$$

we can analogously get the same result.

(c) The event I_k does not occur while J_k does. Here we have a bad model and good estimates at iteration k. We can proceed in the very same way as Case 1c to conclude again that (2.44) holds in any case.

(d) None of the events I_k and J_k occur, thus we have a bad model and bad estimates. Similarly to Case 1d we can bound the increase in f using Lipschitz continuity of $\nabla_x f$, thus

$$f(x_k + s_k) - f(x_k) \leq \frac{C_3 \zeta}{\lambda_k^2}.$$

We use this bound for the difference of ψ_k:

$$\psi_{k+1} - \psi_k \leq \left[\nu C_3 \zeta + (1-\nu) \left(\frac{1}{\gamma^2} - 1 \right) \right] \frac{1}{\lambda_k^2}. \tag{2.54}$$

Now we can bound the expected value of $\psi_{k+1} - \psi_k$ as we did in Case 1; but in Case 2 we only use (2.54), which occurs with probability $(1-\alpha)(1-\beta)$, and (2.44),

which occurs otherwise. Then

$$\begin{aligned}
&\mathbb{E}\left[\Psi_{k+1}-\Psi_k\middle|\mathcal{F}_{k-1}^{M\cdot F},\left\{\|\nabla_x f(X_k)\|<\frac{\zeta}{\Lambda_k}\right\}\right]\leq\\
&\leq\left[\alpha\beta+\alpha(1-\beta)+(1-\alpha)\beta\right](1-\nu)(\gamma^2-1)\frac{1}{\Lambda_k^2}+\\
&+(1-\alpha)(1-\beta)\left[\nu C_3\zeta+(1-\nu)\left(\frac{1}{\gamma^2}-1\right)\right]\frac{1}{\Lambda_k^2}\leq\\
&\leq(1-\nu)(\gamma^2-1)\frac{1}{\Lambda_k^2}+(1-\alpha)(1-\beta)\left[\nu C_3\zeta+(1-\nu)\left(\frac{1}{\gamma^2}-\gamma^2\right)\right]\frac{1}{\Lambda_k^2}.
\end{aligned}$$

By choosing $\alpha\in(0,1)$ and $\beta\in(0,1)$ such that

$$(1-\alpha)(1-\beta)\leq\frac{\frac{1}{\gamma^2}-1}{\frac{1}{\gamma^4}-1+\frac{2C_3\zeta\nu}{\gamma^2(1-\nu)}},$$

we have

$$\mathbb{E}\left[\Psi_{k+1}-\Psi_k\middle|\mathcal{F}_{k-1}^{M\cdot F},\left\{\|\nabla_x f(X_k)\|<\frac{\zeta}{\Lambda_k}\right\}\right]\leq-\frac{1}{2}(1-\nu)\left(\frac{\gamma^2}{\gamma^2-1}\right)\frac{1}{\Lambda_k^2}. \quad (2.55)$$

Finally, combining (2.52) and (2.55), and observing that $1-\gamma^2<\frac{1}{\gamma^2}-1$ we have

$$\mathbb{E}\left[\Psi_{k+1}-\Psi_k\middle|\mathcal{F}_{k-1}^{M\cdot F}\right]\leq-\frac{1}{2}(1-\nu)\left(1-\gamma^2\right)\frac{1}{\Lambda_k^2}.$$

This means that (2.41) holds with $\sigma=\frac{1}{2}(1-\nu)\left(1-\gamma^2\right)>0$. □

The choice of α and β is specified in the following corollary.

Corollary 1. *Let all assumptions of Theorem 1 hold. The statement of Theorem 1 holds if α and β are chosen to satisfy the following conditions:*

$$\frac{\alpha\beta-\frac{1}{2}}{(1-\alpha)(1-\beta)}\geq\frac{\frac{40L_f}{\zeta}+4}{C_1}$$

and

$$(1-\alpha)(1-\beta)\leq\frac{\frac{1}{\gamma^2}-1}{\frac{1}{\gamma^4}-1+\frac{1}{\gamma^2}(40L_f+4\zeta)\max\left\{\frac{4}{\zeta C_1},\frac{16}{\eta_1\eta_2\kappa_H^2},\frac{1}{3\kappa_f}\right\}},$$

with $C_1 = \frac{\kappa_H^2}{32} \max\left\{\frac{K}{K+\kappa_g}, \frac{64\kappa_f}{64\kappa_f+\kappa_g\kappa_H^2}, \frac{2L_\phi}{2L_\phi+\kappa_g}\right\}$ *and* $\zeta = \kappa_g + \eta_2$.

In practice, the probabilities α and β depend on the characteristics of the optimization problem. For a more specific discussion we refer to [24].

The following results can be derived as in [24, Theorem 4.16], [24, Lemma 4.17] and [24, Theorem 4.18], thus for the proof we refer to [24].

Theorem 2. *Let the assumptions of Theorem 1 and Corollary 1 hold. Then the sequence of random iterates generated by Algorithm 5,* $\{X_k\}$*, almost surely satisfies*

$$\liminf_{k\to\infty} \|\nabla_x f(X_k)\| = 0.$$

Lemma 10. *Let the assumptions of Theorem 2 hold. Let* $\{X_k\}$ *and* $\{\Lambda_k\}$ *be the sequences of random iterates and random regularization parameters generated by Algorithm 5. Fix* $\epsilon > 0$ *and define the sequence* $\{K_\epsilon\}$ *consisting of the natural numbers* k *for which* $\|\nabla_x f(X_k)\| > \epsilon$*. Then almost surely*

$$\sum_{k\in\{K_\epsilon\}} \frac{1}{\Lambda_k} < \infty.$$

Theorem 3. *Let the assumptions of Theorem 2 hold. Let* $\{X_k\}$ *be the sequence of random iterates generated by Algorithm 5. Then, almost surely,*

$$\lim_{k\to\infty} \|\nabla_x f(X_k)\| = 0.$$

2.3 MU$^\ell$STREG for finite-sum minimization

In this section, we describe how to adapt Algorithm 5 to the solution of finite-sum minimization problems of the form (2.2) using a multilevel setting with ℓ levels. In particular, in Subsection 2.3.1 we show in detail the algorithmic properties of the MU$^\ell$STREG version to address problem (2.2), while in Subsection 2.3.2 we explicitly show the similarity between Stochastic Variance Reduced Gradient (SVRG) and MU$^\ell$STREG specialized on finite sums.

2.3.1 Algorithmic details

Considering problem (2.2), we assume that $N \gg n$ and we consider hierarchies built just in the sample space, thus $\ell = l$ (see Subsection 2.1.1). We first assume that the computation of the objective function is affordable and we postpone to Subsection 2.4.4 a discussion on the case when N is too large to evaluate the full sum.

Recalling that the objective function in (2.2) is the average of the set of functions $\{f_i\}_{i=1}^N$, we can easily define a hierarchy of approximations by subsampling. In particular, given the number of levels $\ell_{\max} \geq 2$, for every $\ell \in \{1, ..., \ell_{\max}\}$ we define the subsampled function as:

$$f^{S^\ell}(x) = \frac{1}{|\mathcal{S}^\ell|} \sum_{i \in \mathcal{S}^\ell} f_i(x);$$

where $\mathcal{S}^\ell \subseteq \{1, .., N\}$ is a subsample set such that $\emptyset \neq \mathcal{S}^1 \subset ... \subset \mathcal{S}^\ell \subset ... \subset \mathcal{S}^{\ell_{\max}-1} \subset \mathcal{S}^{\ell_{\max}} = \{1, ..., N\}$. In this particular case, R and P are both the identity and all the iterates belong to $\mathbb{R}^n$. We thus drop here the indexes ℓ from the iterates and the steps. We use the functions $\left\{f^{S^\ell}\right\}_{\ell=1}^{\ell_{\max}}$ to define the regularized models $\left\{m^{R,\ell}\right\}_{\ell=1}^{\ell_{\max}}$ that are minimized at each level in a recursive way. Note that each model $m^{R,\ell}$ should be indexed by the index of the iterate at level $\ell + 1$ for which it was defined. We omit this index here to avoid confusion with the index k of the iterate considered to define, given $m^{R,\ell}$, its lower level model. In the notations of Section 2.1, the f^{S^ℓ} corresponds to the ϕ^ℓ.

In particular, at level $1 < \ell \leq \ell_{\max}$, given the objective function of that level $m^{R,\ell}$ and an iterate x_k we define the objective function $m_k^{R,\ell-1}$ at x_k for the lower level $\ell - 1$ as

$$m_k^{R,\ell-1}(s) = \left[m^{R,\ell}\right]^{\mathcal{S}^{\ell-1}}(x_k + s) + (v_k^{\ell-1})^T s + \mathrm{reg}_k^{\ell-1}(s) \tag{2.56}$$

where $\left[m^{R,\ell}\right]^{\mathcal{S}^{\ell-1}}(x_k)$ denotes the subsampled version of $m^{R,\ell}$ evaluated at x_k,

$$v_k^{\ell-1} = \nabla_x m^{R,\ell}(x_k) - \nabla_x \left[m^{R,\ell}\right]^{\mathcal{S}^{\ell-1}}(x_k), \tag{2.57}$$

and $\mathrm{reg}_k^\ell(s) = \frac{1}{2}\lambda_k^\ell \|\nabla_x m^{R,\ell}(x_k)\| \|s\|^2$ with $\lambda_k^\ell > 0$ if $\ell < \ell_{\max}$, and zero otherwise.

At the finest level $\left[m^{R,\ell_{\max}}\right]^{\mathcal{S}^{\ell_{\max}-1}} = \left[f^{\mathcal{S}^{\ell_{\max}}}\right]^{\mathcal{S}^{\ell_{\max}-1}}$ is simply $f^{\mathcal{S}^{\ell_{\max}-1}}$. However, when $\ell < \ell_{\max}$, $m^{R,\ell}$ incorporates also the regularization and the vector v_k^ℓ. Given that these quantities are not defined on a sample set, the subsampled version of $m^{R,\ell}$ differs from $m^{R,\ell}$ just for the term $f^{\mathcal{S}^{\ell-1}}$ that is subsampled on $\mathcal{S}^{\ell-2}$, while the correction and

the regularization vectors remain unchanged. Notice that each time we go down a level we accumulate in $m^{R,\ell}$ a regularization term and a vector v^ℓ.

We report in Algorithm 6 the complete MU$^\ell$STREG algorithm for problem (2.2) and we now discuss its main steps.

Algorithm 6 is recursive and a generic level $\ell \geq 1$ of the hierarchy is described. The main hyperparameters of the algorithm are the number of levels in the hierarchy $\ell_{\max}$ and the cardinality N^ℓ of the subsample sets $\mathcal{S}^\ell$ for every level of the hierarchy. At the very first call MU$^\ell$STREG starts from the top level $\ell_{\max}$ and the objective function is set as $f^{\ell_{\max}}$.

At each iteration k the algorithm either calls itself recursively or performs a fine step at level ℓ, except $\ell = 1$. For $\ell = 1$ the bottom of the hierarchy is reached and no more recursions are allowed. An approximate minimizer of $m_k^{R,1}$ is sought that satisfies (2.10) by the Adaptive-Regularization algorithm with a first-order model (AR1) described e.g. in [21, Sec. 2.4.1] with a regularization parameter weighted by the norm of the gradient of $m_k^{R,1}$ at the current approximation. Notice however that the theoretical results would still hold if the minimization algorithm was replaced by another one, provided that the stopping criterion can be satisfied.

When $\ell > 1$ the algorithm can be called recursively and, if we choose to use the lower-level model, the surrogate minimization problem of the new approximation $m_k^{R,\ell-1}$ is built by sampling a subset of indices $\mathcal{S}^{\ell-1} \subset \mathcal{S}^\ell$ by drawing randomly $N^{\ell-1}$ indices from $\mathcal{S}^\ell$ (Step 4) and MU$^\ell$STREG is recursively called at Step 12 providing the search direction s_k^ℓ.

Steps 15-23 are dedicated to the standard step acceptance rule and regularization parameter update based on the ratio ρ_k^ℓ defined at Step 14. We remark that the condition $\|\nabla m^{R,\ell}(x_k)\| \geq \eta_2/\lambda_k^\ell$ is checked at the beginning of each iteration to save useless computations in case it fails.

The stopping criterion checks if the norm of the gradient is below some tolerance that depends (implicitly) on the level ℓ and on the iteration k. Indeed, when $\ell = \ell_{\max}$ the tolerance is a positive scalar ϵ chosen by the user and we get a classical stopping criterion

$$\|\nabla_x f^{\mathcal{S}^{\ell_{\max}}}(x_k)\| \leq \epsilon. \tag{2.58}$$

Else if $\ell < \ell_{\max}$, we use the stopping condition (2.4). A safeguard is added imposing a

Algorithm 6 MU$^\ell$STREG for finite-sum - MU$^\ell$STREG$\left(x_0, \left\{\left(f^\ell, N^\ell, \epsilon^\ell\right)\right\}_{\ell=1}^{\ell_{\max}}\right)$

Input: $x_0 \in \mathbb{R}^n$, $\{f^\ell\}_{\ell=1}^{\ell_{\max}}$, $f^\ell : \mathbb{R}^n \to \mathbb{R}$ defined on N^ℓ samples with $N^{\ell-1} < N^\ell$, tolerance $\epsilon^\ell > 0$.
Given $0 < \eta_1 \leq \eta_3 < 1$, $\eta_2 > 0$, $0 < \gamma_2 \leq \gamma_1 < 1 < \gamma_3$, $\lambda_{\min} > 0$.

$k = 0$
while the stop criterion for level ℓ is not satisfied **do**
Hierarchy definition
  if $\ell > 1$ **then**
    Build $\mathcal{S}^{\ell-1} \subset \mathcal{S}^\ell$ drawing $N^{\ell-1}$ indices randomly.
  else
    Go to Step 9.
  end if
Model choice
  Choose to go to Step 9 or to Step 10.
Regularized Taylor step
  Define $m_k^\ell(s) = T_k^\ell(s)$ the first-order Taylor series of f^ℓ in x_k. Set $s_k^\ell = -\frac{\nabla f^\ell(x_k)}{\lambda_k^\ell \|\nabla f^\ell(x_k)\|}$. Go to Step 14.
Sub-sampled model
  Compute the correction vector $v_k^{\ell-1}$ as in (2.57) to define the lower level model $\varphi_k^{\ell-1}(s)$ and its regularization $m_k^{R,\ell-1}(s)$ as

$$\varphi_k^{\ell-1}(s) = \left[f^{\ell-1}\right]^{\mathcal{S}^{\ell-1}}(x_k + s) + \left(\nabla_s f^\ell(x_k) - \nabla_x \left[f^\ell\right]^{\mathcal{S}^{\ell-1}}(x_k)\right)^T s;$$

$$m_k^{R,\ell-1}(s) = \varphi_k^{\ell-1}(s) + \frac{1}{2}\lambda_k^\ell \|\nabla_x f^\ell(x_k)\| \|s\|^2.$$

  Recursive call
  Call MU$^\ell$STREG$\left(0, \left\{\left(m_k^{R,j}, N^j, \epsilon^j\right)\right\}_{j=1}^{\ell-1}\right)$ to find approximate solution s^* of the problem

$$\min_{s \in \mathbb{R}^n} m_k^{R,\ell-1}(s),$$

  such that condition (2.4) is satisfied.
  Set $s_k^\ell = s^*$ and $m_k^\ell(s) = \varphi_k^{\ell-1}(s)$.
Step acceptance of trial point
  Compute $\rho_k^\ell := \frac{f^\ell(x_k) - f^\ell(x_k + s_k^\ell)}{m_k^\ell(0) - m_k^\ell(s_k^\ell)}$.
  if $\rho_k \geq \eta_1$ and $\|\nabla_x f^\ell(x_k)\| \geq \eta_2/\lambda_k^\ell$ **then**
    $x_{k+1} = x_k + s_k^\ell$
  else
    $x_{k+1} = s_k^\ell$
  end if
Regularization parameter update
  if $\rho_k^\ell \geq \eta_1$ and $\|\nabla_x f^\ell(x_k)\| \geq \eta_2/\lambda_k^\ell$ **then**

$$\lambda_{k+1}^\ell = \begin{cases} \max\{\lambda_{\min}, \gamma_2 \lambda_k^\ell\}, & \text{if } \rho_k^\ell \geq \eta_3, \\ \max\{\lambda_{\min}, \gamma_1 \lambda_k^\ell\}, & \text{if } \rho_k^\ell < \eta_3 \end{cases}$$

  else
    $\lambda_{k+1}^\ell = \gamma_3 \lambda_k^\ell$.
  end if
  $k = k + 1$
end while

maximum number of iterations.

Notice that the choice of the alternating scheme between coarse and fine steps is left to the user.

2.3.2 Similarity with SVRG

As we already mentioned, MU$^\ell$STREG for problem (2.2) can be interpreted as a variance-reduced method similar to SVRG. An analogous interpretation was given for the line-search based MLVR (MultiLevel Variance Reduction) method presented in [17]. We now show the details of the similarity between the MU$^\ell$STREG and SVRG by writing explicitly the generic update of the iterate x_{k+1} from x_k. We start from SVRG giving also a little explanation which will be useful in the following.

Algorithm 7 SVRG(x_0,f,α,b,m)

Given $x_0 \in \mathbb{R}^n$, the learning rate $\alpha > 0$, the mini-batch size b and an integer m.

1: **for** $k = 0, 1, ...$ **do**
2: Compute the full gradient $\nabla_x f(x_k)$.
3: Set $\tilde{x}_0 = x_k$.
4: **for** $t = 0, ..., m$ **do**
5: Draw randomly the mini-batch $\mathcal{I}_t \subset \{1, .., N\}$, such that $|\mathcal{I}_t| = b$.
6: Define $p_t^{(k)}$, i.e. the $t-$th direction built in x_k as:

$$p_t^{(k)} = \frac{1}{b}\sum_{i\in\mathcal{I}_t} \nabla_x f_i(\tilde{x}_t) + \left(\nabla_x f(x_k) - \frac{1}{b}\sum_{i\in\mathcal{I}_t} \nabla_x f_i(x_k)\right). \tag{2.59}$$

7: Set $\tilde{x}_{t+1} = \tilde{x}_t + \alpha p_t^{(k)}$.
8: **end for**
9: Set $x_{k+1} = \tilde{x}_{m+1}$
10: **end for**

The SVRG method is introduced in [44], and it is described in Algorithm 7. The hyperparameters that characterize SVRG are the number m, the learning rate (or steplength) α, and the mini-batch size b. The parameter m is the number of internal iterations in which the algorithm performs random updates using the gradient aggregation (Steps 4-8), α (used at Step 7) defines the size of the step along the search direction for every update and b is used in Step 5 to select randomly the subsample set $\mathcal{I}_t$.

Given the parameters m, α, b and an iterate x_k we want the explicit expression of

x_{k+1}. Thus, from Step 9 we have $x_{k+1} = \tilde{x}_{m+1}$ and considering the loop in Steps 4–8 we can write

$$\begin{aligned} x_{k+1} &= x_k - \alpha \sum_{t=0}^{m} p_t^{(k)} \overset{(2.59)}{=} \\ &= x_k - \alpha \sum_{t=0}^{m} \left[\frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla_x f_i(\tilde{x}_t) + \left(\nabla_x f(x_k) - \frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla_x f_i(x_k) \right) \right]. \end{aligned}$$

Observing that $\tilde{x}_0 = x_k$ from Step 3, we can write the SVRG update as:

$$x_{k+1} = x_k - \alpha \sum_{t=1}^{m} \left[\frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla_x f_i(\tilde{x}_t) - \frac{1}{b} \sum_{i \in \mathcal{I}_t} \nabla_x f_i(x_k) \right] - \alpha(m+1) \nabla_x f(x_k). \tag{2.60}$$

Now we want to write explicitly the update of x_{k+1} from x_k for MU$^\ell$STREG. Since MU$^\ell$STREG is recursive and does not have a fixed steplength like SVRG the description of the update for MU$^\ell$STREG with a generic number of levels $\ell_{\max}$ becomes very cumbersome; therefore we limit ourselves to the case with two levels MU^{2}STREG. Hence, let us now follow one iteration of MU^{2}STREG for finite-sum, assuming to use only the coarse step. Thus, $\ell_{max} = 2$ and we have fixed the cardinalities of the subsample sets $|\mathcal{S}^1| = N^1 < N^{\ell_{\max}} = N = |\mathcal{S}^2|$, with a certain iterate x_k. Since we start from level $\ell_{\max} = 2$, following Algorithm 6, in particular according to Steps 4 and 10 we draw randomly N^1 samples to define $\mathcal{S}^1 \subset \{1, ..., N\}$ and we define the functions $\varphi_k^1(s)$ and $m_k^{R,1}(s)$ as

$$\varphi_k^1(s) = \frac{1}{N^1} \sum_{i \in \mathcal{S}^1} f_i(x_k + s) + \left[\nabla_x f(x_k) - \frac{1}{N^1} \sum_{i \in \mathcal{S}^1} \nabla_x f_i(x_k) \right]^T s; \tag{2.61}$$

$$m_k^{R,1}(s) = \varphi_k^1(s) + \frac{1}{2} \lambda_k^{(2)} \|\nabla_x f(x_k)\| \|s\|^2. \tag{2.62}$$

In particular, the gradient $\nabla_s m_k^{R,1}(s)$ is

$$\nabla_s m_k^{R,1}(s) = \frac{1}{N^1} \sum_{i \in \mathcal{S}^1} \nabla_x f_i(x_k + s) + \nabla_x f(x_k) - \frac{1}{N^1} \sum_{i \in \mathcal{S}^1} \nabla_x f_i(x_k) + \lambda_k^{(2)} \|\nabla_x f(x_k)\| s; \tag{2.63}$$

Once again, notice that $\nabla_s m_0^{R,1}(0) = \nabla_x f(x_k)$.

Following the recursive step (Step 12) in Algorithm 6, we call MU^{2}STREG on $m_k^{R,1}(s)$ with the initial guess $s_0 = 0 \in \mathbb{R}^n$ and a parameter $\lambda_0^{(1)}$ in order to find the minimizer s^*. Since we are at the bottom level of the hierarchy we use the AR1 method to minimize $m_k^{R,1}(s)$ (as in Step 9), leading to define $s_1 = s_0 + p_0$ with update p_0 defined as

$$p_0 = -\frac{\nabla_s m_0^{R,1}(s_0)}{\lambda_0^{(1)} \|\nabla_s m_0^{R,1}(s_0)\|} = -\frac{\nabla_x f(x_k)}{\lambda_0^{(1)} \|\nabla_x f(x_k)\|}; \tag{2.64}$$

where the last equation derives from the fact that $s_0 = 0$. Let us now assume that we accept the update p_0 according to the acceptance step (Step 14), thus we have $s_1 = s_0 + p_0 = p_0$ and $\lambda_1^{(1)}$. Now we define $s_2 = s_1 + p_1$, again with p_1 defined as

$$\begin{aligned} p_1 &= -\frac{\nabla_s m_0^{R,1}(s_1)}{\lambda_1^{(1)} \|\nabla_s m_0^{R,1}(s_1)\|} = \\ &= -\frac{1}{\lambda_1^{(1)} \|\nabla_s m_0^{R,1}(s_1)\|} \left(\frac{1}{N^1} \sum_{i \in \mathcal{S}^1} \nabla_x f_i(x_k + s_1) + \nabla_x f(x_k) - \right. \\ &\left. - \frac{1}{N^1} \sum_{i \in \mathcal{S}^1} \nabla_x f_i(x_k) + \lambda_k^{(2)} \|\nabla_x f(x_k)\| s_1 \right). \end{aligned}$$

Now, knowing that $s_1 = p_0$ and substituting (2.64) in the last term inside the round brackets and gathering all the coefficients of $\nabla_x f(x_k)$, we obtain

$$\begin{aligned} p_1 = -\frac{1}{\lambda_1^{(1)} \|\nabla_s m_0^{R,1}(s_1)\|} \left[\frac{1}{N^1} \sum_{i \in \mathcal{S}^1} \nabla_x f_i(x_k + s_1) - \right. \\ \left. - \frac{1}{N^1} \sum_{i \in \mathcal{S}^1} \nabla_x f_i(x_k) + \left(1 + \frac{\lambda_k^{(2)}}{\lambda_0^{(1)}} \right) \nabla_x f(x_k) \right]. \end{aligned} \tag{2.65}$$

Assuming to accept also the update p_1, we have now defined s_2 and for the sake of simplicity let us assume that s_2 satisfies the stopping condition (2.10). So s_2 is the minimizer s^* we look for. Recalling that for problem (2.2) there is no hierarchy on the variable space and assuming to accept the update s^* for the iterate x_k, we can set

$$x_{k+1} = x_k + s^* = x_k + s_2 = x_k + s_1 + p_1 = x_k + p_0 + p_1; \tag{2.66}$$

hence, using the expressions (2.64) and (2.65) and gathering again all the coefficients of $\nabla_x f(x_k)$, we have the expression

$$x_{k+1} = x_k - \frac{1}{\lambda_1^{(1)} \|\nabla_s m_0^{R,1}(s_1)\|} \left[\frac{1}{N^1} \sum_{i \in \mathcal{S}^1} \nabla_x f_i(x_k + s_1) - \frac{1}{N^1} \sum_{i \in \mathcal{S}^1} \nabla_x f_i(x_k) \right] - \\ - \left(\frac{\lambda_0^{(1)} + \lambda_k^{(2)}}{\lambda_0^{(1)} \lambda_1^{(1)} \|\nabla_s m_0^{R,1}(s_1)\|} + \frac{1}{\lambda_0^{(1)} \|\nabla_x f(x_k)\|} \right) \nabla_x f(x_k). \tag{2.67}$$

If we compare (2.60) and (2.67), the similarity between SVRG and MU$^\ell$STREG appears quite clear. In both cases in the square brackets, we have a difference between the subsampled gradient computed in an intermediate subiterate, which in (2.60) is $\tilde{x}_t$ while in (2.67) we have $x_k + s_1$, and the subsampled gradient computed in the current iterate x_k. In (2.60) we have a summation on the internal iterates $t = 1, ..., m$ that we do not have in case (2.67), and this is because of our choice to avoid unnecessarily heavy notation, but in the extreme case $m = 1$ the similarity is even more evident. Moreover, in both cases, we have a full gradient contribution. Clearly, the difference in the choice of step length of the two methods (fixed for SVRG, adaptive for MU$^\ell$STREG) is reflected in the scalars that appear in the two expressions.

2.4 Numerical experiments

In this section, we illustrate the performance of MU$^\ell$STREG for the computation of an approximate first-order critical point of the finite-sum minimization problem (2.2).

This section is organized as follows. First, we introduce some implementation details and the problem test set in Subsection 2.4.1, then we study in Subsection 2.4.2 the tuning of the hyperparameters of Algorithm 6, in particular, the number of levels and the sample set cardinalities and we compare the performance of all the variants to the reference one-level method. The method that shows the best performance is a three-level method that we refer to as MU^{3}STREG. In Section 2.3 we compare it to a mini-batch version of SVRG. Finally, we investigate the behavior of MU^{3}STREG when the size of sample size $N^{\max}$ at the finest level is smaller than the full size N.

2.4.1 Implementation issues and test problem set

Algorithm 6 has been implemented in MATLAB R2024a using HPE ProLiant DL560 Gen10 with 4 Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz with 512 Gb RAM[2]. The algorithmic parameters are chosen as follows:

$$\eta_1 = 0.5,\ \eta_2 = 10^{-3},\ \eta_3 = 0.75,\ \gamma_1 = 0.5,\ \gamma_2 = 0.3,\ \gamma_3 = 2,\ \lambda_{\min} = 10^{-4}.$$

The algorithm terminates when condition (2.58) holds with $\epsilon = 10^{-3}$ or 10^4 iterations are performed. Moreover we set $\theta = 10^{-3}$ in (2.10). Also for every recursive call at level $\ell \leq \ell_{\max} - 1$, we set a maximum number of iterations $\text{maxit}_\ell = 5$. Finally, for $\ell > 1$ we set $\lambda_0^\ell = 10^{-4}$ and $\lambda_0^1 = 10^{-3}$. In every test, all the runs are repeated 5 times for different random initial guesses x_0.

Notice that Algorithm 6 is quite generic and allows for different multilevel schemes. Here, we used a recursion scheme that encompasses a fine step after each recursive call, as depicted in Figure 2.2 where the horizontal arrows represent the fine steps. Notice that this involves computing the full gradient of the function f^ℓ (for $\ell > 1$) with increasing computational cost depending on the level ℓ, thus the uses of such step for high ℓ should be limited.

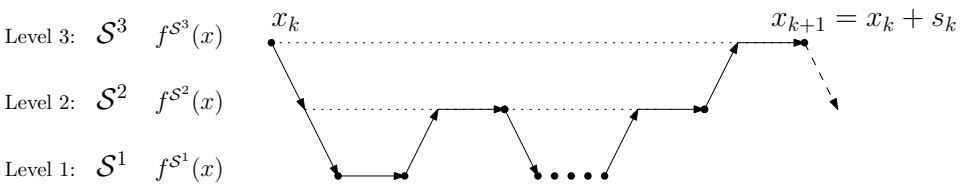

Figure 2.2: Iteration scheme used in our implementation of MU$^\ell$STREG.

SVRG has been implemented in MATLAB too and we chose different configurations for the parameters m, b and α as proposed in [66] for nonconvex problems. We thus set the mini-batch size $b = 10$ and $b = 20$ and $m = N/b$ while we set $\alpha \in \{10^{-2}, 10^{-1}, 0.5\}$. We used the same stopping criterion for SVRG as for MU$^\ell$STREG [3].

In order to compare the efficiency of the various methods, we considered the number of

[2]We kindly acknowledge the Department of Mathematics of the University of Bologna for making the department's HPC resources available for this work.

[3]The stopping criterion is thus checked for SVRG only every m iterations and is checked for MU$^\ell$STREG only at fine level

weighted gradient evaluations performed during the execution: a full-gradient evaluation is counted as 1, while the sub-sampled one as $\frac{N^\ell}{N}$, where $N^\ell = |\mathcal{S}^\ell|$ is the size of the sub-sample set. In the same way, the weighted objective function evaluations are taken into account. Taking into account the size of the gradients n, the same system of gradient weights is used for the objective function and its sub-models, just multiplied by $\frac{1}{n}$. From now on we will consider the sum of the weighted evaluations of gradients and functions as a measure of the efficiency of the method and we will refer to this sum as computational effort or more simply weighted number of evaluations, which will be denoted by #f/g.

Besides the efficiency of the methods, we also take into account the quality of the solutions found. In particular, we focus on the classification accuracy (in percentage) on the testing set that will be denoted by %tA.

Finally, we will also use performance profiles in the forthcoming Figures 2.4 and 2.6. We remind the reader that a performance profile graph $p_A(\tau)$ of an algorithm A at point τ shows the fraction of the test set for which the algorithm is able to solve within a factor of τ of the best algorithm for the given measure [29]. The measure used in Figures 2.4 and 2.6 is the total number of #f/g to get the maximum %tA.

In our experiments, we consider two binary classification problems with different losses. The first problem considers the logistic classification loss with ℓ_2 regularization:

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2N}\sum_{i=1}^{N} \log(1+\exp(-y_i x^T z_i)) + \frac{1}{2N}\|x\|^2, \tag{Pb-LOG}$$

where for every $i = 1,\dots,N$ the pairs $(z_i, y_i) \in \mathbb{R}^n \times \{-1, 1\}$ contain the features vector and the corresponding label. Note that (Pb-LOG) is a strongly convex problem.

The second one is a nonlinear least squares problem with sigmoid loss:

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2N}\sum_{i=1}^{N} \left(y_i - \frac{1}{1+\exp(-y_i x^T z_i)}\right)^2. \tag{Pb-LS}$$

Here $(z_i, y_i) \in \mathbb{R}^n \times \{0, 1\}$, for every $i = 1,\dots,N$.

The tests are performed over four different datasets for binary classification: `MNIST` [51], `MUSH` [56], `A9A` and `IJCNN1` [23]. The data sets are divided into a training set and a testing set as specified in Table 2.1.

Data set	nr. of features (n)	Training set size (N)	Testing set size (N_t)
MNIST	784	60000	10000
MUSH	112	6503	1621
A9A	123	22793	9768
IJCNN1	22	49990	91701

Table 2.1: Data sets with number of features n and number of instances of the training set N and the testing set N_t.

2.4.2 Preliminary parameter tuning: number of levels and sample set cardinalities

Our method is characterized by two parameters that may be problem-dependent: the number of levels $\ell_{\max}$ and the cardinality N^ℓ of each sub-sample set $\mathcal{S}^\ell$, for $\ell = 1, ..., \ell_{\max} - 1$ with $N^{\ell_{\max}} = N$. Clearly, these parameters affect the weight of each gradient and function evaluation during the execution of the method.

In this section, we present the results of the experimental investigation on the influence of these parameters on the performance of the method by comparing the performance of different variants of MU$^\ell$STREG against the one-level version of our algorithm that corresponds in fact to a weighted AR1 method.

In Tables 2.2, 2.4 and 2.6 we consider the weighted evaluations. We report the for both problems (Pb-LOG) and (Pb-LS) the values normalized with respect to the one-level version, for which we indicate also in parenthesis the total number of weighted evaluations. In every column, we underline the best result (minimum number of weighted evaluations) for the multilevel variants and we highlight in italics the results that are worse than those of the one-level version (those corresponding to a factor larger than one). Moreover Tables 2.3, 2.4 and 2.7 report the maximum classification accuracy (in percentage) achieved by the considered methods.

2.4.2.1 Two-level hierarchy

In this section, we fix $\ell_{\max} = 2$, that is $|\mathcal{S}^2| = N^2 = N$, and consider different values for the cardinality of $\mathcal{S}^1$. Since N depends on the dataset, we choose the values of $N^1 = |\mathcal{S}^1|$ proportional to N in order to have a fixed ratio $r := \frac{|\mathcal{S}^1|}{N} \in \{0.5, 0.2, 0.1, 0.05, 0.025, 0.01\}$.

The methods reach convergence for both problems and for every dataset. For the efficiency, Table 2.2 shows that the use of a two-level hierarchy yields improvements in

		Pb-LOG				Pb-LS			
		MNIST	MUSH	A9A	IJCNN1	MNIST	MUSH	A9A	IJCNN1
1-level		1 (1308)	1 (94)	1 (139)	1 (21)	1 (1158)	1 (103)	1 (119)	1 (18)
2-levels	**r= 0.5**	*1.2*	0.6	*2.3*	*1.1*	*1.0*	0.4	*2.4*	*1.2*
	r= 0.2	*1.3*	0.5	*1.6*	0.6	0.6	0.4	*1.6*	0.8
	r= 0.1	0.9	0.4	*1.6*	0.6	0.4	0.2	*1.1*	0.7
	r= 0.05	0.9	0.4	*1.7*	0.6	0.5	0.4	*1.3*	0.7
	r= 0.025	0.7	0.5	*1.3*	0.6	0.3	0.3	*1.2*	0.6
	r= 0.01	0.5	0.4	*1.2*	0.7	0.4	0.3	*1.1*	0.6

Table 2.2: **One-level vs. two-levels: computational effort.** Different variants of two-level methods based on $r = |\mathcal{S}^1|/N$. Value of #f/g to reach convergence, normalized with respect to the one-level version, for which #f/g is reported in parenthesis.

		Pb-LOG				Pb-LS			
		MNIST	MUSH	A9A	IJCNN1	MNIST	MUSH	A9A	IJCNN1
1-level		89.7	98.7	84.7	91.5	89.9	99.4	84.7	91.7
2-levels	**r= 0.5**	89.6	99.1	84.8	91.5	89.6	97.2	84.7	91.7
	r= 0.2	89.5	98.2	84.8	91.5	89.7	97.5	84.7	91.7
	r= 0.1	89.5	98.5	84.8	91.5	89.6	96.2	84.7	91.8
	r= 0.05	89.5	98.1	84.8	91.5	89.6	96.6	84.7	91.8
	r= 0.025	89.6	98.5	84.8	91.5	89.7	96.0	84.7	91.8
	r= 0.01	89.6	98.2	84.8	91.5	89.7	97.3	84.7	91.7

Table 2.3: **One-level vs. two-levels: testing set accuracy**. Different variants of two-level methods based on $r = |\mathcal{S}^1|/N$. Value of %tA at convergence.

lowering the number of evaluations for most of the problems and for most of the values of the cardinality ratios r, and is especially favorable when the ratio of the cardinalities is low.

Notably, for the MUSH dataset for both the convex and nonconvex problems we have a significant reduction in the number of evaluations in each test with two levels. On the other hand, for the A9A dataset the use of our two-level method with any cardinality of $\mathcal{S}^1$ results in even more computational effort than the case with one level. In between these two cases, for the rest of the datasets, we always have a decrease of computational effort with respect to the one-level method with $|\mathcal{S}^1| \leq 0.1N$ (MNIST) or $|\mathcal{S}^1| \leq 0.2N$ (IJCNN1). Note however that the correlation between the ratio $\frac{|\mathcal{S}^1|}{N}$ and the computational effort is not monotonic, meaning that with the decrease of the cardinality of $\mathcal{S}^1$ there is no systematic decrease in the number of evaluations.

From Table 2.3, we note that the classification accuracy reached by MU^{2}STREG is comparable with that obtained with the one-level version, with the only exception of the

MUSH dataset where we have a small increase in the convex case with MU^{2}STREG with $|\mathcal{S}^1| = 0.5N$ and a decrease for all the two-level versions in the nonconvex case.

2.4.2.2 Three-level hierarchy

We now investigate what happens with a deeper hierarchy and we set $\ell_{\max} = 3$ and, as in the previous section, we choose the cardinality of the sub-sampling sets by fixing the same fraction of the number of samples N for all datasets in both problems. Specifically, given $|\mathcal{S}^3| = N$, we fix the cardinality of $\mathcal{S}^2$ such that $\frac{|\mathcal{S}^2|}{N} = 0.1$ and vary the cardinality of $\mathcal{S}^1$ so that $r = \frac{|\mathcal{S}^1|}{N} \in \{0.025, 0.01, 0.005, 0.001\}$.

MU^{3}STREG is now compared with the one-level method and with one version of the two-level method. Taking into account the experiments conducted in Subsection 2.4.2.1, we choose the two-level method with $|\mathcal{S}^2| = N$ and $|\mathcal{S}^1| = 0.1N$ for which the three-level methods tested here are a natural extension.

		Pb-LOG				Pb-LS			
		MNIST	MUSH	A9A	IJCNN1	MNIST	MUSH	A9A	IJCNN1
1-level		1 (1308)	1 (94)	1 (139)	1 (21)	1 (1158)	1 (103)	1 (119)	1 (18)
2-levels r= 0.1		0.9	0.4	*1.6*	0.6	0.4	0.2	*1.1*	0.7
3-levels	**r= 0.025**	0.1	0.4	0.2	0.5	0.2	0.1	0.4	0.4
	r= 0.01	0.1	0.3	0.2	0.4	0.1	0.1	0.4	0.4
	r= 0.005	0.1	0.4	0.1	0.5	0.1	0.2	0.4	0.4
	r= 0.001	0.1	0.5	0.2	0.5	0.1	0.2	0.4	0.5

Table 2.4: **One-level, two-levels vs. three-levels: computational effort**. Different variants of three-level methods based on $r = |\mathcal{S}^1|/N$ ($|\mathcal{S}^2|/N = 0.1$). Value of #f/g to reach convergence, normalized with respect to the one-level version, for which #f/g is reported in parenthesis and the same value for the two-level version with $r = 0.1$.

		Pb-LOG				Pb-LS			
		MNIST	MUSH	A9A	IJCNN1	MNIST	MUSH	A9A	IJCNN1
1-level		89.7	98.7	84.7	91.5	89.9	99.4	84.7	91.7
2-levels r= 0.1		89.5	98.5	84.8	91.5	89.6	96.2	84.7	91.8
3-levels	**r= 0.025**	89.6	98.0	85.0	91.5	89.6	99.1	84.9	91.7
	r= 0.01	89.6	98.8	84.9	91.5	89.6	98.5	84.7	92.3
	r= 0.005	89.7	98.5	84.9	91.5	89.8	99.3	84.7	91.7
	r= 0.001	89.6	97.5	85.0	91.5	89.6	99.1	84.7	91.7

Table 2.5: **One-level, two-levels vs. three-levels: testing set accuracy**. Different variants of three-level methods based on $r = |\mathcal{S}^1|/N$ ($|\mathcal{S}^2|/N = 0.1$), the two-level method with $r = 0.1$ and the one-level version. Value of %tA at convergence.

It can be seen from Table 2.4 that for each problem type (convex or nonconvex) and for each dataset the method that uses the least number of evaluations is always a three-level method. More interestingly, Table 2.4 reveals that the use of MU^{3}STREG results in a significant drop in the number of evaluations needed to achieve convergence with respect to the one and two-level variants. Moreover, Table 2.5 shows that the classification accuracy obtained using the one-level method, MU^{2}STREG and MU^{3}STREG are similar, regardless of problem type and dataset.

2.4.2.3 Five-level hierarchy

The results on the three-level hierarchy shown in Subsection 2.4.2.2 give us good indications for choosing the number of levels $\ell_{\max}$ and the cardinalities of the subsampling sets $\{\mathcal{S}^\ell\}_{\ell=1}^{\ell_max}$. But we want to investigate a little bit further, and in this paragraph we illustrate the results of a test on a five-level version of MU$^\ell$STREG, thus called MU^{5}STREG, and compare it with the one-, two- and three-level versions. Since a five-level hierarchy does not allow us many choices for the cardinality of the sample sets, we chose a single configuration of the hierarchy that could fit all the data sets considered. Thus, the cardinalities of the five sample sets are as follows: $|\mathcal{S}^5| = N$, $|\mathcal{S}^4| = 0.1N$, $|\mathcal{S}^3| = 0.05N$, $|\mathcal{S}^2| = 0.01N$ and $|\mathcal{S}^1| = 0.001N$.

The MU^{5}STREG is compared with those two- and three-level versions that were best for almost all datasets and problems, namely the two-level method with $|\mathcal{S}_2| = N$ and $|\mathcal{S}_1| = 0.1N$ and the three-level method with $|\mathcal{S}_3| = N$, $|\mathcal{S}_2| = 0.1N$ and $|\mathcal{S}_1| = 0.01N$.

Looking at Tables 2.6 and 2.7 we can conclude that the five-level version in most cases is not better than the three-level version. In fact, referring to Table 2.6 we see how the computational cost to achieve convergence of MU^{5}STREG is greater than the three-level version in almost all cases, the only exceptions are for the convex problem (Pb-LOG) with the `MNIST` dataset and for the nonconvex problem (Pb-LS) with the `A9A` dataset; furthermore, looking at Table 2.7, we see how there is no substantial improvement in classification accuracy, in fact where there is an increase of accuracy with respect to three-level method it is at most of the 0.7% (e.g. problem (Pb-LOG) with `MUSH`). The worse performance of MU^{5}STREG compared to MU^{3}STREG could be due to the fact that a five-level hierarchy *warps* too much the original problem. In a hierarchy of five levels, the function $m_k^{R,1}$, defined according to (2.56), surely is very different from the

	Pb-LOG				Pb-LS			
	MNIST	MUSH	A9A	IJCNN1	MNIST	MUSH	A9A	IJCNN1
1-level	1 (1308)	1 (94)	1 (139)	1 (21)	1 (1158)	1 (103)	1 (119)	1 (18)
2-levels r= 0.1	0.87	0.39	*1.65*	0.61	0.43	0.24	*1.10*	0.71
3-levels r= 0.01	0.06	0.33	0.19	0.42	0.11	0.12	0.37	0.40
5-levels	0.04	0.50	0.22	0.87	0.46	0.44	0.26	0.63

Table 2.6: **One-level, two-levels, three-levels vs. five-levels: computational effort**. One five-level hierarchy: $|\mathcal{S}^5| = N$, $|\mathcal{S}^4| = 0.1N$, $|\mathcal{S}^3| = 0.05N$, $|\mathcal{S}^2| = 0.01N$ and $|\mathcal{S}^1| = 0.001N$. Value of #f/g to reach convergence, normalized with respect to the one-level version, for which #f/g is reported in parenthesis and the same value for the two-level version with $r = 0.1$ and for the three-level version with $\mathcal{S}^2 = 0.1$ and $r = 0.01$.

	Pb-LOG				Pb-LS			
	MNIST	MUSH	A9A	IJCNN1	MNIST	MUSH	A9A	IJCNN1
1-level	89.71	98.70	84.74	91.51	89.86	99.44	84.75	91.66
2-levels r= 0.1	89.52	98.52	84.83	91.51	89.64	96.18	84.72	91.80
3-levels r= 0.01	89.55	98.77	84.89	91.49	89.57	98.46	84.68	92.25
5-levels	89.56	99.44	84.86	91.57	89.34	95.74	85.16	92.67

Table 2.7: **One-level, two-levels, three-levels vs. five-levels: testing set accuracy**. One five-level hierarchy: $|\mathcal{S}^5| = N$, $|\mathcal{S}^4| = 0.1N$, $|\mathcal{S}^3| = 0.05N$, $|\mathcal{S}^2| = 0.01N$ and $|\mathcal{S}^1| = 0.001N$; the three-level method with $\mathcal{S}^2 = 0.1$ and $r = 0.01$, the two-level method with $r = 0.1$ and the one-level version. Value of %tA at convergence.

same $m_k^{R,1}$ that we find at the bottom of a hierarchy with only three levels. Probably, the step $s_k^{\ell_{\max}}$ obtained after a full descent of five levels and back provides a smaller improvement than the one provided by a step $s_k^{\ell_{\max}}$ obtained after a descent of only three levels. Hence to reach the maximum accuracy on the testing set MU^{5}STREG needs more iterations and consequently more functions and gradient (weighted) evaluations, resulting in a less efficient method than MU^{3}STREG.

2.4.3 Comparison with SVRG

In this section, we compare MU$^\ell$STREG against a mini-batch version of SVRG on the convex problem (Pb-LOG) and on the nonconvex one (Pb-LS) using the four datasets `MNIST`, `MUSH`, `A9A` and `IJCNN1`. Specifically, for each problem and for each dataset we perform five runs starting from five different random initial guesses for a total of forty numerical tests.

We use the version of MU$^\ell$STREG that gave the best performance for most of the problems in the tuning tests reported in Subsection 2.4.2, that is the three-level

MU^{3}STREG version with the sample cardinalities $N^3 = |\mathcal{S}^3| = N$, $N^2 = |\mathcal{S}^2| = 0.1N$ and $N^1 = |\mathcal{S}^1| = 0.01N$.

MU^{3}STREG and SVRG are compared reporting the maximum classification accuracy on the testing set achieved and the corresponding required computational effort. Moreover, we declare a run as a failure when the achieved classification accuracy is below 80%.

2.4.3.1 Convex problem: logistic classification problem (Pb-LOG)

Here we consider the results of the tests performed on problem (Pb-LOG). Figure 2.3 shows the classification accuracy on the testing set against the number of evaluations for every dataset selecting, for each solver, among the five runs per problem, the one that returns the highest accuracy. Table 2.8 shows mean values of the five runs, instead, together with the standard deviation for the classification accuracy. We only report the runs obtained with $b = 10$ for SVRG (and three choices for the learning rate α) as those obtained with $b = 20$ are rather similar. Both values of the mini-batch size b are considered in Figure 2.4.

On these tests both SVRG and MU^{3}STREG always reach convergence with an accuracy on the testing set higher than 80%, moreover the maximum accuracy for each dataset does not vary much depending on the method applied (see Table 2.8).

Regarding the efficiency of the various methods, looking at the plots in Figures 2.3 we can see that SVRG is quite effective on these convex problems. In these cases, the choice of the step-size is not very critical and a quite large one ($\alpha = 0.5$) can be safely used for all the datasets with the best results. In these experiments, our MU^{3}STREG does not outperform the best version of SVRG, but it shows comparable performance to SVRG with $\alpha = 0.1$ and is (almost) always better than the worst version of SVRG, with the advantage of not requiring the tuning of the step-size.

This is clearly summarized in Figure 2.4, where we show the performance profiles of MU^{3}STREG against the three versions of SVRG with mini-batch size $b = 10$ in the left-hand side plot and $b = 20$ on the right-hand side. Each profile is constructed from the twenty runs performed and is based on the weighted number of gradient and objective function evaluations to achieve maximum accuracy on the testing set. We can see that on average MU^{3}STREG is comparable to/slightly better than SVRG with $\alpha = 0.1$ and

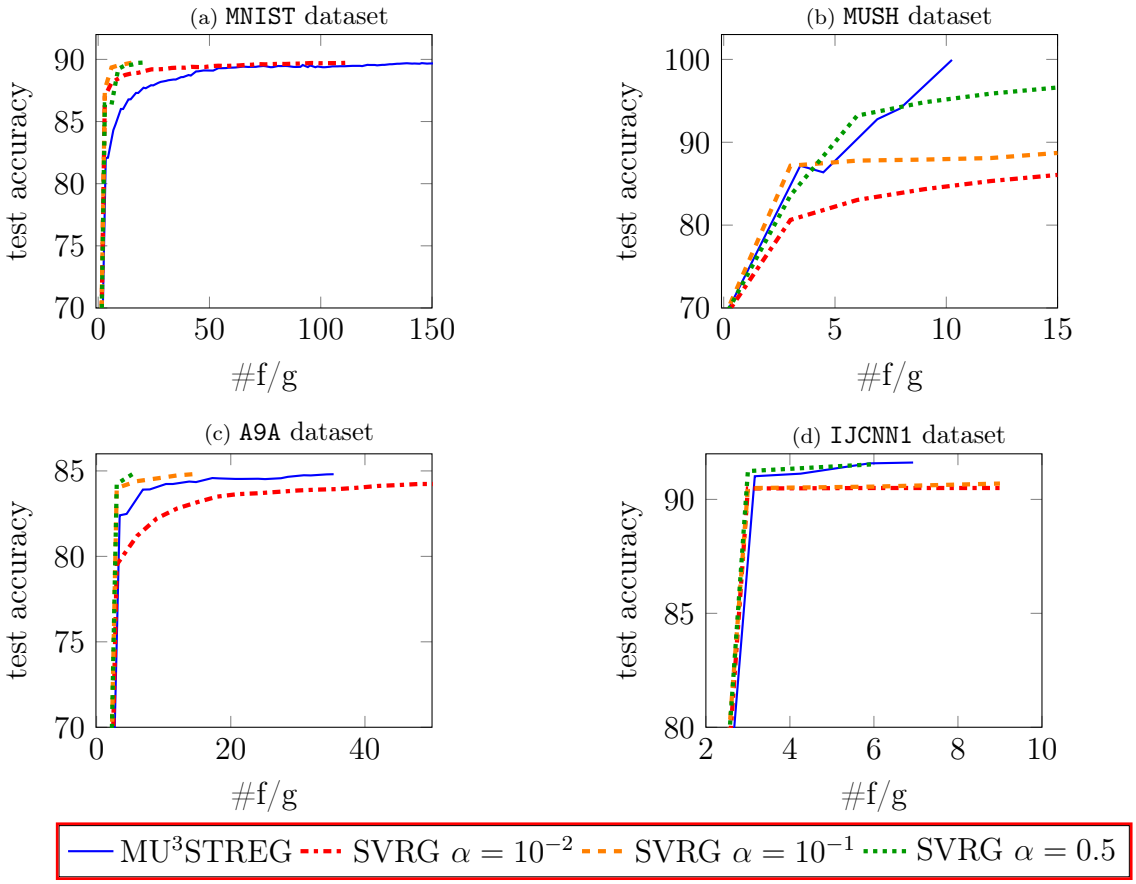

Figure 2.3: (Pb-LOG) Comparison between MU^{3}STREG and SVRG with mini-batch size $b = 10$ on MNIST(2.3a), MUSH(2.3b), A9A(2.3c) and IJCNN1(2.3d) datasets. Plot of classification accuracy on testing set against the number of function and gradient evaluations for the successful run with the highest accuracy for every method.

far better than SVRG with small step-size.

2.4.3.2 Nonconvex problem: nonlinear Least Squares (Pb-LS)

In this section, we report the results of the tests on the nonconvex problem (Pb-LS). As in the previous section, for each dataset, we perform five tests with random initial guesses. Then we show the averaged values in Table 2.9, while in Figure 2.5 we plot for each method the run that gives the maximum accuracy, and the performance profiles in Figure 2.6 take into account the whole 20 runs.

In general, all the methods tested reach convergence but the accuracy reached varies a lot because of the nonconvexity of the problem. In particular, many versions of SVRG

MNIST				
	SVRG $b = 10$			**MU^{3}STREG**
	$\alpha = 0.01$	**$\alpha = 0.1$**	**$\alpha = 0.5$**	
Avg. %tA	89.70	89.73	89.73	89.62
StD %tA	0.02	0.01	0.03	0.10
Avg. # f/g	105.00	14.40	21.60	138.70
StD # f/g	9.25	3.29	2.51	25.97
MUSH				
	SVRG $b = 10$			**MU^{3}STREG**
	$\alpha = 0.01$	**$\alpha = 0.1$**	**$\alpha = 0.5$**	
Avg. %tA	97.42	97.51	97.68	97.88
StD %tA	0.27	0.27	0.20	1.45
Avg. # f/g	989.10	102.57	17.99	13.63
StD # f/g	25.66	2.51	8.48	2.13
A9A				
	SVRG $b = 10$			**MU^{3}STREG**
	$\alpha = 0.01$	**$\alpha = 0.1$**	**$\alpha = 0.5$**	
Avg. %tA	84.74	84.77	84.87	84.75
StD %tA	0.07	0.05	0.06	0.05
Avg. # f/g	144.59	15.60	8.40	25.28
StD # f/g	16.07	1.34	2.51	6.66
IJCNN1				
	SVRG $b = 10$			**MU^{3}STREG**
	$\alpha = 0.01$	**$\alpha = 0.1$**	**$\alpha = 0.5$**	
Avg. %tA	91.50	91.50	91.50	91.54
StD %tA	0.01	0.00	0.02	0.05
Avg. # f/g	250.20	30.00	6.00	8.05
StD # f/g	17.31	0.00	0.00	0.98

Table 2.8: (Pb-LOG) Comparison between MU^{3}STREG and SVRG with mini-batch size $b = 10$. Average of maximum classification accuracy and number of evaluations, with corresponding standard deviation.

find solutions with a classification accuracy lower than 80% and are therefore considered as a failure. The number of failures is reported in Table 2.9 as # **fails**. If the failure occurs for all the initial guesses, the symbol "-" is used. Generally, SVRG fails with large values of the step size α, which are feasible just for the IJCNN1 dataset.

On the other hand, MU^{3}STREG is quite efficient on these nonconvex problems and not only always returns solutions that lead to a classification accuracy greater than 80%, but also always proves to be by far the most efficient method in terms of computational effort to obtain these solutions.

All of this is further summarized in Figure 2.6 in which the performance profiles over the twenty tests of MU^{3}STREG against the three versions of SVRG with $b = 10$ (Figure 2.6, left) and $b = 20$ (Figure 2.6, right) are shown. The advantage of an automatic step

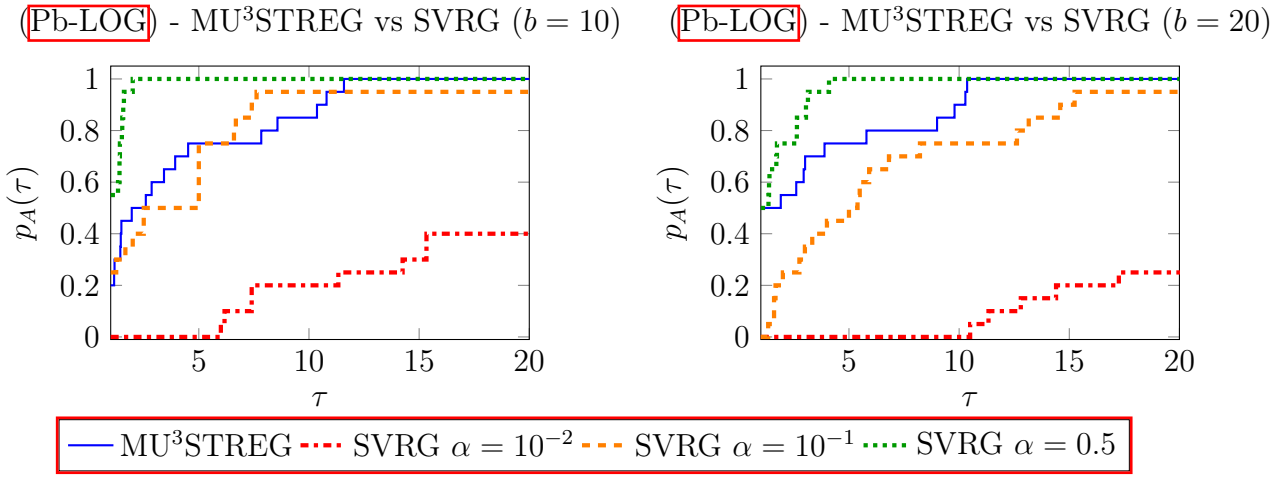

Figure 2.4: (Pb-LOG) Number of weighted evaluations to achieve maximum classification accuracy performance profile: MU^{3}STREG and SVRG with mini-batch size $b = 10$ (left) and $b = 20$ (right) with various stepsizes α.

selection is thus clear in the context of nonconvex problems.

2.4.4 Numerical investigation on the finest sample size

We recall that Algorithm 6 is in fact the adaptation of Algorithm 5 to problem (2.2) assuming that the finest level function $f^{\mathcal{S}^{\ell_{\max}}}$ is the exact objective f in (2.2), that is that N is such that the full sum can be computed. However, the stochastic framework of Algorithm 5 discussed in sections 2.1 and 2.2 is by far more general. Indeed it allows for inexact approximations of f at the finest level, thus allowing for the solution of problems in which the full sample evaluation is not affordable, a situation that is not covered by SVRG convergence theory.

Specifically, in the definition of $\rho_k^{\ell_{\max}}$ at Line 14 the values $f^{\ell_{\max}}(x_k)$ and $f^{\ell_{\max}}(x_k+s_k)$ do not need to coincide with $f(x_k)$ and $f(x_k+s_k)$. Algorithm 6 can thus be called at fine level with a function $f^{\mathcal{S}^{\ell_{\max}}}$ defined on a subset $\mathcal{S}^{\ell_{\max}} \subset \{1,\dots,N\}$, as long as the Taylor model at fine level remains a fully linear model for f, or, even if the full sample set is used to evaluate the gradient and to compute the step, the functions approximations can be evaluated on a smaller subset.

In Table 2.10 we investigate these settings and we report the results obtained using MU^{3}STREG varying

$$N^{\ell_{\max}} = \{1, 0.85, 0.75\}N. \tag{2.68}$$

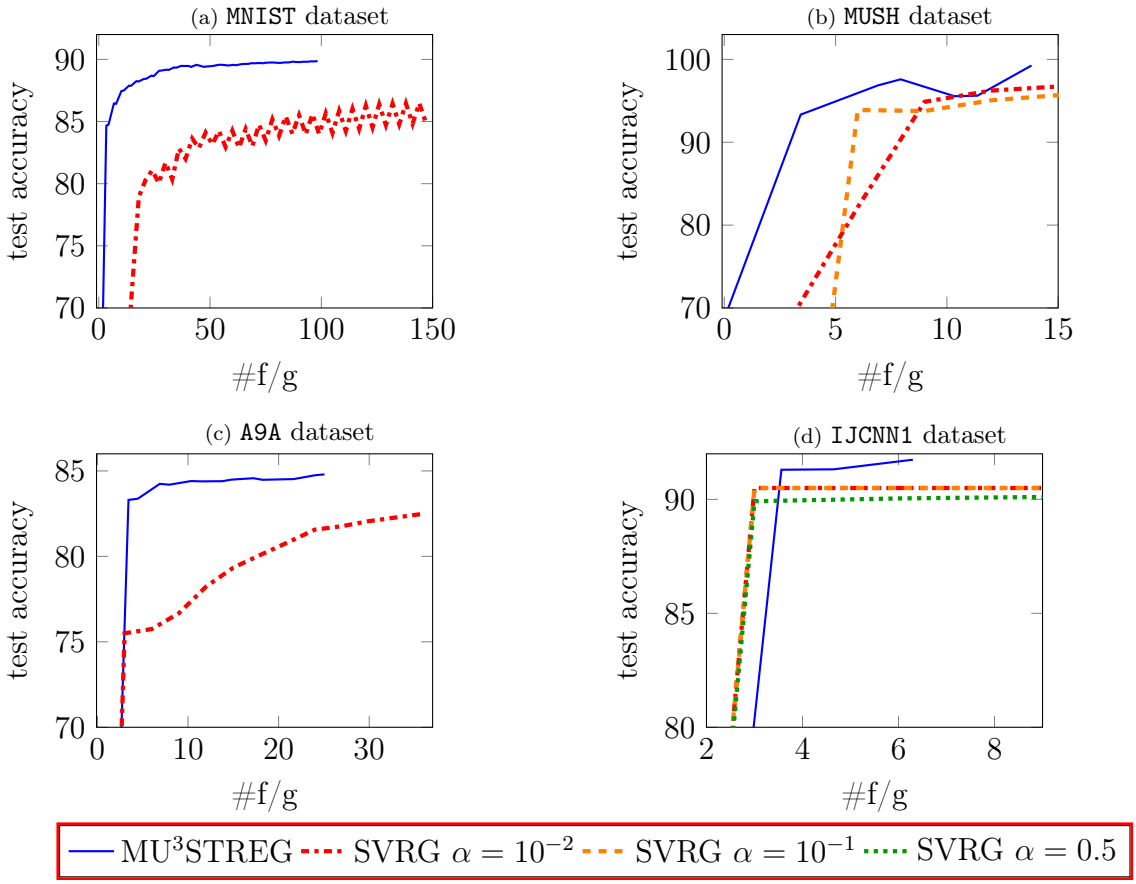

Figure 2.5: (Pb-LS) Comparison between MU³STREG and SVRG with mini-batch size $\mathbf{b} = \mathbf{10}$ on MNIST(2.5a), MUSH(2.5b), A9A(2.5c) and IJCNN1(2.5d).Plot of classification accuracy on testing set against the number of function and gradient evaluations for the successful run with the highest accuracy for every method. The curves that are not plotted correspond to methods that fail for every run for that dataset.

If the full gradient is not evaluated, the stopping criterion (2.58) might not be meaningful. Below we thus use a heuristic stopping test. When (2.58) is satisfied for the first time, after a fine or a coarse step, a new set of $N^{\ell_{\max}}$ randomly chosen samples are drawn and fine steps are taken until (2.58) is satisfied again. In our tests, one additional fine step was sufficient for the stopping criterion to be satisfied.

In Table 2.10 results in the columns with header f are obtained by computing the full gradient (i.e., taking into account all the N samples) at the finest level and using the usual stopping criterion on the gradient norm, while the computation of $\rho_k^{\ell_{\max}}$ involves the objective function averaged on $N^{\ell_{\max}}$ samples as given in (2.68). Differently, results

MNIST							
	SVRG $b = 10$			**SVRG** $b = 20$			**MU^{3}STREG**
	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.5$	
# fails	0	2	5	0	5	5	0
Avg. %tA	90.28	90.23	-	90.43	-	-	89.84
StD %tA	0.01	0.01	-	0.13	-	-	0.03
Avg. # f/g	29646.60	20032.00	-	17843.40	-	-	84.86
StD # f/g	217.25	3156.21	-	6336.56	-	-	7.14
MUSH							
	SVRG $b = 10$			**SVRG** $b = 20$			**MU^{3}STREG**
	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.5$	
# fails	0	0	5	0	0	5	0
Avg. %tA	98.78	98.49	-	98.69	99.27	-	98.04
StD %tA	0.27	0.03	-	0.25	0.55	-	0.84
Avg. # f/g	125.36	885.33	-	341.89	230.33	-	20.23
StD # f/g	19.37	14.75	-	722.65	121.18	-	2.70
A9A							
	SVRG $b = 10$			**SVRG** $b = 20$			**MU^{3}STREG**
	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.5$	
# fails	0	0	5	0	5	5	0
Avg. %tA	84.66	85.15	-	85.00	-	-	84.66
StD %tA	0.02	0.04	-	0.05	-	-	0.10
Avg. # f/g	1310.88	1199.29	-	840.77	-	-	23.61
StD # f/g	15.44	582.12	-	300.51	-	-	4.11
IJCNN1							
	SVRG $b = 10$			**SVRG** $b = 20$			**MU^{3}STREG**
	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.5$	
# fails	0	0	0	0	0	0	0
Avg. %tA	91.68	91.72	90.43	91.68	90.52	89.89	91.69
StD %tA	0.00	0.00	0.01	0.00	0.00	0.01	0.18
Avg. # f/g	2785.20	303.60	76.80	553.87	31.80	34.80	9.28
StD # f/g	22.51	1.34	1.64	1.64	1.64	4.55	0.55

Table 2.9: (Pb-LS) Comparison between MU^{3}STREG and SVRG with mini-batch size $b = 10$ and $b = 20$. Average of maximum classification accuracy reached and number of evaluations, with corresponding standard deviation. The average is evaluated only on the successful tests.

in the columns with header $f, \nabla f$ are obtained by averaging both the objective function and its gradient on $N^{\ell_{\max}}$ samples, and using the proposed heuristic stopping criterion.

As in the previous section, results are averaged over 5 runs (for 5 random initial guesses) in the solution of (Pb-LS) on the 4 data sets.

We can observe that in all cases, the classification accuracy is not affected by the value of $N^{\ell_{\max}}$ and the computational effort mildly varies for the datasets `MUSH`, `A9A` and `IJCNN1`. The only exception is the case of `MNIST`, where the average number of evalua-

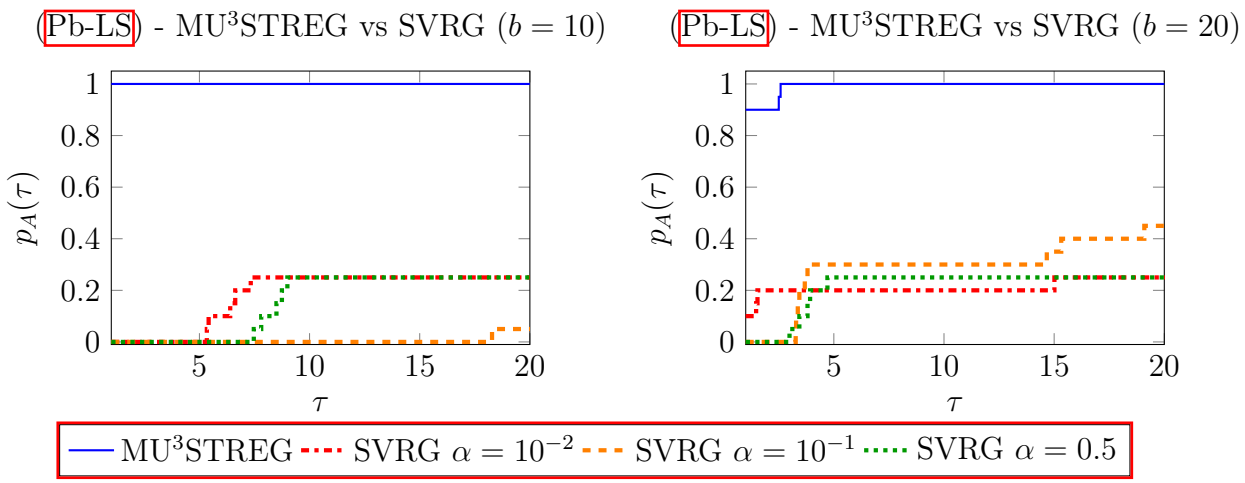

Figure 2.6: (Pb-LS) Number of weighted evaluations to achieve maximum classification accuracy performance profile: MU³STREG and SVRG with mini-batch size $b = 10$ (left) and $b = 20$ (right) with various stepsizes α.

tions #f/g increases as $N^{\ell_{\max}}$ decreases. The 2 fails in the solution of `MUSH` when using inexactness in both gradient and function values (columns header $f, \nabla f$), correspond to the computation of stationary points with an unsatisfactory classification accuracy.

MNIST					
	100%	**85%**		**75%**	
		f	f, ∇f	f	f, ∇f
# fails	0	0	0	0	0
Avg. %tA	89.84	89.85	89.92	89.87	90.01
StD %tA	0.03	0.02	0.03	0.04	0.08
Avg. #f/g	84.86	176.08	200.42	170.08	438.67
StD #f/g	7.14	31.56	51.83	20.84	166.86
MUSH					
	100%	**85%**		**75%**	
		f	f, ∇f	f	f, ∇f
# fails	0	0	0	1	1
Avg. %tA	98.04	97.96	98.37	97.90	98.52
StD %tA	0.84	0.35	0.39	1.03	0.42
Avg. #f/g	20.23	28.53	33.03	31.35	30.58
StD #f/g	2.70	15.04	5.50	8.63	4.01
A9A					
	100%	**85%**		**75%**	
		f	f, ∇f	f	f, ∇f
# fails	0	0	0	0	0
Avg. %tA	84.66	84.74	84.84	84.76	84.85
StD %tA	0.10	0.05	0.10	0.13	0.08
Avg. #f/g	23.61	37.88	32.55	31.07	29.06
StD #f/g	4.11	8.18	5.87	10.92	11.18
IJCNN1					
	100%	**85%**		**75%**	
		f	f, ∇f	f	f, ∇f
# fails	0	0	0	0	0
Avg. %tA	91.69	91.63	91.76	91.67	91.74
StD %tA	0.18	0.08	0.05	0.06	0.06
Avg. #f/g	9.28	30.99	33.76	29.80	29.71
StD #f/g	0.55	6.02	0.76	5.56	2.92

Table 2.10: (Pb-LS) Comparison between MU^{3}STREG varying $N^{\ell_{\max}} = \{1, 0.85, 0.75\}N$. In every column is shown the average of maximum classification accuracy reached by every method and every dataset with corresponding standard deviation and the average number of evaluations with standard deviations.

Conclusions

In this work, we applied Derivative-Free Optimization methods with the possible use of Black-Box functions, to energy production technologies to improve their efficiency and reduce their environmental impact.

In our research project, we focused on gas turbines which are a technology that will play a key role in the short- and medium-term future of the energy transition. In more detail, we focused on optimization methods applied to the cooling system of gas turbines and we analyzed two particular instances of this topic achieving remarkable results.

The first result is the definition from scratch of a framework for optimizing the impingement cooling system for a gas turbine nozzle. We proposed a mathematical formulation of the problem in (1.35), resulting in a mixed variable constrained Black-Box Optimization problem that we numerically addressed using DFO algorithms. While studying a suitable DFO approach for this problem, we also defined and implemented the black-box function NOZZLE which simulates the functioning of the cooling system inside a turbine nozzle and returns as output the parameters to evaluate its efficiency; NOZZLE results in a simple numerical tool for the design of an impingement cooling system. Finally, we combine NOZZLE together with BFO solver in a Derivative-free ℓ_1-penalty method and we validate our approach through numerical tests. The resulting procedure allows for the design of an efficient impingement cooling system and for its improvement without having to rely on the operator's experience and by also reducing the time required with respect to the standard procedure.

The obtained preliminary results form the basis for future developments involving both the black-box function formulation and the optimization method. The NOZZLE function can be improved by starting from a problem with different boundary conditions. In particular, instead of knowing the total mass flow rate $\dot{m}_{tot}$ provided to the cooling system we impose a fixed value for the outlet pressure of the cooling air p_c^{out}. This change

would allow us to remove the constraint on the pressure (1.28) so we do not need to solve the nonlinear equation (1.26), saving some computational effort.

Another improvement of the simulator is to allow the variable *layout* to assume more than two values. Our definition of NOZZLE and the related problem could in fact be treated using a binary variable instead of the categorical variable *layout*, since the latter admits only two values, `inline` and `staggered`. Indeed, there are DFO methods for constrained mixed-integer variable problems that can be adapted to handle binary variables, like the one proposed by Liuzzi *et al.* [53]. Another possible strategy would be to solve two separate constrained Black-Box Optimization problems in which the variable *layout* is fixed to `inline` and `staggered`, respectively, and pick the best solution.

But keeping the variable *layout* as categorical, we leave open many more possibilities for the design of an impingement plate. Clearly, since the Florschuetz model that we used to define NOZZLE admits only two values for *layout*, it is obvious that to admit a layout with values other than `inline` and `staggered` one must overcome the Florschuetz model. Or at least, overcome the use of only Florchuetz's model inside NOZZLE. In fact, while we can think of completely replacing the old Florschuetz model, we can also stand alongside and complement it. In the latter way, we do not give up using a model that has proven to be reliable despite its simplicity. One possible idea is schematically represented in Figure 2.7. Here on the left, we have a representation of the choice we made to define this version of NOZZLE: we have the impingement plate where we use Florschuetz's model, thus we have only two possible layouts, inline and staggered. But what if we split the impingement plate into two regions to allow one layout per region? In this way, considering only the inline and staggered, we can have four possible layouts for the whole plate given by the combinations of inline and staggered on every half-plate, as shown on the right of Figure 2.7. As the figure shows, we could apply Florschuetz's model to the half of the plate that is upstream of the cooling air flow (whose direction is represented by the blue arrow), while for the downstream half, it is necessary to find another model. This is because downstream we are no longer in the exact configuration for which the Florschuetz model was developed, in fact in the first row of jets we will have a nonzero crossflow mass velocity since we will have the contribution coming from the upstream half. Recall that the Florschuetz model was instead developed for a configuration such as that shown on the left in Figure 2.7, in which we have no crossflow coming from upstream of the plate. So a different model needs to be applied to the downstream half.

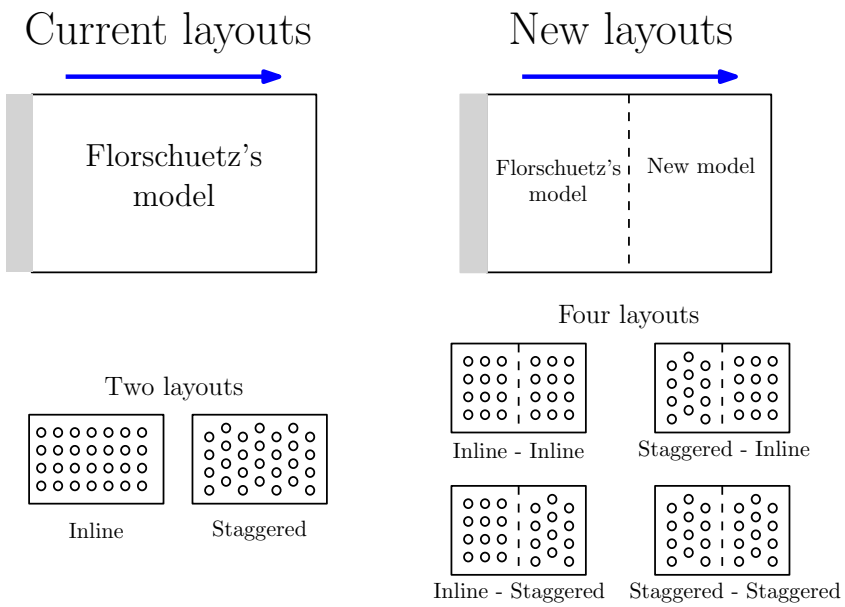

Figure 2.7: **Left:** the current choice used to define NOZZLE with two layouts. **Right:** a hint for a (possible) new idea to allow four layouts.

This different model will be the subject for future research.

Regarding the optimization method, we observed in Section 1.4 that the ℓ_1-penalty BFO method is a local optimization method. However, practitioners often need a global solution. Therefore, the next step will be devoted to the implementation of a global optimization strategy that is suitable for the problem under consideration.

The second contribution of this work is the MU$^\ell$STREG optimization method developed for large problems with a noisy objective function. MU$^\ell$STREG was originally conceived as a method to be applied to a validation phase of the design of a complete cooling system. During this validation, we typically have data-matching between the results of a numerical simulation of the cooling system and empirical data. Typically these are large-scale problems and are affected by random uncertainties either in the measured data or in the simulation processes. We have proposed a new framework for the multilevel solution of stochastic problems, assuming that the stochastic objective function can be represented at different levels of accuracy. Our framework encompasses both hierarchies in the variables space and in the function space and it is, to our knowledge, the first stochastic framework for multilevel methods, that are currently limited to the deterministic case.

The proposed method MU$^\ell$STREG is a new multilevel stochastic gradient method based on adaptive regularization that generalizes the AR1 method [21] and we propose a stochastic convergence analysis for it. This convergence theory is the first stochastic convergence study both for multilevel methods and for adaptive regularization methods.

We show that MU$^\ell$STREG can be interpreted as a variance reduction method for finite-sum minimization problems and we numerically compare it to a mini-batch version of SVRG. We show the advantage of our automatic step selection in the context of nonconvex problems. We also investigate the practical advantages of the stochastic framework over the deterministic one, which allows for the solution of finite-sum problems without the need to evaluate the function/gradient over the full sample set. This makes our method feasible also for problems defined over very large sample sets, a situation that is not covered by the convergence theory of standard variance reduction methods.

Supported by these theoretical and experimental results, in the near future we aim to continue the experimentation by applying MU$^\ell$STREG on data-fitting problems to validate the design of the entire network of cooling systems of a gas turbine so that we can test MU$^\ell$STREG on a stochastic optimization problem that allows us to define a hierarchy on both the space of variables and the approximations of the objective function.

Bibliography

[1] M. A. Abramson, "Mixed variable optimization of a load-bearing thermal insulation system using a filter pattern search algorithm," *Optim. Eng.*, vol. 5, pp. 157–177, 2004.

[2] M. A. Abramson, C. Audet, J. W. Chrissis, and J. G. Walston, "Mesh adaptive direct search algorithms for mixed variable optimization," *Optim. Lett.*, vol. 3, pp. 35–47, 2009.

[3] S. Alarie, C. Audet, A. E. Gheribi, M. Kokkolaras, and S. Le Digabel, "Two decades of blackbox optimization applications," *EURO J. Comput. Optim.*, vol. 9, p. 100 011, 2021.

[4] Z. Allen-Zhu, "Natasha: Faster non-convex stochastic optimization via strongly non-convex parameter," in *International Conference on Machine Learning*, PMLR, 2017, pp. 89–97.

[5] C. Audet and J. E. Dennis Jr, "Pattern search algorithms for mixed variable programming," *SIAM J. Optim.*, vol. 11, no. 3, pp. 573–594, 2001.

[6] C. Audet and J. E. Dennis Jr, "A pattern search filter method for nonlinear programming without derivatives," *SIAM J. Optim.*, vol. 14, no. 4, pp. 980–1010, 2004.

[7] C. Audet and J. E. Dennis Jr, "A progressive barrier for derivative-free nonlinear programming," *SIAM J. Optim.*, vol. 20, no. 1, pp. 445–472, 2009.

[8] C. Audet and W. Hare, *Derivative-free and Blackbox Optimization* (Springer Series in Operations Research and Financial Engineering). Cham, Switzerland: Springer, 2017.

[9] C. Audet and M. Kokkolaras, "Blackbox and derivative-free optimization: Theory, algorithms and applications," *Optim. Eng.*, vol. 17, pp. 1–2, 2016.

[10] A. Bandeira, K. Scheinberg, and L. Vicente, "Convergence of trust-region methods based on probabilistic models," *SIAM J. Optim.*, vol. 24, no. 3, pp. 1238–1264, 2014.

[11] S. Bellavia, G. Gurioli, B. Morini, and P. L. Toint, "Trust-region algorithms: Probabilistic complexity and intrinsic noise with applications to subsampling techniques," *EURO J. Comput. Optim.*, vol. 10, p. 100 043, 2022.

[12] S. Bellavia, N. Krejić, B. Morini, and S. Rebegoldi, "A stochastic first-order trust-region method with inexact restoration for finite-sum minimization," *Comput. Optim. Appl.*, vol. 84, no. 1, pp. 53–84, 2023.

[13] A. S. Berahas, O. Sohab, and L. N. Vicente, "Full-low evaluation methods for derivative-free optimization," *Optim. Methods Softw.*, vol. 38, no. 2, pp. 386–411, 2023.

[14] E. H. Bergou, Y. Diouane, V. Kungurtsev, and C. W. Royer, "A stochastic levenberg–marquardt method using random models with complexity results," *SIAM/ASA J. Uncert. Quant.*, vol. 10, no. 1, pp. 507–536, 2022.

[15] E. G. Birgin, J. L. Gardenghi, J. M. Martínez, S. A. Santos, and P. L. Toint, "Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models," *Math. Program.*, vol. 163, pp. 359–368, 2017.

[16] L. Bottou, F. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, 2018.

[17] V. Braglia, A. Kopaničáková, and R. Krause, "A multilevel approach to training," *arXiv preprint arXiv:2006.15602*, 2020.

[18] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A multigrid tutorial*. SIAM, 2000.

[19] A. Bűrmen, J. Olenšek, and T. Tuma, "Mesh adaptive direct search with second directional derivative-based hessian update," *Comput. Optim. Appl.*, vol. 62, pp. 693–715, 2015.

[20] H. Calandra, S. Gratton, E. Riccietti, and X. Vasseur, "On high-order multilevel optimization strategies," *SIAM J. Optim.*, vol. 31, no. 1, pp. 307–330, 2021.

[21] C. Cartis, N. I. M. Gould, and P. L. Toint, *Evaluation complexity of algorithms for nonconvex optimization*. MOS-SIAM Series on Optimization, 2022.

[22] C. Cartis and R. Roberts, "Scalable subspace methods for derivative-free nonlinear least-squares optimization," *Math. Program.*, vol. 199, pp. 461–524, 2023.

[23] C. Chang and C. Lin, "LIBSVM: A library for support vector machines," *tist*, vol. 2, 27:1–27:27, 3 2011, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[24] R. Chen, M. Menickelly, and K. Scheinberg, "Stochastic optimization using a trust-region method and random models," *Math. Program.*, vol. 169, pp. 447–487, 2018.

[25] L. Cocchi, F. Marini, M. Porcelli, and E. Riccietti, "Black-box optimization for the design of a jet plate for impingement cooling," *Optim. Eng.*, 2025. DOI: `10.1007/s11081-025-09981-0`.

[26] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-free Optimization* (MPS-SIAM Series on Optimization). Philadelphia, USA: SIAM, 2009.

[27] C. Davis and W. Hare, "Exploiting known structures to approximate normal cones," *Math. Oper. Res.*, vol. 38, no. 4, pp. 665–681, 2013.

[28] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," *Adv. Neural Inf. Process. Syst.*, vol. 27, 2014.

[29] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Math. Program.*, vol. 91, pp. 201–213, 2002.

[30] N. Echebest, M. L. Schuverdt, and R. P. Vignau, "An inexact restoration derivative-free filter method for nonlinear programming," *J. Comput. Appl. Math.*, vol. 36, pp. 693–718, 2017.

[31] G. Fasano, G. Liuzzi, S. Lucidi, and F. Rinaldi, "A linesearch-based derivative-free approach for nonsmooth constrained optimization," *SIAM J. Optim.*, vol. 24, no. 3, pp. 959–992, 2014.

[32] L. W. Florschuetz, D. E. Metzger, D. I. Takeuchi, and R. A. Berry, "Multiple jet impingement heat transfer characteristic: Experimental investigation of in-line and staggered arrays with crossflow," Tech. Rep., 1980.

[33] L. W. Florschuetz, D. E. Metzger, D. I. Takeuchi, and R. A. Berry, "Jet array impingement with crossflow-correlation of streamwise resolved flow and heat transfer distributions," Tech. Rep., 1981.

[34] L. W. Florschuetz, C. R. Truman, and D. E. Metzger, "Streamwise flow and heat transfer distributions for jet array impingement with crossflow," in *Turbo Expo: Power for Land, Sea, and Air*, American Society of Mechanical Engineers, vol. 79634, 1981.

[35] R. W. Fox, P. J. Pritchard, and A. T. McDonald, *Introduction to Fluid Mechanics*. John Wiley & Sons, 2010, ISBN: 9780470547557. [Online]. Available: https://books.google.it/books?id=kiFJRQAACAAJ.

[36] S. Gratton, A. Kopaničáková, and P. L. Toint, "Multilevel objective-function-free optimization with an application to neural networks training," *SIAM J. Optim.*, vol. 33, no. 4, pp. 2772–2800, 2023.

[37] S. Gratton, V. Mercier, E. Riccietti, and P. L. Toint, "A block-coordinate approach of multi-level optimization with an application to physics-informed neural networks," *Comput. Optim. Appl.*, 2024.

[38] S. Gratton, A. Sartenaer, and P. L. Toint, "Recursive trust-region methods for multiscale nonlinear optimization," *SIAM J. Optim.*, vol. 19, pp. 414–444, 2008.

[39] S. Gratton and P. L. Toint, "S2mpj and cutest optimization problems for matlab, python and julia," *arXiv preprint arXiv:2407.07812*, 2024.

[40] S. Gratton and L. N. Vicente, "A merit function approach for direct search," *SIAM J. Optim.*, vol. 24, no. 4, pp. 1980–1998, 2014.

[41] Y. Ha, S. Shashaani, and R. Pasupathy, "Complexity of zeroth-and first-order stochastic trust-region algorithms," *arXiv preprint arXiv:2405.20116*, 2024.

[42] J. Han, S. Dutta, and S. Ekkad, *Gas turbine heat transfer and cooling technology*. CRC press, 2012.

[43] F. P. Incropera, D. P. DeWitt, T. L. Bergman, and A. S. Lavine, *Incropera's Principles of Heat and Mass Transfer*. John Wiley & Sons, Incorporated, 2017, ISBN: 9781119382911. [Online]. Available: https://books.google.it/books?id=PGIAMQAACAAJ.

[44] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," *Adv. in Neural Information Proc. Sys.*, vol. 26, 2013.

[45] M. Kokkolaras, C. Audet, and J. E. Dennis Jr, "Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system," *Optim. Eng.*, vol. 2, pp. 5–29, 2001.

[46] A. Kopaničáková and R. Krause, "Globally Convergent Multilevel Training of Deep Residual Networks," *SIAM J. Sci. Comput.*, vol. 0, no. 0, S254–S280, 2022. DOI: 10.1137/21M1434076.

[47] J. Larson, M. Menickelly, and S. M. Wild, "Derivative-free optimization methods," *Acta Numer.*, vol. 28, pp. 287–404, 2019.

[48] G. Lauga, A. Repetti, E. Riccietti, N. Pustelnik, P. Gonçalves, and Y. Wiaux, "A multilevel framework for accelerating usara in radio-interferometric imaging," in *2024 32nd European Signal Processing Conference (EUSIPCO)*, 2024, pp. 2287–2291. DOI: 10.23919/EUSIPCO63174.2024.10715263.

[49] G. Lauga, E. Riccietti, N. Pustelnik, and P. Gonçalves, "Iml fista: A multilevel framework for inexact and inertial forward-backward. application to image restoration," *SIAM J. Imaging Sc.*, vol. 17, no. 3, pp. 1347–1376, 2024.

[50] S. Le Digabel and S. M. Wild, "A taxonomy of constraints in black-box simulation-based optimization," *Optim. Eng.*, pp. 1–19, 2023.

[51] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, https://keras.io/api/datasets/mnist/.

[52] R. M. Lewis and V. Torczon, "A direct search approach to nonlinear programming problems using an augmented lagrangian method with explicit treatment of linear constraints," in *Technical Report of the College of William and Mary*, 2010, pp. 1–25.

[53] G. Liuzzi, S. Lucidi, and F. Rinaldi, "Derivative-free methods for mixed-integer constrained optimization problems," *J. Optim. Theory Appl.*, vol. 164, pp. 933–965, 2015.

[54] S. Lucidi, V. Piccialli, and M. Sciandrone, "An algorithm model for mixed variable programming," *SIAM J. Optim.*, vol. 15, no. 4, pp. 1057–1084, 2005.

[55] F. Marini, M. Porcelli, and E. Riccietti, *A multilevel stochastic regularized first-order method with application to training*, 2024. arXiv: 2412.11630 [math.OC]. [Online]. Available: https://arxiv.org/abs/2412.11630.

[56] *Mushroom*, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C5959T, 1981.

[57] S. G. Nash, "A multigrid approach to discretized optimization problems," *Optim. Methods Softw.*, vol. 14, no. 1-2, pp. 99–116, 2000.

[58] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[59] "Nowpac: A provably convergent derivative-free nonlinear optimizer with path-augmented constraints," *arXiv preprint arXiv:1403.1931*, 2014.

[60] V. Picheny, R. B. Gramacy, S. M. Wild, and S. Le Digabel, "Bayesian optimization under mixed constraints with a slack-variable augmented lagrangian," *Adv. in Neural Information Proc. Sys.*, vol. 29, 2016.

[61] M. Porcelli and P. L. Toint, "Bfo, a trainable derivative-free brute force optimizer for nonlinear bound-constrained optimization and equilibrium computations with continuous and discrete variables," *ACM Trans. Math. Software*, vol. 44, no. 1, pp. 1–25, 2017.

[62] M. Porcelli and P. L. Toint, "Exploiting problem structure in derivative free optimization," *ACM Trans. Math. Software*, vol. 48, no. 1, pp. 1–25, 2022.

[63] M. Pourbagian, B. Talgorn, W. G. Habashi, M. Kokkolaras, and S. Le Digabel, "Constrained problem formulations for power optimization of aircraft electrothermal anti-icing systems," *Optim. Eng.*, vol. 16, pp. 663–693, 2015.

[64] T. Pourmohamad, "Combining multivariate stochastic process models with filter methods for constrained optimization," Ph.D. dissertation, UC Santa Cruz, 2016.

[65] M. J. D. Powell, *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer, 1994.

[66] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "Stochastic variance reduction for nonconvex optimization," in *International conference on machine learning*, PMLR, 2016, pp. 314–323.

[67] R. G. Regis and S. M. Wild, "Conorbit: Constrained optimization by radial basis function interpolation in trust regions," *Optim. Methods Softw.*, vol. 32, no. 3, pp. 552–580, 2017.

[68] F. Rinaldi, L. Vicente, and D. Zeffiro, "Stochastic trust-region and direct-search methods: A weak tail bound condition and reduced sample sizing," *SIAM J. Optim.*, vol. 34, no. 2, pp. 2067–2092, 2024.

[69] C. W. Royer, O. Sohab, and L. N. Vicente, "Full-low evaluation methods for bound and linearly constrained derivative-free optimization," *Comput. Optim. Appl.*, pp. 1–37, 2024.

[70] P. R. Sampaio and P. L. Toint, "Numerical experience with a derivative-free trust-funnel method for nonlinear optimization problems with general nonlinear constraints," *Optim. Methods Softw.*, vol. 31, no. 3, pp. 511–534, 2016.

[71] J. C. Spall, "Stochastic optimization," in *Handbook of Computational Statistics: Concepts and Methods*, J. E. Gentle, W. K. Härdle, and Y. Mori, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 173–201, ISBN: 978-3-642-21551-3. DOI: 10.1007/978-3-642-21551-3_7. [Online]. Available: https://doi.org/10.1007/978-3-642-21551-3_7.

[72] N. Zuckerman and N. Lior, "Jet impingement heat transfer: Physics, correlations, and numerical modeling," in ser. Advances in Heat Transfer, G. A. Greene, P. A. Hartnett†, A. Bar-Cohen, and Y. I. Cho, Eds., vol. 39, Elsevier, 2006, pp. 565–631. DOI: https://doi.org/10.1016/S0065-2717(06)39006-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065271706390065.