Hledání optimálního tvaru stěn matematického modelu proudění krve v problematice úplného kavopulmonálního cévního napojení

Optimal shape design of walls of blood flow mathematical model focusing on the total cavopulmonary connection

Master thesis

Author:	**Bc. Jan Bureš**
Supervisor:	**doc. Ing. Radek Fučík, Ph.D.**
Consultant:	**MUDr. Mgr. Radomír Chabiniok, Ph.D.**
Academic year:	2024/2025

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bureš**   Jméno: **Jan**   Osobní číslo: **494688**

Fakulta/ústav: **Fakulta jaderná a fyzikálně inženýrská**

Zadávající katedra/ústav: **Katedra matematiky**

Studijní program: **Matematické inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Optimální tvar stěn idealizovaného úplného kavopulmonálního spojení.

Název diplomové práce anglicky:

Optimal wall geometry of an idealized total cavopulmonary connection.

Pokyny pro vypracování:

1. S použitím dostupné literatury a konzultací s odborníky sestavte matematický model cévního proudění v problematice úplného kavopulmonálního spojení (TCPC) založený na mřížkové Boltzmannově metodě (LBM) a navrhněte vhodně parametrizovanou testovací úlohu, která aproximuje charakteristiky a funkčnost systému TCPC.
2. Formulujte optimalizační úlohu zaměřenou na hledání optimálního tvaru stěn s využitím parametricky popsané geometrie testovací úlohy z bodu 1. Pokuste se navrhnout vhodnou účelovou funkci použitelnou v rámci studované problematiky.
3. Proveďte rešerši optimalizačních metod vhodných pro řešení problémů charakterizovaných zvýšenou časovou náročností potřebnou pro vyhodnocení účelové funkce.
4. Pro simulaci proudění vhodně upravte výpočetní kód LBM vyvíjený na KM FJFI ČVUT v Praze. Navrhněte a implementujte obecný optimalizační rámec, jehož součástí bude výpočetní kód LBM. Využijte volně dostupný software nebo sám implementujte vybrané metody matematické optimalizace z bodu 3.
5. Aplikujte metody matematické optimalizace k hledání optimálního řešení pro uvažovanou optimalizační úlohu z bodu 2. Diskutujte získané výsledky a výpočetní náročnost jejich získání.

Seznam doporučené literatury:

[1] T. Krüger, et al., The lattice Boltzmann method: Principles and Practice. Springer International Publishing, 2017.
[2] Z. Guo, S. Chang, Lattice Boltzmann method and its application in engineering. World Scientific, 2013.
[3] J. D. Anderson, Computational Fluid Dynamics. McGraw-Hill series in mechanical engineering. McGraw-Hill Professional, 1995.
[4] F. M. Rijnberg, et al., Energetics of blood flow in cardiovascular disease. Circulation,137(22), 2018, 2393–2407.
[5] S. Boyd, L. Vandenberghe, Convex optimization. Cambridge University Press, 2004.
[6] C. Audet a W. Hare. Derivative-free and blackbox optimization. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, Cham, Switzerland, 1.edice, 2017.

Jméno a pracoviště vedoucí(ho) diplomové práce:

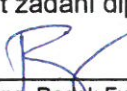**doc. Ing. Radek Fučík, Ph.D.   katedra matematiky   FJFI**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:
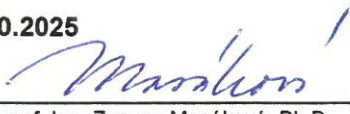
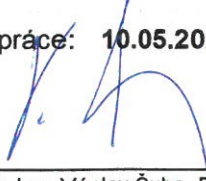**Mudr. Mgr. Radomír Chabiniok, Ph.D.   University of Texas Southwestern Medical Center, USA**

Datum zadání diplomové práce: **31.10.2023**   Termín odevzdání diplomové práce: **10.05.2024**

Platnost zadání diplomové práce: **31.10.2025**

_____
doc. Ing. Radek Fučík, Ph.D.
podpis vedoucí(ho) práce

_____
prof. Ing. Zuzana Masáková, Ph.D.
podpis vedoucí(ho) ústavu/katedry

_____
doc. Ing. Václav Čuba, Ph.D.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

_____   _____
Datum převzetí zadání   Podpis studenta

Chapter 1

Mathematical optimization

Mathematical optimization, also known as mathematical programming, is a broad field that covers various disciplines, including linear and nonlinear optimization, convex programming, integer programming, and more. The goal of this chapter is to summarize the key concepts and techniques relevant to the scope of this work. Specifically, we will focus on methods for solving optimization problems with constraints and those where the objective function is generally unknown and its evaluation is computationally expensive. Classic methods applicable to unconstrained optimization, such as the Davidon-Fletcher-Powell [1] and Broyden-Fletcher-Goldfarb-Shanno [2] algorithms, while fundamental, fall outside the scope of this work and details on these methods can be found for instance in [3].

This chapter will focus on methods that address the specific challenges posed by constrained and generally unknown objective functions. First, a general optimization problem is defined, followed by an introduction to basic techniques for solving constrained problems. Next, black-box optimization methods are discussed. Finally, the proposed optimization framework used in this work is presented.

1.1 General optimization problem

Let $m, n, q \in \mathbb{N}$. Define the continuous functions $f : \mathbf{D} \to \mathbb{R}$, $\boldsymbol{g} : \mathbf{D} \to \mathbb{R}^m$, $\boldsymbol{h} : \mathbf{D} \to \mathbb{R}^q$, where $\mathbf{D} = \mathrm{Dom}\,(f) \cap \mathrm{Dom}\,(\boldsymbol{g}) \cap \mathrm{Dom}\,(\boldsymbol{h})$, i.e., $\mathbf{D}$ is the intersection of the domains of the given functions. Next, define the set

$$\mathbf{X} = \{\, \boldsymbol{x} \in \mathbf{D} \subseteq \mathbb{R}^n \mid \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0} \wedge \boldsymbol{h}(\boldsymbol{x}) = \mathbf{0} \,\}, \tag{1.1}$$

where the inequality $\boldsymbol{g} \leq \mathbf{0}$ and equality $\boldsymbol{h} = \mathbf{0}$ are understood component-wise. The general goal of mathematical optimization is to solve the problem

$$\min_{\boldsymbol{x} \in \mathbf{X}} f(\boldsymbol{x}). \tag{1.2}$$

The function f being minimized is called the objective function, $\mathbf{D}$ is referred to as the domain of the problem, and $\mathbf{X}$ is called the set of admissible solutions of the problem [3]. Note that, henceforth, f denotes only the objective function and not the distribution function discussed in Chapter **??**.

When classifying optimization problems, we refer to what are known as constraints. These are determined by the definition of the set $\mathbf{X}$, i.e., the equality and inequality conditions for the functions $\boldsymbol{g}$ and $\boldsymbol{h}$, and by the domain $\mathbf{D}$. Constraints defined by $\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0} \wedge \boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}$ are called explicit constraints, while those determined by the domain $\mathbf{D}$ are called implicit constraints.

The optimal solution of the problem (1.2) is denoted by $\boldsymbol{x}^\star \in \mathbf{X}$ and is defined as

$$\boldsymbol{x}^\star = \operatorname*{argmin}_{\boldsymbol{x} \in \mathbf{X}} f(\boldsymbol{x}). \tag{1.3}$$

Note that the optimal solution may not be unique, and we refer to the set of all optimal solutions as the optimal set. It is also important to recognize that the search for an optimal solution can equivalently be formulated as finding the maximum of the function $-f$ over the same set $\mathbf{X}$, enabling the use of the same techniques for solving maximization problems [3, 4].

1.2 Solving constrained problems

We will now consider the problem 1.2, where the set of admissible solutions is given by 1.1. In this section, we describe how to generally solve this problem by converting it into a sequence of unconstrained optimization problems. There exist several other methods for solving constrained problems, but here we will focus on the penalty and the barrier methods.

1.2.1 Penalty methods

As mentioned earlier, penalty methods convert constrained optimization problems into unconstrained ones. The fundamental principle of penalty methods is to incorporate the conditions that define the set of admissible solutions by adding a penalty term to the objective function, which reflects the degree of violation of those conditions [3]. The penalty function is defined as a continuous scalar function on $\mathbb{R}^n$ that satisfies

$$p(\boldsymbol{x}) = 0, \quad \forall \boldsymbol{x} \in \mathbf{X},$$
$$p(\boldsymbol{x}) > 0, \quad \text{otherwise.}$$

(1.4)

A typical choice for the penalty function is, e.g.,

$$p(\boldsymbol{x}) = \sum_{j=1}^{m} \left(\max \left\{ g_j(\boldsymbol{x}), 0 \right\} \right)^2 + \sum_{j=1}^{q} h_j(\boldsymbol{x})^2.$$

(1.5)

Using the penalty function p, we construct the modified objective function

$$\phi(\boldsymbol{x}, r) = f(\boldsymbol{x}) + r p(\boldsymbol{x}),$$

(1.6)

where $r > 0$ is called the penalty coefficient [3]. From the definition of the penalty function, it can be seen that the values of the modified objective function $\phi(\boldsymbol{x}, r)$ differ from the original function f only for such $\boldsymbol{x}$ that violate the specified condition.

To solve the constrained problem, we iteratively construct a new term in an increasing sequence of penalty coefficients $r_1, r_2, \ldots$, for which we solve the unconstrained optimization problem for the modified function $\phi(\boldsymbol{x}, r_k)$. The optimal solution $\boldsymbol{x}_k$ of this unconstrained problem is then used as the starting point for the next iteration. This process is repeated until the condition $r_k p(\boldsymbol{x}_k) < \varepsilon$ for some $\varepsilon > 0$ is satisfied. When this condition is met, $\boldsymbol{x}_k$ can be considered a sufficiently accurate approximation of the solution to the constrained problem. It should be noted that penalty methods allow searching for an optimal solution outside the set of admissible solutions during the iterations [4], and thus are categorized as exterior point methods. A key assumption of penalty methods is that the domain of the problem satisfies $\mathbf{D} = \mathbb{R}^n$.

The construction of the sequence of penalty parameters and the principle of the penalty method are illustrated in Figure 1.1 on a trivial example of minimizing the function $f(x) = 0.5x$ with the constraint $g(x) = 4 - x \le 0$. The modified objective function in this case takes the form

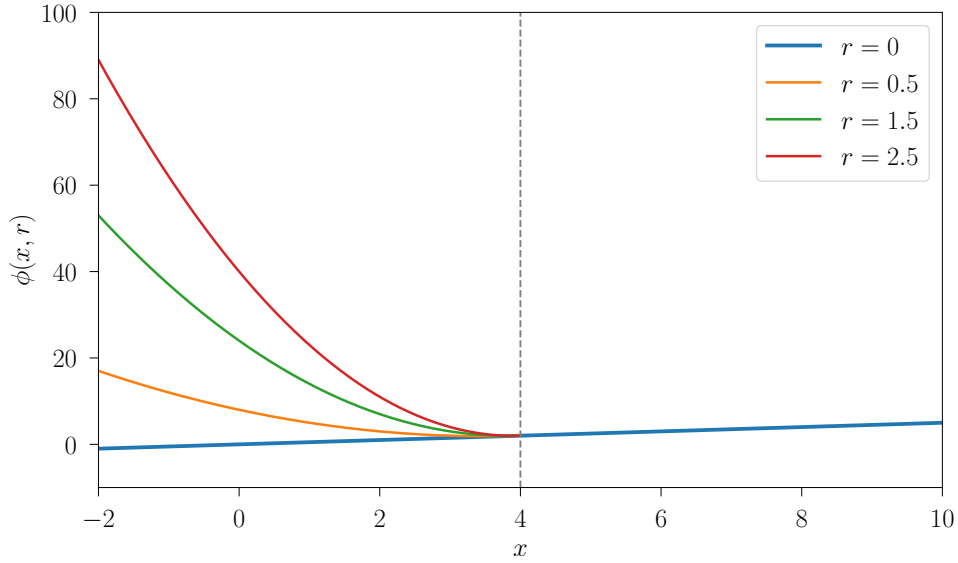$$\phi(x, r) = 0.5x + r \left(\max\{4 - x, 0\} \right)^2.$$

(1.7)

Figure 1.1: An illustration of the penalty method applied to minimizing the function $f(x) = 0.5x$ with the constraint $g(x) = 4 - x \leq 0$. Different shapes of the modified objective function $\phi(x, r)$ depending on the value of the penalty parameter r are distinguished by color. The condition defining the set of admissible solutions is indicated by the gray dashed line. The set of admissible solutions lies in the half-plane to the right of this gray dashed line.

1.2.2 Barrier method

The barrier method operates on a principle similar to that of the penalty method, but its main difference is that, during the iterative process of finding an optimal solution to a constrained problem, it ensures that the solution estimates always remain within the interior of the feasible set, which is defined as

$$\mathbf{X}^{\text{o}} = \left\{ \boldsymbol{x} \in \mathbf{D} \subseteq \mathbb{R}^n \,\middle|\, \boldsymbol{g}(\boldsymbol{x}) < \mathbf{0} \right\}. \tag{1.8}$$

Methods that satisfy this condition are generally referred to as interior-point methods [4].

As with penalty methods, we account for the conditions defining the feasible set by adding a new term to the objective function, which reflects the degree to which these conditions are violated. The barrier function B is a continuous scalar function on $\mathbf{X}^{\text{o}}$ that satisfies the condition

$$(\exists j \in \{1, 2, \ldots, m\})(\lim_{\substack{\boldsymbol{x} \to \boldsymbol{y} \\ \mathbf{X}^{\text{o}}}} g_j(\boldsymbol{x}) = 0) \Rightarrow \lim_{\substack{\boldsymbol{x} \to \boldsymbol{y} \\ \mathbf{X}^{\text{o}}}} B(\boldsymbol{x}) = +\infty. \tag{1.9}$$

A typical choice for the barrier function is the logarithmic barrier function

$$B(\boldsymbol{x}) = -\sum_{j=1}^{m} \ln\left(-g_j(\boldsymbol{x})\right), \tag{1.10}$$

or the reciprocal barrier function

$$B(\boldsymbol{x}) = -\sum_{j=1}^{m} \frac{1}{g_j(\boldsymbol{x})}. \tag{1.11}$$

Using the barrier function B, we construct the modified objective function

$$\phi(\boldsymbol{x}, r) = f(\boldsymbol{x}) + rB(\boldsymbol{x}), \tag{1.12}$$

where $r > 0$ [4].

To solve the constrained problem using the barrier method, at each iteration we construct a new term in a strictly decreasing sequence of positive parameters $r_1, r_2, \ldots$, for which we solve the unconstrained optimization problem for the modified function $\phi(\boldsymbol{x}, r_k)$. The vector $\boldsymbol{x}_k = \mathrm{argmin}_{\boldsymbol{x} \in \mathbf{X}^\circ}(f(\boldsymbol{x}) + r_k B(\boldsymbol{x}))$, obtained by optimizing the unconstrained problem, is used as the starting point for the next iteration. This process is repeated until the condition $r_k < \varepsilon$ for some $\varepsilon > 0$ is satisfied, at which point $\boldsymbol{x}_k$ is considered a sufficient approximation to the solution of the constrained problem [4].

The construction of the parameter sequence $(r_k)_{k \in \mathbb{N}}$ and the principle of the barrier method are illustrated in Figure 1.2 using the example of minimizing the function $f(x) = 0.5x$ with the constraint $g(x) = 4 - x \leq 0$ and the choice of the reciprocal barrier function. The modified objective function in this case is

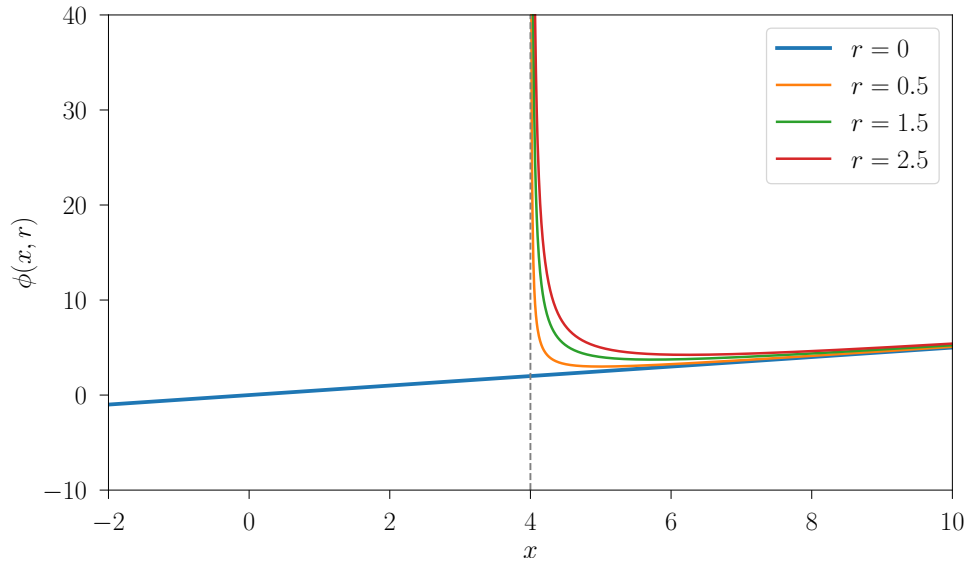$$\phi(x, r) = 0.5x - \frac{r}{4 - x}. \tag{1.13}$$

Figure 1.2: An illustration of the barrier method applied to minimizing the function $f(x) = 0.5x$ with the constraint $g(x) = 4 - x \leq 0$. Different shapes of the modified objective function $\phi(x, r)$ depending on the value of the barrier parameter r are distinguished by color. The condition defining the set of feasible solutions is indicated by the gray dashed line. The set of feasible solutions lies in the half-plane to the right of this gray dashed line.

Finally, we define a specific choice of the barrier function B_∞ called the extreme barrier function [5]. The extreme barrier function mirrors the asymptotic behavior of the aforementioned barrier functions and is given by

$$B_\infty(\boldsymbol{x}) = 0, \quad \forall \boldsymbol{x} \in \mathbf{X},$$
$$B_\infty(\boldsymbol{x}) = +\infty, \quad \text{otherwise.} \tag{1.14}$$

The modified objective function is then given by

$$f_\infty(\boldsymbol{x}) = f(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathbf{X},$$
$$f_\infty(\boldsymbol{x}) = +\infty, \quad \text{otherwise.} \tag{1.15}$$

1.3   Black-box optimization

In practice, we often need to optimize an objective function f whose exact form and derivative are unknown. This is common in numerical simulations, where the function can only be evaluated at specific points. Moreover, evaluating the function at a point may be difficult, time-consuming, or computationally expensive. As a result, standard optimization algorithms are not well-suited for these problems.

The discipline that deals with problems where the objective function (or constraints) is given by a so-called black-box[1], is called black-box optimization (hereafter referred to as BBO). In BBO, it is typically not assumed that the objective function is continuous or differentiable [5, 6, 7].

It is worth noting that black-box optimization is often confused with derivative-free optimization (DFO), which encompasses methods and techniques for objective functions whose derivatives are unknown or difficult to compute [5, 6, 8]. These two disciplines share many common characteristics, but they differ primarily in that, within DFO, the formula for calculating the derivative of the objective function may still be known. Furthermore, BBO includes heuristic methods, whereas DFO focuses mainly on methods that can be reliably analyzed mathematically in terms of convergence and stopping criteria, which is often not possible for BBO methods [5]. Therefore, although the terms BBO and DFO are often used interchangeably, in this work, we will treat them as two distinct disciplines [5].

Additionally, it should be noted that various classifications of methods within BBO can be found in the literature. In this work, we will adhere to the classification presented in [5], distinguishing between heuristic methods, direct search methods, and methods based on surrogate models. Each of these classes will be briefly described in this section.

1.3.1   Heuristic Methods

Heuristic optimization methods often rely on different predefined rules or even trial and error when seeking the solution of an optimization problem. These methods usually do not guarantee optimal solutions, but they are often effective for finding near-optimal results in a reasonable amount of time. Heuristic methods include genetic algorithms, detailed in [5], along with various other heuristic approaches.

In this section, however, we will focus on a different widely used heuristic method, the Nelder-Mead method, also known as the simplex method[2][3][10].

The Nelder-Mead method finds a solution to an optimization problem by iteratively constructing simplexes. The process begins by initializing a starting simplex. The objective function is then evaluated at each vertex of this simplex. In each subsequent iteration, the simplex is transformed in order to move closer to the position of the sought stationary point of the objective function. The transformation of the simplex involves manipulating its points using predefined operations – expansion, reflection, contraction (inner and outer), and shrinking, which are schematically illustrated in Figure 1.3.

---

[1]In programming, a black-box refers to a system whose internal mechanisms are unknown to the user. This means that the user generally has access only to the system's input and output [5].

[2]A simplex in $\mathbb{R}^n$ is defined as a bounded convex polytope (a generalization of a polyhedron to any dimension) with a non-empty interior and exactly $n + 1$ vertices [5].

[3]The term "simplex method" more commonly refers to the algorithm used to find the optimal solution in linear programming. This algorithm was developed by George Dantzig [9].

(a) initial simplex   (b) expansion   (c) reflection

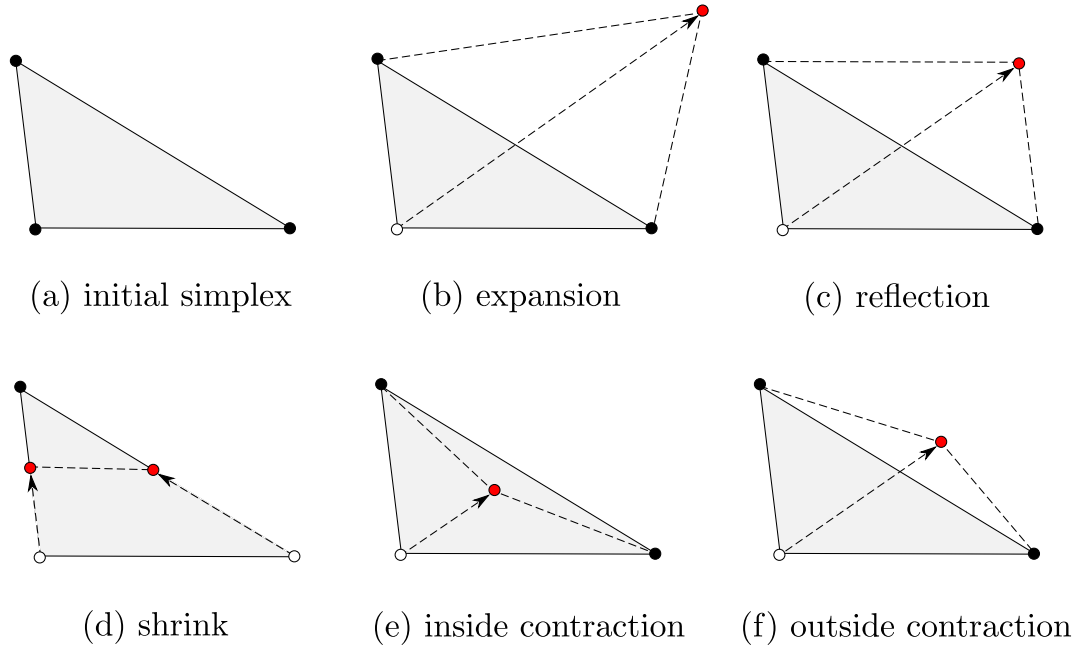(d) shrink   (e) inside contraction   (f) outside contraction

Figure 1.3: A schematic representation of the operations used to transform simplexes in the Nelder-Mead method. The vertices generated by applying each operation are shown in red. For clarity, the operations are depicted in $\mathbb{R}^2$.

The transformations performed during each iteration are determined by comparing the function values at the vertices of the simplex. The newly formed simplex shares either exactly one vertex or exactly n vertices with the simplex from the preceding iteration. The algorithm continues to iteratively transform the simplex until a stopping condition (specified by the user) is met [5]. Details of the Nelder-Mead method, including the algorithm's description and the choice of stopping condition, are discussed in [5, 6, 10].

The heuristic nature of the Nelder-Mead method stems from the fact that its principle is based on a somewhat random search of the space using predefined rules. Several iterations of space exploration using simplexes, for a specific choice of initial simplex and a specific function, are shown in Figure 1.4. While the convergence of this method has been proven, it is not guaranteed that the method will always converge to a stationary point [5]. It should be noted that the Nelder-Mead method was primarily developed for unconstrained optimization problems, but it can be adapted for constrained optimization problems using the techniques described in section **??**.

Algorithm 1 Nelder-Mead algorithm

Require: Initial simplex $Y^0 = \{y^0, y^1, \ldots, y^n\}$, function $f : \mathbb{R}^n \to \mathbb{R}$, parameters δ^e, δ^{oc}, δ^{ic}, γ, iteration counter $k \leftarrow 0$

Ensure: Approximate minimum of f

1: **procedure** NELDER-MEAD(Y^0)
2:      Reorder Y^k so that $f(y^0) \le f(y^1) \le \cdots \le f(y^n)$
3:      Set $f_{\text{best}}^k = f(y^0)$
4:      **while** stopping condition not met **do**

Reflection

5:          Compute centroid $x^c = \frac{1}{n} \sum_{i=0}^{n-1} y^i$
6:          Set reflection point $x^r = x^c + \delta^r(x^c - y^n)$
7:          Compute $f^r = f(x^r)$
8:          **if** $f_{\text{best}}^k \le f^r < f(y^{n-1})$ **then**
9:             Set $Y^{k+1} = \{y^0, y^1, \ldots, y^{n-1}, x^r\}$
10:         Increment $k \leftarrow k + 1$ and continue
11:         **end if**

Expansion

12:         **if** $f^r < f_{\text{best}}^k$ **then**
13:            Set expansion point $x^e = x^c + \delta^e(x^r - x^c)$
14:            Compute $f^e = f(x^e)$
15:            **if** $f^e < f^r$ **then**
16:              Set $Y^{k+1} = \{y^0, y^1, \ldots, y^{n-1}, x^e\}$
17:            **else**
18:              Set $Y^{k+1} = \{y^0, y^1, \ldots, y^{n-1}, x^r\}$
19:            **end if**
20:            Increment $k \leftarrow k + 1$ and continue
21:         **end if**

Contraction

22:         **if** $f^r \ge f(y^n)$ **then**
23:            **Outside Contraction:** Compute $x^{oc} = x^c + \delta^{oc}(x^c - y^n)$
24:            Compute $f^{oc} = f(x^{oc})$
25:            **if** $f^{oc} < f(y^n)$ **then**
26:              Set $Y^{k+1} = \{y^0, y^1, \ldots, y^{n-1}, x^{oc}\}$
27:            **else**
28:              **Shrink:** Set $Y^{k+1} = \{y^0, y^0 + \gamma(y^1 - y^0), \ldots, y^0 + \gamma(y^n - y^0)\}$
29:            **end if**
30:            Increment $k \leftarrow k + 1$ and continue
31:         **else**
32:            **Inside Contraction:** Compute $x^{ic} = x^c + \delta^{ic}(x^c - y^n)$
33:            Compute $f^{ic} = f(x^{ic})$
34:            **if** $f^{ic} < f(y^n)$ **then**
35:              Set $Y^{k+1} = \{y^0, y^1, \ldots, y^{n-1}, x^{ic}\}$
36:            **else**
37:              Set $Y^{k+1} = \{y^0, y^0 + \gamma(y^1 - y^0), \ldots, y^0 + \gamma(y^n - y^0)\}$
38:            **end if**
39:            Increment $k \leftarrow k + 1$ and continue
40:         **end if**
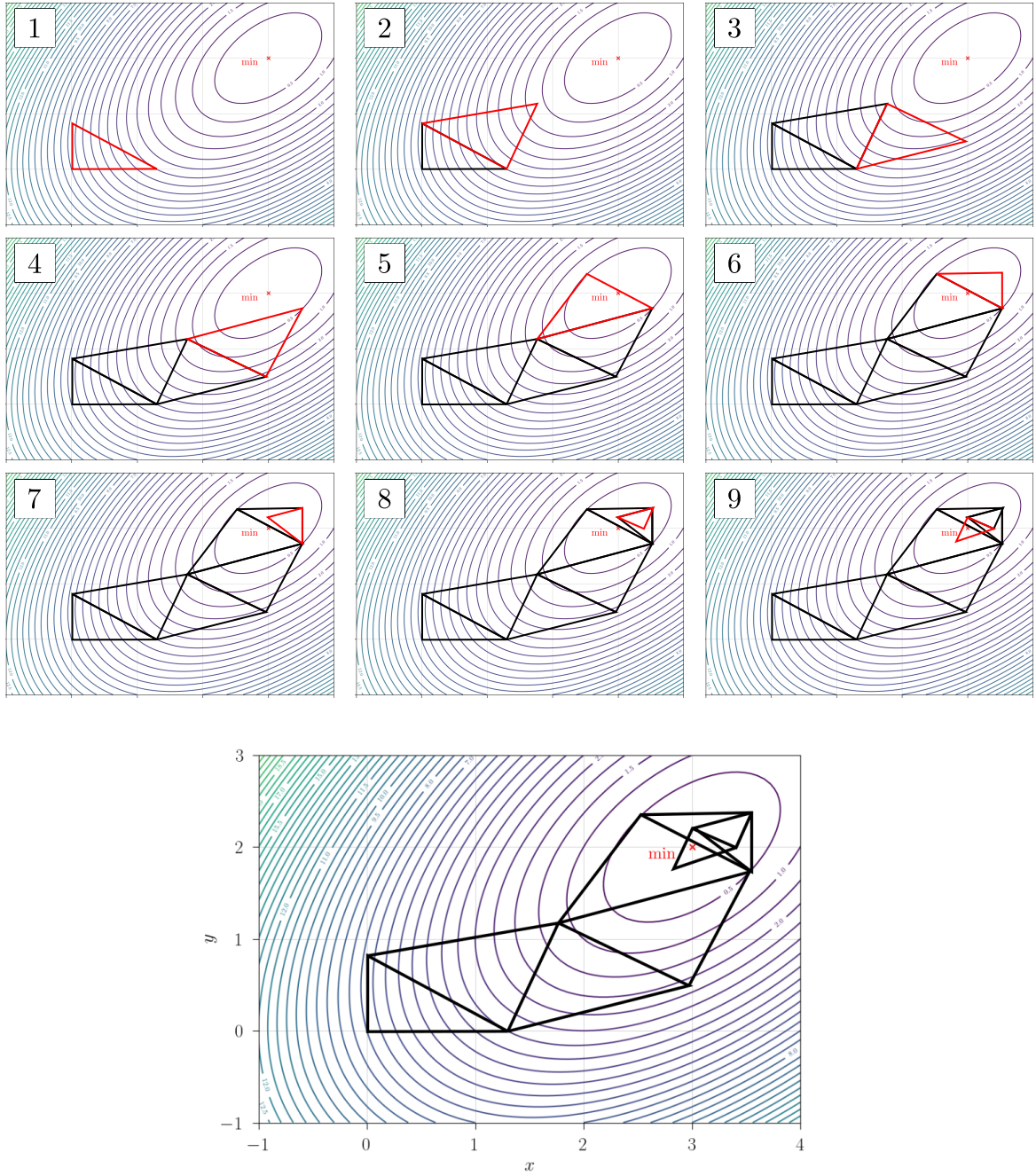41:   **end while**
42: **end procedure**

Figure 1.4: Several iterations of the Nelder-Mead method for a specific choice of the initial simplex when minimizing the function $x^2 - 4x + y^2 - y - xy + 7$, with the minimum at the point (3,2) marked by a red cross.

1.3.2 Direct search methods

Among direct search methods, we will describe the *generalized pattern search* (hereafter GPS) method [11] and the *mesh adaptive direct search* (hereafter MADS) method [12].

To describe the GPS algorithm, it is necessary to define a mesh, which is used to describe the search sets within the GPS algorithm. Let $\mathbf{G} \in \mathbb{R}^{n \times n}$ be invertible and $\mathbf{Z} \in \mathbb{Z}^{n \times p}$, $n, p \in \mathbb{N}$. Assume that every vector from $\mathbb{R}^n$ can be expressed as a linear combination of the columns of matrix $\mathbf{Z}$ (treated as vectors),

such that all the coefficients in this linear combination are non-negative. Furthermore, let $\mathbf{D} = \mathbf{GZ}$. The mesh $\mathbf{M}$ generated by $\mathbf{D}$ centered at point x is defined as

$$\mathbf{M} = \{x + \delta\,\mathbf{D}y \,|\, y \in \mathbb{N}^p\}, \tag{1.16}$$

where δ is called the mesh size parameter [5, 11]. In each iteration of the GPS algorithm, the shape of the mesh generally changes, as it is always centered at the point representing the best estimate in that iteration, and the size of the mesh step also changes. Let x_k and δ_k represent the estimate of the solution and the mesh size in the k-th iteration, respectively. We then define the mesh in the k-th iteration, denoted as $\mathbf{M}_k$, as

$$\mathbf{M}_k = \{x_k + \delta_k\,\mathbf{S}y \,|\, y \in \mathbb{N}^p\}. \tag{1.17}$$

Note that the columns of matrix $\mathbf{D}$, as defined above, can be interpreted as the possible directions in which the GPS algorithm searches the optimization space [5, 11]. Examples of search directions and meshes generated by different matrices $\mathbf{G}$ and $\mathbf{Z}$ are presented in Figure 1.5.
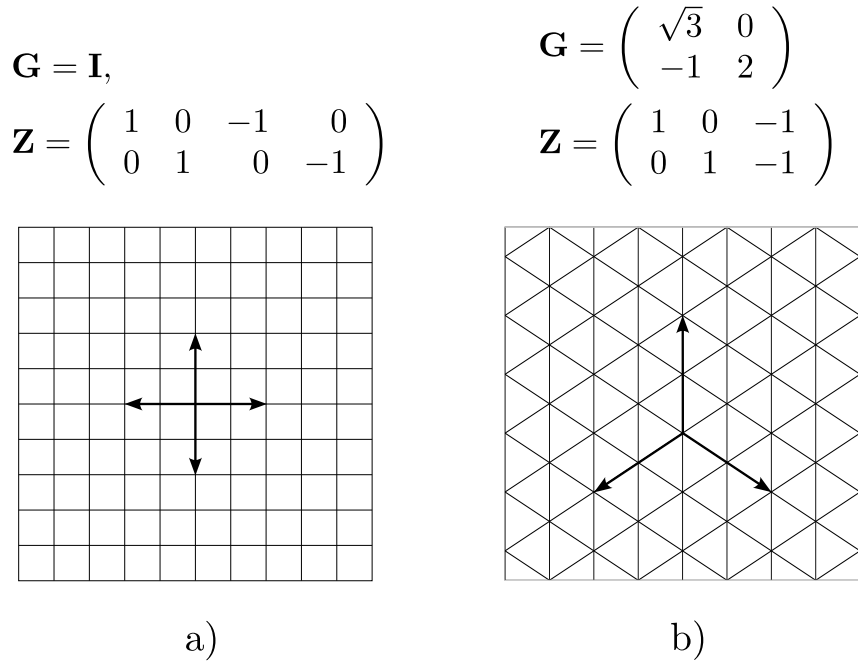
$$\mathbf{G} = \mathbf{I},$$

$$\mathbf{Z} = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}$$

$$\mathbf{G} = \begin{pmatrix} \sqrt{3} & 0 \\ -1 & 2 \end{pmatrix}$$

$$\mathbf{Z} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix}$$

a)   b)

Figure 1.5: Examples of search directions and meshes in $\mathbb{R}^2$ with obtained by different choices of $\mathbf{G}$ and $\mathbf{Z}$. The mesh points are at the intersections of the lines, the arrows represent possible search directions.

After initializing the necessary starting parameters, each iteration of the GPS algorithm is divided into two main steps. The first step is called the search step. During the search step, a finite set S_k of candidate mesh points, selected according to a strategy specified by the user, is evaluated by computing the objective function at each one of the points. If none of the evaluated points represents an improvement over the value $f(x_k)$, the poll step follows. In the poll step, the objective function is evaluated at all neighboring mesh points of x_k. If none of the evaluated points represents an improvement over the value $f(x_k)$, we set $x_{k+1} = x_k$ and decrease the mesh size, i.e., $\delta_{k+1} < \delta_k$. However, if a point that improves

the estimate of the solution is found in either the search or the poll step, this point is set as x_{k+1}, and the mesh size is increased, i.e., $\delta_{k+1} > \delta_k$ [5, 11].

The changes described above define a new mesh $\mathbf{M}_k$ in each iteration. The mesh changes throughout the GPS algorithm. The algorithm terminates when $\delta_{k+1} < \varepsilon$ for some user-specified $\varepsilon > 0$. It can be shown that the mesh step converges to zero, and under appropriate assumptions, the solution estimates converge to a stationary point of the objective function. Details can be found in [5]. It should be noted that the convergence of GPS has been proven for unconstrained problems [5]. The full GPS algorithm is presented is presented in Algorithm 2.

Algorithm 2 Generalized Pattern Search (GPS) for unconstrained optimization

---

Require: Function $f : \mathbb{R}^n \to \mathbb{R}$, initial point x^0, initial mesh size parameter δ^0, positive spanning matrix D, mesh size adjustment parameter $\tau \in (0, 1)$, stopping tolerance ϵ_{stop}, iteration counter $k \leftarrow 0$

Ensure: Approximate solution x^*

1: **procedure** GPS(x^0)
2:     **while** $\delta^k > \epsilon_{\text{stop}}$ **do**

---

1. Search

3:         Define a finite subset S^k of the mesh M^k
4:         **if** $f(t) < f(x^k)$ for some $t \in S^k$ **then**
5:             Set $x^{k+1} \leftarrow t$ and $\delta^{k+1} \leftarrow \tau^{-1}\delta^k$
6:             **continue**
7:         **else**
8:             Go to Poll step
9:         **end if**

---

2. Poll

10:         Select a positive spanning set $D^k \subseteq D$
11:         Define $P^k = \{x^k + \delta^k d : d \in D^k\}$
12:         **if** $f(t) < f(x^k)$ for some $t \in P^k$ **then**
13:             Set $x^{k+1} \leftarrow t$ and $\delta^{k+1} \leftarrow \tau^{-1}\delta^k$
14:         **else**
15:             x^k is a mesh local optimizer
16:             Set $x^{k+1} \leftarrow x^k$ and $\delta^{k+1} \leftarrow \tau\delta^k$
17:         **end if**

---

3. Termination

18:         **if** $\delta^{k+1} \leq \epsilon_{\text{stop}}$ **then**
19:             **terminate**
20:         **else**
21:             Increment $k \leftarrow k + 1$ and continue
22:         **end if**
23:     **end while**
24: **end procedure**

We will also briefly mention the MADS method, which represents an improvement over the GPS method. Unlike GPS, the MADS algorithm allows the exploration of the objective function's values in directions that form a dense subset in $\mathbb{R}^n$ during the poll step [5, 6]. This generalization improves the

convergence of the algorithm and allows for proving the convergence of MADS even for constrained problems, where the objective function is modified into an extreme barrier function 1.15. Details regarding the MADS algorithm and its operation can be found, for example, in [5].
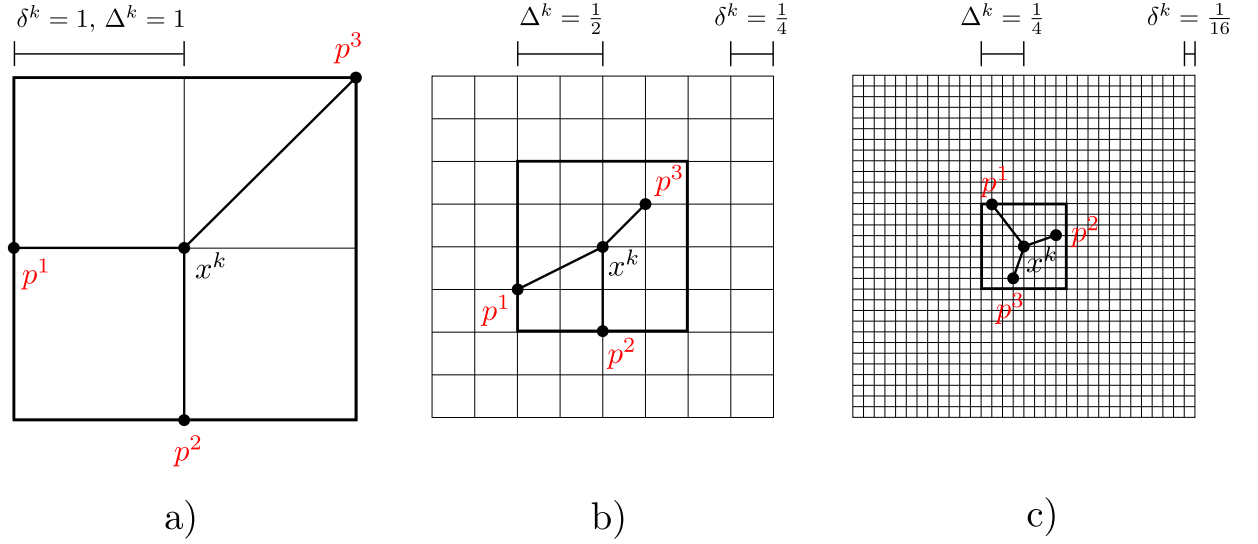

Figure 1.6: Examples of meshes and frames $\mathbb{R}^2$ for different values of δ^k and Δ^k

---
Algorithm 3 Mesh Adaptive Direct Search (MADS)

Require: Function $f_\Omega : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$, initial point $x^0 \in \Omega$, initial frame size parameter Δ^0, positive spanning matrix D, mesh size adjustment parameter $\tau \in (0, 1)$, stopping tolerance ϵ_{stop}, iteration counter $k \leftarrow 0$

Ensure: Approximate solution x^*
 1: **procedure** MADS(x^0)
 2:     **while** $\Delta^k > \epsilon_{\text{stop}}$ **do**
---
1. Parameter Update

 3:         Set the mesh size parameter $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$
---
2. Search

 4:         Define a finite set $S^k \subset M^k$ such that:
 5:         **if** $f_\Omega(t) < f_\Omega(x^k)$ for some $t \in S^k$ **then**
 6:             Set $x^{k+1} \leftarrow t$ and $\Delta^{k+1} \leftarrow \tau^{-1}\Delta^k$
 7:             **continue**
 8:         **else**
 9:             Go to Poll step
10:         **end if**
---
3. Poll

11:         Select a positive spanning set D_{Δ^k} and define:
12:         $P^k = \{x^k + \delta^k d : d \in D_{\Delta^k}\}$, a subset of the frame F^k with extent Δ^k
13:         **if** $f_\Omega(t) < f_\Omega(x^k)$ for some $t \in P^k$ **then**
14:             Set $x^{k+1} \leftarrow t$ and $\Delta^{k+1} \leftarrow \tau^{-1}\Delta^k$
15:         **else**
16:             Set $x^{k+1} \leftarrow x^k$ and $\Delta^{k+1} \leftarrow \tau\Delta^k$
17:         **end if**
---
4. Termination

18:         **if** $\Delta^{k+1} \leq \epsilon_{\text{stop}}$ **then**
19:             **terminate**
20:         **else**
21:             Increment $k \leftarrow k + 1$ and continue
22:         **end if**
23:     **end while**
24: **end procedure**

1.3.3 Optimization Using a Surrogate Model

In cases where evaluating the objective function at a specific point is time-consuming or computationally expensive, it can be useful to employ a surrogate for the objective function during optimization. We define a surrogate model of the given problem as the problem

$$\min_{x \in \tilde{\mathbf{X}}} \tilde{f}(x), \tag{1.18}$$

where

$$\tilde{\mathbf{X}} = \left\{ x \in \mathbf{D} \subseteq \mathbb{R}^n \,\middle|\, \tilde{g}(x) \leq 0 \wedge \tilde{h}(x) = 0 \right\}, \tag{1.19}$$

and the functions $\tilde{f}, \tilde{g}$, and $\tilde{h}$ have characteristics similar to those of the functions f, g, and h in the original problem. The characteristics of $\tilde{f}, \tilde{g}$, and $\tilde{h}$ are intentionally left undefined, reflecting the fact that the surrogate model does not necessarily need to be an accurate approximation of the original problem [7, 5, 8]. A good approximative model may not always be a suitable surrogate for optimization purposes, a situation illustrated in Fig. 1.7.
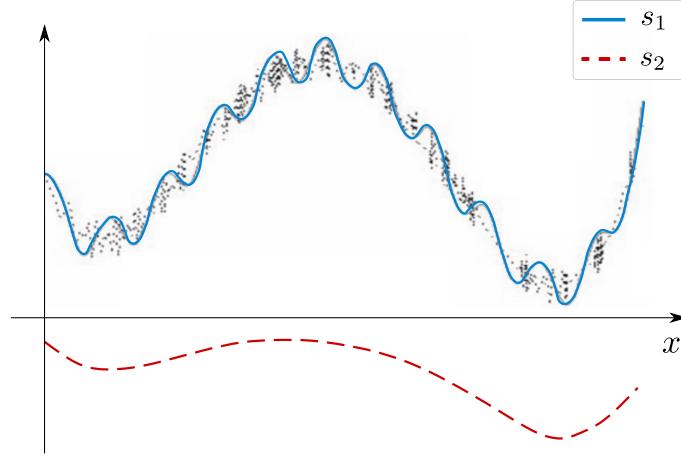

Figure 1.7: An illustration of two surrogate models, s_1 and s_2. The black points represent noisy values of the objective function. While using surrogate model s_1 represents a better choice for approximating the function, it is not suitable for optimization since s_1 contains many undesirable stationary points that the original objective function does not have. On the other hand, while surrogate model s_2 is not as accurate in approximating the function's values, it is a better choice for optimization because the stationary points of s_2 are almost identical to those of the optimized objective function.

Using a surrogate model in optimization is often part of a larger optimization method. Surrogate models can, for example, be used within GPS and MADS methods described in section 1.3.2, where, during the exploration step, we first evaluate the surrogate function $\tilde{f}$ at the same points, sort these values, and then use the sorted set of points to evaluate the original function f. This potentially allows us to significantly reduce the time required to complete the exploration step, as sorting the points increases the probability of finding a better estimate of the solution at one of the first examined points [5]. Surrogate models can also be used within other methods to accelerate the process, and their application is discussed in detail in [7, 5].

Bibliography

[1] R. Fletcher and M. J. D. Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, 6(2):163–168, August 1963.

[2] C. G. BROYDEN. The convergence of a class of double-rank minimization algorithms. *IMA Journal of Applied Mathematics*, 6(3):222–231, 1970.

[3] Dimitri Bertsekas. *Nonlinear programming*. Athena Scientific, September 2016.

[4] David G Luenberger and Yinyu Ye. *Linear and nonlinear programming*. International series in operations research & management science. Springer, New York, NY, 3 edition, July 2008.

[5] Charles Audet and Warren Hare. *Derivative-free and blackbox optimization*. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, Cham, Switzerland, 1 edition, December 2017.

[6] Jeffrey Larson, Matt Menickelly, and Stefan M. Wild. Derivative-free optimization methods. *Acta Numerica*, 28:287–404, May 2019.

[7] Stéphane Alarie, Charles Audet, Aïmen E. Gheribi, Michael Kokkolaras, and Sébastien Le Digabel. Two decades of blackbox optimization applications. *EURO Journal on Computational Optimization*, 9:100011, 2021.

[8] Oliver Kramer, David Echeverría Ciaurri, and Slawomir Koziel. Derivative-free optimization. In *Computational Optimization, Methods and Algorithms*, pages 61–83. Springer Berlin Heidelberg, 2011.

[9] George B. Dantzig. Origins of the simplex method, June 1990.

[10] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, January 1965.

[11] Charles Audet and J. E. Dennis. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, January 2002.

[12] Charles Audet and J. E. Dennis. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(1):188–217, January 2006.