POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

BOSCH

In collaboration with
Robert Bosch GmbH Corporate Research

Global optimization of pulse patterns for an electrical drive via Set Membership methods

Tesi di Laurea Magistrale in
Automation and Control Engineering - Ingegneria dell'Automazione

Authors: **Giulio Montecchio & Mattia Alborghetti**

Student ID: 960856, 953649
Advisor: Prof. Lorenzo Fagiano
Co-advisors: Stefan Gering, Maximilian Manderla, Martin Loehning
Academic Year: 2022-23

Acknowledgements

We would like to express our deepest gratitude to Professor Lorenzo Fagiano for believing in our passion, and for the precious advice that he provided in this project.

We could not have undertaken this journey without Stefan, Max, and Martin. Major thanks to you for choosing us and guiding us throughout this mesmerizing experience in Bosch. Together with the entire PACE team, they provided us with precious support and opened a fascinating window into the world of corporate research. Thanks should also go to all the students in the research center: Salim, Alessandro, Matteo, Leonardo, Lena, Annabelle, Ranbir, Dominik, Till, Kenzo, Daniel and Weibin. Sharing ideas, insights on different projects, free and hard time made the working environment stimulating and exciting, but also pleasant and relaxed. Our stay in Germany really felt like home.

Abstract

The optimization of pulse patterns is a crucial problem in the field of modulation techniques in electrical drives. This problem is challenging due to its high dimensionality and non-convexity. Set Membership Global Optimization (SMGO) is an innovative optimization method that is characterized by its global nature. Its functioning is black-box and data-driven, hence an explicit model of the cost function is unnecessary.

This work explores the application of SMGO to the optimization of pulse patterns. The objective of the optimization consists in the minimization of the distortion of the currents flowing in the coils of the electric motor.
To enhance the performance of SMGO, modifications are introduced, including a novel trust region strategy, an adaptive tuning in the exploration-exploitation trade-off, and a smart generation of candidate points. These changes aim to improve the efficiency and robustness of the SMGO method, in its most general application.

After conducting a thorough analysis, it is found that the state-of-the-art method, employing gradient-based optimization and randomized multiple starting points, is a better solution to obtain an Optimal Pulse Pattern. However, the thesis concludes that SMGO is still an effective method and could be useful for similar problems, especially when non-differentiable models are taken into account.

Moreover, the developed modifications to the algorithm are found to be effective in several benchmarks, thus ultimately improving the global optimizer.

Keywords: Optimization, Global, Set membership, SMGO, Black-box, Data-driven, Modulation, Optimal Pulse Pattern, OPP, Electrical drive.

Abstract in lingua italiana

L'ottimizzazione dei pulse patterns (noti anche come angoli memorizzati) è un problema cruciale nell'ambito delle tecniche di modulazione degli azionamenti elettrici. Questa ottimizzazione è particolarmente impegnativa, a causa della elevata dimensionalità e non convessità della specifica applicazione. Il Set Membership Global Optimization (SMGO) è un algoritmo di ottimizzazione innovativo che si caratterizza per la sua natura globale. Il suo funzionamento è black-box e basato sui dati, quindi indipendente dalla formulazione della funzione di costo.

Questa tesi esplora l'applicazione di SMGO per l'ottimizzazione dei pulse patterns.
L'obiettivo considerato in questo lavoro consiste nella minimizzazione della distorsione delle correnti che scorrono nelle bobine del motore elettrico.
Per migliorare le prestazioni di SMGO, sono state introdotte diverse modifiche, tra cui una nuova strategia per la regione di fiducia, una regolazione adattiva del trade-off tra sfuttamento-esplorazione e una generazione intelligente di punti candidati. Questi cambiamenti mirano a migliorare l'efficienza e la robustezza di SMGO nella sua implementazione più generale.

Dopo aver condotto un'analisi approfondita, è emerso che il metodo correntemente usato in industria, che consiste nell'utilizzo di un'ottimizzazione basata sul gradiente con molteplici inizializzazioni randomiche, è una soluzione migliore per ottenere dei pulse patterns ottimali. Tuttavia, la tesi conclude che SMGO è comunque un metodo efficace e potrebbe essere utile nello stesso ambito, specialmente se vengono integrati al problema nuovi fattori, come l'introduzione di un modello non differenziabile, in grado di rendere i metodi correnti inutilizzabili.

In aggiunta, le modifiche implementate all'algoritmo sono risultate efficaci su numerose funzioni di benchmark, comprovando un generale miglioramento del suddetto solver globale.

Parole chiave: Ottimizzazione, Globale, Set Membership, SMGO, Pulse pattern, Black-box, OPP, Pulse pattern ottimale, Angoli memorizzati, Azionamenti elettrici.

Contents

1 | Context and introduction

The technological advancement and the commitment to decarbonization of recent years brought new challenges of electromobility into focus. Among them, increasingly tight requirements on efficiency play a crucial role. In addition to optimal electric machine design, it is fundamental to apply an optimal control technique. The components and systems which convert electricity into power in the drive system of an electric vehicle are known as electric drive.

Electric drives have been extensively studied for decades, and they often involve the use of a phase current controller. This controller operates in continuous time and outputs voltage signals that are then translated into a series of voltage steps, by means of a certain modulation scheme. The actuation of these voltage steps is finally exerted through the inverter switches. However, traditional modulations are not considered to be optimal in terms of minimizing losses and maximizing efficiency across the entire operating range of the machine.

Optimal Pulse Pattern (OPP) is an advanced control technique that entails a synchronous modulation scheme; the sequence of switching instants is fixed on each motor revolution. The use of OPP in the automotive field has significant potential. By implementing these patterns, losses in the WLTC (*Worldwide harmonized Light vehicles Test Cycle*) can be greatly reduced. Additionally, OPP provides a flexible modulation method that can be used across a wide range of operating conditions, eliminating the need to switch between multiple modulation methods.

The optimal control problem of determining an optimal pulse pattern is a non-convex global one, and its formulation and solution is topic of research. In this thesis, we approach the problem with the *Set Membership Global Optimization* (SMGO), an innovative method characterized by its data-driven, global, and model-free nature. The lack of dependence on explicit mathematical models makes it a powerful tool for real-world applications, where the objective function is absent or too complex to be explicitly employed in optimization.

1.1. State of the art

Among modulation techniques, OPP is a well established method in industrial applications with constant speed. It is used to achieve very low current distortions during steady-state operation of medium-voltage applications. However, recently OPP has grasped the interest of the automotive industry. In this field, the most commonly used control methods are the *Direct Torque Control* (DTC) and *Field Oriented Control* (FOC) in conjunction with asynchronous *Pulse Width Modulation* (PWM) [5]. These methods can guarantee good dynamic performance, which is vital for the electric motors of vehicles; however, they are not intended to be optimal in terms of loss. The advantages that can be obtained through OPP have been already presented in [1]. In [11] it is shown how the implementation of a Model Predictive Controller that interpolates optimal pulse patterns across the operating region can provide good dynamic performances. A comparison between this *Model Predictive Pulse Pattern Control* (MP3C) and established methods is offered also in [10]. In [5] the combination of OPP with different modulation techniques in an adaptive PWM control shows good efficiency. Similar results are confirmed also by [15], in a different setting.
In all these works, OPP controllers involve the offline computation of an optimal pulse pattern. The optimal pulse pattern is different at each operating point and it is obtained as the solution of a challenging optimization problem. It is a constrained optimization with a multi-objective, non-convex and black-box cost function. This cost function is characterized by many local minima and it could be expensive in terms of time. At the state of the art, the optimal pulse pattern problem is solved through traditional methods, namely a multistart gradient-based approach. With this approach a local gradient-based optimization is initialized in different regions of the search space, approximating the value of the gradient with the finite difference method. However, this method requires a high number of function evaluations; these function evaluations are not fully utilized, since they do not lead to the choice of the next initialization.

In this thesis, the optimization problem is tackled with *Set Membership Global Optimization*, which is anticipated to be an efficient strategy for data-driven optimizations [22, 23]. SMGO is designed to find the global optimum of a function without requiring any knowledge of the function's derivatives. It is particularly useful in situations where the function is complex or expensive to evaluate. In [21, 22], the method is contextualized in the global optimization framework. Summarizing [21], the approaches to similar problems are divided into five different classes: swarm-based, generation-based, direct search, model-based and Lipschitz-based. SMGO belongs to the last class. This class of

methods is composed by sequential algorithms that make use of the information provided by an estimate of the Lipschitz constant of the cost function in order to choose the next sampling point. The lack of dependence on explicit mathematical models makes them a powerful tool for real-world applications, where the objective function may be poorly understood or difficult to model [24]. This method applies well to the problem at hand for different reasons:

1. The cost function is highly non-convex and contains many local minima; the quest for the global one is crucial.
2. It can deal with a medium/high number of optimization variables.
3. Since the optimization is run offline, memory issues can be neglected.
4. The presence of different tuning parameters allows wise use of the budget of function evaluations.
5. It can be easily upgraded to real-time applications.
6. It can deal with non-smooth cost functions.
7. It can be used for problems with expensive cost functions.

Since it is a newly developed optimization algorithm, the application of SMGO algorithm is still in a flexible, experimental phase. Some of its components can be adapted to better suit the specific problem. As a consequence, the application to the OPP problem can result in valuable insights into the algorithm's mechanics. The findings of this study have the potential to enhance and refine the optimization process.

1.2. Contribution of the thesis

The objective of the thesis is twofold. On the one hand, the application of SMGO offers a new global solution for OPP computation. On the other hand, it allows us to assess the performance of this method on an active industrial problem. Moreover, it is customary for the OPP to be solved for different numbers of optimization variables [1]. Therefore, this study offers insight into the quality of SMGO on different dimensions, without requiring a reformulation of the main structure of the problem.
The analysis that we conduct on the combination of SMGO with the OPP problem leads to the development of new components of the algorithm. Such components enhance the overall behavior of SMGO.

The thesis is structured as follows:

- **Chapter 2**: We introduce the general theory behind the model of the motor, the electric drive and the pulse pattern modulation technique. We then formalize the optimal problem at hand. We define the optimization variables, the cost function and the constraints involved.
- **Chapter 3**: We introduce SMGO, with an overview of both the theoretical principle and the practical implementation.
- **Chapter 4**: We extend and modify SMGO to account for linear constraints. Thanks to the innovative modifications adopted in this chapter, SMGO can be profitably applied to the class of pulse pattern optimization problems.
- **Chapter 5**: The main contribution to SMGO provided by this thesis is presented here. We introduce some changes to the method that consistently improve its performance. The advances provided by these alternative mechanisms are commented especially for the OPP case, but proved to be effective also on a set of 14 different benchmark functions. The results of this support study are reported in Appendix A.
- **Chapter 6**: We offer a novel study on the application of SMGO to the OPP problem, and we compare its results and performance with those of two commonly used optimization methods. This comparison provides a deeper understanding of the strengths and weaknesses of SMGO with respect to state-of-the-art methods.
- **Chapter 7**: We collect here the main conclusions of this study and gather the most promising outlooks.

2 | Model of the control problem

This chapter describes the mathematical model of the optimization under study, i.e. the control of an electrical drive. An electrical drive typically consists of three main components: the power supply, the electric motor, and the control system. While the power supply is out of the scope of this thesis, the motor and the control system need to be properly described, to allow the formulation of the OPP control problem.

In Section 2.1 we introduce the dynamical model of the electric machine, with further insight into the simulation technique that is implemented. In Section 2.2 we offer a brief introduction to the control system, with a focus on the pulse pattern technique in an inverter synchronous modulation scheme. In Section 2.3 we report the results of the simulation of the complete model. Finally, in Section 2.4 the optimal control problem behind the computation of an OPP is formulated.

2.1. PMSM machine

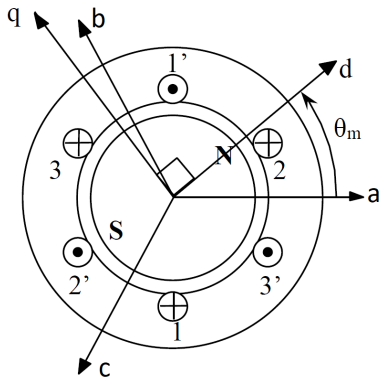


Figure 2.1: Diagram of a brushless motor.

The term *Permanent Magnet Synchronous Machine* (PMSM) refers to an Alternating Current (AC) electric machine constituted by a cylindric stator with symmetric three-phase windings and a rotor provided with a permanent magnet, that generates the rotating

magnetic flux. In the past, the scope of these drives was mainly low-power applications (e.g. drone propellers, CD players, ...) due to the limitations imposed by the high cost and size of the permanent magnets. However, given the high reliability and the ideal absence of rotor losses, their importance is growing in many fields like numerically controlled machine tools, industrial automation, robotics, light traction, heavy traction, and wind generation [4]. Compared to other machines that present coils on the rotor to generate the magnetic flux, PMSMs have a higher specific power and lower inertia due to the simpler rotor structure.

2.1.1. Fundamentals

Let $t \in \mathbb{R}$ be the continuous time variable. Variables $v_a(t)$, $v_b(t)$, $v_c(t)$ and $i_a(t)$, $i_b(t)$, $i_c(t)$ represent respectively voltages and currents applied to the stator windings.
The stator equations of a PMSM are

$$\begin{cases} v_a(t) = R_a i_a(t) + \dfrac{d\Psi_a(t)}{dt} & \text{(2.1a)} \\ v_b(t) = R_b i_b(t) + \dfrac{d\Psi_b(t)}{dt} & \text{(2.1b)} \\ v_c(t) = R_c i_c(t) + \dfrac{d\Psi_c(t)}{dt}. & \text{(2.1c)} \end{cases}$$

Variables Ψ_a, Ψ_b, Ψ_c are the magnetic fluxes concatenated to each coil.
In this thesis, we assume the stator to be symmetric, so that the phase resistances R_a, R_b, R_c are identical for each coil. Moreover, we limit our study to the steady-state condition of a single operating point. For this reason, all the resistances are set equal to the same constant R_s (where s stands for *stator*).
The current-flux relationship, in the case of a general anisotropic PMSM, are

$$\begin{cases} \Psi_a(t) = L_{\text{ss}}(\theta_{\text{m}})\, i_a + M_{\text{ss}}(\theta_{\text{m}})\, i_b + M_{\text{ss}}\left(\theta_{\text{m}} - \dfrac{2}{3}\pi\right) i_c \\ \qquad\quad + \psi_{\text{pm}}(\theta_{\text{m}}) & \text{(2.2a)} \\ \Psi_b(t) = L_{\text{ss}}\left(\theta_{\text{m}} - \dfrac{2}{3}\pi\right) i_b + M_{\text{ss}}(\theta_{\text{m}})\, i_a + M_{\text{ss}}\left(\theta_{\text{m}} + \dfrac{2}{3}\pi\right) i_c \\ \qquad\quad + \psi_{\text{pm}}\left(\theta_{\text{m}} - \dfrac{2}{3}\pi\right) & \text{(2.2b)} \\ \Psi_c(t) = L_{\text{ss}}\left(\theta_{\text{m}} - \dfrac{4}{3}\pi\right) i_c + M_{\text{ss}}\left(\theta_{\text{m}} - \dfrac{2}{3}\pi\right) i_a + M_{ss}\left(\theta_{\text{m}} + \dfrac{2}{3}\pi\right) i_b \\ \qquad\quad + \psi_{\text{pm}}\left(\theta_{\text{m}} - \dfrac{4}{3}\pi\right), & \text{(2.2c)} \end{cases}$$

where $\theta_{\mathrm{m}}(t)$ is the rotor position, L_{ss} is the auto inductance and M_{ss} is the mutual inductance.
The flux linked with the stator windings varies accordingly to:

$$\Psi_{\mathrm{pm}}(\theta_{\mathrm{m}}(t)) = \hat{\Psi}_{\mathrm{pm}} cos(\theta_{\mathrm{m}}(t)). \tag{2.3}$$

It is sinusoidal with respect to the rotor position $\theta_{\mathrm{m}}(t)$ and its amplitude is given by the flux of the permanent magnet $\hat{\Psi}_{\mathrm{pm}}$.

For analysis and control of the electric motors, it is standard to convert a balanced **three-phase** winding machine into a fictitious one with **two-phase** quadrature electric quantities, represented in Figure 2.1. This conversion involves the so-called *Park and Clarke transformation*, also known as *Space Phasor Formula* [20]. There are two different conventions of this transformation: *amplitude/magnitude invariant* or *power invariant*. In this work, an amplitude invariant convention is adopted, since it is more common in industrial practice. Therefore, from the combination of equations (2.1), (2.2) and (2.3), one can derive the simpler model 2.4, as presented in [4, 17]. It is conventional to characterize the two-phase reference system with the letters d and q, with the axis d aligned and **rotating** with the North of the permanent magnets. This gives the name to the well-known *dq-model* of the machine described below:

$$\begin{cases} v_d(t) = R_s i_d(t) + L_d \dot{i}_d(t) - \omega_s L_q i_q(t) & \text{(2.4a)} \\ v_q(t) = R_s i_q(t) + L_q \dot{i}_q(t) + \omega_s \left(L_d i_d(t) + \hat{\Psi}_{\mathrm{pm}} \right) & \text{(2.4b)} \\ \Psi_d(t) = L_d i_d(t) + \hat{\Psi}_{\mathrm{pm}} & \text{(2.4c)} \\ \Psi_q(t) = L_q i_q(t), & \text{(2.4d)} \end{cases}$$

where the *dot* operator indicates the time derivative operation and ω_s is the electrical speed, which does not depend on time, according to the steady-state hypothesis.
The winding d with magnetic axis in the direction d is crossed by the current i_d and supports the flux Ψ_d through a self-inductance L_d, that is constant and different from the self-inductance L_q of the winding q, because the PMSM under study is not isotropic. A remarkable advantage of the dq-model is that the inductances are no longer functions of the mechanical angle.
From the power balance of the machine, completely described in [4], the expression

$$T_e(t) = \frac{3}{2} N_p \left[(L_d - L_q)\, i_d(t) i_q(t) + \hat{\Psi}_{\mathrm{pm}} i_q(t) \right] \tag{2.5}$$

of the electrical torque can be retrieved. The value N_p defines the number of pole pairs of the machine and the coefficient 3/2 is due to the amplitude invariant transformation.

2.1.2. Dynamical model

Equations (2.4) are rearranged to obtain the state space representation of the system:

$$\begin{cases} \dot{\Psi}_d(t) = -\dfrac{R_s}{L_d}\Psi_d(t) + \omega\Psi_q(t) + v_d(t) + \dfrac{R_s}{L_d}\hat{\Psi}_{\text{pm}} & \text{(2.6a)} \\ \dot{\Psi}_q(t) = -\omega\Psi_d(t) - \dfrac{R_s}{L_q}\Psi_q(t) + v_q(t) & \text{(2.6b)} \\ i_d(t) = \dfrac{1}{L_d}\Psi_d(t) - \dfrac{1}{L_d}\hat{\Psi}_{\text{pm}} & \text{(2.6c)} \\ i_q(t) = \dfrac{1}{L_q}\Psi_q(t). & \text{(2.6d)} \end{cases}$$

We define the state vector $\boldsymbol{\xi}$, the input vector $\boldsymbol{u}$ and the output vector $\boldsymbol{y}$:

$$\boldsymbol{\xi}(t) = \begin{bmatrix} \Psi_d(t) \\ \Psi_q(t) \end{bmatrix} \quad \boldsymbol{u}(t) = \begin{bmatrix} u_d(t) \\ u_q(t) \\ \hat{\Psi}_{\text{pm}} \end{bmatrix} \quad \boldsymbol{y}(t) = \begin{bmatrix} i_d(t) \\ i_q(t) \end{bmatrix}. \tag{2.7}$$

The state space model of the system can now be written as

$$\begin{cases} \dot{\boldsymbol{\xi}}(t) = A\boldsymbol{\xi}(t) + B\boldsymbol{u}(t) & \text{(2.8a)} \\ \boldsymbol{y}(t) = C\boldsymbol{\xi}(t) + D\boldsymbol{u}(t), & \text{(2.8b)} \end{cases}$$

with the matrices

$$A = \begin{bmatrix} -R_s/L_d & \omega \\ -\omega & -R_s/L_d \end{bmatrix} \tag{2.9}$$

$$B = \begin{bmatrix} 1 & 0 & R_s/L_d \\ 0 & 1 & 0 \end{bmatrix} \tag{2.10}$$

$$C = \begin{bmatrix} 1/L_d & 0 \\ 0 & 1/L_q \end{bmatrix} \tag{2.11}$$

$$D = \begin{bmatrix} 0 & 0 & -1/L_d \\ 0 & 0 & 0 \end{bmatrix} \tag{2.12}$$

The system turns out to be a second-order linear time-invariant system.

The input-output relationship $G(s)$ is obtained as

$$G(s) = C(sI - A)^{-1}B + D, \tag{2.13}$$

where s is the *Laplace operator* [2]. We can write

$$y(t) = G(s)u(t), \tag{2.14}$$

where the transfer function is

$$G(s) = \begin{bmatrix} g_{11}(s) & g_{12}(s) & g_{13}(s) \\ g_{21}(s) & g_{22}(s) & g_{23}(s) \end{bmatrix} \tag{2.15}$$

and g_{ik} is the scalar transfer function from input k to output i.

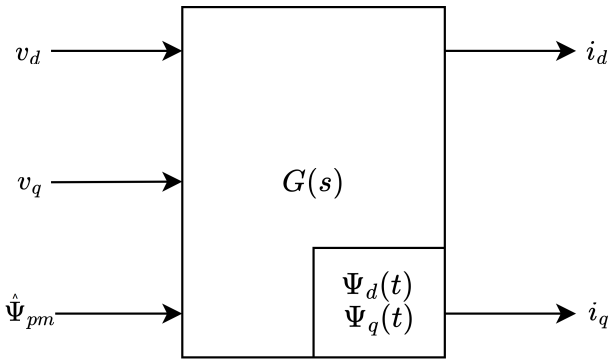


Figure 2.2: Schematic diagram of input-output relationship.

2.1.3. Multivariable frequency response

The steady-state condition allows to easily use the input-output relationship to simulate the dynamical model. Compared to the numerical integration, the frequency analysis shortens the computational time, avoiding the transient simulation of the state variables.

The *fundamental theorem of the frequency response* can be extended in a multivariable case as follows [25]. Replacing the Laplace operator s with $j\omega$, $g_{ik}(j\omega)$ represents the sinusoidal response from input k to output i. To be more specific, imagine that we apply to input channel k a scalar sinusoidal signal given by

$$u_k(t) = u_{k0} \sin(\omega t + \alpha_k). \tag{2.16}$$

If the input signal is persistent, then the corresponding persistent output signal in channel

i is also a sinusoid with the same frequency

$$y_i(t) = y_{i0} \sin(\omega t + \beta_i). \tag{2.17}$$

The amplification (gain) and the phase shift may be obtained from the complex number $g_{ik}(j\omega)$ as follows

$$\frac{y_{i0}}{u_{k0}} = |g_{ik}(j\omega)|, \tag{2.18}$$

$$\beta_i - \alpha_k = \angle g_{ik}(j\omega). \tag{2.19}$$

In phasor notation, using Euler's formula for complex numbers, we may compactly represent the sinusoidal time response described in (2.16) - (2.18) by

$$y_i(\omega) = g_{ik}(j\omega)u_k(\omega), \tag{2.20}$$

where

$$u_k(\omega) = u_{k0}e^{j\alpha_k} \tag{2.21}$$

and

$$y_i(\omega) = y_{i0}e^{j\beta_i}. \tag{2.22}$$

Here, the use of ω (and not $j\omega$) as the argument of $u_k(\omega)$ and $y_i(\omega)$ implies that these are complex numbers, representing at each frequency ω the magnitude and phase of the sinusoidal signals in (2.16) and (2.17). The overall response to simultaneous input signals of the same frequency in several input channels is, by the superposition principle for linear systems, equal to the sum of the individual responses, and we have from (2.20)

$$y_i(\omega) = g_{i1}(j\omega)u_1(\omega) + g_{i2}(j\omega)u_2(\omega) + \cdots = \sum_k g_{ik}(j\omega)d_k(\omega) \tag{2.23}$$

or in matrix form

$$y(\omega) = G(j\omega)u(\omega). \tag{2.24}$$

In this application, the elements of (2.24) are

$$u(\omega) = \begin{bmatrix} u_1(\omega) \\ u_2(\omega) \\ u_3(\omega) \end{bmatrix} = \begin{bmatrix} v_d(\omega) \\ v_q(\omega) \\ \hat{\Psi}_{\mathrm{pm}}(\omega) \end{bmatrix} \tag{2.25}$$

and

$$y(\omega) = \begin{bmatrix} y_1(\omega) \\ y_2(\omega) \end{bmatrix} = \begin{bmatrix} i_d(\omega) \\ i_q(\omega) \end{bmatrix}. \tag{2.26}$$

In order to have a simplified notation later, for a generic frequency ω_i the previous vectors are written as

$$u(\omega_i) = U^i = \begin{bmatrix} U_1^i \\ U_2^i \\ U_3^i \end{bmatrix} \tag{2.27}$$

and

$$y(\omega_i) = Y^i = \begin{bmatrix} Y_1^i \\ Y_2^i \end{bmatrix} \tag{2.28}$$

Given the linearity of the system, the superimposition principle holds.
We can now define the *fundamental frequency* as

$$\omega_0 = \frac{2\pi}{T}, \tag{2.29}$$

where T is the relative period. Under the mild assumption of periodic signals, every entry of the vector $u(t)$ can be decomposed as the sum of N harmonic signals:

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{N} U_1^n e^{jn\omega_0 t} \\ \sum_{n=1}^{N} U_2^n e^{jn\omega_0 t} \\ \sum_{n=1}^{N} U_3^n e^{jn\omega_0 t} \end{bmatrix}. \tag{2.30}$$

The index n indicates the n-multiple of the fundamental frequency. The entries of the second right-hand term of (2.30) are a truncation of the theoretical infinite series. The equation (2.30) holds under the assumption of a large N.
The three sequences of complex coefficients $U_1^{1\dots N}$, $U_2^{1\dots N}$, $U_3^{1\dots N}$ are the frequency spectra of the input $u_1(t)$, $u_2(t)$, $u_3(t)$. They are obtained by means of the Fourier Transform. We remark that every single coefficient in these spectra represents the contribution of the harmonic of order n. The sum of all the elements of these spectra corresponds to the original signal.
In the `MATLAB` simulation, the function `fft` (Fast Fourier Transform) is adopted [3].

Therefore, in order to get the spectrum of the output vector $y(t)$, the equation (2.24) has to be repeated N times, one for every multiple of the fundamental frequency ω_0:

$$\begin{bmatrix} Y^1 \\ Y^2 \\ \vdots \\ Y^N \end{bmatrix}^T = \begin{bmatrix} G(j\omega_0)U^1 \\ G(j2\omega_0)U^2 \\ \vdots \\ G(jN\omega_0)U^N \end{bmatrix}^T . \tag{2.31}$$

Starting from the spectrum $Y^1, \ldots, Y^N$, the inverse Fourier transform (command `ifft` in `MATLAB`) is used to reconstruct the output signal in the time domain:

$$y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{N} Y_1^n e^{jn\omega_0 t} \\ \sum_{n=1}^{N} Y_2^n e^{jn\omega_0 t} \end{bmatrix}. \tag{2.32}$$

On a side note, the operation (2.31) is implemented using matrices of dimension 3: The function `pagemtimes` combines matrix and element-wise products resulting in a remarkable speed up of the frequency response computation.

2.1.4. Parameters of the model

The framework of this thesis will be limited to a single steady-state condition of the machine. In the electrical drive field, we refer to each steady-state condition as an *operating point* or *operating condition*. Each operating point is univocally determined by the mechanical speed Ω_m and the electrical torque T_e.

The electrical speed ω_s of the dq-model (2.4) is directly linked to the mechanical speed Ω_m by the relationship

$$\omega_s = \frac{2\pi\Omega_\mathrm{m}}{60N_p}, \tag{2.33}$$

with Ω_m expressed in *rpm*. Notice that ω_s corresponds to the fundamental frequency ω_0 treated in Section 2.1.3.

For a given operating point, the parameters L_d, L_q, and $\hat{\Psi}_\mathrm{pm}$ of the dq-model (2.4) can be considered known. Indeed, there exists vast documentation in the industry, and very accurate models of the PMSM in the automotive field are already available. On top of that, the influence of temperature, aging and any other factors can be neglected, and we assume the parameters to be functions of the operating conditions only.

In this thesis, we deal only with continuous and periodic signals. The latter feature allows us to limit our analysis to a single period. Nevertheless, in the simulation of a model,

the continuous signal must be discretized with a sampling time T_s. Also in this case, there exists a data sheet of sampling times commonly used. This sampling time is an important knob since it compromised the accuracy of the model with the computational and memory burden.
In our study, this choice influences the cost function of the optimal control problem, which is based on the model simulation. Further details on this consequence are given in Section 2.4.2.

2.2. Modulation techniques

The actuation of PMSM is realized through voltage signals applied to the coils. In order to control the input voltage for the machine, a power electronic device is needed. The device under study is a two-level DC-AC (Direct Current - Alternate Current) inverter; This means that it can provide two different values of voltage as output [19]. A schematic of the inverter is shown in Figure 2.3.

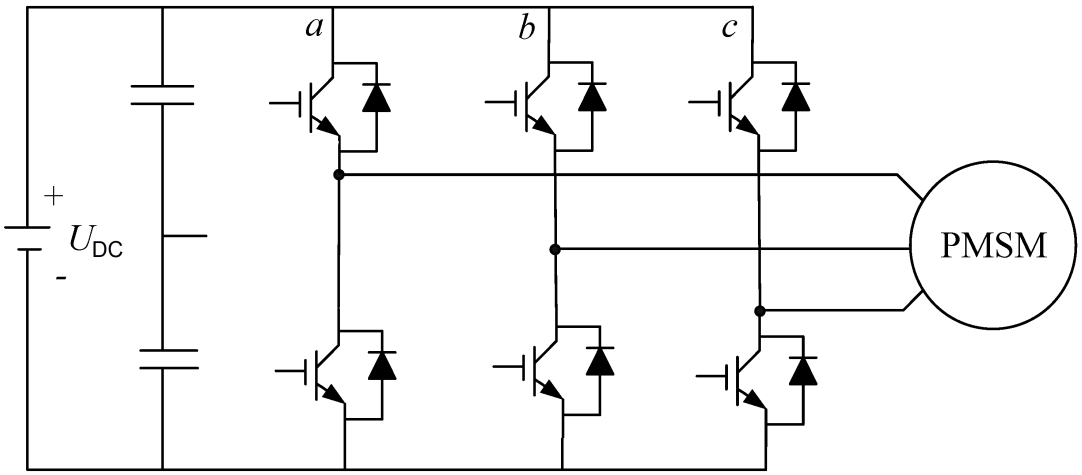


Figure 2.3: Typical topology of three-phase voltage source inverter. *Credits to* [28].

The key components of an inverter are the semiconductor devices, typically transistors, which act as controllable switches. They are used to control the flow of current through the inverter. A sequence of switches determines the output waveform of the voltage at the inverter legs. We remark here that two consecutive switches on the same transistor cannot happen arbitrarily close, but some time is needed, namely the minimum switching time $t_{\min}$. This is a physical limitation of the semiconductor device of the inverter. This parameter is important for the constraints of the optimal control problem and it is further treated in Section 2.4. The process that controls these switches is named *modulation*.

Every modulation technique involves a specific rationale that determines whether the switch is open or closed at any given time; the basic principles are explained in [14].

Among the most used modulation techniques, the old-fashioned Carrier-Based PWM (CBPWM) is not applicable to high-power applications like load traction: At high speed, a large number of inverter switches is required, causing a prohibitive power loss.

Nowadays, the state of the art for an electrical drive in automotive applications is the so-called Space-Vector PWM (SVPWM), an asynchronous modulation method that is strictly related to an outer control, e.g. the *Field Oriented Control* (FOC) [5]. However, SVPWM cannot exploit the maximum potential of the input voltage on the output. This concept is quantitatively represented by a low modulation index, namely the ratio between the fundamental frequency of the generated waveform and the input DC voltage. In addition, this modulation is far away from having a small *Total Harmonic Distortion* (THD), feature that is vastly treated in Section 2.4.2. To overcome the limitations of these well-established modulation techniques, in this paper, the problem is tackled by the use of Optimal Pulse Patterns modulation.

2.2.1. Optimal pulse pattern

The optimal pulse pattern technique is based on the assumption that there is a unique mathematical relationship between the amplitude of the fundamental frequency and the various harmonics, and the switching angles or instants. The sequence of switches forms a pattern of pulses, that gives the name to the method.
Unlike CBPWM and SVPWM, OPPs abandon more general switching criteria in favor of a specific choice of each switching instant. OPPs belong to the class of synchronous modulation schemes. The sequence of switching angles over an electrical rotation is obtained as the solution of an optimal control problem; The formulation of such a problem for our case study will be treated in Section 2.4.

The implementation of OPP is already an established method in steady-state medium voltage applications. The application of this method in the case of different speeds has to face the following practical limitations [12]:

1. The online computation of OPPs is demanding and time-consuming.
2. The implementation of OPPs computed offline requires a significant amount of storage capacity in the controller memory.
3. At high pulse numbers, the incentive to use OPPs diminishes since the harmonic benefit of OPPs over established modulation methods is significantly reduced.

Nevertheless, in recent publications [5, 15], the OPP has been validated as an interesting possibility for the automotive industry, especially for certain regions of the operating space [1]. The most promising control structure is the implementation of OPPs computed offline for different steady states, where the transition from one steady state to another is managed by an outer controller [11].

In a synchronous modulation scheme, the switching signal in each electrical rotation is identical, therefore it is sufficient to define it over one electrical period. The number of switches per period is an integer number.
In this work, we opted for an angle-based representation of the pulse pattern. Therefore, in all the representations of the pulse patterns, we put on the x-axis the electric rotor angle φ_{el} and we represent the single electrical period as an interval of length 2π. We remark here that a similar characterization could be made with respect to the time.

Before entering the description of the waveform, it is necessary to specify what is the voltage level after the first switch. This voltage can take two values:

$$U_{\text{init}} \in \{-0.5U_{\text{DC}},\, 0.5U_{\text{DC}}\} \tag{2.34}$$

where U_{DC} is the constant DC voltage that supplies the power converter.

We define now as *precommutation angle* σ_0 a shift of the whole sequence of switching angles in the electric period. Therefore, the sequence of pulses that characterizes each specific pulse pattern is now contained in the interval $[\sigma_0, \sigma_0 + \pi)$.

When it comes to the computation of an optimal pulse pattern, the quarter- and half-wave symmetry over the single electric period is exploited [1]. In this way, only a fourth of the period of each pulse pattern has to be optimized.

- **Half-period symmetry**: The synchronous switching signal for one period can be divided into quarter 1 for the interval $[\sigma_0,\, \sigma_0 + \pi)$ and quarter 2 for the interval $[\sigma_0 + \pi,\, \sigma_0 + 2\pi)$. In quarter 2 the switching signal is the switching signal of quarter 1 inverted in the voltage, i.e.,

$$u\left(\varphi_{\text{el}}\right) = -u\left(\sigma_0 + \pi + \varphi_{\text{el}}\right), \qquad \text{for } \varphi_{\text{el}} \in \left[\sigma_0,\, \sigma_0 + \pi\right), \tag{2.35}$$

  Notice that the enforcement of this symmetry establishes two fixed switches in the pulse pattern, one at σ_0 and the second one at $\sigma_0 + \pi$.
  In Figure 2.4 quarter 1 is highlighted by the green boxes and quarter 2 by the blue boxes.

- **Quarter-period symmetry**: The first and second period of a half-period symmetric switching signal are axially symmetric to $\sigma = \sigma_0 + \pi/2$ and $\sigma = \sigma_0 + 3\pi/2$, respectively, i.e.,

$$u(\varphi_{\text{el}}) = u(\sigma_0 + \pi - \varphi_{\text{el}}), \quad \text{for } \varphi_{\text{el}} \in [\sigma_0, \sigma_0 + \pi/2), \tag{2.36}$$

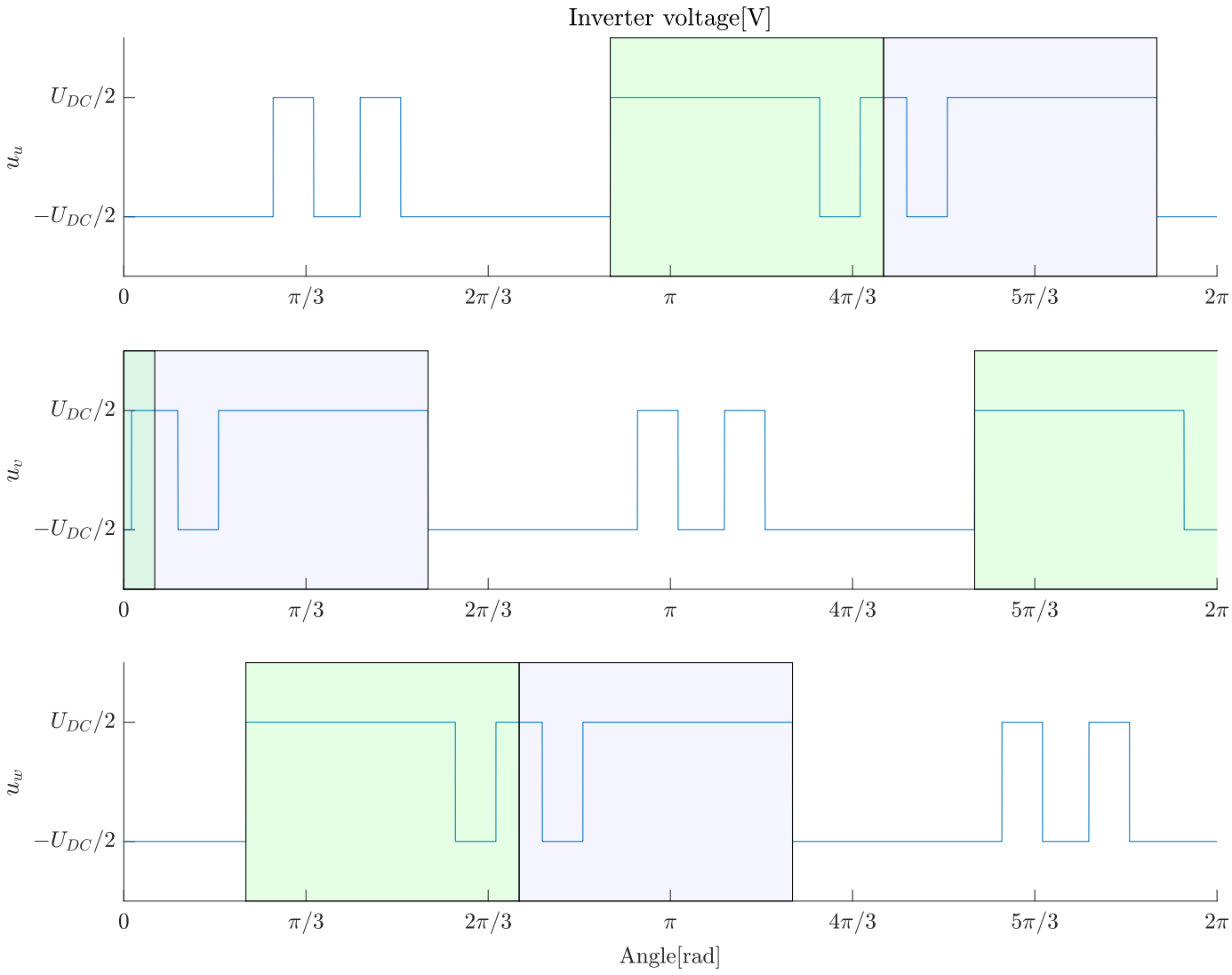


Figure 2.4: Half and quarter period symmetry of the three inverter voltages with $l = 2$. They are obtained through symmetry from the pattern in the green box.

A pulse pattern is then completely described by the voltage level U_{init}, a precommutation angle σ_0, and a sequence of l switching angles, all defined in the first quarter phase. Each

pulse pattern corresponds to the voltage signal of a single terminal of the inverter. In the other two terminals, the sequence of switching angles is the same, but one is delayed of $\frac{2}{3}\pi$ and the other is anticipated of the same quantity. An example of three pulse patterns built exploiting these symmetries is represented in Figure 2.4.
The three voltages of the terminals are combined in the star center of the inverter, producing a *star center voltage*. For the pulse pattern of Figure 2.4, the value of the star center voltage over the electric rotation is represented in Figure 2.5.

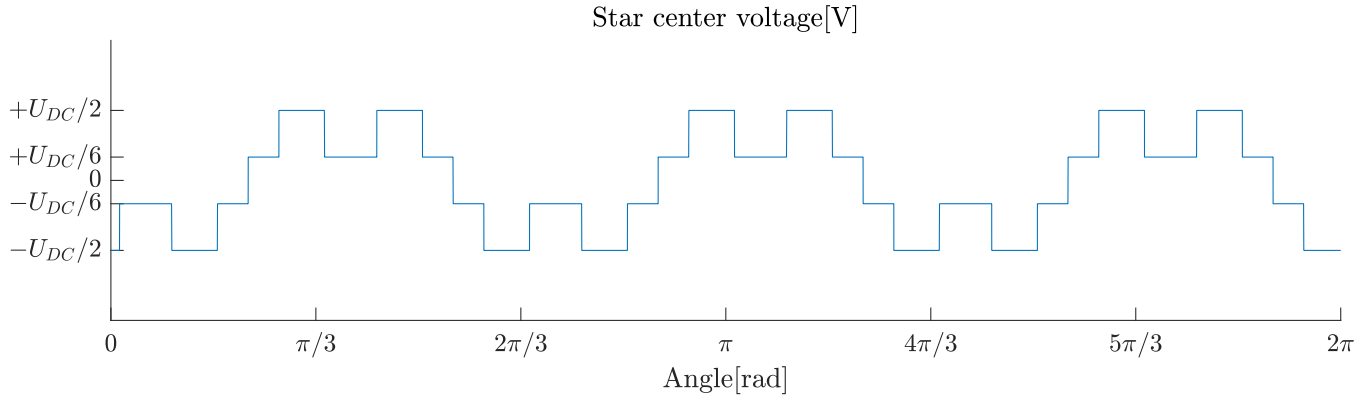


Figure 2.5: Star center voltage over an electric period.

Each coil of the PMSM is then connected to one of the terminals and to the star center. Given the shifts of $\frac{2}{3}\pi$ applied in the definition of the three terminal voltages, the resulting phase signals are properly shifted, so that they preserve the three-phase structure. Figure 2.6 shows the phase voltage corresponding to the pulse pattern of two switches represented so far. Notice that it is obtained as the sum of the voltages in the first plot of Figure 2.4 and the one in Figure 2.5.

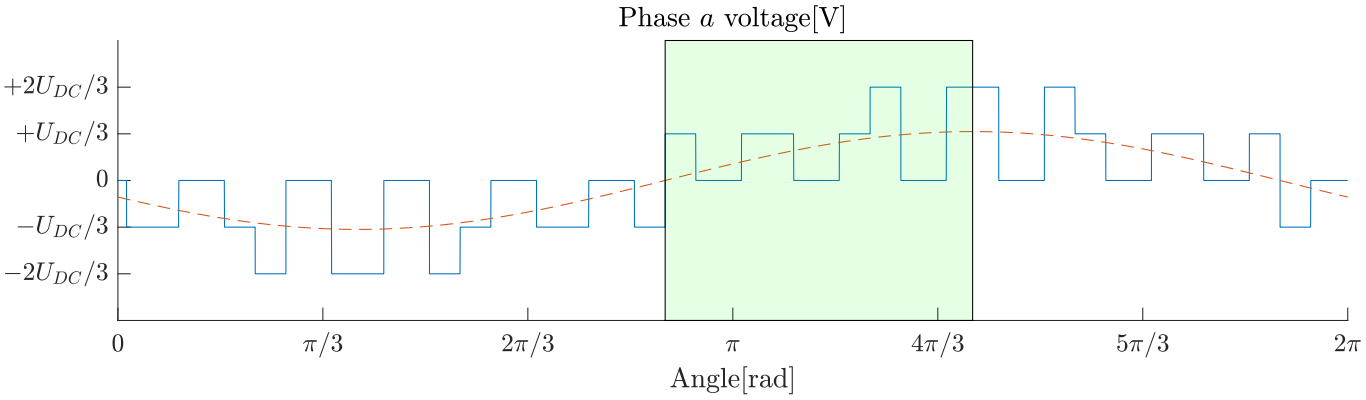


Figure 2.6: Voltage and first harmonic (red dashed line) of phase a over an electric period. The green box starts at the zero voltage crossing of the first harmonic and highlights the quarter-phase symmetry of the signal.

We remark now that the precommutation angle offers a useful degree of freedom in the formulation of the problem since it can be physically interpreted as the phase shift of the fundamental voltage generated by the circuit, as shown in Figure 2.6.

2.3. Simulation of the electric motor and drive

This Section shows how the electrical drive is simulated given a specific pulse pattern, in combination with some simulation results. In Figure 2.7 the main computational flow is provided in a block diagram representation, denoting by square the algebraic relationships and by a circle the dynamical ones. The following graphs are obtained by applying the Optimal Pulse Pattern depicted in 2.4.

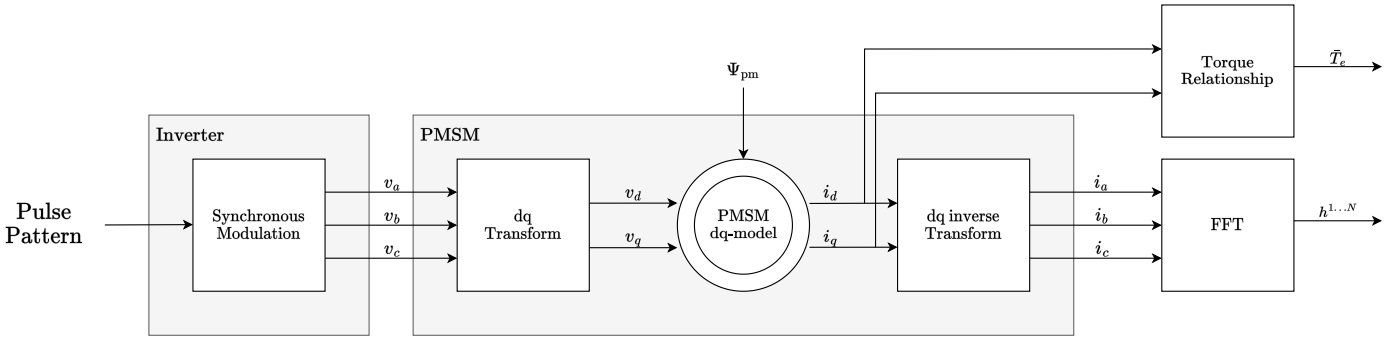


Figure 2.7: Application of the model of the machine and the electrical drive. The diagram depicts the non-linear dynamical relation from the pulse pattern to the mean electric torque $\bar{T}_e$ and the harmonic content of the phase current $h^{1\dots N}$.

Firstly, the synchronous modulation recreates the phase voltage signals from the pulse pattern. It can be seen as the simulation of the inverter device.

After, an interface for the dynamical model of the machine is built, using the dq direct and inverse transformations to obtain phase currents. The dq-framework is very valuable for understanding the behavior of the machine and its control, which is traditionally decoupled for axis d and axis q in FOC. An insight into the electrical quantities in this framework is revealed in Figure 2.8. In particular, the upper plots report the dq-voltages and the lower plots report the dq-current and the relative first harmonic components.

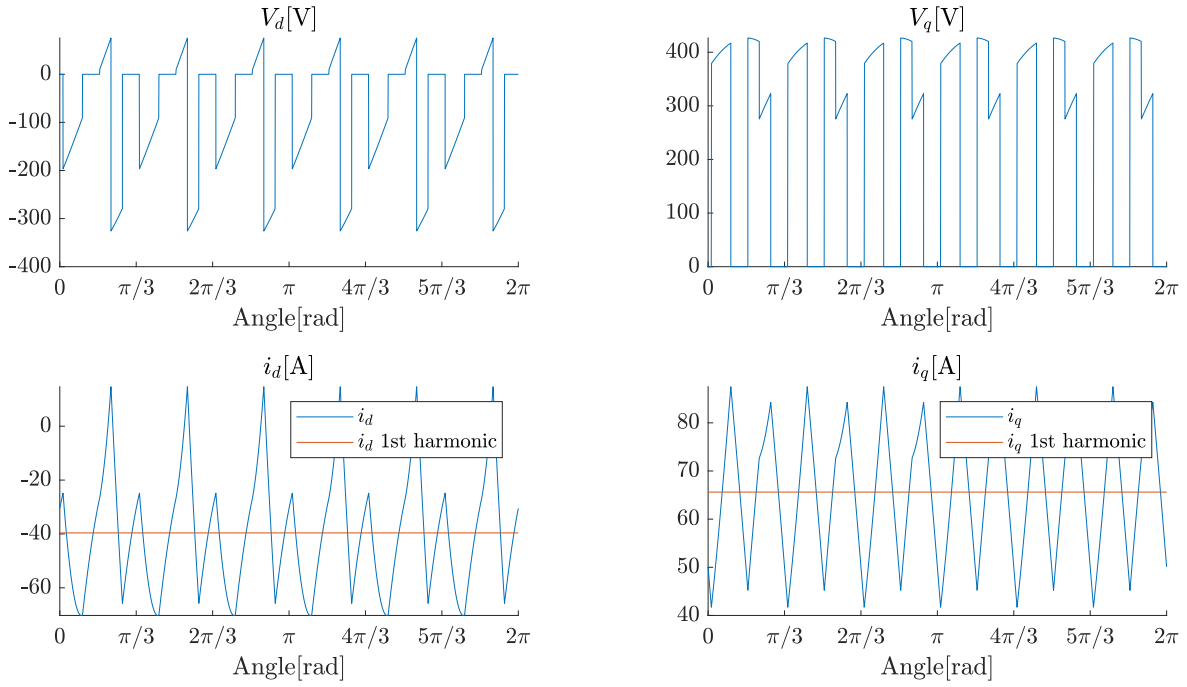


Figure 2.8: [Voltages and currents in the rotating dq reference frame, over an electric period. Red lines represent the first sinusoidal harmonic of the dq currents, that in this reference frame are constants.

In addition, the currents in the dq frame are used to compute the dispensed electrical torque T_e, depicted in Figure 2.9, according to the torque relationship 2.5. Then, the average of the torque over the electrical period $\bar{T}_e$ can be used as a constraint to ensure the operating point requirement.
In the end, the currents flowing in the machine windings are presented in 2.10. These signals are analyzed with a Fourier Transform, in order to obtain the harmonic spectrum magnitude.

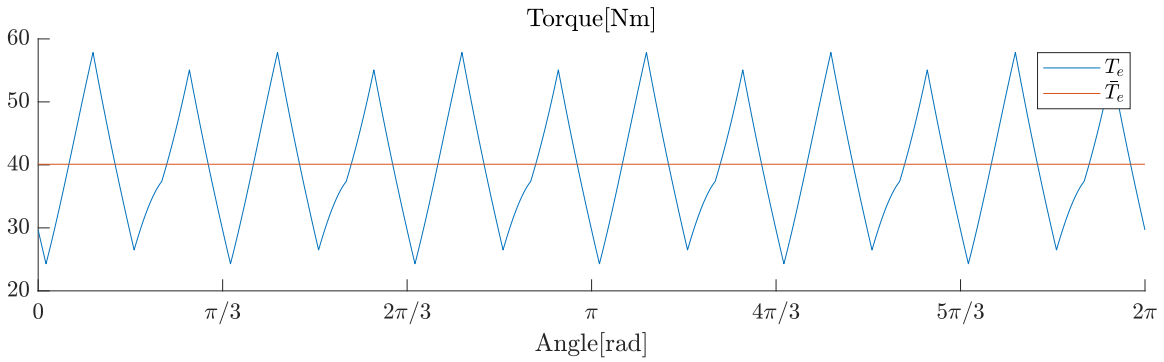


Figure 2.9: Electrical torque over an electric period. The red line represents the average value.

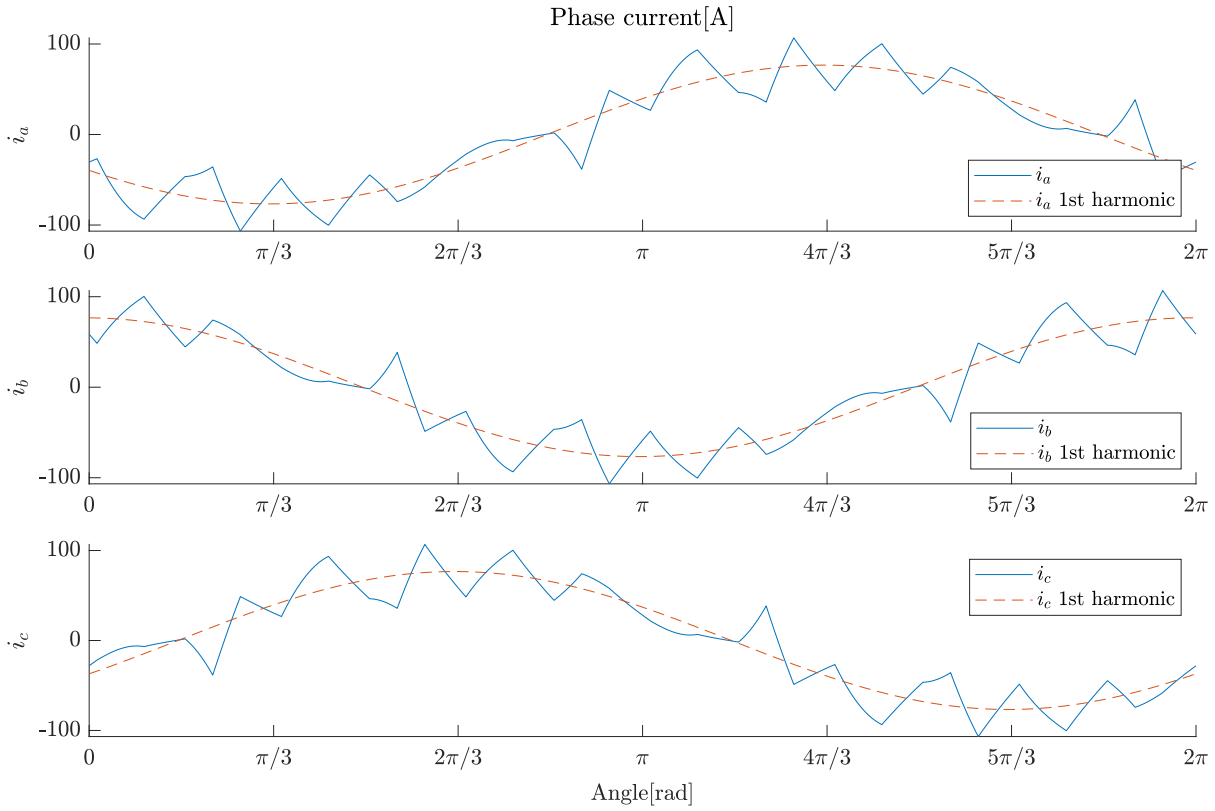


Figure 2.10: Three-phase currents flowing in the machine coils over an electric period. $2/3\pi$ phase shift can be easily seen from the red dotted lines representing the first harmonics for each coil.

Similar to the already stated notation used for (2.30), the phase current spectrum can be written as:

$$i(t) = \begin{bmatrix} i_a(t) \\ i_b(t) \\ i_c(t) \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{N} H_1^n e^{jn\omega_0 t} \\ \sum_{n=1}^{N} H_2^n e^{jn\omega_0 t} \\ \sum_{n=1}^{N} H_3^n e^{jn\omega_0 t} \end{bmatrix}, \tag{2.37}$$

where $H_1^{1...N}$, $H_2^{1...N}$, $H_3^{1...N}$ are the complex coefficients of the frequency spectra of the current and $h_1^{1...N}$, $h_2^{1...N}$, $h_3^{1...N}$ are the relative magnitudes. This frequency domain analysis of the three-phase currents will be useful later in Section 2.4.2 to build the cost function for the optimization problem. For this specific purpose, only the spectrum amplitude will be considered. Moreover, the amplitude content is equal for all the phases

$$h_1^{1...N} = h_2^{1...N} = h_3^{1...N} = h^{1...N}, \tag{2.38}$$

and it is presented in Figure 2.11.

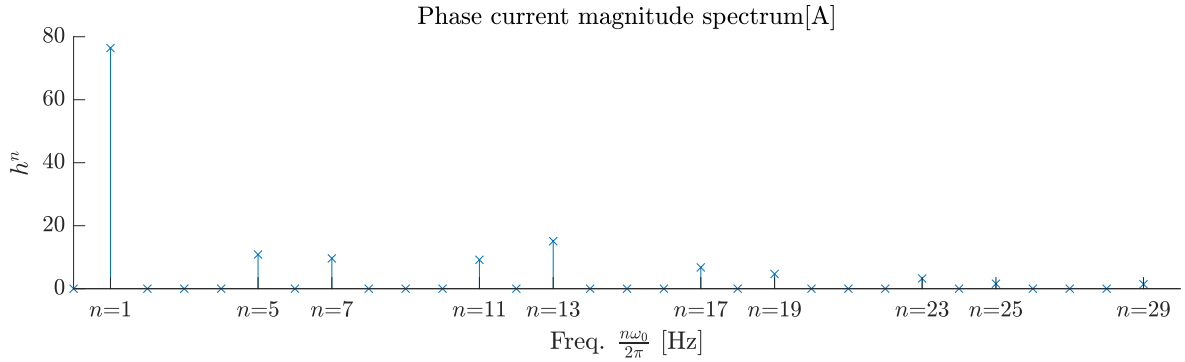


Figure 2.11: Simulated phase current harmonic spectrum.

2.3.1. The dynamical behavior of the machine

The phase voltages v_a, v_b, v_c display a standard inverter harmonic content and in Figure 2.12 just one among the three phases is presented (for symmetry reasons, the three spectra are identical). The even-order harmonics and the harmonics with an order multiple of three are not present, because of the periodic synchronous modulation and the three-phase configuration.
Notice that the objective of the power converter is to realize an ideal sinusoidal voltage, that would have just a first harmonic. However, the true realization, that must undergo a modulation technique, produces additional high harmonics, which have to be considered as pollution.

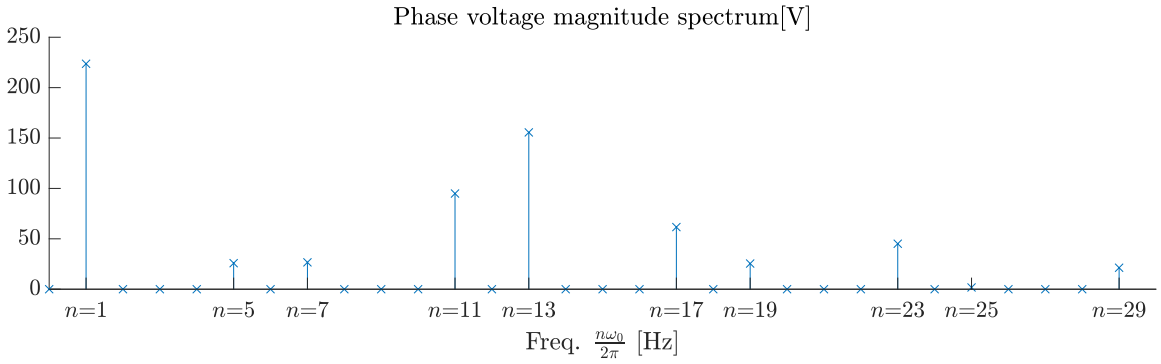


Figure 2.12: Simulated phase voltage harmonic spectrum.

The phase voltages and the phase currents correspond respectively to the input and the output of the PMSM. Comparing the relative spectra of Figures 2.11 and 2.12, it can be concluded that overall the motor acts as a classic low-pass filter. The system linearity preserves the actual values of the frequencies and the current amplitude is reduced according to the frequency response described in Section 2.1.3.

2.4. Optimization problem formulation

In order to formulate the optimal control problem, first we define in Section 2.4.1 the optimization variables, then in Section 2.4.2 we define the cost function of the problem. In Section 2.4.3 we introduce the constraints of the problem. The complete optimization problem is then assembled in Section 2.4.4.

2.4.1. Optimization variables

The optimization variables of the optimal control problem that we are formulating need to be chosen carefully. They have to cover comprehensively the whole set of possible pulse patterns, but they must be limited to keep the dimension of the problem low. As discussed in Section 2.2.1, to define OPPs we construct a switching signal over the first quarter of one of the pulse patterns. Then, to define the complete electrical rotation in all phases it is sufficient to define:

1. The precommutation angle σ_0 determines also the first switching angle of the sequence.
2. The sequence of switching angles on the quarter period.
3. The voltage level after the precommutation angle U_{init}.

The first element of this list has already been defined in Section 2.2.1. For what concerns the second element, different approaches are available.
It is possible to use a sequence with the angles defined with respect to the zero of the electric rotation. In this case, we should limit their values to the interval $[\sigma_0, \sigma_0 + \pi/2)$, and they must be in increasing order.
Alternatively, each angle can be defined as an increment with respect to the previous one. In this case, the sequence does not have to be in increasing order, but we should apply a constraint to the sum of the increment angles so that it is less than $\pi/2$.
Finally, it is possible to define each switching angle as the increment from the precommutation angle. In this case, their value must be limited to the interval $[0, \pi/2)$ and they must be in increasing order.
Out of these three possible approaches, the third shows the good advantage of constant bounds on each switching angle. This feature is appealing for optimization methods where the search space must be constant and previously defined, like SMGO. Another advantage of this formulation stands in the fact that σ_0 is the phase delay of the phase voltage. In fact, if one wants to keep the same shape of the phase voltage, but with a different phase, it is possible to simply change σ_0, instead of re-optimizing the whole pulse pattern.

Therefore, we define the l switching angles in the quarter phase after the precommutation σ_0 as $\bar{\sigma_1}, \ldots, \bar{\sigma_l}$. We can equivalently represent the sequence of switching angles using the variables

$$\sigma_i = \bar{\sigma_i} - \sigma_0 \quad \text{for} \quad i = 1, \ldots, l. \tag{2.39}$$

Notice that the first switch occurs at σ_0. We can then collect these continuous variables in the vector

$$\boldsymbol{\sigma} = [\sigma_0, \sigma_1, \ldots, \sigma_f]\,. \tag{2.40}$$

The initial voltage U_{init} could be introduced as an additional boolean optimization variable, such that $U_{\text{init}} \in \{-0.5\,U_{\text{DC}}, 0.5\,U_{\text{DC}}\}$. This structure essentially requires solving twice the optimization problem, using only $\boldsymbol{\sigma}$ as optimization variable.
Differently, in this work, the optimization problem is solved just one time, fixing

$$U_{\text{init}} = 0.5\,U_{\text{DC}} \tag{2.41}$$

and bounding the precommutation angle in the interval

$$\sigma_0 \in [-\pi, +\pi)\,. \tag{2.42}$$

This is equivalent to the solution of two optimization problems. Indeed, the following proposition is valid:

Proposition 2.1. *A shift by π of the whole pulse pattern, possible thanks to the precommutation angle, is equivalent to a change in the sign of the starting voltage U_{init}.*

Proof. This property is due to the half-period symmetry and the periodicity of the pulse pattern. Let consider $\varphi_{\text{el}} \in [0, 2\pi)$ and the pulse pattern $u_s(\varphi_{\text{el}})$. For the sake of simplicity, we assume $\sigma_0 = 0$, but the proof holds also for different values.
A change of U_{init} in the pulse pattern leads to the pulse pattern

$$u_{f1}(\varphi_{\text{el}}) = -u_s(\varphi_{\text{el}}).$$

Instead, if we apply to u_s the shift $\sigma_0 = \pi$, the new pulse pattern is

$$u_{f2}(\varphi_{\text{el}}) = u_s(\sigma_0 + \varphi_{\text{el}}) = u_s(\pi + \varphi_{\text{el}}) = -u_s(2\pi + \varphi_{\text{el}})$$

where the last step is due to symmetry (2.35). Since the signal is periodic with period 2π, we can rewrite it as $-u_s(\varphi_{\text{el}})$, proving the equivalence. □

In order to take into account all the possible pulse patterns, either the first optimization variable must cover an interval of at least 2π or both signs for the starting voltage must be taken into account. In the last case, σ_0 can be limited to a stricter interval, thanks to a priori knowledge on the specific operating condition.
Thanks to Proposition 2.1, the optimization variables of the problem at hand are limited to the entries of the vector $\boldsymbol{\sigma}$.

In this work, the optimization problem is formulated with a fixed number of l switches, and therefore a fixed number of optimization variables. The number of switching angles l could also be considered an optimization variable. Under the assumption that the cost function is decreasing with respect to the number of the optimization variables (further detail in Section 2.4.2), the problem is iteratively solved with an increasing value of l, in order to improve the solution. Moreover, this approach allows to directly test how the optimization algorithm in use behaves with increasing dimensions. In Chapter 6, we will see that a big number of optimization variables on such a high non-linear problem turns out to be impracticable to be solved with global methods.

2.4.2. Cost Function

The cost function composition is the most interesting and creative task in the formulation of any kind of optimization problem because it has to be representative of the objective behavior we want to optimize and at the same time compliant with the optimization method used to solve the problem.

The goal of OPPs is to minimize the total losses of the electric drive system. Some of the sources of powertrain losses are the switching losses, incurred by electronic power converters, and the copper and iron losses, associated with higher-order harmonics of the rotating magnetic field in the electric machine. These are contingent upon the operating conditions of the vehicles. Nevertheless, a relevant contribution is given by the deviation of the realized physical electrical quantities (currents, voltages, and fluxes) from the desired fundamental ones. As a matter of fact, the harmonic pollution introduced by a component can seriously spread into the electric network, producing a cascading pollution distribution.

For this reason, the cost function in this work is based on the *Total Harmonic Distortion* (THD) concept. This methodological choice follows the literature on this topic [7, 10], as much as recent trails of research [1], and even if it is a simplified approach, still represents a valid and relevant starting point. The resulting framework can then be easily extended to a more complex loss model.

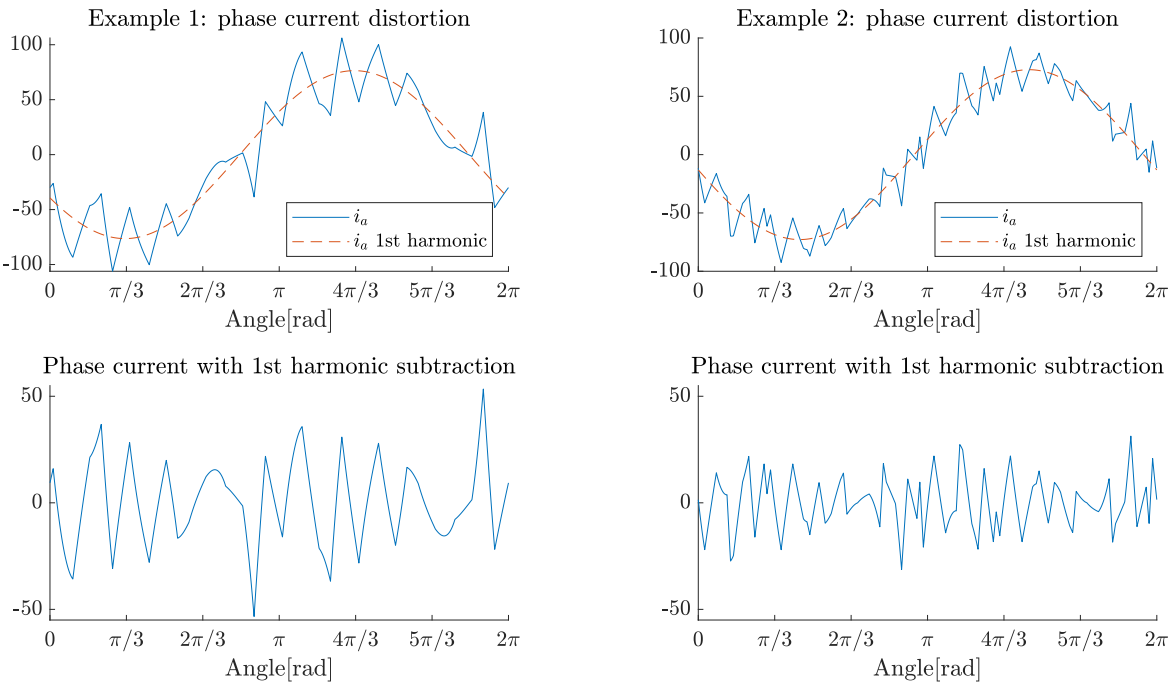


Figure 2.13: Examples of different harmonic distortions.

The THD is a performance criterion computed with the spectrum of a signal. In particular, it compares the magnitude of the first harmonic with the magnitude of the higher harmonic content. One of the possible mathematical formulations of such a concept is

$$\text{THD} = \frac{\sqrt{\sum_{i=2}^{N}(h^i)^2}}{h^1}, \tag{2.43}$$

where h^i is the amplitude of the i-th spectrum harmonic of the signal under exam.

In this work, we apply this index to the signal of the phase current, since the current distortion plays a fundamental role in the functioning of a PMSM. Note that a different choice could have been to apply it to the dq-model currents I_d and I_q, but we opted for the phase current because it has a physical meaning. From now on, we indicate with $h^i(\boldsymbol{\sigma})$ the magnitude of the i-th order harmonic component of the phase current.

We abandon formulation (2.43) in favor of the multi-objective cost function

$$F(\boldsymbol{\sigma}) = \sqrt{\sum_{i=2}^{N}(h^i(\boldsymbol{\sigma}))^2} + Qh^1(\boldsymbol{\sigma}) \tag{2.44}$$

and we do this for a double reason. First of all, to avoid numerical issues related to the

denominator approaching zero value. In the second instance, the second term enforces a soft constraint on the upper limit of the torque, as explained in Section 2.4.3. In fact, leveraging on h^1 at the denominator, the solver encourages h^1 to high values. This implies a high value of the torque, far from the desired one, especially for working points with low torque and hence low current. The weight parameter Q is empirically tuned to have the same order of magnitude for both terms, but still express the first term, which is the primary objective, as bigger than the second.

It is possible to see that the cost function (2.44) does not account in any way for the switching losses. For this reason, an increasing number of switching l can be considered beneficial for the minimization at hand. This consideration is corroborated by the best value found in the resolution of the optimal problem for dimension $D = l + 1$ going up to 20, as shown below.

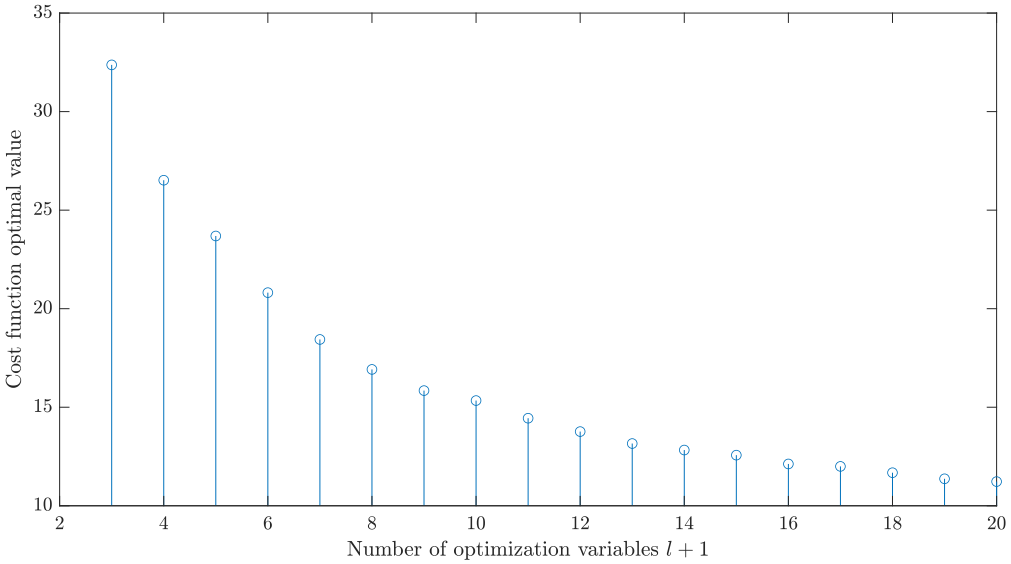


Figure 2.14: Optima of the cost function for different $D = l+1$. These results are obtained using multistart gradient-based optimization.

On the shape of the cost function

The cost function (2.44) is a continuous function defined on the domain of continuous variables $[\sigma_0, \sigma_1, \ldots, \sigma_f] \in \mathbb{R}^{l+1}$. However, the model of the system is simulated with a sampling time T_s. This means that a change in one of the optimization variables lower than T_s does not affect the simulation, and consequently does not affect the cost function. As a result, the cost function is a piecewise constant function, with a more or less finer partition, according to the discrete-time step T_s used to sample the input signal during a

single electric rotation (see Section 2.1.4).

2.4.3. Constraints

This optimal control problem is characterized by constraints of different nature. First, there are linear constraints on the optimization variables. These are due to the specific formulation chosen in 2.4.1, but also to a physical constraint due to the inverter nature, which is the minimum pulse time $t_{\min}$. The semiconductors in each phase of the inverter cannot switch arbitrarily fast, but they need some time after every switch before they can be switched again. To account for this, this minimum pulse time is translated into a proper minimal pulse angle width $\sigma_{\min}$ (expressed in rad) and is accounted in the constraints of the problem.
Therefore, the bounds on the optimization variables are

$$\sigma_0 \in [-\pi, +\pi), \tag{2.45a}$$

$$\sigma_i \in \left[i\,\sigma_{\min}, \frac{\pi}{2} - \frac{(l+1-i)\sigma_{\min}}{2} \right] \quad \text{for} \quad i = 1, ..., l. \tag{2.45b}$$

Notice that (2.45b) ensures the pulse pattern to be within a quarter-period since we take advantage of the symmetries. Moreover, the difference between the lower bound on σ_1

$$\sigma_1 > \sigma_{\min} \tag{2.46}$$

and the upper bound on σ_f

$$\sigma_f < \frac{\pi}{2} - \frac{\sigma_{\min}}{2} \tag{2.47}$$

is due to the quarter phase symmetry. Indeed, if (2.47) is strictly respected, in the full pattern σ_f differs of $\sigma_{\min}$ from the following switching.
The formulation chosen in 2.4.3 requires also that the switching angles must be in increasing order. We simply write

$$\sigma_i + \sigma_{\min} \leq \sigma_{i+1} \quad \text{for} \quad i = 1, \ldots, l-1 \tag{2.48}$$

Notice that this set of constraints does not involve the precommutation angle σ_0, which can be freely chosen within the bounds (2.45a). The presence of $\sigma_{\min}$ ensures that no switching occurs too close to another one.

The OPP control problem is a versatile formulation. For what concerns the non-linear constraints, it is possible to account for a variety of different physical issues, e.g. the

maximum current flowing in the coils or a limit on the voltage ripple [1].
In this study, the only non-linear constraint explicitly applied is the one on the operating condition. In fact, we want to optimize the pulse pattern for a specific steady state, given by the couple Ω_{m} and T_{req}. While the speed is implicitly defined in the model of the PMSM, the torque is imposed.
Enforcing a restriction on torque for every time instant requires a big number of constraints. Therefore, we opt for imposing a constraint on the mean torque value $\bar{T}_e(\boldsymbol{\sigma})$. The choice of the cost function in Section 2.4.2 supports this choice, since the minimization of the current harmonic distortion leads to a small torque ripple.
The most accurate idea to maintain the desired steady-state condition would be to enforce an equality constraint on the mean torque $\bar{T}_e(\boldsymbol{\sigma})$ that the model outputs. However, the equality constraint compromises the performance of the solver. Therefore, we opt for a non-linear inequality constraint on the mean torque

$$\bar{T}_e(\boldsymbol{\sigma}) \geq T_{\mathrm{req}} \tag{2.49}$$

Notice that (2.49) enforce only a lower limit on the mean torque. The upper limit is compelled as a soft constraint, thanks to the second term of (2.44).

From a formal point of view, the fact that the cost function (2.44) depends on the optimization variables can be considered as a non-linear constraint. To point out this dependency, we first define

$$\boldsymbol{h} = \left[h^1, \ldots, h^N\right] \tag{2.50}$$

as the vector that contains all the amplitude of the N-truncated series of current harmonics. We can now write

$$\boldsymbol{h}(\boldsymbol{\sigma}) = F_{\mathrm{ed}}(\boldsymbol{\sigma}), \tag{2.51}$$

where the F_{ed} function basically entails the complete electrical drive model, that relates the amplitude current h^i with the optimization variables $\boldsymbol{\sigma}$. The function $F_{\mathrm{ed}}(\boldsymbol{\sigma})$ is an articulated algebraic and dynamic relationship, described in detail in Figure 2.7.
Notice that, in the solution of the optimization problem, this constraint is implicitly respected in the computation of the cost function. Hence, it does not have to be taken into account by the solver.

2.4.4. Optimization problem

The overall optimal control problem turns out to be a continuous non-convex constrained optimization. The nature of OPP is to find an optimal modulation to a given steady-state working point, therefore the problem can be solved offline and the result can be stored and used during online control. The offline computation ensures the necessary time to explore the multiple local minima that affect this problem.

$$\min_{\boldsymbol{\sigma}=[\sigma_0,\sigma_1,\ldots,\sigma_f]} F(\boldsymbol{\sigma}) = \sqrt{\sum_{i=2}^{N}(h^i(\boldsymbol{\sigma}))^2} + Qh^1(\boldsymbol{\sigma}) \tag{2.52a}$$

subject to

$$\boldsymbol{h}(\boldsymbol{\sigma}) = F_{\text{ed}}(\boldsymbol{\sigma}) \tag{2.52b}$$

$$\sigma_0 \geq -\pi \tag{2.52c}$$

$$\sigma_0 \leq \pi \tag{2.52d}$$

$$\sigma_1 \geq \sigma_{\min} \tag{2.52e}$$

$$\sigma_f \leq \frac{\pi}{2} - \frac{\sigma_{\min}}{2} \tag{2.52f}$$

$$\sigma_j + \sigma_{\min} \leq \sigma_{j+1} \quad \text{for} \quad j = 1,\ldots,l-1 \tag{2.52g}$$

$$\bar{T}_e(\boldsymbol{\sigma}) \geq T_{\text{req}} \tag{2.52h}$$

3 | Set Membership Global Optimization

The Set Membership Global Optimization (SMGO) is an innovative method characterized by its data-driven, global, and derivative-free nature. The following description is a summary as close as possible to the dissertation [21], included here for self-consistency.

3.1. Problem statement

The objective is to minimize a scalar cost function $f(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subset \mathbb{R}^D$ is assumed to be a compact and convex search set, from now referred as *search space*. The points in the search space are subject to the inequality constraints g_s, $s = 1, ..., S$. The functions f and g_s, $s = 1, ..., S$ are unknown, but the following assumptions hold:

Assumption 1. *Functions f and g_s, $s = 1, ..., S$ are Lipschitz continuous over $\mathcal{X}$, with unknown Lipschitz constants $\gamma, \rho_1, ..., \rho_S$:*

$$\begin{aligned} f &\in \mathcal{F}(\gamma) \\ g_1 &\in \mathcal{F}(\rho_1) \\ &\vdots \\ g_S &\in \mathcal{F}(\rho_S) \end{aligned} \tag{3.1}$$

where

$$\mathcal{F}(\gamma) \doteq \{h : |h(\boldsymbol{x_1}) - h(\boldsymbol{x_2})| \leq \gamma \|\boldsymbol{x_1} - \boldsymbol{x_2}\|, \forall \boldsymbol{x_1}, \boldsymbol{x_2} \in \mathcal{X}\}. \tag{3.2}$$

In other words, a function is Lipschitz continuous if its rate of change is bounded by a constant multiple of the norm of the distance between any two points in its domain. In this work, $\|\cdot\|$ symbolizes the 2-norm (Euclidean norm).

Assumption 2. *Functions f and g_s, $s = 1, ..., S$ are not known in closed form, but can be evaluated at any point $\boldsymbol{x}^{(n)} \in \mathcal{X}$ without noise or disturbance (exact evaluation):*

$$\begin{aligned} z^{(n)} &= f(\boldsymbol{x}^{(n)}) \\ c_1^{(n)} &= g_1(\boldsymbol{x}^{(n)}) \\ &\vdots \\ c_S^{(n)} &= g_S(\boldsymbol{x}^{(n)}). \end{aligned} \tag{3.3}$$

Using the convention that a constraint g_s is satisfied at a point $\boldsymbol{x}$ if $g_s(\boldsymbol{x}) \geq 0$, we define the set of points that fulfill g_s:

$$\mathcal{G}_s \doteq \{\boldsymbol{x} \in \mathcal{X} \mid g_s(\boldsymbol{x}) \geq 0\}. \tag{3.4}$$

The existence of a finite feasible region is guaranteed:

Assumption 3. *Considering the feasible set $\mathcal{G} \doteq \mathcal{X} \cap \left\{\cap_{s=1}^{S} \mathcal{G}_s\right\}$, we have*

$$\mathcal{L}(\mathcal{G}) > 0, \tag{3.5}$$

where $\mathcal{L}$ stands for the Lebesgue measure.

The last assumption states that there is a finite non-zero measure (*Lebesgue measure*) for the feasible set. The Lebesgue measure is also known as "D volume"; e.g., for $D = 2$ it is the area, while for $D = 3$ it is the three-dimensional volume.
The combination of these assumptions ensures that at least one global minimizer $\boldsymbol{x}^*$ exists in $\mathcal{X}$:

$$\boldsymbol{x}^* \in \mathcal{X} \doteq \{\boldsymbol{x} \in \mathcal{G} \mid \forall \boldsymbol{x}' \in \mathcal{G}, f(\boldsymbol{x}') \geq f(\boldsymbol{x})\}, \tag{3.6}$$

with the corresponding minimum cost $z^* = f(\boldsymbol{x}^*)$.

3.2. Algorithm

SMGO is implemented through a sequential algorithm: The algorithm performs a well-defined sequence of tasks upon the arrival of a new data point from evaluation. Each function evaluation produces the tuple $\left(\boldsymbol{x}^{(i)}, z^{(i)}, \boldsymbol{c}^{(i)}\right)$, so we take for granted that the evaluations of the constraint functions are coupled, i.e. they are performed simultaneously to the evaluation of the cost. Each data point can be referred to as *sample*. The collection of the points sampled by the optimization algorithm is defined as

$$\boldsymbol{X}^{\langle n \rangle} \doteq \left\{ \left(\boldsymbol{x}^{(1)}, z^{(1)}, \boldsymbol{c}^{(1)}\right); \left(\boldsymbol{x}^{(2)}, z^{(2)}, \boldsymbol{c}^{(2)}\right); \ldots; \left(\boldsymbol{x}^{(n)}, z^{(n)}, \boldsymbol{c}^{(n)}\right) \right\}, \tag{3.7}$$

where $n \in \mathbb{N}$ is the number of data points, which corresponds to the number of total function evaluations. With a slight abuse of notation, the notation $\boldsymbol{x}^{(i)} \in \boldsymbol{X}^{\langle n \rangle}$ means that the tuple $\left(\boldsymbol{x}^{(i)}, z^{(i)}, \boldsymbol{c}^{(i)}\right)$ is contained in the data set $\boldsymbol{X}^{\langle n \rangle}$.

The tasks performed by the algorithm are enumerated as follows:

1. **Set membership model update**: The information provided by the new sample is used to refine the current surrogates of the cost and constraint functions.
2. **Candidate points generation**: New points of the search space are added to the candidate points.
3. **Exploitation**: Investigates the candidate points that could correspond to the optimum.
4. **Exploration**: Investigates the candidate points belonging to a region of the search space where the cost is uncertain.

These steps are revised one by one in the following sections.

3.2.1. Set membership model update

The optimization at hand is based on *Set Membership* (SM) nonlinear function approximation [18]. This approximation can be seen as a cheap surrogate model of the black-box function. In this model, each point of the search space is associated with a lower and upper bound: These represent the tightest bounds, on the basis of the available data and prior assumptions, on the possible values of the underlying function. The difference between these two bounds can be considered an uncertainty measure.

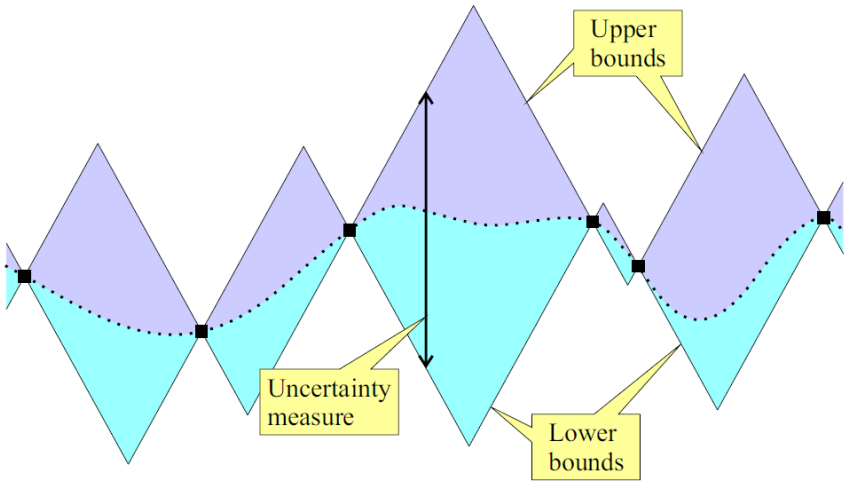


Figure 3.1: Set Membership-based bounds on a 1D function. *Credits to [21].*

At every iteration $n = 1, \dots, N_{\max}$, a point $x^{(n)} \in \mathcal{X}$ is sampled, resulting in a tuple $\left(\boldsymbol{x}^{(n)}, z^{(n)}, \boldsymbol{c}^{(n)}\right)$. This tuple is appended to the existing data set:

$$\boldsymbol{X}^{\langle n \rangle} = \boldsymbol{X}^{\langle n-1 \rangle} \cup \left(\boldsymbol{x}^{(n)}, z^{(n)}, \boldsymbol{c}^{(n)}\right). \tag{3.8}$$

Using the updated data set $\boldsymbol{X}^{\langle n \rangle}$, we compute the estimates of the Lipschitz constants γ and ρ_s, respectively denoted as $\tilde{\gamma}^{\langle n \rangle}$ and $\tilde{\rho}_s^{\langle n \rangle}, s = 1, \dots, S$ accordingly to (3.2):

$$\tilde{\gamma}^{\langle n \rangle} = \max_{\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \in \boldsymbol{X}^{\langle n \rangle}} \left(\frac{\left| z^{(i)} - z^{(j)} \right|}{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)} \right\|}, \tilde{\gamma}^{\langle n-1 \rangle} \right), \tag{3.9}$$

$$\tilde{\rho}_s^{\langle n \rangle} = \max_{\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \in \boldsymbol{X}^{\langle n \rangle}} \left(\frac{\left| c_s^{(i)} - c_s^{(j)} \right|}{\left\| \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)} \right\|}, \tilde{\rho}_s^{\langle n-1 \rangle} \right). \tag{3.10}$$

These formulas are applicable starting from $n = 2$, i.e. the second iteration; $\tilde{\gamma}^{\langle 1 \rangle} > 0$ and $\tilde{\rho}_s^{\langle 1 \rangle} > 0$ are small but finite initial values for the respective Lipschitz constants, which can be available from initial information or estimated. These update laws establish that the estimates of the Lipschitz constants are monotonously increasing.
The new sampled point allows updating also the lower and upper bounds of the estimated cost function on a general point $\boldsymbol{x} \in \mathcal{X}$:

$$\bar{f}^{\langle n \rangle}(\boldsymbol{x}) = \min_{k \in \{1, \dots, n\}} \left(z^{(k)} + \bar{\gamma}^{\langle n \rangle} \left\| \boldsymbol{x} - \boldsymbol{x}^{(k)} \right\| \right) \tag{3.11}$$

as upper bound and

$$\underline{f}^{\langle n \rangle}(\boldsymbol{x}) = \max_{k \in \{1,\dots,n\}} \left(z^{(k)} - \bar{\gamma}^{\langle n \rangle} \left\| \boldsymbol{x} - \boldsymbol{x}^{(k)} \right\| \right) \tag{3.12}$$

as lower bound. We also obtain a central approximation of the cost function

$$\tilde{f}^{\langle n \rangle}(\boldsymbol{x}) = \frac{1}{2} \left(\bar{f}^{\langle n \rangle}(\boldsymbol{x}) + \underline{f}^{\langle n \rangle}(\boldsymbol{x}) \right) \tag{3.13}$$

and its corresponding uncertainty measure

$$\lambda_f^{\langle n \rangle}(\boldsymbol{x}) = \bar{f}^{\langle n \rangle}(\boldsymbol{x}) - \underline{f}^{\langle n \rangle}(\boldsymbol{x}). \tag{3.14}$$

Analogous bounds can be computed for each constraint g_s, resulting in the quantities $\bar{g}_s^{\langle n \rangle}(\boldsymbol{x})$, $\underline{g}_s^{\langle n \rangle}(\boldsymbol{x})$, $\tilde{g}_s^{\langle n \rangle}(\boldsymbol{x})$, and $\lambda_{g,s}^{\langle n \rangle}(\boldsymbol{x})$. If the estimated constraint function $\tilde{g}_s^{\langle n \rangle}(\boldsymbol{x}) > 0$, the constraint is considered satisfied. The region where all the constraints are satisfied provides an estimate of the feasible region, as depicted in Figure 3.2.

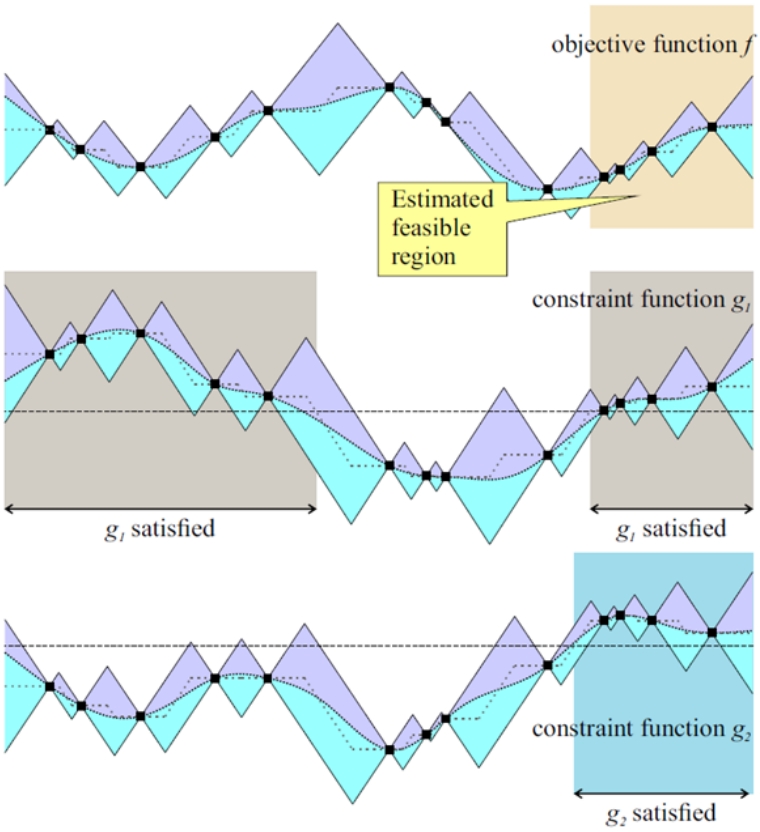


Figure 3.2: SM-based bounds on a 1D function with two constraints. The regions where the constraints are greater than 0 (horizontal line) are highlighted. *Credits to [21].*

3.2.2. Candidate points generation

The SM-bounds are computed for a set of candidate points $\boldsymbol{E}^{\langle n \rangle} \subset \mathcal{X}$. This set can be either initialized or not, but after the first iteration is systematically updated based on the existing data. The generation technique is flexible: The standard and most tested technique is the *spider web generation*, introduced in [24] as an alternative to the *midpoints-based generation* [23].

Initialization

At the beginning of the optimization, it is possible (but not necessary) to populate the set $\boldsymbol{E}^{\langle n \rangle}$ with some initial candidate points. In [21] the case of hyperrectangular $\mathcal{X}$ is treated. In order to have a uniform distribution in the search space, the points are selected from the Sobol distribution [26]. The Sobol distribution is a quasi-random low-discrepancy sequence of points in a unit hypercube of dimension D. This method preserves the repeatability of the algorithm and confers a good coverage of $\mathcal{X}$.

Midpoints-based generation

For this simple candidate points generation, $\mathcal{X}$ is assumed to be a polytope, with vertex set $\boldsymbol{V} \doteq \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_V\}$, where $V > D$. In this case, the candidate points are all the midpoints between any pair of samples and/or vertices, i.e.

$$\boldsymbol{E}^{\langle n \rangle} = \left\{ \frac{\boldsymbol{x}^{(i)} + \boldsymbol{x}^{(j)}}{2} \mid \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)} \in \boldsymbol{X}^{\langle n \rangle} \cup \boldsymbol{V} \right\}. \tag{3.15}$$

This candidate points generation method is simple in concept and implementation. However, it is restricted only to polytopic $\mathcal{X}$. The number of candidate points that will be generated is $Vn + \frac{n(n-1)}{2}$, being V the number of vertices. This highlights the drawback of the method: For hyperrectangle search spaces, the number of vertices V grows exponentially w.r.t. D. This issue limits the method to use cases with relatively low D (up to around 10).

Spider web generation

This points generation technique is a modified version of the midpoints-based one and is applicable for a generalized convex search space $\mathcal{X}$. For each sample $\boldsymbol{x}^{(i)}$, we generate the candidate points:

1. by gridding along each segment to other samples $\boldsymbol{x}^{(j)} \in \boldsymbol{X}^{(n)} \setminus \boldsymbol{x}^{(i)}$;
2. by gridding from $\boldsymbol{x}^{(i)}$, along the coordinate directions to the bounds of $\mathcal{X}$.

The first criterion is similar to the midpoints-based one, except that the generation on the segments linking to the vertices is no longer involved. We define $\boldsymbol{Y}_{\circ}^{(i)}$ as the set of candidate points generated at iteration i from $\boldsymbol{x}^{(i)}$ to each other sample $\boldsymbol{x}^{(j)}$. We can write

$$\boldsymbol{Y}_{\circ}^{(i)} = \left\{ \boldsymbol{x}^{(i)} + \frac{k}{\bar{B}} \boldsymbol{x}^{(j)} \mid k \in \{1, \dots, \bar{B} - 1\} \right\}, \tag{3.16}$$

with $1/\bar{B}$ being the gridding granularity of the points generation.
For the second points generation criterion, we denote the coordinate directions $\pm\hat{\boldsymbol{a}}_d$ with $d = 1, \dots, D$, and define

$$\begin{aligned} b_{+d}^{(i)} = & \max_{b \in [0,\infty)} b \\ & \text{s.t. } \boldsymbol{x}^{(i)} + b\hat{\boldsymbol{a}}_d \in \mathcal{X}, \end{aligned} \tag{3.17}$$

$$\begin{aligned} b_{-d}^{(i)} = & \max_{b \in [0,\infty)} b \\ & \text{s.t. } \boldsymbol{x}^{(i)} - b\hat{\boldsymbol{a}}_d \in \mathcal{X}. \end{aligned} \tag{3.18}$$

The formulas above describe the lengths of the segments emanating from $\boldsymbol{x}^{(i)}$ towards the borders of $\mathcal{X}$ along dimension d, in the positive and negative coordinate directions. This is a convex constrained optimization, but in the case $\mathcal{X}$ is a hyperbox, it can be circumvented. The candidate points generated along the coordinate direction are described as

$$\boldsymbol{Y}_{+d}^{(i)} = \left\{ \boldsymbol{x}^{(i)} + \frac{k}{\bar{B}} b_{+d}^{(n)} \hat{\boldsymbol{a}}_d \mid k \in \{1, \dots, \bar{B} - 1\} \right\}, \tag{3.19}$$

$$\boldsymbol{Y}_{-d}^{(i)} = \left\{ \boldsymbol{x}^{(i)} - \frac{k}{\bar{B}} b_{-d}^{(n)} \hat{\boldsymbol{a}}_d \mid k \in \{1, \dots, \bar{B} - 1\} \right\}. \tag{3.20}$$

Hence, for each sampled point $\boldsymbol{x}^{(i)}$, we generate a set of candidate points as follows:

$$\boldsymbol{Y}^{(i)} = \left(\bigcup_{d=1}^{D} \left\{ \boldsymbol{Y}_{+d}^{(i)}, \boldsymbol{Y}_{-d}^{(i)} \right\} \right) \cup \boldsymbol{Y}_{\circ}^{(i)}. \tag{3.21}$$

Finally, the overall set of candidate points is

$$\boldsymbol{E}^{\langle n \rangle} = \bigcup_{i=1}^{n} \boldsymbol{Y}^{(i)}. \tag{3.22}$$

A 2D example of a spider web generation is shown in Figure 3.3.
This method results in a total of $n(\bar{B}-1)\left(2D+\frac{n-1}{2}\right)$ candidate points. The growth of generated candidate points is polynomial w.r.t. n, but is linear w.r.t. D. Compared to the midpoints-based generation, this method is computationally advantageous for higher D, especially for hyperrectangular $\mathcal{X}$. Furthermore, the spider web generation holds broader applicability, because it does not require a polytopic $\mathcal{X}$, but can be applied to any convex and compact space.

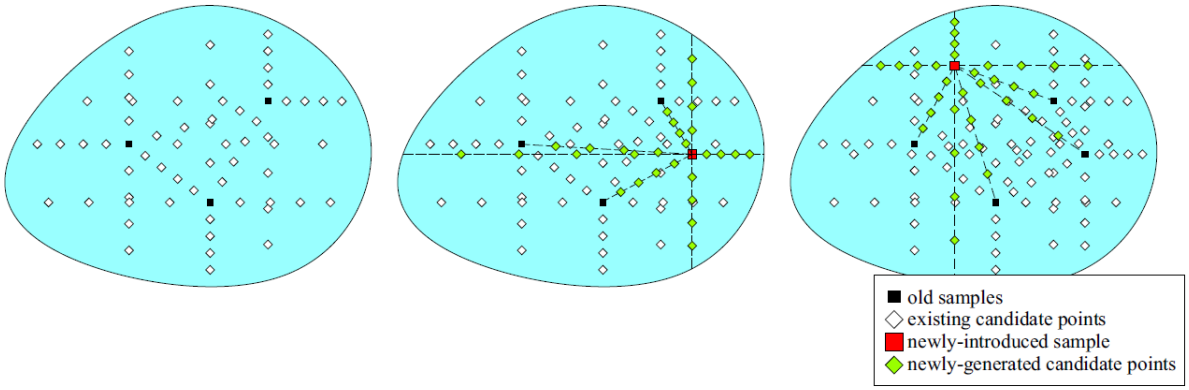


Figure 3.3: Spider web candidate points generation in a 2D case. *Credits to [21].*

3.2.3. Exploitation

During exploitation, the algorithm tries to improve the current optimum $z^{*\langle n \rangle}$. We define the current best feasible sample as

$$\left(\boldsymbol{x}^{*\langle n \rangle}, z^{*\langle n \rangle}, \boldsymbol{c}^{*\langle n \rangle}\right) = \arg \min_{\boldsymbol{x}^{(i)} \in \boldsymbol{X}^{\langle n \rangle}} z^{(i)} \tag{3.23a}$$

$$\text{s.t. } \boldsymbol{c}^{(i)} \geq 0. \tag{3.23b}$$

In other words, it is the sample in $\boldsymbol{X}^{\langle n \rangle}$ with the lowest objective value $z^{(i)}$ among the feasible ones. We can also define the *optimality gap* δ, which is basically the difference between the current best-sampled value and the true optimal value:

$$\delta^{\langle n \rangle} = z^{*\langle n \rangle} - z^{*}. \tag{3.24}$$

The quest for a better optimum is influenced by the following ideas:

- During exploitation, we expect to find a better optimum close to the current one. For this reason, the search is limited to a small trust region $\mathcal{T}^{\langle n \rangle}$.
- The SM-based model carries information about how low the cost function could possibly be at a certain candidate point. The combination of $\tilde{f}^{\langle n \rangle}(\boldsymbol{x})$ and $\lambda_f^{\langle n \rangle}$ provides a metric of the expected improvement.
- The feasibility at a certain candidate point can be inferred by the SM-based model of the constraint functions g_s. The combination of $\underline{g}_s^{\langle n \rangle}(\boldsymbol{x})$ and $\tilde{g}_s^{\langle n \rangle}(\boldsymbol{x})$ indicates whether the point is predicted to fulfill g_s upon sampling.

Therefore, the selection of a new sample $\boldsymbol{x}_\theta^{\langle n \rangle}$ is formulated as

$$\boldsymbol{x}_\theta^{\langle n \rangle} = \arg \min_{\boldsymbol{x} \in \boldsymbol{E}^{\langle n \rangle} \cap \mathcal{T}^{\langle n \rangle}} \left(\tilde{f}^{\langle n \rangle}(\boldsymbol{x}) - \beta \lambda_f^{\langle n \rangle}(\boldsymbol{x}) \right) \tag{3.25a}$$

$$\text{s.t. } \Delta \tilde{g}_s^{\langle n \rangle}(\boldsymbol{x}) + (1 - \Delta) \underline{g}_s^{\langle n \rangle}(\boldsymbol{x}) \geq 0, \quad s = 1, \ldots, S. \tag{3.25b}$$

The user-defined weighting β introduces a trade-off between minimizing the central approximation $\tilde{f}^{\langle n \rangle}$, and maximizing the uncertainty $\lambda_f^{\langle n \rangle}$, to gain more information about the hidden function.
The exploitation aims to sample a feasible point. For this reason, the constraint (3.25b) is included in the surrogate problem. The left-hand expression in (3.25b) is a metric designed to predict the satisfaction of any unsampled point x w.r.t. constraint g_s. This metric entails the parameter Δ. Firstly introduced in [24], this parameter is named *risk factor* and it affects the cautiousness of the algorithm w.r.t. feasibility. This tuning offers a useful feature for real-time optimization that always needs to be provided with a feasible point. When $\Delta = 0$, the candidate points considered feasible are limited to the points where the lower bounds $\underline{g}_s^{\langle n \rangle}(\boldsymbol{x}), s = 1, \ldots, S$ are all greater than 0: This is the most stringent condition. Instead, $\Delta = 1$ is the more risky condition, in the sense that it's easier to find unfeasible points as a result of the exploitation.

If there exists a solution $\boldsymbol{x}_\theta^{\langle n \rangle}$ to the surrogate problem (3.25), before being sampled, it undergoes an expected improvement test. This test assesses if the lower bound of the cost function $\underline{f}^{\langle n \rangle}$ is lower than the current optimum $z^{*\langle n \rangle}$ by a certain threshold:

$$\underline{f}^{\langle n \rangle} \left(\boldsymbol{x}_\theta^{\langle n \rangle} \right) \leq z^{*\langle n \rangle} - \eta, \tag{3.26}$$

where $\eta = \alpha \tilde{\gamma}^{\langle n \rangle}$ is referred to as the *expected improvement threshold*, and $\alpha > 0$ is a tuning parameter. Figure 3.4 offers a visualization of (3.26).

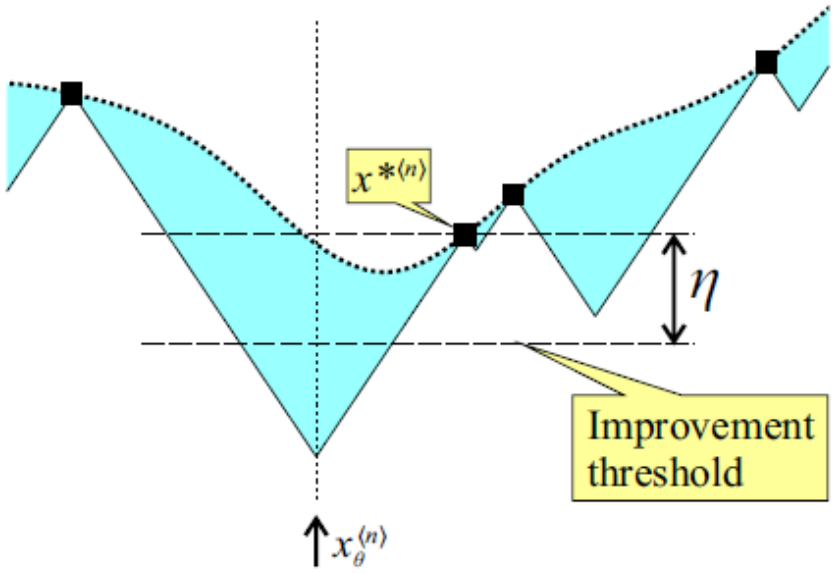


Figure 3.4: Expected improvement threshold η in a 1D case. *Credits to [21].*

On the one hand, if the exploitation candidate $x_\theta^{\langle n\rangle}$ satisfies (3.26), it will be assigned for evaluation at the next iteration $n+1$:

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}_\theta^{\langle n\rangle}. \tag{3.27}$$

On the other hand, if the exploitation optimization (3.25) is unfeasible or if $\boldsymbol{x}_\theta^{\langle n\rangle}$ fails the expected improvement test (3.26), the algorithm will proceed to attempt an exploration of the search space. This means that the criterion for selecting the next sampling point changes. The expected improvement test is necessary to ensure the theoretical properties of the algorithm, as we recall in Section 3.3.

Trust region

The trust region is a ball $\mathcal{T}^{\langle n\rangle}$ centered at the current optimal point $\boldsymbol{x}^{*\langle n\rangle}$ with radius $v^{\langle n\rangle}$. A trust region exists as soon as $\boldsymbol{x}^{*\langle n\rangle}$ exists:

$$\mathcal{T}^{\langle n\rangle} = \left\{\boldsymbol{x} \in \mathcal{X} \mid \left\|\boldsymbol{x} - \boldsymbol{x}^{*\langle n\rangle}\right\|_\infty \leq v^{\langle n\rangle}\right\}. \tag{3.28}$$

Any norm can be used to declare (3.28), e.g. 1- or 2-norm, as to adapt to the shape of the search set $\mathcal{X}$. The size of $\mathcal{T}^{\langle n+1\rangle}$ is updated according to the resulting sampled value

$z^{(n+1)}$, as follows:

$$v^{\langle n+1\rangle} = \begin{cases} \max\left(\underline{v}, \kappa v^{\langle n\rangle}\right) & \text{if exploitation on } n \text{ failed,} \\ \min\left(\bar{v}, \frac{1}{\kappa} v^{\langle n\rangle}\right) & \text{if exploitation on } n \text{ was successful.} \end{cases} \tag{3.29}$$

where $\kappa < 1$ is a shrinking factor. In summary, $\mathcal{T}^{\langle n+1\rangle}$ is shrunk (until a very small radius $\underline{v}$), if no better optimum was found, i.e., there is no improvement w.r.t. current best $z^{*\langle n\rangle}$, or if exploration was done instead. Conversely, $\mathcal{T}^{\langle n+1\rangle}$ is enlarged (up to $\bar{v}$) when the new sample from exploitation has a posteriori returned a new best point. This definition slightly differs from the theoretical one provided in [21, 24], but it describes accurately the implementation.

The size change of the trust region $\mathcal{T}^{\langle n\rangle}$ imitates the level of confidence that we have in a certain region. If the region provides good samples, the trust region is kept large to avoid cutting too many candidate points out of the exploitation phase. Conversely, when the exploitation fails, a shrinkage of the trust region facilitates the conversion into an exploration.

We remark that the trust region serves two purposes:

1. It filters the candidate point set $\boldsymbol{E}^{\langle n\rangle}$ in order to exploit just a neighborhood of the current optimum.

2. It casts a set of additional points, namely the *cloud points*. These points need to be characterized by the bounds of the SM-based model, so they add some computational burden. However, they introduce a higher variety of search directions in the exploitation phase.

In the standard implementation, the cloud points are obtained as elements of a Sobol distribution. The Sobol distribution is a good coverage of the neighborhood of the current best sample, but it limits the trust region to a D-hyperrectangular shape. Since it is not guaranteed that the cloud points belong to $\mathcal{X}$, this condition must be checked on every member of the cloud.

3.2.4. Exploration

To acquire more information regarding the shape of the unknown functions in the whole search space, the SM-based model is used to perform exploration. In this case, we want to sample the point $\boldsymbol{x}_{\phi}^{\langle n \rangle}$ with the highest uncertainty, such that upon sampling at iteration $n+1$, the uncertainty $\lambda_f^{\langle n+1 \rangle}\left(\boldsymbol{x}_{\phi}^{\langle n \rangle}\right)$ sets to zero, and the uncertainty in its vicinity also decreases. In the exploration routine, we select $\boldsymbol{x}_{\phi}^{\langle n \rangle}$ as the solution of the optimization problem

$$\boldsymbol{x}_{\phi}^{\langle n \rangle} = \arg \max_{\boldsymbol{x} \in E^{\langle n \rangle}} \Phi^{\langle n \rangle}(\boldsymbol{x}). \tag{3.30}$$

The function $\Phi^{\langle n \rangle}(\boldsymbol{x})$ is referred to as the *exploration merit function*. It is composed by the terms

$$\Phi^{\langle n \rangle}(\boldsymbol{x}) \doteq \phi^{\langle n \rangle}(\boldsymbol{x}) + k\left(\tau^{\langle n \rangle}(\boldsymbol{x})\right), \tag{3.31}$$

where the first term on the right-hand part of the equation is responsible for the exploration behavior, while the second term guarantees the global convergence of the proposed algorithm, as will be shown in Section 3.3. The term $\tau^{\langle n \rangle}(\boldsymbol{x})$ is an iteration count initialized at the generation of the candidate point, i.e. the age of the candidate point. Function $k(\cdot)$ is any is continuous and strictly increasing function, with $\lim_{\tau \to \infty} k(\tau) = +\infty$ and $k(0) = 0$.

The design of $\phi^{\langle n \rangle}(\boldsymbol{x})$ is flexible. We opted for the one developed in [24]. It tries to take into consideration the following factors:

- The importance of prioritizing the most remote candidate points with respect to the existing samples.
- When exploring, it is crucial to choose areas where uncertainty with respect to the objective is higher. This approach is applied only in regions that are estimated feasible.
- When no feasible points are found, the algorithm should focus on searching in regions where feasible solutions are more likely to be found. For instance, in the case of multiple constraints, the algorithm should prioritize regions in which more constraints have already been fulfilled.

These factors are combined in

$$\phi^{\langle n \rangle}(\boldsymbol{x}) \doteq \zeta(\boldsymbol{x})\left((1-\Delta) w_{\lambda}(\boldsymbol{x}) + \Delta w_{\pi}(\boldsymbol{x}) w_g(\boldsymbol{x})\right), \tag{3.32}$$

where

$$\zeta = \min_{\left(\boldsymbol{x}^{\langle n\rangle}, z^{(n)}, c^{(n)}\right) \in \boldsymbol{X}^{\langle n\rangle}} \left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}\right\|, \tag{3.33a}$$

$$w_\lambda(\boldsymbol{x}) = \begin{cases} \lambda_f^{\langle n\rangle}(\boldsymbol{x}) & \text{if } \Delta\tilde{g}_s^{\langle n\rangle}(\boldsymbol{x}) + (1-\Delta)\underline{g}_s^{\langle n\rangle}(\boldsymbol{x}) \geq 0, s = 1, \dots, S \\ 0 & \text{otherwise,} \end{cases} \tag{3.33b}$$

$$w_\pi(\boldsymbol{x}) = \sum_{s=1}^{S} \frac{\lambda_{g,s}^{\langle n\rangle}(\boldsymbol{x})}{\tilde{\rho}_s^{\langle n\rangle}}, \tag{3.33c}$$

$$w_g(\boldsymbol{x}) = 2^{\sum \mathbb{1}_2(\boldsymbol{x})}, \tag{3.33d}$$

$$\mathbb{1}_s(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \tilde{g}_s^{\langle n\rangle}(\boldsymbol{x}) \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.33e}$$

At the end of the exploration, the selected sample is assigned for evaluation at the next iteration $n+1$:

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}_\phi^{\langle n\rangle}. \tag{3.34}$$

3.3. Theoretical Analysis

In this section, the convergence of SMGO is treated. The convergence of a global optimization is achieved if, given a required sub-optimality measure $\varepsilon > 0$, it produces a ε-suboptimal cost w.r.t. the global one z^*, and does it in a finite number of evaluations. The global convergence property is defined in [21] as follows:

Definition 1. *An optimization algorithm is globally convergent, if*

$$\forall \varepsilon > 0,\ \boldsymbol{x}^{(1)} \in \mathcal{X},\ \exists\, n_\varepsilon < \infty : z^{*\langle n_\varepsilon\rangle} \leq z^* + \varepsilon.$$

The definition above asserts that given any initial starting point $\boldsymbol{x}^{(1)}$ within the search space $\mathcal{X}$, the algorithm will attain an ε-suboptimal point in a finite number of evaluations n_ε. However, in the absence of knowledge regarding the true Lipschitz constants, it is impossible to determine an upper bound on the evaluations required to ensure ε-suboptimality. The foundation of this theoretical convergence is the ability of the algorithm to generate a progressively dense distribution of samples in the search space. The argument of a dense points generation will be the object of Theorem 3.1.
Before proceeding with it, we introduce a preliminary lemma that deals with the behavior of the exploitation routine.

Lemma 1. *SMGO will switch to the exploration routine after a finite number of successive exploitation evaluations.*

We clarify that the aforementioned lemma is established not based on the location of the exploitation point, but rather on the condition of expected improvement, based on a finite α. Therefore, the proximity of the exploitation point to the optimum is irrelevant, as long as it falls within the estimated feasible region and its sampling results in the removal of a finite volume from $\boldsymbol{E}^{\langle n \rangle}$.
Moreover, the lemma demonstrates how a higher (lower) improvement threshold factor α leads to larger (smaller) hyperballs being removed from $\boldsymbol{E}^{\langle n \rangle}$ at every exploitation sampling, resulting in a decreased (increased) upper bound on the number of iterations required to switch to exploration.
As a result of the mechanism of the algorithm, exploration is performed if exploitation fails. As $n \to \infty$, the exploration routine will run infinitely often. Thus, the convergence properties of SMGO depend on whether the exploration routine can satisfy Definition 1, given that there are no guarantees on exploitation convergence.
We present a theorem outlining the conditions required for a candidate points generation scheme, with an exploration merit function (3.31), to guarantee convergence.

Theorem 3.1. *Consider at any iteration n the set of unsampled candidate points $\boldsymbol{E}^{\langle n \rangle}$ generated by SMGO, given the set of sampled points $\boldsymbol{X}^{\langle n \rangle}$. With Assumption 1, SMGO is globally convergent (Definition 1), if $\boldsymbol{E}^{\langle n \rangle}$ satisfies both of the following conditions:*

1. *For any $\boldsymbol{x}^{(i)} \in \boldsymbol{X}^{\langle n \rangle}$ and any half-space $\mathcal{H}$ with $\boldsymbol{x}^{(i)}$ on its boundary,*

$$(\mathcal{H} \cap \mathcal{X}) \cap \boldsymbol{E}^{\langle n \rangle} \neq \varnothing$$

2. *Consider any half-space $\mathcal{H}$, with $\boldsymbol{x}^{(i)} \in \boldsymbol{X}^{\langle n \rangle}$ on its boundary. Now take a sample $\boldsymbol{x}^{(j)}$ such that*

$$\boldsymbol{x}^{(j)} = \arg \min_{\boldsymbol{x} \in \boldsymbol{X}^{\langle n \rangle} \cap \mathcal{H}} \left\| \boldsymbol{x} - \boldsymbol{x}^{(i)} \right\|$$

If $\boldsymbol{x}^{(j)}$ exists, then it should happen that

$$\left(\mathcal{B}\left(\boldsymbol{x}^{(i)}, \left\| \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)} \right\| \right) \cap \mathcal{H} \right) \cap \boldsymbol{E}^{\langle n \rangle} \neq \varnothing,$$

where $\mathcal{B}\left(\boldsymbol{x}^{(i)}, \left\| \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)} \right\| \right)$ is the ball centered in $\boldsymbol{x}^{(i)}$ with radius $\left\| \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)} \right\|$.

The readers interested in the proof are redirected to [21].

Algorithm 3.1 Set Membership Global Optimization

Input: Initial point $\boldsymbol{x}^{(1)}$, search space $\mathcal{X}$, Lipschitz constants estimates $\tilde{\gamma}^{\langle 1\rangle} = \underline{\gamma}$, $\tilde{\rho}_s^{\langle 1\rangle} = \underline{\rho}$, maximum number of iteration $N_{\max}$, parameters α, β, $\Delta, \bar{B}$.

while *iteration n within the budget N_{max}* **do**
  Evaluate the objective f and constraints g_s at $\boldsymbol{x}^{(n)}$, add the resulting sample $\left(\boldsymbol{x}^{(n)}, z^{(n)}, \boldsymbol{c}^{(n)}\right)$ to the set $\boldsymbol{X}^{\langle n\rangle}$
  Update the SM-model
  Update the Lipschitz constants $\tilde{\gamma}^{\langle n\rangle}, \tilde{\rho}_1^{\langle n\rangle}, \ldots, \tilde{\rho}_S^{\langle n\rangle}$ according to (3.9)-(3.10),
  update the current best sample $\left(\boldsymbol{x}^{*\langle n\rangle}, z^{*\langle n\rangle}, \boldsymbol{c}^{*\langle n\rangle}\right)$ from $\boldsymbol{X}^{\langle n\rangle}$ (3.23a),
  update the candidate points $\boldsymbol{E}^{\langle n\rangle}$ (3.22).
  Exploitation routine
  Update the trust region size $v^{\langle n\rangle}$ (3.29) and cast the cloud points in the trust region $\mathcal{T}^{\langle n\rangle}$, solve (3.25) to choose the candidate exploitation point $\boldsymbol{x}_\theta^{\langle n\rangle}$.
  if $\boldsymbol{x}_\theta^{\langle n\rangle}$ exists and expected improvement condition (3.26) is met **then**
    Assign test point for next iteration $\boldsymbol{x}^{(n+1)} \leftarrow \boldsymbol{x}_\theta^{\langle n\rangle}$.
  else
    Exploration routine
    Solve (3.30) to compute $\boldsymbol{x}_\phi^{\langle n\rangle}$.
    Assign test point for next iteration $\boldsymbol{x}^{(n+1)} \leftarrow \boldsymbol{x}_\theta^{\langle n\rangle}$.
  end if
  Go to next iteration $n \leftarrow n + 1$.
end while
Output: Final optimal point and value: return the best sample $\left(\boldsymbol{x}^{*\langle N_{\max}\rangle}, z^{*\langle N_{\max}\rangle}, \boldsymbol{c}^{*\langle N_{\max}\rangle}\right)$ from the set $\boldsymbol{X}^{\langle N_{\max}\rangle}$.

4 | SMGO for Optimal Pulse Pattern

In this Chapter, we detail the modifications made to SMGO to suit the industrial optimization problem presented and formulated in Section 2.4. In Section 4.1, we investigate how to enforce the linear constraints that ensure an increasing sequence of properly spaced angles. After that, we treat some precautions that must be integrated into the SMGO algorithm. In Section 4.2, the tuning of SMGO is described.

4.1. Enforcement of linear constraints

The optimization variables of such a problem are collected as

$$\boldsymbol{\sigma} \doteq [\sigma_0, \sigma_1, ..., \sigma_f]\,. \tag{4.1}$$

The dimensionality of our optimization problem is $D = l + 1$, where l is the number of switching angles that we are considering. The bounds on each single optimization variable have been treated in Section 2.4.3 and are reported in (2.45). However, these bounds guarantee neither the ordering of the switching angles nor the minimum pulse width $\sigma_{\min}$ between them, formally required by constraint (2.52g).
The enforcement of these restrictions as *black-box* constraints g_i would be a poor choice: It would lead to a significant increase in computational burden and could also compromise the performance of the algorithm, e.g. influencing its exploration with the uncertainty of too many functions. On top of that, the search space described by the hyperbox (2.45) is avoidably large and includes redundant pulse patterns.

The acknowledgment of the linear nature of said constraints paves the way to a more efficient solution. With the definition of $\boldsymbol{\sigma}_{\text{sa}} \doteq [\sigma_1, ..., \sigma_f]$ as the sequence of merely switching angles (ruling out the precommutation) and $\boldsymbol{\sigma}_{\min}$ as the column vector obtained

stacking $l-1$ times the value $\sigma_{\min}$, constraint (2.52g) is rewritten as:

$$M\,\boldsymbol{\sigma}_{\text{sa}} \leq \boldsymbol{\sigma}_{\min} \quad \text{with} \quad M = \begin{bmatrix} 1 & -1 & & & \\ & \ddots & \ddots & & \\ & & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{(l-1)\times l}. \tag{4.2}$$

The set of feasible switching angle sequences can then be written as

$$\mathcal{P}_{\text{sa}} = \left\{ \boldsymbol{\sigma}_{\text{sa}} \in \mathbb{R}^l | M\,\boldsymbol{\sigma}_{\text{sa}} \leq \boldsymbol{\sigma}_{\min} \right\}. \tag{4.3}$$

Combining this writing with the knowledge that each switching angle is bounded, we obtain the definition of polytope contained in [13]. Moreover, the introduction of a new switching angle introduces only one new vertex to the set of vertices.

Remark. *The polytope $\mathcal{P}_{sa}$ is a simplex by construction: In fact, it is contained in $\mathbb{R}^l$ and is characterized by $l+1$ linearly independent vertices. It can be described as the convex hull of said vertices [13].*

The extension of (4.3) to include the precommutation angle σ_0 is straightforward:

$$\mathcal{P} = \left\{ \boldsymbol{\sigma} \in \mathbb{R}^{l+1} | M^e\,\boldsymbol{\sigma} \leq \boldsymbol{\sigma}^{\boldsymbol{e}}_{\min} \right\}, \tag{4.4}$$

$$\text{where} \quad M^e = \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & M \end{bmatrix} \in \mathbb{R}^{l\times(l+1)} \quad \text{and} \quad \boldsymbol{\sigma}^{\boldsymbol{e}}_{\min} = \begin{bmatrix} 0 \\ \boldsymbol{\sigma}_{\min} \end{bmatrix} \in \mathbb{R}^l.$$

The resulting polytope is a prism, i.e. the product of a polytope with an interval. More precisely, the prism $\mathcal{P}$ is the result of the product of the l-simplex $\mathcal{P}_{\text{sa}}$ and the interval $[-\pi, \pi]$, where the precommutation angle σ_0 is defined. As a consequence, it has $2(l+1)$ vertices. A visualization of such prism in the case $D=3$ is offered in Figure 4.1.

The prism $\mathcal{P}$ is a convex and compact set. It is a subpolytope of the D-hyperbox defined by (2.45). This means that its $2D$ vertices are a subset of the 2^D vertices of the hyperbox. Limiting the search space of SMGO to the prism $\mathcal{P}$ would implicitly enforce the linear constraints (4.2), but also reduce the Lebesgue measure of the search space [13]. In order to do that, it is sufficient to account for candidate points that are only inside $\mathcal{P}$. Checking all the candidate and cloud points generated throughout the algorithm and filtering out the points that are generated outside of $\mathcal{P}$ is cumbersome and impractical, especially for the cloud of initial points and for the points cast in the trust region. This consideration, in conjunction with the properties of the prismatic search space $\mathcal{P}$, motivated some slight modifications of the algorithm mechanics.

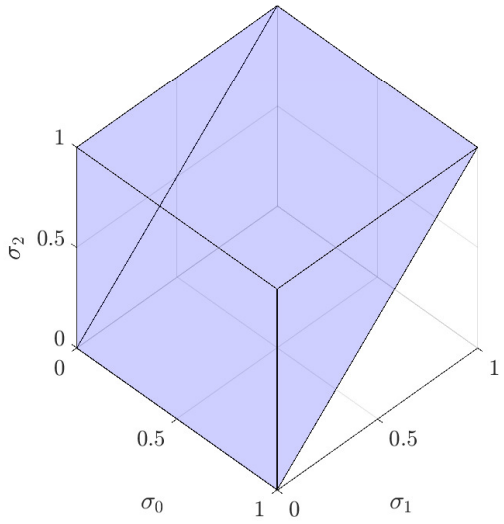


Figure 4.1: Plot of the prism for a normalized 3D search space.

4.1.1. Generation along coordinate directions

According to the spider web generation described in Section 3.2.2, the generation of new candidate points along coordinate direction requires the solution of the optimization problems described by equations (3.17) and (3.18). Thanks to the structure of the search space, this optimization problem can be circumvented.
Given a sample $\boldsymbol{\sigma}^{(n)} = \left[\sigma_0^{(n)}, \sigma_1^{(n)}, ..., \sigma_f^{(n)}\right]$, the lengths of the segments starting from $\boldsymbol{\sigma}^{(n)}$ and finishing on the border of $\mathcal{X}$ along the coordinate direction i can be written as

$$b_{+d}^{(i)} = \begin{cases} \pi - \sigma_0^{(n)} & \text{for } i = 0 \\ \sigma_{i+1}^{(n)} - \sigma_i^{(n)} & \text{for } i = 1, \dots, l-1 \\ \frac{\pi}{2} - \frac{\sigma_{\min}}{2} - \sigma_f^{(n)} & \text{for } i = l, \end{cases} \tag{4.5}$$

$$b_{-d}^{(i)} = \begin{cases} \sigma_0^{(n)} + \pi & \text{for } i = 0 \\ \sigma_1^{(n)} - \sigma_{\min} & \text{for } i = 1 \\ \sigma_i^{(n)} - \sigma_{i-1}^{(n)} & \text{for } i = 2, \dots, l. \end{cases} \tag{4.6}$$

4.1.2. Initialization and trust region

The original technique to generate the initial points or the cloud of points for the trust region makes use of the Sobol distribution. As mentioned in Section 3.2.2, these points are generated inside a unitary hyperbox and do not respect the linear constraints described by (4.2).
Filtering out the points of a Sobol sequence is extremely expensive, especially for high dimensions. For example, out of the first million points of the 10-dimensional Sobol sequence, only 7 respect the constraints.

An alternative approach leverages the structure of the search space. As per the definition of polytopes [13], it is possible to express the search set with vertices $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_V\}$ as a convex combination of its vertices:

$$\mathcal{P} := \left\{ \sum_{i=1}^{V} q_i \boldsymbol{v_i} \mid q_1, \dots, q_V \geq 0, \sum_{i=1}^{V} q_i = 1 \right\}. \tag{4.7}$$

We can then univocally map each point of the search space into a precise vector of weights $\boldsymbol{q} = [q_1, \dots, q_V]$. The problem of finding a uniform distribution is then shifted into the weights space.
A first approach would be to sample the weights from a univariate and uniform probability distribution in $[0, 1]$ and then normalize them with respect to their sum, to obtain a unitary sum. Nonetheless, due to the central limit theorem, the outcome of this procedure is a multivariate Gaussian distribution. As a result, the distribution no longer provides good coverage of the search space. On the contrary, it concentrates at its center, especially as the dimensionality increases.

A better idea would be to generate the weights using a multivariate probability distribution, such that the condition of a unitary sum is a priori satisfied. For this reason, we consider the symmetric Dirichlet distribution. As described in [9], its density function can be written as

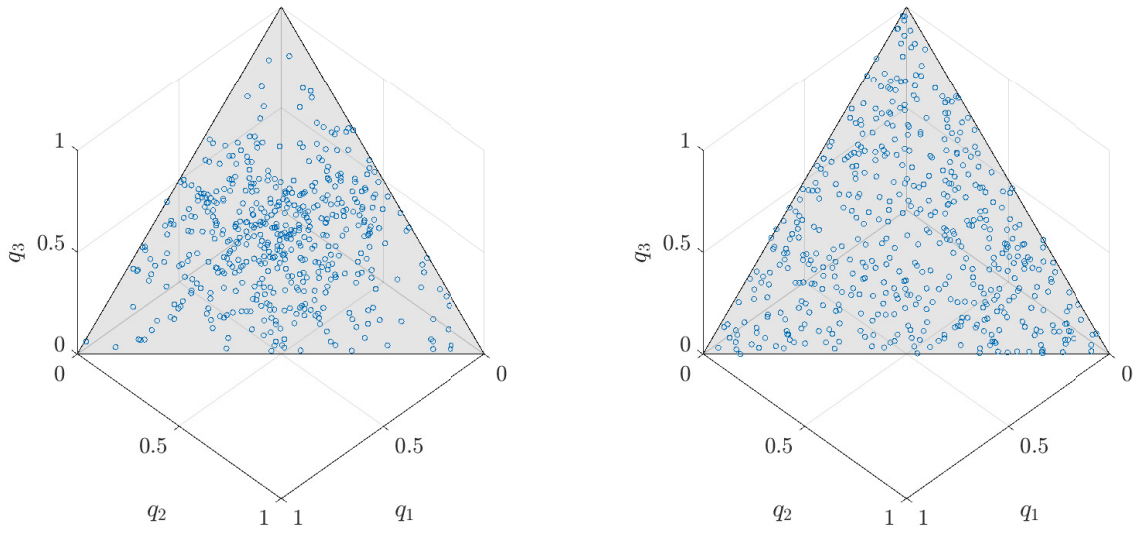
$$\mathrm{Dir}(\boldsymbol{q}, a) = \frac{\Gamma(V\,a)}{\Gamma(a)^V} \prod_{i=1}^{V} q_i^{a-1}, \tag{4.8}$$

where $\Gamma(a)$ is the gamma distribution with shape parameter a.
A single set of weights $\boldsymbol{q}$ sampled from a Dirichlet distribution describes a point that belongs to a standard simplex, i.e. the $V-1$ simplex that has as vertices the standard unit vectors of $\mathbb{R}^V$. When $a = 1$, if we sample multiple $\boldsymbol{q}$ from the Dirichlet distribution, we cover the standard simplex homogeneously. This can be seen in Figure 4.2; the points obtained through Dirichlet offer better coverage of the standard simplex than those

obtained from a univariate distribution and normalized with respect to their sum.

Now that we have a method to obtain a set of weights distributed uniformly on a standard simplex, we can use the linear combination of these weights with the vertices of a simplex to obtain a uniform distribution of points inside the latter.
This method is relevant for the prismatic search space $\mathcal{P}$, as it is obtained by taking the product polytope of a simplex and an interval, and can therefore be applied to both of these components as per equation (4.7).

Additionally, the Dirichlet distribution provides a set of cloud points centered around the current optimum. In doing so, we use instead of the corners, a fraction of the vectors linking the current sample to the corners. The size of these generating vectors can be adjusted to increase or decrease the size of the trust region as needed.
The number of points N_{cld} is set as a tuning parameter of the algorithm. It specifies both the number of points to be cast during the initialization and the number of points contained by every single instance of the trust region.
We remark that the points obtained through the Dirichlet distribution are no longer deterministic, as emphasized in [21] for the Sobol distribution. Similarly to what is done with the Sobol sequence in `MATLAB`, it is possible to save in the memory a set of weights and always use it, or fix the seed of the gamma distribution involved. This feature is neglected since it is of no interest to the case study.



(a) Weights sampled from uniform distribution. (b) Weights sampled from Dirichlet distribution.

Figure 4.2: Comparison of the distribution of 500 samples on the standard simplex.

4.2. Tuning of SMGO

In this Section, we explain the criteria followed in the tuning of SMGO for the OPP problem.

4.2.1. Risk factor

The OPP computation is an offline optimization where the major achievement is finding the global minimum among all the local ones. Indeed, at the end of the optimization, it is not important the number of feasible points that we have. Differently from a real-time scenario [21, 24], it is also not relevant to always find a feasible solution throughout the optimization. These considerations on the application of our problem lead to the choice of a risky algorithm, with $\Delta = 1$.
This means that the algorithm explores more, and does not limit itself to a single region in the case of a search set where the set of feasible points is not complex. From the analysis of the cost delivered in Chapter 5, we see that this is relevant to our problem, at least for the 3D case.

4.2.2. Minimum distance

As already mentioned in Section 2.4.2, the cost function is piecewise constant. As a consequence, the sampling of a new point $x^{\langle n \rangle}$ that is too close to a point already sampled is useless since simply wastes a cost evaluation. For this reason, a minimum distance $\underline{d}$ should be set, so that if the new point does not satisfy the criterion

$$\left\| \boldsymbol{x}^{\langle n \rangle} - \boldsymbol{x}^{(i)} \right\| \leq \underline{d} \quad \forall \boldsymbol{x}^{(i)} \in \boldsymbol{X}^{\langle n \rangle}, \tag{4.9}$$

it should not be sampled.
The concept of a minimum distance and the check (4.9) are already present in the algorithm, but $\underline{d}$ is set to a default value of 10^{-9}.
To better suit the algorithm to our specific problem, we set $\underline{d}$ as a tuning parameter. In addition to that, the generation of new candidate points is properly limited, to avoid clusters that clutter regions where the cost function is constant by construction.

To do so, we compute the sampling angular distance

$$\sigma_s = T_s\,\omega_s, \tag{4.10}$$

which is simply the angle traveled in T_s.
Now one can set the minimum distance

$$\underline{d} = \frac{\sigma_s}{2}. \tag{4.11}$$

We then perform a check similar to (4.9) on the points that are being added to the set of candidates.
Also, the minimum radius of the trust region $\underline{v}$ is adapted to the value of $\underline{d}$ so that the radius of the smallest trust region is still bigger than the minimum distance.

4.2.3. Other parameters

The choice of the parameter α regulates the trade-off between exploitation and exploration. The tuning of this parameter is object of further discussion in Chapter 5.
The value $\bar{B}$ controls the number of candidate points introduced on a certain iteration of the optimization. Notice that this amount is not fixed, but increases with the iteration. For the first trials of the algorithm, it is left to the default value of $\bar{B} = 4$. However, a new candidate point generation, which is introduced in 5, replaces this tuning parameter with a more direct one.
Parameter β is left to the default value of 0.1.

5 | Enhancing SMGO

This Chapter evaluates the basic setup of the OPP optimization problem and discusses the initial findings in Section 5.1. The study leads to some general observations about certain aspects of SMGO. In Section 5.2, the trust region concept is investigated and enhanced. In Section 5.3, a new approach to adjust the exploration-exploitation trade-off is introduced and tested. Section 5.4 presents a new candidate point mechanism that achieves the same results in significantly less time. Finally, we provide a brief commentary on the general applicability of these enhancements, supported by the results in Appendix A.

5.1. Early results

The set-up of SMGO for the OPP computation introduced in Chapter 4 is now tested on the simplest case that preserves the prismatic search space, i.e. the optimization of two switching angles ($D = 3$). In this case, it is easy to find the global minimum through a large number of different gradient-based optimization. The value of the optimum and the optimal point are reported in Table 5.1. A visualization of the cost function in the normalized space can be seen in Figure 5.1.

z	σ_0	σ_1	σ_2
32.373	2.796	1.203	1.436

Table 5.1: Best feasible point in the 3D OPP problem.

The testing procedure involves selecting different values of the tuning parameter, α, which governs the exploration-exploitation trade-off. When α is low we prioritize exploitation, while when α is high we favor exploration. We conduct 10 trials for each α value to assess the consistency of the optimizer, with a budget of 500 evaluations that can be allocated according to α. On the first tests, SMGO delivered poor results. Table 5.2 reports the best, worst and mean optimum found by the ten trials, the mean number of exploited points, the number of trials that converged in the neighborhood of the global optimum,

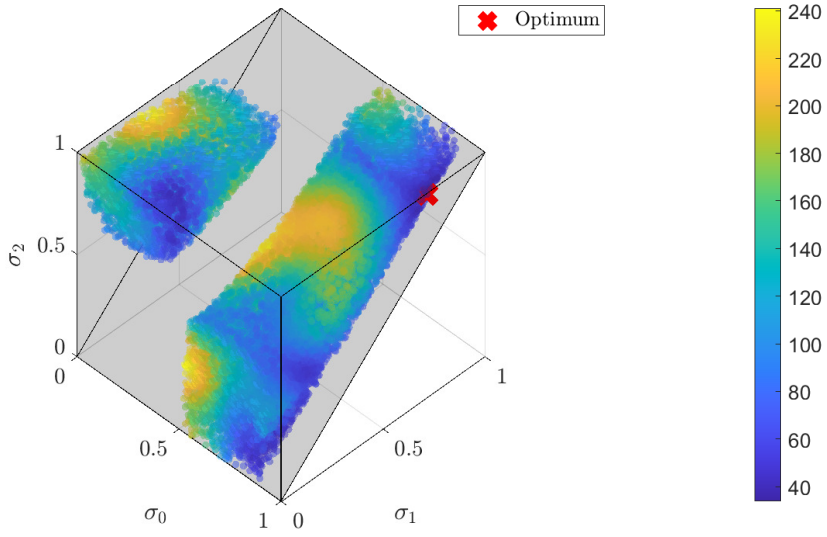


Figure 5.1: Colormap of the cost function in the feasible regions. The red cross is the global optimum.

and the mean *self-time* required by one trial. The self-time is the amount of time required by the computations of the algorithm itself, ignoring the time spent to evaluate the cost function.

α	Best	Worst	Mean	Exploitation	Converged	Time[s]
0.0005	32.379	45.129	36.240	369.1/500	6/10	65.2
0.001	32.397	40.821	36.563	293.5/500	5/10	66.6
0.005	32.528	41.289	37.646	108.4/500	4/10	69.4
0.01	33.265	42.850	38.339	66.8/500	4/10	68.8
0.05	34.016	45.331	41.175	13.9/500	7/10	67.8

Table 5.2: Original SMGO with different tuning of α.

Table 5.2 provides valuable insights into the optimization results. Firstly, none of the settings of α resulted in consistent convergence across all 10 trials. For each α value, at least three trials failed to reach the region containing the optimum. In other words, they converged to suboptimal local minima. This suggests that the effectiveness of the method is heavily dependent on the choice of the initial point, or alternatively, it could be said that the convergence rate is inadequate.
Secondly, the effects of the tuning of α can be observed. The obvious effect is that the number of exploited points within the budget of 500 function evaluations decreases as α

increases. Additionally, we notice that the best optimum value worsens. This is due to a drawback of the definition of α as the tuning parameter responsible for the trade-off. In fact, when we increase the value of α we not only favor exploration over exploitation, but also exclude from the search space a certain ball around the current best point. This issue is tackled in Section 5.3.

5.2. Extended trust region

Among the failed trials, the following behavior is observed: The algorithm keeps exploiting a small region around the first feasible point, eventually reaching one local minimum, but wasting too many iterations doing so. This misbehavior is known in optimization as *over-expoloitation*. Before entering the details of this issue, a remark on the nature of the exploitation and the exploration is due.

Remark. *during exploitation, the only aim of the algorithm is to find a better minimum; during exploration, the only aim of the algorithm is to reduce the overall uncertainty of the SM-model.*

As a direct consequence, one can infer that the optimum cannot be found using only exploration, because the exploration merit function does not advocate points with a lower cost function.
The rationale of the shrinkage of the trust region, as explained in Section 3.2.3, limits the exploitation mode to a small ball around the current optimum. If the global one is on a different region of the search space, as it could happen in the case study of $D = 3$, the algorithm can get closer to the right region only through exploration. This implies that the probability of locating the global minimum is solely reliant on the level of uncertainty present in the area where it is situated. Only when an exploration-based sampling discovers a better solution in that area, the trust region will encompass the zone containing the global minimum. This feature deeply affects the rate of convergence of the algorithm.

In this study, we adopt a new approach for the trust region. In order to facilitate the ability of the algorithm to transition from a local minimum to a better one, we eliminate the idea of a trust region serving as a restriction on the exploitation area. Consequently, during the entire course of the optimization, the algorithm performs exploitation selecting points that can improve the current optimum from among all candidate points, as well as an additional set of cloud points created around the optimum. The cloud points are contained within a ball that expands and contracts based on the criteria of the trust region

as defined in the original SMGO formulation. This approach enhances the algorithm's ability to move from a local minimum to an improved solution.
Moreover, the computation of the cloud of points around the optimum is suspended when the size of the trust region is minimal. This indicates that the algorithm has already sufficiently exploited the current optimum region and should shift its focus to other areas. By implementing this measure, the algorithm can save computational effort.

α	Best	Worst	Mean	Exploitation	Converged	Time[s]
0.0005	32.378	40.825	33.541	268.9/500	9/10	48.9
0.001	32.394	40.881	35.141	250.5/500	7/10	48.7
0.005	32.462	42.625	36.322	82.5/500	6/10	50.6
0.01	33.027	43.626	36.341	49.1/500	8/10	49.3
0.05	35.732	50.681	39.824	14.9/500	9/10	49.2

Table 5.3: SMGO with extended trust region, for different tuning of α.

The effects of the extended trust region are shown in Table 5.3. We notice how the number of converged trials is generally increased, as well as the mean optimum. Nevertheless, the influence of alpha is still clearly visible, both in the refinement of the best optimum and in the number of exploited points. Finally, the column corresponding to the time shows a significant decrease.

Algorithm 5.1 Exploitation with extended trust region

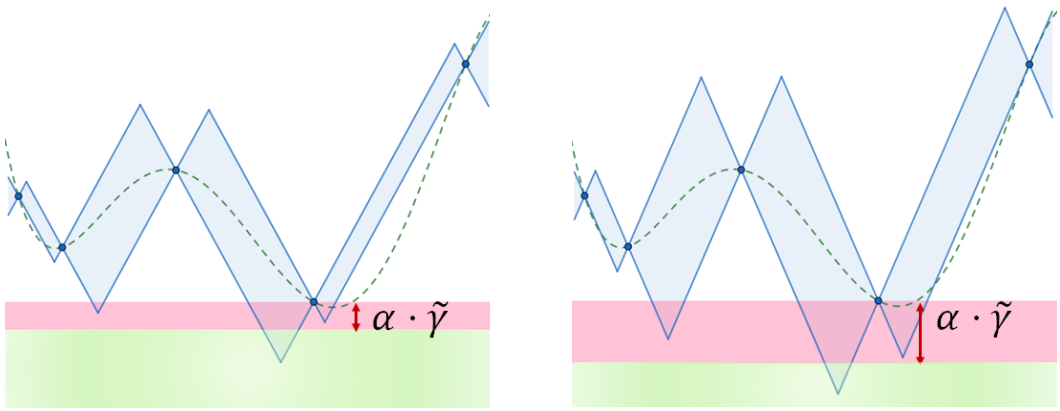
while *iteration* n *within the budget* N_{max} **do**
  Perform steps prior to exploitation, as stated in Algorithm 3.1
  if *the new best sample* $\boldsymbol{x}^{*\langle n \rangle}$ *is updated* **then**
    Activate trust region $\mathcal{T}^{\langle n \rangle}$ with maximum size $\bar{v}$ around $\boldsymbol{x}^{*\langle n \rangle}$.
  end if
  if The trust region $\mathcal{T}^{\langle n \rangle}$ is not active **then**
    Exploitation (or exploration) on $\boldsymbol{E}^{\langle n \rangle}$.
    Assign test point for next iteration $\boldsymbol{x}^{(n+1)} \leftarrow \boldsymbol{x}_{\theta}^{\langle n \rangle} \left(\text{ or } \boldsymbol{x}_{\phi}^{\langle n \rangle} \right)$.
  else
    if *the current size* $v^{\langle n \rangle} > \underline{v}$ **then**
      if *exploitation on* n *was successful* **then**
        The size of $\mathcal{T}^{\langle n+1 \rangle}$ is increased $v^{\langle n+1 \rangle} = \min\left(\bar{v}, \frac{1}{\kappa} v^{\langle n \rangle}\right)$.
      else
        The size of $\mathcal{T}^{\langle n+1 \rangle}$ is decreased $v^{\langle n+1 \rangle} = \max\left(\underline{v}, \kappa v^{\langle n \rangle}\right)$.
      end if
      Exploitation (or exploration) on $\mathcal{T}^{\langle n \rangle} \cup \boldsymbol{E}^{\langle \boldsymbol{n} \rangle}$.
      Assign test point for next iteration $\boldsymbol{x}^{(n+1)} \leftarrow \boldsymbol{x}_{\theta}^{\langle n \rangle} \left(\text{ or } \boldsymbol{x}_{\phi}^{\langle n \rangle} \right)$.
    else
      Deactivate trust region $\mathcal{T}^{\langle n \rangle}$.
    end if
  end if
  Go to next iteration $n \leftarrow n + 1$.
end while

5.3. Adaptive alpha

The trade-off between exploitation and exploration constitutes a crucial feature when it comes to the rate of convergence of SMGO.

Remark. *The exploration-exploitation trade-off depends on the user's objectives. If the goal is to obtain a refined solution, the algorithm should prioritize exploitation and use its budget accordingly. Instead, if the aim is to ensure that the search space has been thoroughly explored, the compromise should lean towards exploration. The nature of this trade-off has been extensively studied in the literature on global optimization. An overview of the topic is offered in [27].*

The tuning parameter that gives control over this issue is α. Combined with the estimated Lipschitz constant $\tilde{\gamma}$, it provides a threshold for the exploitation, as stated in (3.26). The thresholds for two different SM-models of the same cost function are represented by the red region in Figure 5.2. The points that can be sampled through exploitation are those with a lower bound which is beneath the value of the current optimum minus the threshold. In other words, only the point with a lower bound in the green region of Figure 5.2.



(a) Threshold with $\tilde{\gamma}_1$. (b) Threshold with $\tilde{\gamma}_2 > \tilde{\gamma}_1$.

Figure 5.2: Expected improvement threshold for two SM-models of the same cost function. The points with lower bound in the green region can be exploited.

Always from the same Figure, we see that the lower bound in the neighborhood of the current minimum cannot overcome the threshold. This happens because the lower bounds of the candidate points in the neighborhood of the optimum belong all to the optimum hypercone [24]. For whatever Lipschitz constant $\tilde{\gamma}$, a value below the threshold can be achieved only starting from a distance α, ruling out from the exploitation all the points

inside such ball.

Remark. *The value of α sets a limit for exploitation. More precisely, it sets a limit on the proximity to the current optimum.*

As a result, an increase of α prevents the optimization from properly refining the optimum, even when the SM-model is confident that the global optimum is near the current best point.
Another drawback of the original implementation of the tuning parameter α is that is not clear how many iterations of exploitation will be performed out of the full budget. Moreover, this relationship is highly dependent on the specific cost function. For these reasons, the tuning of α could require a trial and error procedure, before the user finds the value that better suits his purpose.

In this thesis, a new approach is proposed. Instead of using a fixed α, we could define a different criterion to balance exploitation and exploration. By selecting a different tuning parameter, we can control more directly the trade-off between the two, and at the same time decoupling the side effect of the fixed α, which sets a limit on the precision of the optimum.

5.3.1. Ratio PI controller

A balanced trade-off between exploitation and exploration makes fair use of the budget that we have, be it a limit on time or on the number of cost function evaluations. Thus, we can consider the exploration-to-exploitation ratio as an alternative metric to evaluate this trade-off:

$$R = \frac{N_\phi}{N_\theta}, \tag{5.1}$$

where N_ϕ is the number of exploration performed and N_θ is the number of exploitation. We can define a desirable ratio R_{ref} which represents a use of the budget that reflects the objectives of the optimization. Notice that we do not want to enforce to the algorithm an exploitation every R_{ref} exploration. This would be inappropriate since it would not account for the specific situation in which the algorithm finds itself. For example, it would be useless to enforce an exploitation when the whole set of candidate points appears to have lower bounds higher than the current optimum. Instead, we want α to settle on a value that is good for the current SM model and the current set of candidate points.

Remark. *With this technique, we substituted the degree of freedom provided by α with a reference for the ratio R_{ref}. This reference ratio is now responsible for the exploration-to-exploitation trade-off. A higher value prioritizes finding the global minimum, while a*

lower one prioritizes obtaining a more refined local optimum.

We define the error e at iteration n as

$$e(n) = R_{\text{ref}} - R(n), \tag{5.2}$$

where $R(n)$ is the ratio obtained from the number of iterations spent in exploration and exploitation, until iteration n. The value of $\alpha(n)$ is then obtained from the law

$$\alpha(n) = \max\left(K_P\, e(n) + K_I\, \Sigma_e(n),\ \alpha_{\min}\right), \tag{5.3}$$

where

$$\Sigma_e(n) = \begin{cases} \Sigma_e(n-1), & \text{if } \alpha(n-1) = \alpha_{\min}, \\ \Sigma_e(n-1) + e(n-1) & \text{otherwise.} \end{cases} \tag{5.4}$$

Equation (5.3) can be seen as a discrete PI controller with a lower saturation. The following elements are a part of it:

1. K_P and K_I, that are the proportional and the integral gain, and they are empirically tuned.
2. $\Sigma_e(n)$, that is the integral error.
3. $\alpha_{\min}$, which corresponds to the lower saturation of the controller. It must be greater than 0, otherwise, the exploitation could sample points that, according to the current SM-model, can only be worse than the current optimum. It can also be user-defined, especially if there is no need for a very well refined global optimum, but is sufficient to find the region where it lies.

We remark that in this ratio controller we did not define an upper saturation $\alpha_{\max}$. However, the implementation of an upper saturation would amend the overshooting of the control variable α. Therefore, $\alpha_{\max}$ should be set to the lowest value of α that excludes all the candidate points from exploitation. such value changes with the iteration, since it depends on the candidate points currently available, and should be updated accordingly.

When the output of the controller reaches the lower saturation limit, the error between the set-point R_{ref} and the actual value may continue to accumulate in $\Sigma_e(n)$, leading to what is called *windup*.
Notice that, since the rationale behind α adaptive is assigned to be a PI controller, the general issues concerning control theory are reflected in the optimization problem. In this case, the windup would heavily delay the exploration phase, falling back into the over-

exploitation and affecting the rate of convergence. An anti-windup technique is needed to address this issue, by preventing the integrator term from accumulating further when the output is saturated. This can be achieved by *clamping* the integrator term, as it is done in the update law (5.4).

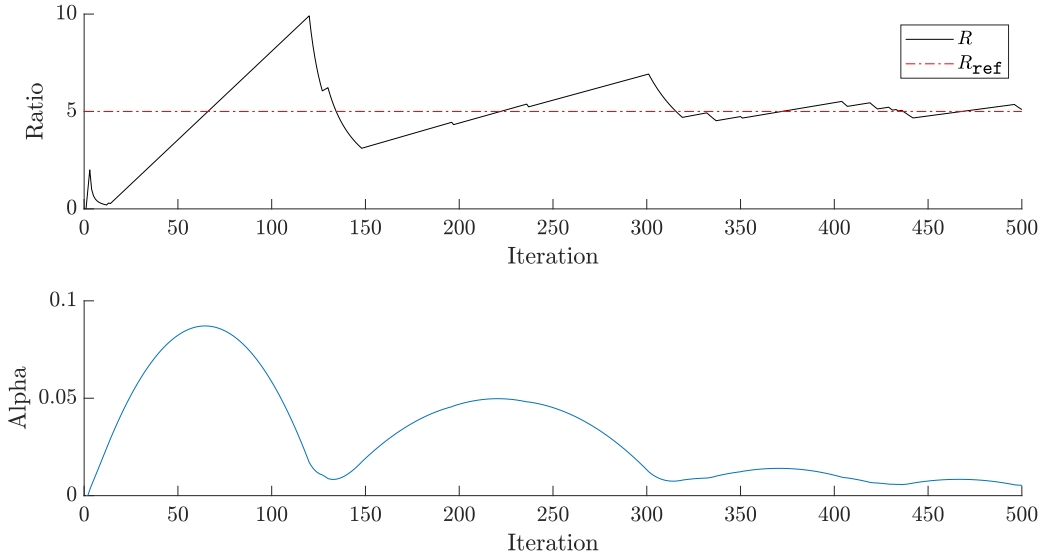


Figure 5.3: Reference tracking and output of the R PI controller.

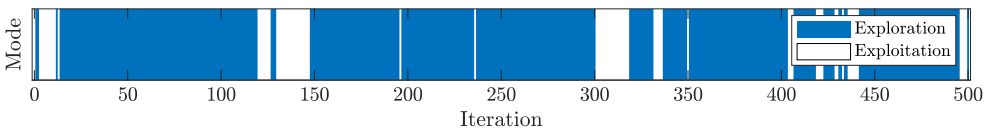


Figure 5.4: Colormap of exploration and exploitation throughout the whole optimization.

Figure 5.3 offers the plots of the reference tracking and the output variable of such controller. We can see how when the value of R overcomes the reference value, the value of α is decreased, to favor an exploitation - viceversa when the value of R is below R_{ref}. One can notice also that the effect of α on the ratio R is neither straightforward nor immediate. For example, on some iterations R decreases (i.e. an exploitation is performed) despite the increasing α; on others R keeps increasing despite the decreasing α. This happens because none of the candidate points currently available satisfies the threshold.

Remark. *The quality of the reference tracking of R is not relevant to the success of the trial. A bad tracking could mean that the choice of the reference R_{ref} is poor or that the candidate points set is not populated enough, but also that the global minimum was found early.*

Indeed, the primary purpose of this structure is not to achieve good tracking, but to adapt to varying situations in terms of the estimate of the Lipschitz constant and the availability of candidate points. The reason why we define it *adaptive* α is now evident. In Figure 5.4 we can see how the adaptive α technique prevents over-exploitation. The number of iterations on which exploitation is performed is distributed over the entire budget. Table 5.4 contains the results of the same performance test of the previous sections. Each row corresponds to a different setting of the tuning parameter R_{ref}. For clarity, in the table we use the indicator

$$\Xi = \frac{1}{R_{\text{ref}} + 1} \tag{5.5}$$

that symbolizes the percentage of the budget of iterations that is reserved for exploitation.

Ξ[%]	**Best**	**Worst**	**Mean**	**Exploitation**	**Converged**	**Time[s]**
50%	32.391	45.864	34.417	191.8/500	9/10	44.8
33%	32.379	32.525	32.459	157.2/500	10/10	45.6
17%	32.458	34.454	32.988	70.8/500	10/10	46.8
9%	32.507	41.413	34.960	38.2/500	9/10	47.9

Table 5.4: SMGO with extended trust region and adaptive α, for different values of Ξ.

We can list the following observations:

- The consistency is generally increased: in the second and third rows the optimization converges to the optimal region on all ten trials.
- For all settings, the optimization is capable of finding the optimum with sufficient accuracy. There is no longer a limit on the accuracy of the optimum.
- The value of the ratio R is close to the reference one, but not exactly equal (from top to bottom, the theoretical value should be: 250/500, 166/500, 83/500, 45/500).

Threshold as optimality gap

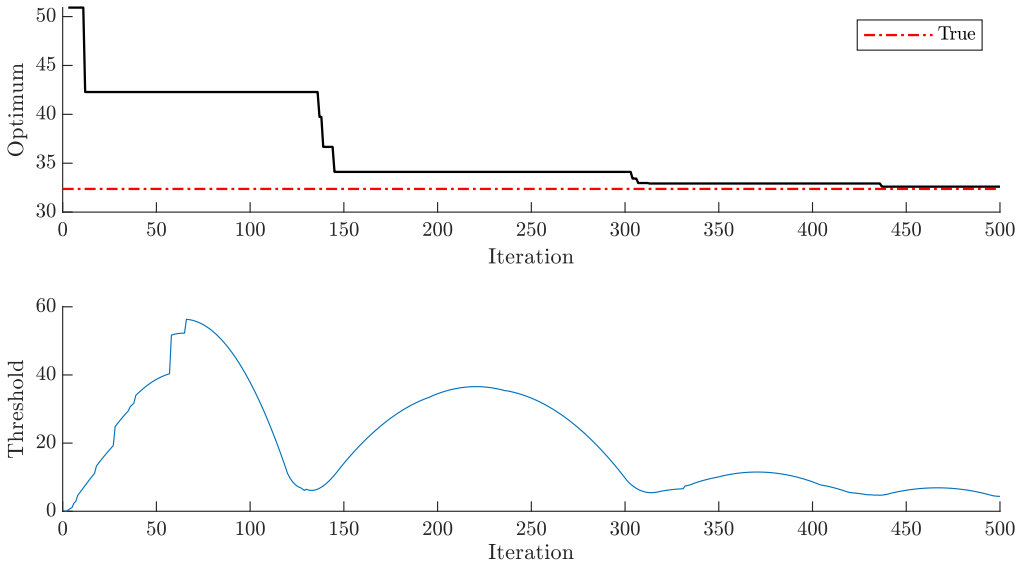


Figure 5.5: Best point (upper plot) and improvement threshold (lower plot) over iteration. The red dotted line is the true optimum.

Figure 5.5 illustrates the optimization trial corresponding to the PI realization discussed in the previous section. The upper plot marks the descent of the best point towards the true optimum, while the lower tracks and threshold over iteration. We can notice that the second plot appears to be a scaled plot of α of Figure 5.3, but not identical. Recalling that the threshold is defined as $\alpha\tilde{\gamma}$, the difference is due to the update of $\tilde{\gamma}$. The updates happen mainly in the first 70 iterations.

The value of the threshold over iteration n offers an insightful metric. It could be considered as an upper bound to the optimality gap $\delta^{\langle n \rangle}$ from (3.24), but two conditions must be met.

1. The value of the estimate $\tilde{\gamma}$ must correspond to the real Lipschitz constant γ.
2. The set of candidate points must include the coordinate of the optimum x^*.

Nevertheless, even when these two conditions are not met, the threshold is a useful indicator when assessing the overall performance of the optimization. By looking at it, we can see how convinced SMGO is about the current optimum, given the current candidate point set $\boldsymbol{E}^{\langle n \rangle}$. We can use this piece of information to understand if the tuning of R_{ref} is adequate, or we can gauge the quality of the candidate points generation mechanism.

5.4. Sunburst point generation

The default candidate point generation technique is the *spider web generation*. A drawback of this technique is that the number of candidate points is polynomial with the number of iterations n, as stated in Section 3.2.2. This increase is mirrored also in the memory demand. This is not very suited for the offline computation of the OPP, since we want to perform as much sampling as possible, to inject as much information as possible into the SM-model.
In this thesis, we provide a new generation technique, that takes into account the following considerations:

- The new points that are added to the data set of candidate points must differ from the candidate points that already are inside it.
- The number of candidate points should increase linearly with the iteration.
- The cloud of candidate points should get finer with the iteration, but just in the interesting regions.
- The generation mechanism should be compliant with Theorem 3.1.

Keeping in mind these principles, we can define a new rationale for the candidate point generation. At every iteration, we add the midpoints of the segments linking the last sample $\boldsymbol{x}^{(n)}$ to a fixed number of other points.
The database $\boldsymbol{E}^{\langle n \rangle}$ is enlarged at each iteration with

$$\boldsymbol{Y}_s^{(n)} = \left\{ \frac{\boldsymbol{x}^{(n)} + \boldsymbol{p}^{(j)}}{2} \quad \forall \quad \boldsymbol{p} \in \mathcal{W}^{(n)} \right\}, \tag{5.6}$$

so that

$$\boldsymbol{E}^{\langle n \rangle} = \boldsymbol{E}^{\langle n-1 \rangle} \cup \boldsymbol{Y}_s^{(n)}. \tag{5.7}$$

The set of points $\mathcal{W}^{(n)}$ contains two kinds of endpoint $\boldsymbol{p}$:

1. Endpoints along the coordinate directions.
2. The N_{cdpt} closest candidate points.

The first kind depends on the mode used to sample $\boldsymbol{x}^{(n)}$, according to the implementation explained in Algorithm 5.2.
This generation technique allows also to easily integrate into the algorithm a check on the minimum distance, as explained in 4.2.2. In fact, it is enough to remove from the

endpoint set $\mathcal{W}^{(n)}$ all the endpoints $\boldsymbol{p}$ that do not respect the condition

$$\left\|\boldsymbol{x}^{(n)} - \boldsymbol{p}\right\| \leq \underline{d}. \tag{5.8}$$

A comparison of this candidate point mechanism with the spider web is offered in Figures 5.6 and 5.7. The name *Sunburst point generation* is inspired by the shape of the segments linking the new point to the closest one.

The number of candidate points added at each iteration is now fixed and it is equal to $2D + N_{\text{cdpt}}$. The tuning parameter N_{cdpt} substitutes the parameter $\bar{B}$, offering to the user a degree of freedom: A higher value of N_{cdpt} makes the algorithm more memory demanding and time consuming, but allows to choose the new sample from a denser set of candidates.

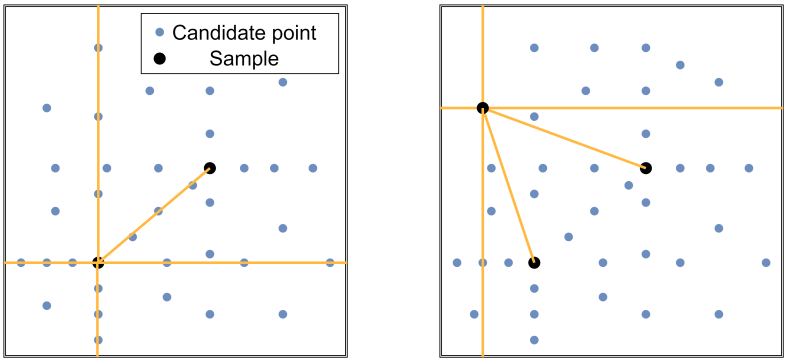


Figure 5.6: Two consecutive iterations of spider web generation. Candidate points are generated on the orange segments.

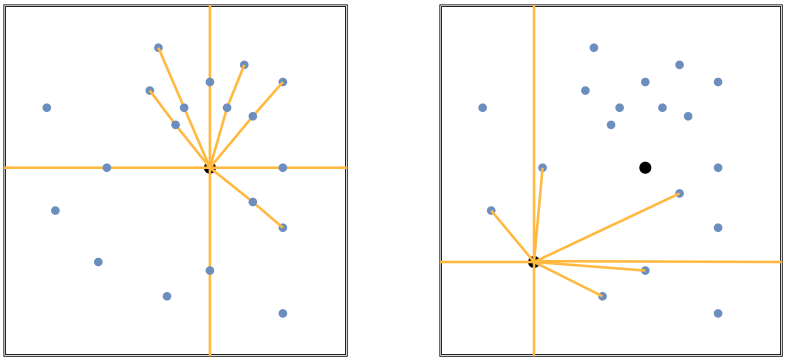


Figure 5.7: Two consecutive iterations of sunburst generation. Candidate points are generated on the orange segments.

The presence of endpoints along the coordinate direction makes this generation mechanism compliant with Theorem 3.1, guaranteeing that the whole search space will be eventually covered.
On top of that, the distribution of candidate points in the search space gets finer only in the regions that are repeatedly sampled, which correspond to the most interesting ones.

From Table 5.5, we see that with the new candidate points generation technique the time consumed by the algorithm is greatly reduced, and the quality of the results is only slightly compromised. For example, in the last row, the number of trials that converge within the budget is reduced. Nevertheless, this reduction of the time consumed proves to be vital in the offline optimization of pulse patterns with more than 5 switches.

Ξ[%]	**Best**	**Worst**	**Mean**	**Exploitation**	**Converged**	**Time[s]**
50%	32.380	40.870	33.627	179.7/500	9/10	2.1
33%	32.382	40.828	33.358	151.0/500	9/10	2.0
17%	32.389	33.019	32.574	69.5/500	10/10	2.0
9%	32.845	47.571	38.481	27.0/500	7/10	1.7

Table 5.5: SMGO with extended trust region, adaptive α and sunburst generation.

Algorithm 5.2 Sunburst candidate point generation

while *iteration* n *within the budget* N_{max} **do**
  if *the sample* $\boldsymbol{x}^{(n)}$ *was found in exploitation mode* **then**
    Find the N_{cdpt} closest point and collect them in $\mathcal{W}^{(n)}$.
    Compute along the coordinate directions $\pm\hat{\boldsymbol{a}}_d$ with $d = 1, \dots, D$ the points at distance $v^{\langle n \rangle}$ (3.29) and add them to $\mathcal{W}^{(n)}$.
  else
    Find the N_{cdpt} closest point and collect them in $\mathcal{W}^{(n)}$.
    Compute along the coordinate directions $\pm\hat{\boldsymbol{a}}_d$ with $d = 1, \dots, D$ the points at distance $\frac{b^{(i)}_{\pm d}}{2}$ with (4.5) and (4.6), add them to $\mathcal{W}^{(n)}$.
  end if
  Remove identical points, points closer than $\underline{d}$ and points equal to $\boldsymbol{x}^{(n)}$ from $\mathcal{W}^{(n)}$.
  Generate new candidate points (5.6).
end while

5.5. On the generality of the enhancement

Despite being developed on the OPP problem, the three improvements of the previous Sections are valid for any application. To check if the enhancement they can bring into the algorithm is general, they have been tested one by one on a set of 14 benchmark functions, taken from [24].
The summary of each test and the changes in performance are documented in Appendix A. All the tests are initialized from the same points. The other features of the algorithm are kept identical. The test has been run for two different settings of the risk factor.

- **For $\Delta = 1$**: The new version of SMGO, i.e. SMGO with all the new components, performs as well as the original algorithm in almost all tests except three, but in way less time.
- **For $\Delta = 0.2$**: This is the default value of Δ. The new version of SMGO performs better than the original algorithm in seven tests, and it performs worse in just one.

We remark that in the tests where the performance of the new version of SMGO is worse, the original algorithm uses a large fraction of the budget for exploitation. The introduction of the new components favor exploration, but for some function using the whole budget in exploitation could be a better choice. This information can still be considered in the tuning of the new SMGO.

6 | Comparative study and performance tests

This Chapter presents some comparative results of SMGO against other optimization algorithms. In Section 6.1, we report the results of the application of SMGO to the 6D optimization. This is useful to get a better idea of the **rate of convergence** and the robustness of this optimization method for dimensions higher than three. In the following Sections, the results of the same problem obtained with other methods are reported.
In particular, SMGO is compared with gradient-based multistart and Bayesian optimization. The former represents the state-of-the-art in the industry, and its set-up in the OPP framework is explained in Section 6.2. The latter is a well-established black-box global optimization method, and it is introduced in Section 6.3.
All the computations are performed on a 24-core AMD Epyc 7402 2.8GHz, 1024GB.

6.1. SMGO

The principles and settings of this method have been largely discussed in the previous Chapters. In this Section, we simply want to report the results obtained on the six dimensions optimization problem, i.e., the quest for an optimal pulse pattern with five switching angles.

6.1.1. Results

Figure 6.1 offers an interesting overview of the rate of convergence of the optimizer, following the descent of the optimum on the ten trials, represented by the ten different lines.
This overview is paired with Figure 6.2, which gives an idea of the time required by this method. As expected, the time increases polynomially with the iterations [24].

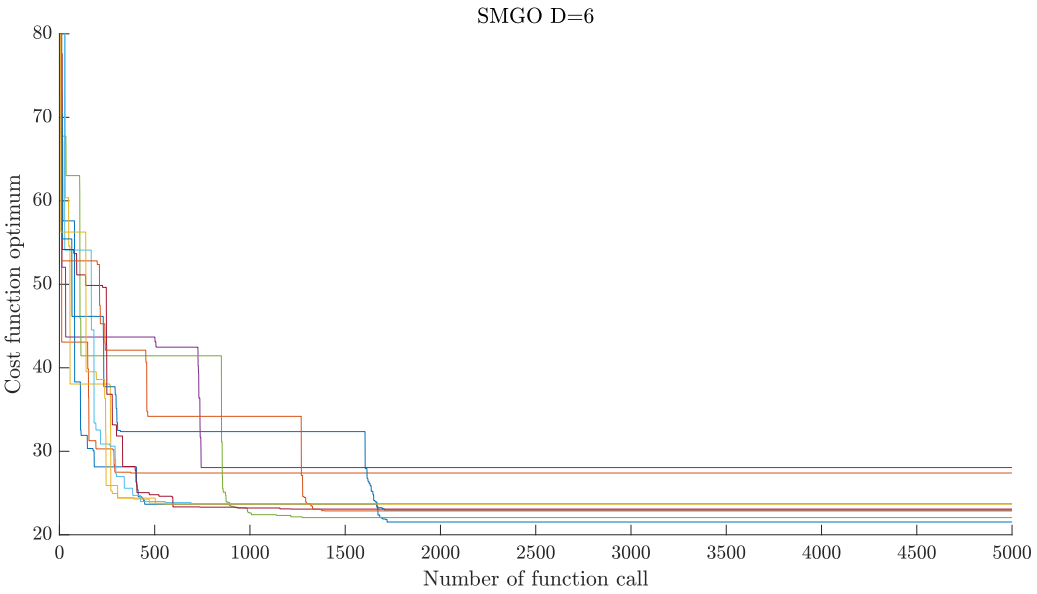


Figure 6.1: Optimum of ten trials of SMGO for the OPP computation with $D = 6$.

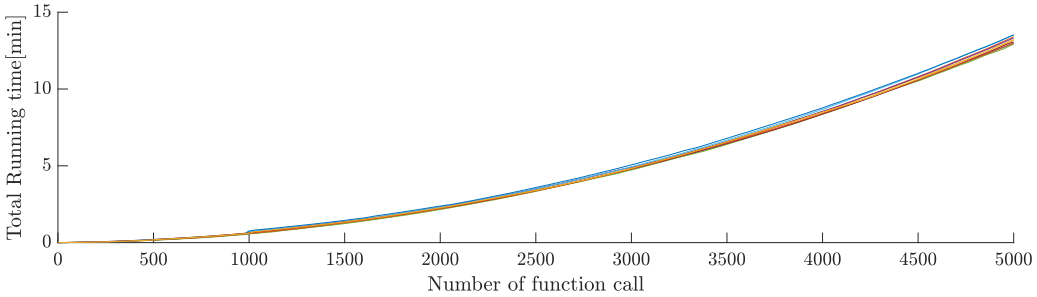


Figure 6.2: Total time of ten trials of SMGO for the OPP computation with $D = 6$.

test	$\boldsymbol{\sigma}_0$	$\boldsymbol{\sigma}_1$	$\boldsymbol{\sigma}_2$	$\boldsymbol{\sigma}_3$	$\boldsymbol{\sigma}_4$	$\boldsymbol{\sigma}_5$	**cost**	**const**
1	-0.349	0.186	0.893	0.983	1.345	1.479	21.53	0.216
2	-0.344	0.082	0.191	0.428	0.620	0.847	27.40	0.086
3	-0.357	0.092	1.177	1.268	1.394	1.509	23.71	0.004
4	2.798	0.065	0.315	0.781	0.962	1.491	28.05	0.303
5	-0.347	0.228	0.851	0.976	1.409	1.512	22.06	0.196
6	-0.347	0.100	1.179	1.324	1.415	1.498	23.72	0.060
7	-0.349	0.098	1.184	1.300	1.379	1.487	23.08	0.211
8	-0.347	0.075	0.437	0.588	1.188	1.342	22.96	0.058
9	2.792	0.066	0.115	1.210	1.387	1.529	22.86	0.050
10	-0.349	0.085	1.153	1.263	1.384	1.499	23.68	0.125

Table 6.1: Optimal points of SMGO for OPP with $D = 6$.

The consistency of the optimum found over the ten trials can be inferred from Table 6.1. It is easy to see that not all the trials converge to the same optimum, especially because there exist multiple minima with a similar cost.

Note that the optimal pulse patterns obtained in Table 6.1 can be grouped in two sets, according to their precommutation angle, which assumes mainly two values that differ by roughly π. The optimal pulse patterns of these two groups provide similar results, but cannot be identical.

6.2. Gradient-based multistart

With the name *gradient-based* methods we refer to a family of methods that make use of the first- and second-order optimality conditions. They are very powerful, but they exploit only local information in the quest for a minimum. This means that if they are applied to non-convex optimization problems, they could get stuck in local minima. If we want to use this approach to find the global optimum of a non-convex cost function, we must repeat the gradient-based optimization multiple times, with different initial conditions. This structure gives the name *multistart* to the method.

The starting points could be chosen randomly or using prior knowledge of the problem. In the case of an electrical drive problem, for example, a suitable initial point can be chosen according to the pulse pattern of traditional modulation techniques.
In this thesis, the initialization point is drawn from the Dirichlet random distribution, discussed in Chapter 4. This distribution guarantees that every gradient-based is initialized on a pulse pattern with switching angles in increasing order, so that the linear constraints on the initial point are respected.

The optimizer is `fmincon` of `MATLAB`. Among the manifold of gradient-based methods, this is a Sequentially Quadratic Programming (SQP) that makes use of the quasi-newton method Broyden–Fletcher–Goldfarb–Shanno (BFGS) [16]. The cost function is black-box, so *Exact Newton* or *Gauss-Newton* cannot be taken into account. BFGS tries to approximate the Hessian matrix using only information on the gradient, in order to check if the best point meets the second-order optimality condition. For this reason, it is the easiest and most straightforward method to be applied, and also the default for the `MATLAB` function `fmincon`.

The computation of the gradient is done with the central difference approach. This method approximates the value of the components of the gradient of the cost function in a slightly more expensive way, using two function evaluations for each coordinate direction,

but this is usually compensated by a faster rate of convergence, due to the accuracy of the gradient [6]. Therefore, given a cost function $f(\boldsymbol{x})$, the computation of the component along direction $\hat{\boldsymbol{a}}_d$ of the gradient $\nabla_x f(\boldsymbol{x})^T$ is executed as

$$\nabla_x f(\boldsymbol{x})^T \hat{\boldsymbol{a}}_d \approx \frac{f(\boldsymbol{x} + \mu\hat{\boldsymbol{a}}_d) - f(\boldsymbol{x} - \mu\hat{\boldsymbol{a}}_d)}{2\mu}, \tag{6.1}$$

where μ is the step size of the perturbation applied to compute the gradient.
We recall here that in the OPP problem we have a piecewise constant function. For this reason, when we define the step size μ in (6.1), it is necessary to account for the partition of the search space. If μ is too low, the first optimality condition is met in every point of the search space. Hence, recalling the definition of σ_s introduced in Section 4.2, we define

$$\mu = \sigma_s. \tag{6.2}$$

With the perturbation set according to (6.2), the THD-based cost function proves to be quasi-smooth. Indeed, the discontinuities due to the nature of the cost function do not affect the computation of the gradient, which proves to be quasi-continuous.

6.2.1. Results

In this study, the multistart optimization is executed up to a search space of dimension twenty. In Figure 6.3 is presented an example of ten trials of multistart, in the case of six optimization variables. One trial of the multistart optimization algorithm consists of a sequential execution of `fmincon` with different initial condition. When the previous instance of `fmincon` ends for a stopping condition, a new one is prompted. The test collects ten trials, each colored line corresponding to one of these.

Figure 6.3 depicts how the cost function optimum is improved over the budget of function calls. This plot gives an idea of the rate of convergence of the algorithm with respect to the number of iterations and it is useful for a comparison with the other global methods, but this interpretation must be evaluated with some warnings. Indeed, in the sequence of optima only the minima found at the end of each instance of `fmincon` are considered. This gives each trial the shape of a linear interpolation. This justifies also the fact that each trial begins at around 500 function calls: This is the first minimum available, i.e. the optimum found on the first instance of `fmincon` for each one of the ten trials.

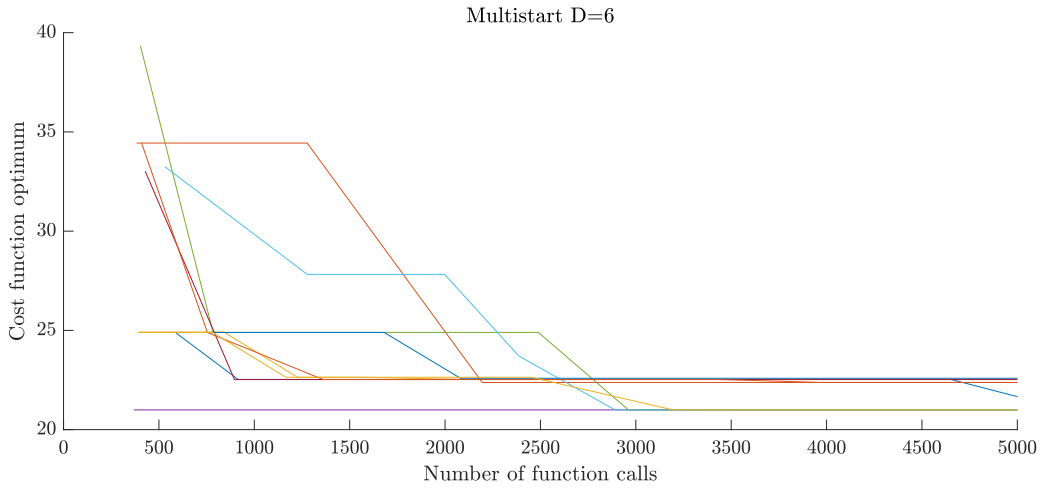


Figure 6.3: Optimum of ten trials of gradient-based multistart for the OPP computation with $D = 6$.

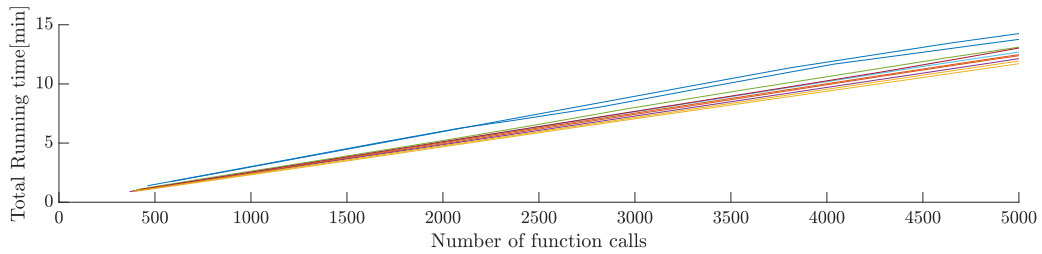


Figure 6.4: Total time of ten trials of gradient-based multistart for the OPP computation with $D = 6$.

test	$\boldsymbol{\sigma}_0$	$\boldsymbol{\sigma}_1$	$\boldsymbol{\sigma}_2$	$\boldsymbol{\sigma}_3$	$\boldsymbol{\sigma}_4$	$\boldsymbol{\sigma}_5$	**cost**	**const**
1	2.791	0.243	0.293	1.127	1.252	1.485	21.64	-0.012
2	-0.349	0.092	1.194	1.335	1.465	1.545	23.03	-0.003
3	-0.349	0.228	0.422	0.609	0.853	0.978	23.71	-0.002
4	-0.350	0.091	1.191	1.315	1.412	1.507	22.53	-0.000
5	-0.352	0.047	0.106	0.144	1.238	1.447	24.90	-0.002
6	-0.350	0.091	1.174	1.271	1.359	1.482	22.94	-0.016
7	-0.350	0.091	1.189	1.313	1.411	1.507	22.53	-0.007
8	-0.349	0.090	1.191	1.325	1.422	1.510	22.59	-0.084
9	2.790	0.070	0.124	1.208	1.363	1.512	22.38	-0.004
10	-0.349	0.190	0.900	0.989	1.362	1.492	21.00	-0.007

Table 6.2: Optimal points of gradient-based multistart for OPP with $D = 6$.

It is important to note that a negative slope gives only an intuition on the overall improvement given by that iteration of the local optimization, but it is not representative of the intermediate function evaluations. In other words, a negative slope says that the previously found minimum is improved in the following initialization. It is also worth noticing that a flat line means that the optimum is not improved: this could mean that the following initializations ended up in a worse local minimum, or an unfeasible point. For this reason, is not possible to infer from the graph how many initial conditions are used in every single run.

As expected, the total running time (Figure 6.4) grows linearly with the number of function calls. The time represented in this plot is the sum of the time required to evaluate the cost function, together with the algorithm self-time. This last one is almost constant for every iteration. The number of iterations is proportional to the number of function calls. On a side note, `fmincon` evaluates the cost function and constraint function separately. Both require, as the most relevant computational effort, the electrical drive model simulation. This implies making the same simulation twice. We notice that this limit is intrinsic of the `MATLAB` implementation, and could be avoided in a different application. Later, in Section 6.4, the function evaluation time is halved, so that the comparison of the time is fair.

Table 6.2 summarizes the outcome of the OPP computation for $D = 6$ with a budget of 5000 function evaluations. For every trial, the best optimal point is reported. Notice that, differently from SMGO, the `MATLAB` function `fmincon` considers the point feasible if the entry of the column **const** is negative. We see from Table 6.1 that the optima found are generally more accurate than the one found by SMGO. Nevertheless, not all trials converge to the same minimum.

6.3. Bayesian optimization

Bayesian optimization is a well-established global black-box optimization method, like SMGO. Nevertheless, they approximate the value of the cost function with two different surrogate models. SMGO uses the SM-model, based on the Lipschitz constant, while Bayesian fits a Gaussian process on the evaluated samples.
An overview of the solver and the principles behind is presented in [8]. Notice that `MATLAB bayesopt` can take into account linear constraints, so that a reformulation of such constraints is not required.

6.3.1. Results

Extremely consistent results are obtained on the three optimization variables test, with a very fast convergence rate, shown in Figure 6.5. These graphs represent the evolution of ten trials of Bayesian optimization. Differently from the multistart case, on every iteration the plot is representative of the best point up to that iteration.

Table 6.3 shows the results of the 3D Bayesian optimization. The convergence to the global optimum is reached in all trials. The same conclusions on the value of the cost function and on the feasibility of the result can be drawn as in the previous paragraph. However, the computational time for this method is polynomial with respect to the number of function calls. Indeed, more function calls mean more sampled points, which makes the fitting of the Gaussian process iteratively harder. As a consequence, going up with the iterations, the constant time for the function evaluation sum up to a dramatically high algorithm self-time. This keeps happening, even if the number of points involved in the fitting is limited to a certain budget, which is an option offered by `bayesopt`. Apparently, no Kriging schemes, that could make the fitting problem lighter, are implemented in the official `MATLAB` function.

From a theoretical point of view, documentation on `bayesopt` states that is possible to reach up to twenty optimization variables. Nevertheless, the application to the 6D OPP computation highlights the limits of the method. Indeed, in Figure 6.7 and Table 6.4, we can see that 2000 function calls are still not enough to have all the trials converge to the same result. However, already at this stage, the time required is prohibitive (about 50 hours per trial, as visible in Figure 6.8).

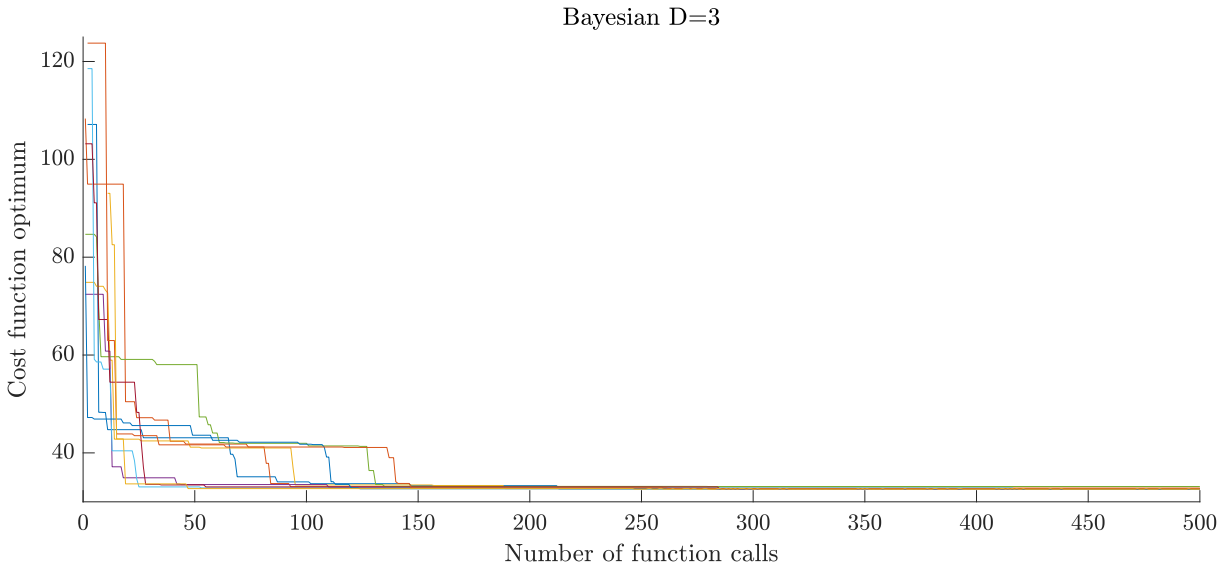


Figure 6.5: Optimum of ten trials of Bayesian for the OPP computation with $D = 3$.

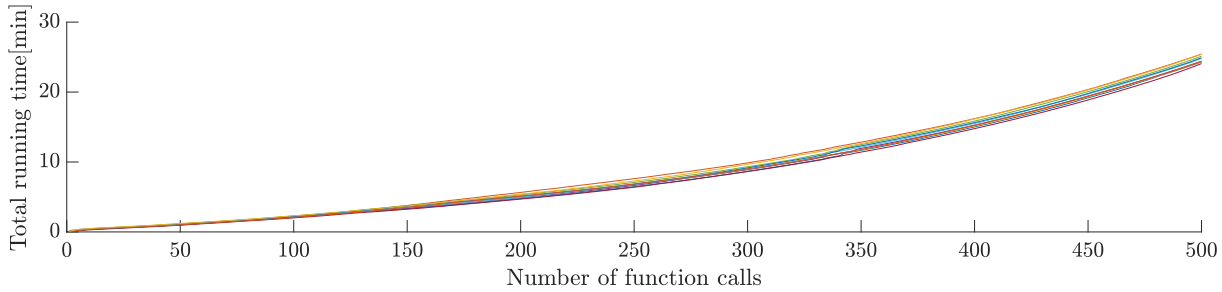


Figure 6.6: Total time of ten trials of Bayesian for the OPP computation with $D = 3$.

test	σ_0	σ_1	σ_2	cost	const
1	2.794	1.200	1.428	32.66	-0.454
2	2.792	1.198	1.436	32.81	-0.731
3	2.792	1.198	1.431	32.66	-0.454
4	2.794	1.205	1.433	32.67	-0.124
5	2.793	1.196	1.427	32.66	-0.454
6	2.790	1.195	1.428	32.66	-0.454
7	2.791	1.196	1.428	32.66	-0.454
8	2.793	1.195	1.427	32.66	-0.454
9	2.792	1.193	1.425	32.70	-0.447
10	2.793	1.190	1.422	32.70	-0.447

Table 6.3: Optimal points of Bayesian optimization for OPP with $D = 3$.

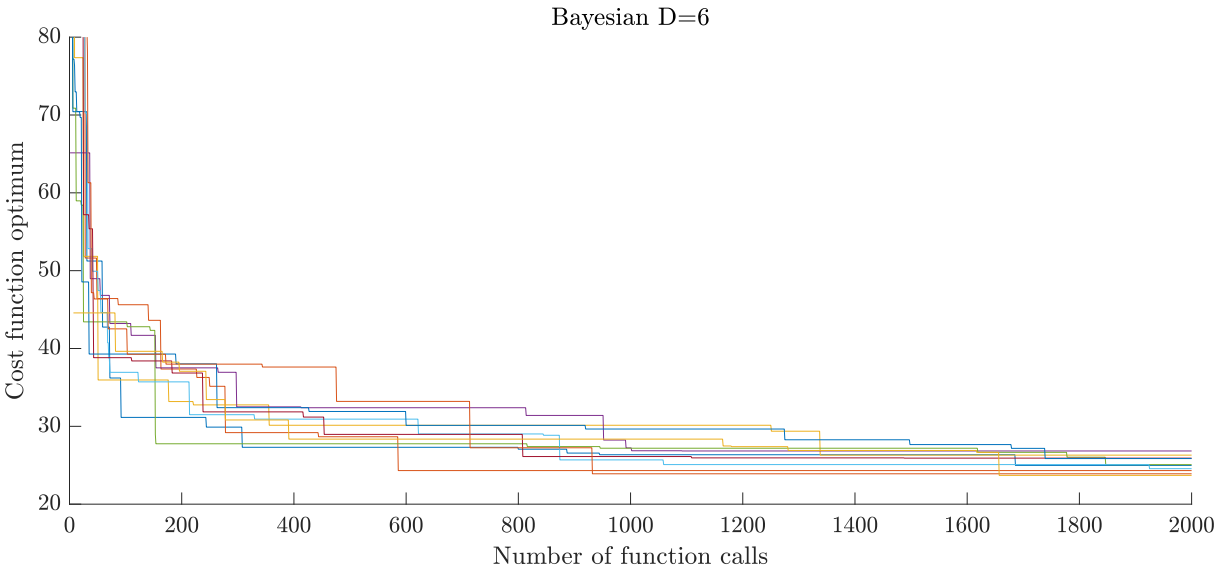


Figure 6.7: Optimum of ten trials of Bayesian for the OPP computation with $D = 6$.

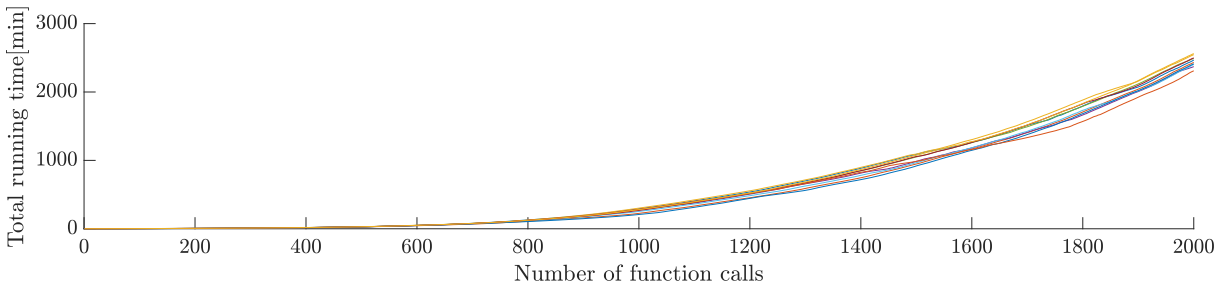


Figure 6.8: Total time of ten trials of Bayesian for the OPP computation with $D = 6$.

test	$\boldsymbol{\sigma}_0$	$\boldsymbol{\sigma}_1$	$\boldsymbol{\sigma}_2$	$\boldsymbol{\sigma}_3$	$\boldsymbol{\sigma}_4$	$\boldsymbol{\sigma}_5$	**cost**	**const**
1	2.730	0.074	0.114	1.203	1.347	1.509	24.98	-7.127
2	2.774	0.068	0.120	1.183	1.311	1.499	24.32	-0.503
3	2.792	0.297	0.316	1.140	1.298	1.482	26.30	-0.927
4	2.789	0.164	0.200	1.185	1.295	1.468	26.84	-0.547
5	2.758	0.085	0.227	0.840	0.928	1.458	25.08	-3.538
6	2.749	0.093	0.135	1.222	1.371	1.512	24.56	-5.201
7	-0.403	0.102	1.210	1.285	1.340	1.467	25.92	-6.452
8	2.758	0.221	0.262	1.126	1.278	1.503	25.86	-4.785
9	2.750	0.244	0.288	1.130	1.251	1.472	23.90	-6.214
10	2.749	0.236	0.295	1.118	1.229	1.490	23.70	-4.033

Table 6.4: Optimal points of Bayesian optimization for OPP with $D = 6$.

On top of that, the mechanism to enforce linear constraints meets some issues when dealing with the OPP problem for dimensions higher than six: At the beginning, the algorithm tests a distribution of points on the whole search space. Among these, the ones that satisfy the linear constraints are chosen for the following iterations. However, if the region that is feasible for the linear constraints is small, it is tough to get enough feasible points to initialize the algorithm. This leads `bayesopt` to abort the optimization.
Unfortunately, this is the case in the OPP search space. As seen in Chapter 4, the feasible region is a prism, contained in the hyperbox defined by the bounds of the variables. The fraction of space occupied by this prism with respect to the full hyperbox search space gets exponentially smaller with the increase of D [13].

The combination of `bayesopt` with the technique introduced in Section 4.1, as much as the modification of some parts of the algorithm to circumvent this implementation limit, have been explored but later abandoned, since incidental for the topic of the thesis.

For this reason, and also for the impracticable demand of time, the Bayesian optimization shows inadequacy for the OPP optimization problem, especially when dealing with a large number of switching angles. It still offers a good comparison up to $D = 6$.

6.4. Comparison of the methods

In order to assess the competitiveness of SMGO in the OPP framework, this Section compares it to the previously described methods, i.e., Bayesian optimization and gradient-based multistart. As a characteristic of the OPP problem, the study is conducted on different dimensions D.

In this comparative study, every algorithm is tested for ten trials, with a given number of function evaluations. The first chart that we report summarizes in a synthetic way the results obtained with the different methods. The square represents the mean value of the optima found over all the trials, with the bracket delimited by the largest and the smallest optimum. The blue numbers tell the average amount of local gradient-based optimizations carried out in the multistart optimization.
The second chart helps visualize the performance of each method in terms of time. Each bar represents the time of one of the three methods compared. For bars out of scale, the number is reported on top.

Remark. *The study of the performance of these three methods for a given budget of cost evaluations, allows us to draw conclusions on the computational burden of the optimization independently from the time requested by a single cost evaluation.*

6.4.1. Comparison in 3 dimensions

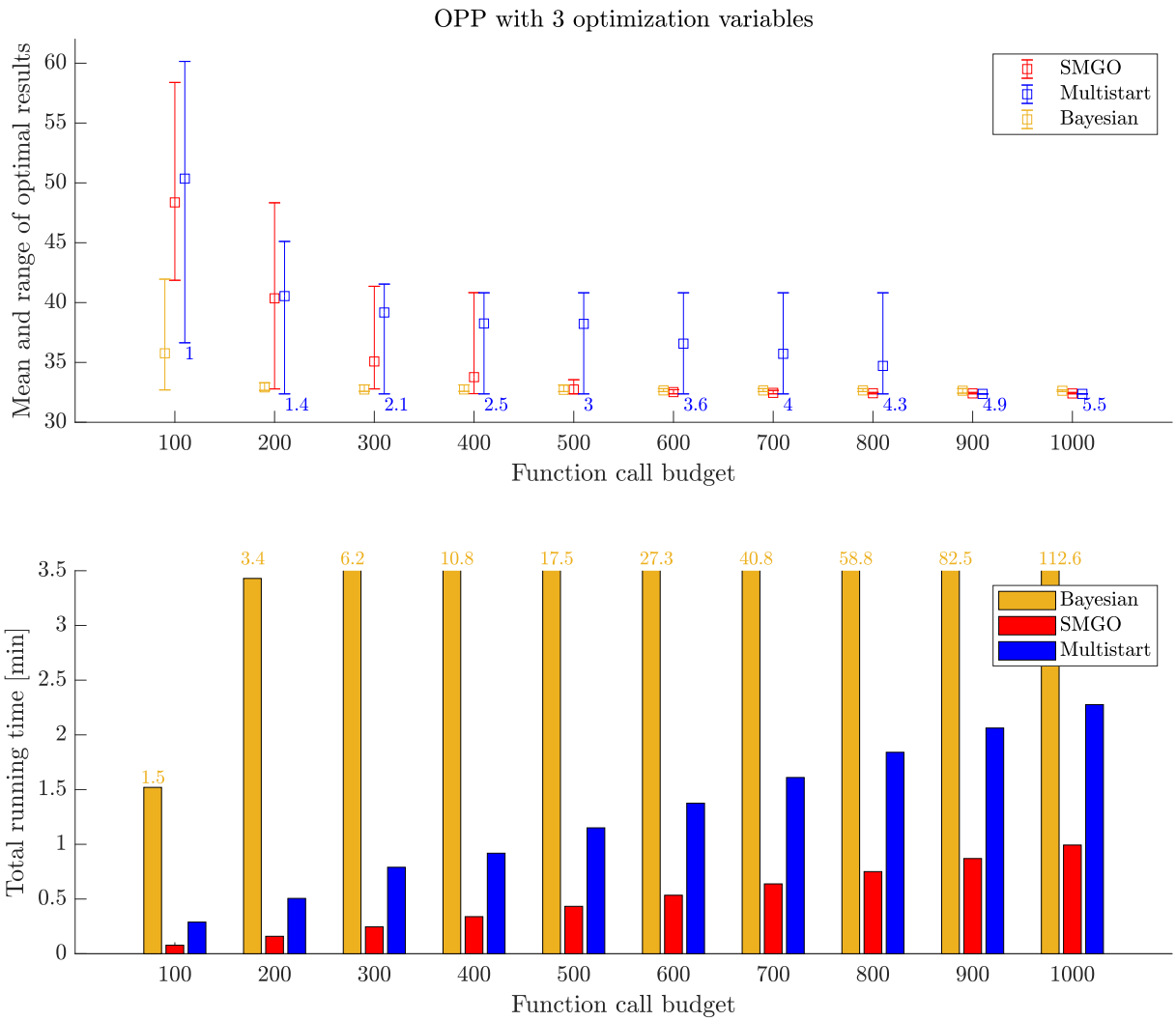


Figure 6.9: Comparison charts of different optimizer for $D = 3$. On the first chart, the square is the mean, the bracket is the range and the blue number is average the number of multistart initializations.

Figure 6.9 contains the results for the $3D$ case. All the solvers reach the same global minimum, with a reasonable budget of function evaluations. SMGO converges with a smaller number of function evaluations compared to multistart. Bayesian employs even fewer evaluations, but it requires way more computational time, one order of magnitude higher than SMGO. Overall, SMGO for this low dimensional case can be considered the best one, in the trade-off between computational burden and convergence rate.

The blue numbers of the first chart bring to the following remark.

Remark. *In multistart optimization, the whole budget is just sufficient to initialize the gradient-based optimization in a few different points.*

One can argue that, for this reason, the convergence to the global optimum is compromised. However, in this work, it is preferred to leave enough budget for each local gradient-based optimization. We point out also that using central difference method, a substantial number of function evaluations are employed to estimate the gradient at each iteration.

6.4.2. Comparison in 6 dimensions

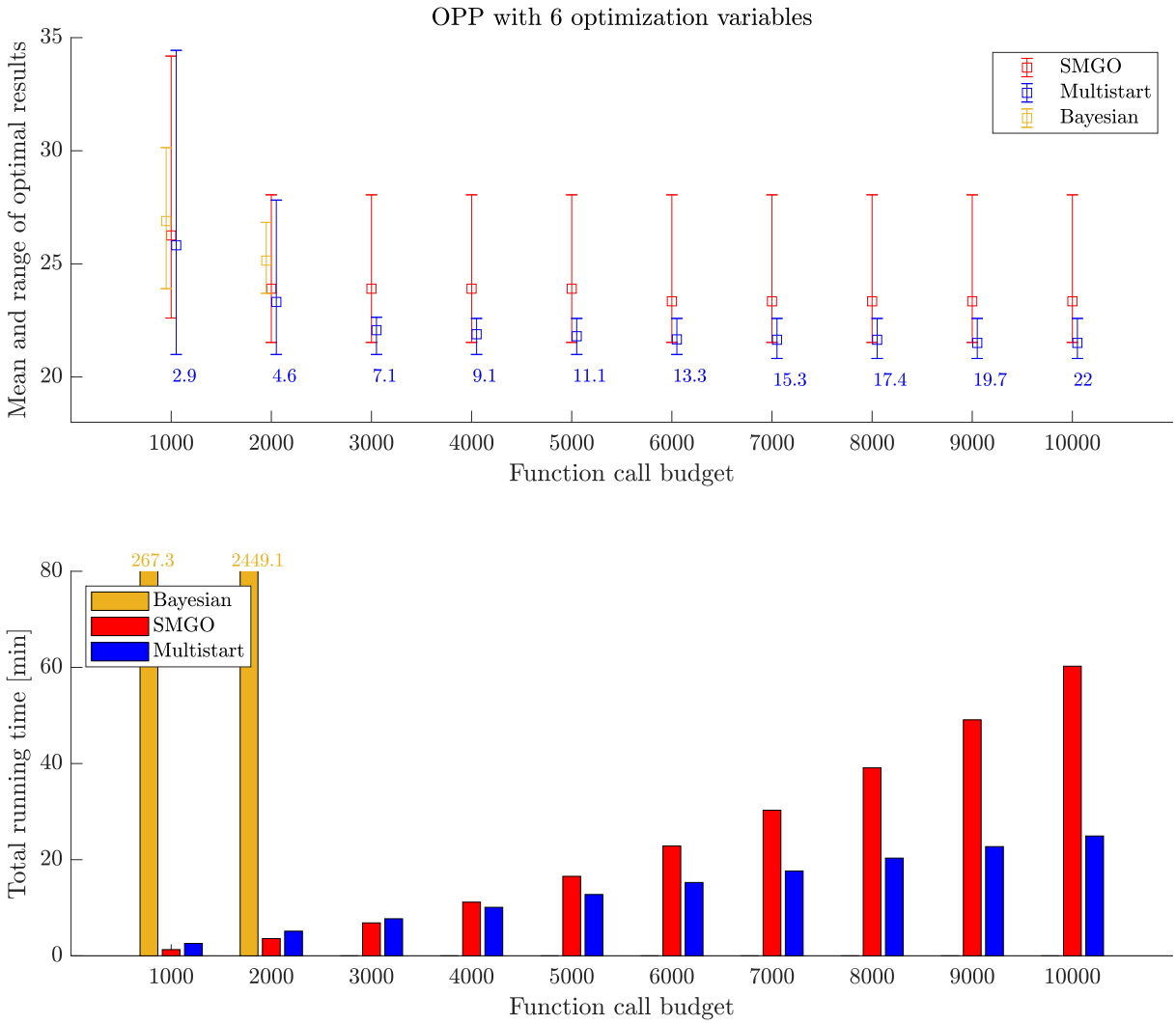


Figure 6.10: Comparison charts of different optimizer for $D = 6$. On the first chart, the square is the mean, the bracket is the range and the blue number is the average number of multistart initializations.

It Is known that the OPP problem requires more than three optimization variables to have interesting results, especially for higher motor speed. Up to five switching angles, SMGO maintains a good quality-efficiency trade-off and result. However, starting from $D = 6$, SMGO performance deteriorates. In the following are presented the comparative charts with $\boldsymbol{\sigma} \in \mathbb{R}^6$.

In this framework, all the solvers struggle to converge consistently to the same final optimum, as was shown in the previous Sections.
The range of optima found by SMGO is comparable to the range of multistart for small budget (2000 function calls), but with larger budgets the performance of SMGO does not improve. The right half of the first chart shows how at least one of the trials cannot improve its best result. For large budgets, multistart provides a lower average of found minima.

In the second chart, the break-even point for the computational time can be seen. After that, SMGO starts to be computationally more demanding than multistart.
Nevertheless, in the class of global methods, SMGO is still able to deal with high dimensionality, while Bayesian has to stop on a smaller budget of function evaluations, for the reasons explained in 6.3.
Therefore, despite performing not as well as multistart in the THD OPP problem, SMGO can still be considered as a viable global optimizer in high dimensional problems, for the simple fact that is still able to provide results.

6.4.3. Comparison in 20 dimensions

In this study, the number of optimization variables has been brought up to twenty optimization variables. The two usual charts are reported in Figure 6.11. The conclusions that can be drawn are the same in the 6D case. Firstly, notice that Bayesian optimization is not even applied for this dimension. For budgets of a few hundred cost function evaluations, SMGO provides slightly better performance in terms of mean and variability of the resulting optima, and also in terms of computational burden. Nonetheless, with larger budgets, the roles are reversed: multistart can converge very reliably to a better minimum, in short time.

In the end, the gap between SMGO and multistart results can be appreciated in terms of electrical drive performance. Figures 6.12 and 6.13 compare the current distortion and the torque ripple obtained with SMGO, on the left, and multistart, on the right. The OPP producing these results are chosen as the result of the trials with a cost value close to the mean one.

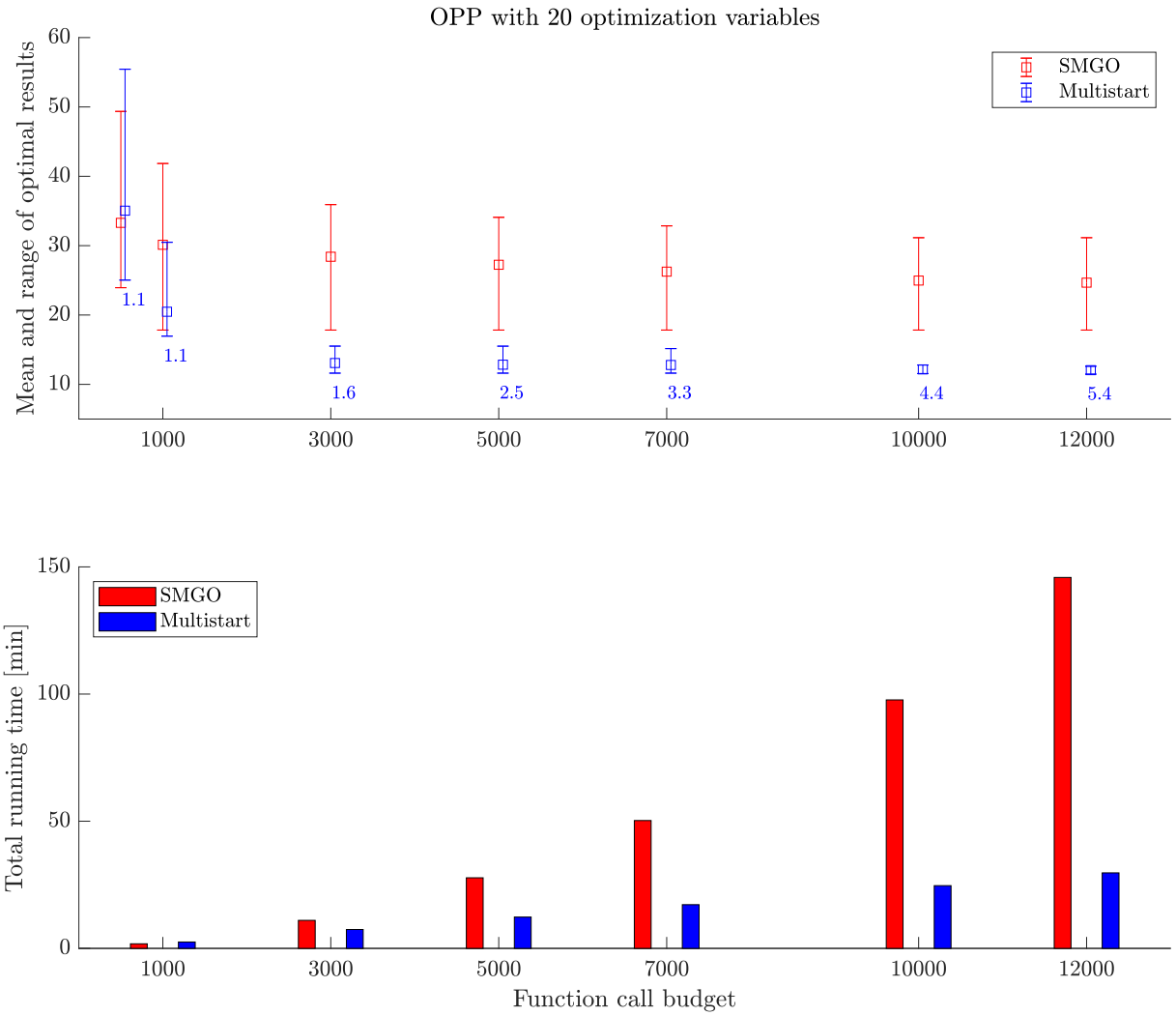


Figure 6.11: Comparison charts of different optimizer for $D = 20$. On the first chart, the square is the mean, the bracket is the range and the blue number is the average number of multistart initializations.

In order to obtain the same average torque, multistart solution requires a lower current amplitude with respect to the SMGO solution, and it provides also a restrained THD. Moreover, the OPP found with multistart has a smaller torque ripple.

Looking at Figure 6.11, we notice that multistart can converge with just between 5 and 6 random initializations of local gradient-based optimization.
Therefore, the problem at hand can be considered smooth enough. Normally, for this kind of problem, gradient-based methods are the best choice.

All the observations elaborated so far concern the THD-based problem introduced in Section 2.4. Nevertheless, the OPP can be computed starting from different problems.
A cost function with a more accurate loss model or different constraints can lead to the

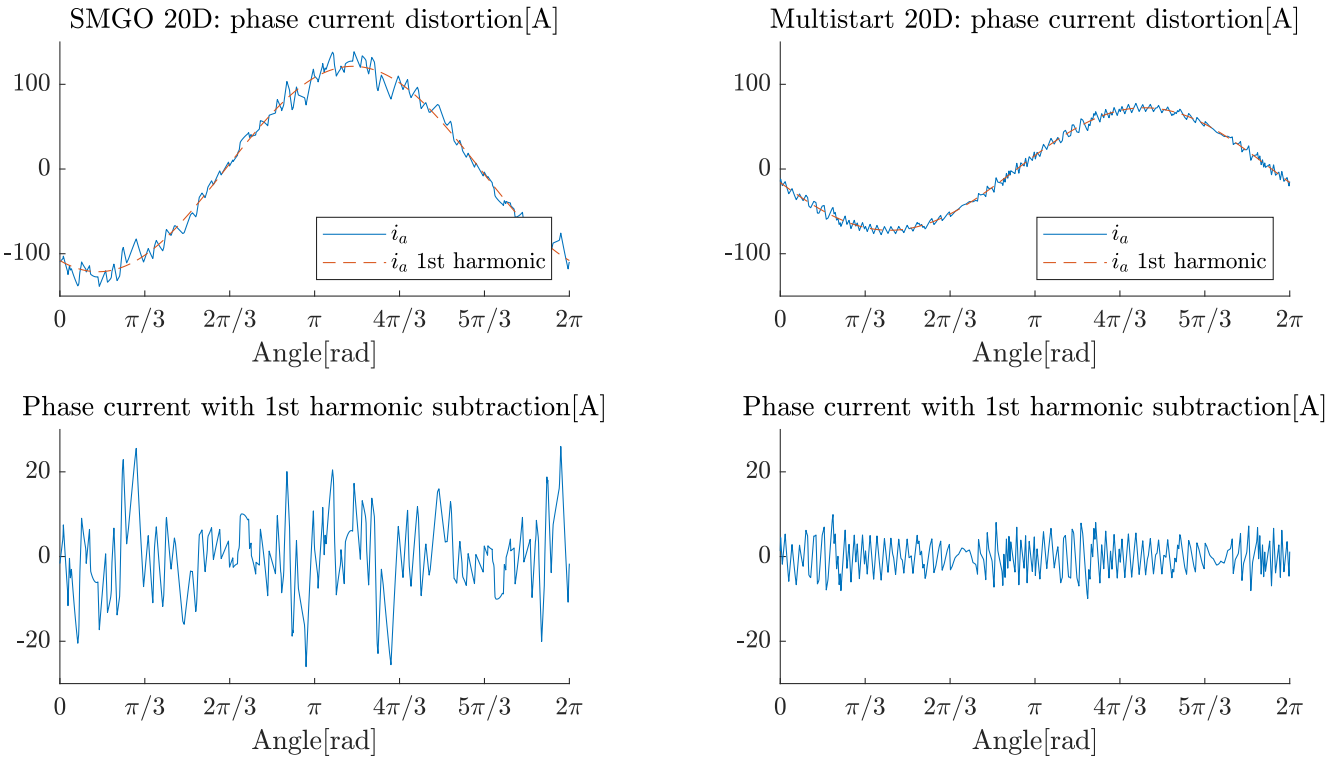


Figure 6.12: Distortion of SMGO OPP (on the left) vs distortion of the multistart OPP (on the right) in the 20D case.

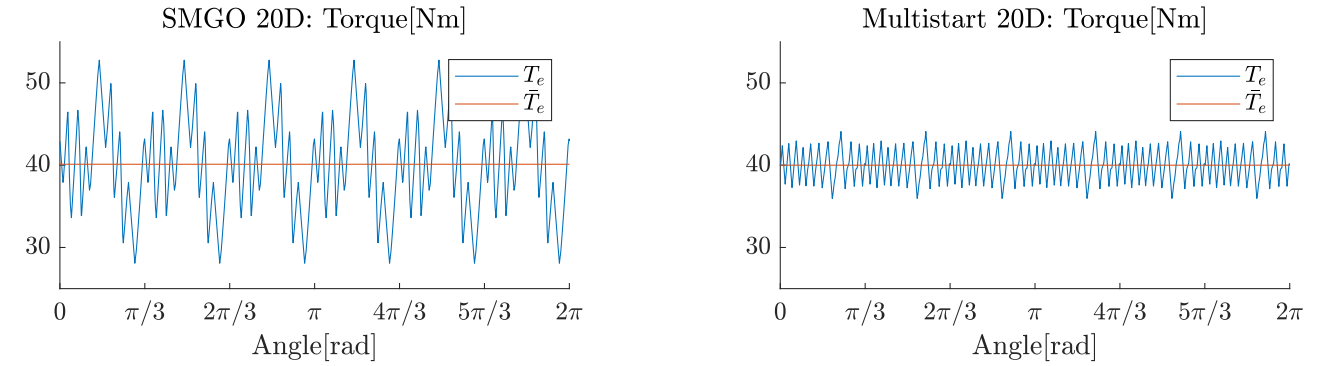


Figure 6.13: Torque ripple of SMGO OPP (on the left) vs ripple of the multistart OPP (on the right) in the 20D case.

failure of gradient-based methods, due to higher non-convexity and more discontinuity. Hence, SMGO could still be a superior method with respect to gradient-based for different formulations of the OPP problem.

6.4.4. On the time for the cost evaluation

The main drawback of the application of gradient-based multistart optimization in the case of high dimensional black-box cost functions is a large amount of function evaluation that this method requires. However, for cheap cost function, this disadvantage can be neglected.
The cost function of the THD OPP problem is very cheap to evaluate since it requires about 40 ms. This feature, together with the quasi-smoothness of the problem, makes multistart the ideal solver for this problem.
In light of these facts, pure SMGO is not suited to solve THD-based OPPs.

7 | Conclusions and future developments

The Optimal Pulse Patterns are specific drive modulations able to dramatically reduce the harmonic distortion introduced by the power converters in the electrical machines for steady-state operating points. They can also be used as starting points to build up dynamic controls (e.g. MP3C) that manage to deal with transients between different steady-state motor operations.

The OPP optimization problem at hand is based on a THD-based cost function and enforces a single inequality constraint on torque. We combine it with a new solver, the Set Membership Global Optimization. SMGO is a black box global optimization method, which is tested for the first time on the OPP problem. In this thesis, new ideas for the SMGO algorithm are proposed and analyzed. Thanks to the *extended trust region*, the *adaptive alpha* and the *sunburst candidate points generation* mechanisms, the solver becomes faster and less memory demanding. These components proved to be a general enhancement and will be soon implemented in the `MATLAB` toolbox.

For low dimensions of this specific problem, SMGO appears to be a good option. It is faster than Bayesian and multistart, and the budget of function evaluations it requires is in-between the two methods.
However, in practical terms, Optimal Pulse Patterns require a high number of switching angles, that correspond to a high number of optimization variables. In this case, SMGO does not perform better than traditional gradient-based multistart, because the formulated problem is quasi-smooth.

Nevertheless, SMGO remains appealing for two branches of optimization:

- **offline optimization** in case of few optimization variables, expensive and long cost function evaluation or highly non-smooth problems.
- **online optimization** where the real bottleneck is a low budget of available cost and constraint function evaluations.

The work described in this thesis offers a stimulating insight into SMGO, and highlights some interesting topics that could be objective of further research.
The main outlooks are the following:

- **Hybridization with gradient-based method**. It can be approached in different ways. The most intuitive could be to use SMGO for exploration, and then use the points it finds to initialize a local gradient-based optimization. In this way, instead of using a random distribution, the points will be chosen based on the surrogate model. This can be seen as a smart initialization of multistart, that takes into account all the information previously retrieved. It combines the convergence property of SMGO with a very efficient exploitation stage, where we make use of knowledge of the gradient.
- **More complex OPP problem formulation**. The THD minimization confirmed to be an improvement for the functioning of the motor, also on the test bench. However, it is not the only efficiency criterion that can be accounted for when treating optimal pulse patterns. There exist more accurate loss models and functional constraints that can be integrated into the model of the electrical drive. As already mentioned, this change in the structure of the optimization brings non-smoothness into the problem. The non-smoothness can disrupt the efficacy of gradient-based methods. In this case, SMGO could still be a viable solution.
- **Algorithm parallelization**. SMGO is a sequential algorithm that makes use of a large database. In the algorithm, some mechanisms depend on the points already sampled, while some others are independent. Therefore, the algorithm could be enhanced to be executed on parallel cores. Parallelization could increase its speed and make it more competitive.

Bibliography

[1] F. Berkel, M. Löhning, and S. Reimann, *Loss optimal pulse patterns for electrical drives*, tech. rep., Robert Bosch GmbH, 2022.

[2] P. Bolzern, R. Scattolini, and N. Schiavoni, *Fondamenti di controlli automatici*, Collana di istruzione scientifica, McGraw-Hill Companies, 2008.

[3] E. O. Brigham and R. E. Morrow, *The fast fourier transform*, IEEE Spectrum, 4 (1967), pp. 63–70.

[4] F. C. Dezza, *AC brushless*. http://castellidezza.faculty.polimi.it, 2017. Politecnico di Milano.

[5] K. Douzane, L. Rinehart, C. Keraudren, S. Rodhain, and F. Tahiri, *Inverter and motor efficiency increase with fpcu implementing optimized pulse pattern methods*, in 34th International Electric Vehicle Symposium and Exhibition, Nanjing, Jiangsu, 2021.

[6] L. Fagiano, *Constrained Numerical Optimization for Estimation and Control*. Lecture Notes, Politecnico di Milano, September 2021.

[7] R. Fotouhi, L. Leitner, R. Kennel, and H. du Toit Mouton, *An efficient method to calculate optimal pulse patterns for medium voltage converters*, in IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society, 2014, pp. 1221–1226.

[8] P. I. Frazier, *A tutorial on Bayesian optimization*, 2018.

[9] B. A. Frigyik, A. Kapila, and M. R. Gupta, *Introduction to the dirichlet distribution and related processes*, tech. rep., University of Washington, 2010.

[10] T. Geyer, *Model Predictive Control of High Power Converters and Industrial Drives*, John Wiley and Sons, Inc., 2016.

[11] T. Geyer, N. Oikonomou, G. Papafotiou, and F. D. Kieferndorf, *Model predictive pulse pattern control*, IEEE Transactions on Industry Applications, 48 (2012), pp. 663–676.

[12] T. Geyer and V. Spudić, *Carrier-based model predictive pulse pattern control*, in 2018 IEEE Energy Conversion Congress and Exposition (ECCE), 2018, pp. 4024–4031.

[13] M. Henk, J. Richter-Gebert, and G. Ziegler, *Handbook of discrete and computational geometry*, in Basic properties of convex polytopes, J. E. Goodman, J. O'Rourke, and C. D. Tóth, eds., CRC Press LLC., third ed., 2017, ch. 15.

[14] D. Holmes and T. Lipo, *Pulse Width Modulation for Power Converters: Principles and Practice*, IEEE Press Series on Power and Energy Systems, John Wiley and Sons, Inc., 2003.

[15] E. Kontodinas, A. Kraemer, H.-D. Endres, S. Wendel, P. Karamanakos, and J. Bonifacio, *An experimental assessment of modulation methods for drive trains used in electric vehicles*, in IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society, 2022.

[16] J. Lv, S. Deng, and Z. Wan, *An efficient single-parameter scaling memoryless broyden-fletcher-goldfarb-shanno algorithm for solving large scale unconstrained optimization problems*, IEEE Access, 8 (2020), pp. 85664–85674.

[17] F. L. Mapelli, *Il motore sincrono a magneti permanenti (motore brushless)*, 2015.

[18] M. Milanese and C. Novara, *Set membership identification of nonlinear systems*, Automatica, 40 (2004), pp. 957–975.

[19] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power Electronics. Converters, Applications and Design*, John Wiley and Sons, Inc., third ed., 2003.

[20] C. J. O'Rourke, M. M. Qasim, M. R. Overlin, and J. L. Kirtley, *A geometric interpretation of reference frames and transformations: dq0, clarke, and park*, IEEE Transactions on Energy Conversion, 34 (2019), pp. 2070–2083.

[21] L. J. Sabug, *On data-driven optimization in the design and control of autonomous systems*, PhD thesis, Politecnico di Milano, 2023.

[22] L. J. Sabug, F. Ruiz, and L. Fagiano, *On the use of set membership theory for global optimization of black-box functions*, in 2020 59th IEEE Conference on Decision and Control (CDC), 2020, pp. 3586–3591.

[23] L. J. Sabug, F. Ruiz, and L. Fagiano, *SMGO: A set membership approach to data-driven global optimization*, Automatica, 133 (2021), p. 109890.

[24] L. J. Sabug, F. Ruiz, and L. Fagiano, *SMGO-Δ: Balancing caution and reward in global optimization with black-box constraints*, Information Sciences, 605 (2022), pp. 15–42.

[25] S. Skogestad and I. Postlethwaite, *Multivariable frequency response analysis*, John Wiley and Sons, Inc., Hoboken, NJ, USA, 2005, pp. 68–71.

[26] I. Sobol', *On the distribution of points in a cube and the approximate evaluation of integrals*, USSR Computational Mathematics and Mathematical Physics, 7 (1967), pp. 86–112.

[27] M. Črepinšek, S.-H. Liu, and M. Mernik, *Exploration and exploitation in evolutionary algorithms: A survey*, ACM Comput. Surv., 45 (2013).

[28] C. Zhang, Q. Guo, L. Li, M. Wang, and T. Wang, *System efficiency improvement for electric vehicles adopting a permanent magnet synchronous motor direct drive system*, Energies, 10 (2017).

A | Enhanced SMGO on test functions

In this Appendix, the algorithmic components introduced in Chapter 5 are tested on a set of benchmark optimization problems. The different settings that are tested are:

- **old** - Original SMGO, with default tuning.
- **ETR** - SMGO with Extended Trust Region.
- **SCG** - SMGO with Sunburst Candidate points Generation.
- **Aα** - SMGO with Adaptive Alpha.
- **new** - SMGO with all the new components.

By differentiating the effects given by each component, it is easy to understand the main contribution of every edit.
Each row shows the performance metrics of a method, and each column represents a different metric. The used metrics are:

1. **Best opt**: the best objective value found by the method over ten trials.
2. **Worst opt**: the worst objective value found by the method over ten trials.
3. **Mean opt**: the mean of the best objective values found by the method over ten trials.
4. **Explts**: the mean number of exploited points over ten trials.
5. **Feas**: the mean number of feasible points found by the method over ten trials.
6. **Time[s]**: the self-time in seconds needed for the method to complete one trial. The time required for the evaluation of the objective is not accounted.

all the tests have been initialized on the same points. The other features of the algorithm are kept identical, and set to the default value. The values of the parameters are collected in the table A.1.

When an entry of the three first columns is reported to be equal to *Inf*, it means that

α	$\bar{B}$	R_{ref}	N_{cdpt}	β	$\alpha_{\min}$
0.005	4	5	50	0.1	0.0005

Table A.1: Parameters adopted on the benchmark functions.

no feasible value was found. Out of the set of tests in [24], functions G06 and G08 are particularly challenging, since the feasible region is very small.
The performance is evaluated according to the **Mean opt**. The other columns are available for further insight. The different settings of SMGO have been tested for two different risk factors. We can collect the observations for the two values of Δ.

- **For $\Delta = 1$**: In the majority of tests, the new SMGO performs similarly to the original, but in much less time. It provides worse results in just three tests. However, out of these three, the test on STYB_D10 can be neglected. Indeed, this benchmark function is unconstrained: Since Δ is set equal to 1, the exploration merit function is equal to 0 on all points. Therefore, the exploration is obviously less performing than the exploitation, because it does not follow any useful criterion.
- **For $\Delta = 0.2$**: The new version of SMGO outperforms the original algorithm in seven tests, and it performs worse in just one.

In the following tables, when the performance of the new SMGO and the performance of the old one are consistently different, the best method out of the two is highlighted.

Table A.2: Results for T1 ($D = 2$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	6.02e-01	6.14e-01	6.07e-01	21.6/500	273.1/500	68.9
ETR	6.02e-01	6.11e-01	6.05e-01	20.9/500	271.2/500	63.3
SCG	6.03e-01	6.23e-01	6.10e-01	21.6/500	277.0/500	8.0
Aα	6.00e-01	6.07e-01	6.01e-01	23.7/500	272.9/500	68.9
new	6.00e-01	6.03e-01	6.01e-01	23.7/500	273.4/500	5.0

Table A.3: Results for T2 ($D = 2$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	2.54e-01	2.68e-01	2.61e-01	22.1/500	36.6/500	54.6
ETR	2.54e-01	2.74e-01	2.64e-01	22.9/500	34.6/500	48.3
SCG	2.56e-01	2.80e-01	2.65e-01	20.9/500	35.3/500	6.6
Aα	2.53e-01	2.55e-01	2.54e-01	45.1/500	45.8/500	54.9
new	2.53e-01	2.57e-01	2.55e-01	30.5/500	40.2/500	4.1

Table A.4: Results for T3 ($D = 2$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-2.00e+00	-2.00e+00	-2.00e+00	56.3/500	357.8/500	52.5
ETR	-2.00e+00	-2.00e+00	-2.00e+00	57.6/500	358.6/500	47.0
SCG	-2.00e+00	-2.00e+00	-2.00e+00	55.5/500	358.6/500	6.7
Aα	-2.00e+00	-2.00e+00	-2.00e+00	83.3/500	366.9/500	51.0
new	-2.00e+00	-1.98e+00	-2.00e+00	47.2/500	354.8/500	4.3

Table A.5: Results for STYB ($D = 2$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-7.83e+01	-7.83e+01	-7.83e+01	124.0/500	269.0/500	68.4
ETR	-7.83e+01	-7.81e+01	-7.83e+01	205.6/500	285.2/500	59.0
SCG	-7.83e+01	-7.83e+01	-7.83e+01	115.5/500	273.7/500	7.8
Aα	-7.83e+01	-7.83e+01	-7.83e+01	82.8/500	254.3/500	66.3
new	-7.83e+01	-7.80e+01	-7.83e+01	64.4/500	245.4/500	4.9

Table A.6: Results for STYB ($D = 10$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-3.21e+02	-3.20e+02	-3.21e+02	263.2/500	500.0/500	58.1
ETR	-3.61e+02	-3.00e+02	-3.24e+02	409.4/500	500.0/500	42.1
SCG	-3.21e+02	-3.21e+02	-3.21e+02	252.5/500	500.0/500	16.1
Aα	-3.19e+02	-3.15e+02	-3.17e+02	83.3/500	500.0/500	61.1
new	-3.11e+02	-3.05e+02	-3.05e+02	67.0/500	500.0/500	5.3

Table A.7: Results for G04 ($D = 5$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-3.07e+04	-3.02e+04	-3.04e+04	221.8/500	248.5/500	128.8
ETR	-3.08e+04	-3.06e+04	-3.07e+04	329.3/500	197.7/500	122.7
SCG	-3.06e+04	-3.02e+04	-3.04e+04	228.3/500	255.6/500	19.2
Aα	-3.05e+04	-3.00e+04	-3.03e+04	81.6/500	276.9/500	138.3
new	-3.05e+04	-3.03e+04	-3.05e+04	75.4/500	295.5/500	13.5

Table A.8: Results for G05 ($D = 4$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	5.21e+03	5.53e+03	5.36e+03	75.1/500	20.0/500	126.9
ETR	5.13e+03	5.43e+03	5.26e+03	33.4/500	13.7/500	119.0
SCG	5.13e+03	5.38e+03	5.25e+03	68.0/500	18.3/500	17.2
Aα	5.27e+03	5.45e+03	5.35e+03	83.5/500	23.5/500	124.5
new	5.14e+03	5.47e+03	5.27e+03	44.9/500	17.8/500	12.6

Table A.9: Results for G06 ($D = 2$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-2.66e+03	Inf	Inf	2.6/500	0.1/500	65.3
ETR	-5.26e+03	Inf	Inf	2.2/500	0.4/500	60.9
SCG	Inf	Inf	Inf	0.0/500	0.0/500	7.4
Aα	-6.27e+03	Inf	Inf	8.2/500	1.7/500	69.0
new	Inf	Inf	Inf	0.0/500	0.0/500	5.9

Table A.10: Results for G08 ($D = 2$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-9.55e-02	-9.12e-02	-9.40e-02	108.9/500	88.4/500	64.5
ETR	-9.57e-02	-8.71e-02	-9.24e-02	203.6/500	138.6/500	58.0
SCG	-9.55e-02	-9.31e-02	-9.46e-02	104.4/500	89.5/500	7.6
Aα	-9.55e-02	-8.51e-02	-9.13e-02	82.7/500	64.0/500	64.2
new	-9.25e-02	-7.84e-02	-8.65e-02	82.8/500	51.7/500	4.9

Table A.11: Results for G09 ($D = 7$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	7.08e+02	2.69e+03	1.18e+03	218.8/500	189.4/500	114.5
ETR	9.27e+02	1.33e+03	1.10e+03	390.7/500	133.4/500	98.2
SCG	7.72e+02	2.28e+03	1.18e+03	199.5/500	170.3/500	19.3
Aα	7.50e+02	1.46e+03	1.03e+03	75.9/500	102.3/500	122.7
new	1.08e+03	8.12e+03	3.30e+03	90.5/500	79.6/500	12.8

Table A.12: Results for G10 ($D = 8$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	1.22e+04	Inf	Inf	15.3/500	5.2/500	162.1
ETR	9.18e+03	Inf	Inf	18.0/500	1.0/500	146.9
SCG	1.24e+04	Inf	Inf	24.0/500	6.3/500	25.9
Aα	1.08e+04	Inf	Inf	20.2/500	8.5/500	163.2
new	1.01e+04	Inf	Inf	18.5/500	7.0/500	16.0

Table A.13: Results for G12 ($D = 3$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-9.86e-01	-9.23e-01	-9.70e-01	109.3/500	74.8/500	55.3
ETR	-9.94e-01	-9.24e-01	-9.71e-01	124.1/500	57.1/500	49.5
SCG	-9.86e-01	-9.23e-01	-9.61e-01	94.1/500	57.2/500	7.9
Aα	-9.94e-01	-9.48e-01	-9.70e-01	82.3/500	64.9/500	57.1
new	-1.00e+00	-8.81e-01	-9.52e-01	67.1/500	34.2/500	4.8

Table A.14: Results for G23 ($D = 9$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-4.13e+03	-3.89e+03	-4.09e+03	101.0/500	306.3/500	89.6
ETR	-5.61e+03	-3.89e+03	-5.13e+03	424.3/500	426.9/500	70.3
SCG	-4.11e+03	-3.37e+03	-3.88e+03	90.7/500	199.5/500	19.0
Aα	-4.13e+03	-3.61e+03	-3.99e+03	73.3/500	278.9/500	92.3
new	-5.20e+03	-2.13e+03	-4.56e+03	48.2/500	226.5/500	9.3

Table A.15: Results for G24 ($D = 2$) with $\Delta = 1.00$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-5.49e+00	-5.35e+00	-5.44e+00	44.1/500	284.1/500	69.9
ETR	-5.50e+00	-5.43e+00	-5.48e+00	30.9/500	280.6/500	63.7
SCG	-5.48e+00	-5.31e+00	-5.41e+00	45.0/500	292.6/500	8.7
Aα	-5.51e+00	-5.47e+00	-5.50e+00	36.0/500	280.4/500	68.2
new	-5.51e+00	-5.45e+00	-5.49e+00	30.3/500	289.4/500	4.9

Table A.16: Results for T1 ($D = 2$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	6.08e-01	6.41e-01	6.26e-01	13.9/500	290.6/500	68.1
ETR	6.12e-01	6.43e-01	6.31e-01	14.3/500	290.7/500	60.7
SCG	6.29e-01	6.49e-01	6.37e-01	13.6/500	291.2/500	8.3
Aα	6.03e-01	6.30e-01	6.08e-01	25.2/500	296.9/500	69.0
new	6.00e-01	6.15e-01	6.04e-01	21.2/500	281.0/500	5.0

Table A.17: Results for T2 ($D = 2$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	2.89e-01	3.56e-01	3.21e-01	6.8/500	54.9/500	53.6
ETR	2.70e-01	3.74e-01	3.31e-01	6.1/500	53.6/500	47.5
SCG	2.78e-01	3.41e-01	3.12e-01	9.0/500	38.8/500	6.7
Aα	2.59e-01	3.44e-01	2.78e-01	23.8/500	66.5/500	54.4
new	2.53e-01	2.55e-01	2.55e-01	30.0/500	77.5/500	4.0

Table A.18: Results for T3 ($D = 2$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-2.00e+00	-2.00e+00	-2.00e+00	56.1/500	459.8/500	52.4
ETR	-2.00e+00	-1.97e+00	-1.99e+00	42.7/500	458.2/500	47.4
SCG	-2.00e+00	-2.00e+00	-2.00e+00	53.6/500	469.5/500	6.6
Aα	-2.00e+00	-2.00e+00	-2.00e+00	83.2/500	462.2/500	50.4
new	-2.00e+00	-1.98e+00	-1.99e+00	42.0/500	426.0/500	4.2

Table A.19: Results for STYB ($D = 2$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-7.83e+01	-7.83e+01	-7.83e+01	118.6/500	410.4/500	66.9
ETR	-7.83e+01	-7.82e+01	-7.83e+01	184.2/500	418.1/500	58.5
SCG	-7.83e+01	-7.83e+01	-7.83e+01	108.6/500	425.1/500	7.9
Aα	-7.83e+01	-7.83e+01	-7.83e+01	83.1/500	398.6/500	65.6
new	-7.83e+01	-7.82e+01	-7.83e+01	71.0/500	348.2/500	4.7

Table A.20: Results for STYBD ($D = 10$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-3.35e+02	-3.20e+02	-3.22e+02	262.8/500	500.0/500	58.2
ETR	-3.41e+02	-2.36e+02	-3.01e+02	361.7/500	500.0/500	42.8
SCG	-3.44e+02	-2.77e+02	-3.16e+02	271.0/500	500.0/500	16.3
Aα	-3.19e+02	-2.77e+02	-3.08e+02	84.6/500	500.0/500	60.8
new	-3.05e+02	-2.38e+02	-2.63e+02	93.1/500	500.0/500	5.6

Table A.21: Results for G04 ($D = 5$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-3.03e+04	-2.99e+04	-3.02e+04	96.4/500	445.7/500	143.8
ETR	-3.06e+04	-2.99e+04	-3.03e+04	117.4/500	450.8/500	128.7
SCG	-3.05e+04	-2.97e+04	-3.01e+04	106.9/500	435.3/500	21.0
Aα	-3.05e+04	-3.01e+04	-3.03e+04	84.0/500	446.9/500	142.8
new	-3.07e+04	-3.03e+04	-3.05e+04	82.8/500	291.8/500	14.4

Table A.22: Results for G05 ($D = 4$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	5.59e+03	6.64e+03	5.90e+03	16.3/500	20.2/500	136.2
ETR	5.53e+03	6.23e+03	5.85e+03	17.2/500	21.2/500	123.1
SCG	5.53e+03	7.54e+03	6.10e+03	11.8/500	14.1/500	18.6
Aα	5.29e+03	6.19e+03	5.71e+03	54.8/500	57.4/500	131.4
new	5.14e+03	5.32e+03	5.20e+03	40.2/500	23.4/500	13.3

Table A.23: Results for G06 ($D = 2$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	Inf	Inf	Inf	0.0/500	0.0/500	66.0
ETR	Inf	Inf	Inf	0.0/500	0.0/500	60.4
SCG	Inf	Inf	Inf	0.0/500	0.0/500	8.0
Aα	Inf	Inf	Inf	0.0/500	0.0/500	75.9
new	Inf	Inf	Inf	0.0/500	0.0/500	5.5

Table A.24: Results for G08 ($D = 2$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-9.20e-02	-3.24e-03	-3.94e-02	4.1/500	8.7/500	67.0
ETR	-9.33e-02	-1.44e-02	-4.61e-02	2.1/500	6.8/500	62.0
SCG	-9.41e-02	-2.18e-03	-3.63e-02	2.1/500	5.7/500	8.4
Aα	-9.58e-02	-2.64e-02	-6.44e-02	83.3/500	85.1/500	66.5
new	-9.44e-02	-7.13e-02	-8.69e-02	82.3/500	60.3/500	5.1

Table A.25: Results for G09 ($D = 7$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	2.51e+03	Inf	Inf	0.0/500	1.6/500	129.4
ETR	2.51e+03	Inf	Inf	0.0/500	1.6/500	113.9
SCG	7.40e+03	Inf	Inf	0.0/500	1.2/500	21.4
Aα	7.66e+03	Inf	Inf	50.2/500	51.6/500	127.5
new	9.41e+02	3.83e+04	6.35e+03	86.1/500	125.3/500	12.0

Table A.26: Results for G10 ($D = 8$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	Inf	Inf	Inf	0.0/500	0.0/500	159.4
ETR	Inf	Inf	Inf	0.0/500	0.0/500	143.8
SCG	Inf	Inf	Inf	0.0/500	0.0/500	27.8
Aα	Inf	Inf	Inf	0.0/500	0.0/500	174.2
new	1.05e+04	Inf	Inf	25.2/500	9.5/500	18.3

Table A.27: Results for G12 ($D = 3$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-9.94e-01	-9.42e-01	-9.73e-01	13.1/500	42.2/500	57.8
ETR	-9.93e-01	-9.14e-01	-9.71e-01	9.5/500	37.8/500	50.7
SCG	-9.99e-01	-8.89e-01	-9.59e-01	9.4/500	29.5/500	8.3
Aα	-9.94e-01	-9.51e-01	-9.74e-01	81.1/500	107.1/500	56.9
new	-9.93e-01	-9.24e-01	-9.60e-01	60.5/500	37.4/500	4.8

Table A.28: Results for G23 ($D = 9$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-4.13e+03	-4.07e+03	-4.13e+03	80.5/500	456.3/500	90.9
ETR	-5.61e+03	-4.52e+03	-5.31e+03	422.0/500	465.1/500	75.6
SCG	-4.13e+03	-3.37e+03	-3.88e+03	74.7/500	327.5/500	20.2
Aα	-4.13e+03	-4.08e+03	-4.13e+03	68.0/500	454.7/500	90.6
new	-5.08e+03	-4.60e+03	-4.88e+03	52.8/500	262.2/500	9.3

Table A.29: Results for G24 ($D = 2$) with $\Delta = 0.20$.

Mode	Best opt	Worst opt	Mean opt	Explts	Feas	Time[s]
old	-5.39e+00	-4.96e+00	-5.24e+00	21.6/500	255.1/500	69.1
ETR	-5.46e+00	-4.98e+00	-5.17e+00	18.3/500	255.6/500	60.7
SCG	-5.40e+00	-4.97e+00	-5.17e+00	17.8/500	251.9/500	8.4
Aα	-5.46e+00	-5.17e+00	-5.28e+00	52.3/500	273.4/500	67.7
new	-5.50e+00	-5.45e+00	-5.48e+00	28.7/500	295.9/500	5.4

List of Figures

List of Tables

List of Symbols

Variable	Description	Unit
t	continuous time variable	[s]
$\frac{d}{dt}, dot$	derivative with respect to time	
$v_a(t),\ v_b(t),\ v_c(t)$	voltage applied to machine coil a, b, c	
	or phase voltage a, b, c	[V]
$i_a(t),\ i_b(t),\ i_c(t)$	current flowing into machine coil a, b, c	
	or phase current a, b, c	[V]
$\Psi_a(t),\ \Psi_b(t),\ \Psi_c(t)$	magnetic flux concatenated with machine coil a, b, c	[Wb]
$R_a,\ R_b,\ R_c,\ R_s$	stator resistance	[Ω]
$\theta_{\mathrm{m}}(t)$	rotor position	[rad]
L_{ss}	coil self-inductance	[H]
M_{ss}	coil mutual inductance	[H]
$\hat{\psi}_{\mathrm{pm}}$	permanent magnet flux	[Wb]
$v_d(t),\ v_q(t)$	voltage applied to the fictitious coil d, q	[V]
$i_d(t),\ i_q(t)$	current flowing into the fictitious coil d, q	[A]
$\Psi_d(t),\ \Psi_q(t)$	magnetic flux concatenated with the fictitious coil d, q	[Wb]
$L_d,\ L_q$	self-inductance of the fictitious coil d, q	[H]
ω_s	electrical speed	[rad/s]
N_p	number of pole pairs	
$T_e(t)$	electrical torque generated by the machine	[Nm]
$\bar{T}_e$	average value of electrical torque over an electrical period	[Nm]
T_{req}	requested electrical torque	[Nm]
$\boldsymbol{\xi}(t)$	dynamical model state vector	
$\boldsymbol{u}(t)$	dynamical model input vector	
$\boldsymbol{y}(t)$	dynamical model output vector	
s	Laplace operator	

Variable	Description	Unit
j	imaginary unit	
ω_0	fundamental frequency	[rad/s]
T	electric period	[s]
N	maximum number of fundamental frequency multiples	
$U_1^{1\dots N}, U_2^{1\dots N}, U_3^{1\dots N}$	complex coefficients of the input model frequency spectra	
$Y_1^{1\dots N}, Y_2^{1\dots N}$	complex coefficients of the output model frequency spectra	
$H_1^{1\dots N}, H_2^{1\dots N}, H_3^{1\dots N}$	complex coefficients of the phase current frequency spectra	
$h_1^{1\dots N}$, $h_2^{1\dots N}$, $h_3^{1\dots N}$	module of complex coefficients $H_1^{1\dots N}, H_2^{1\dots N}, H_3^{1\dots N}$	
$\boldsymbol{h}$	vector of the phase current harmonics amplitudes	
Ω_m	mechanical speed	[rpm]
T_s	sampling time	[s]
t_min	semiconductor minimum switching time	[s]
φ_el	generic rotor angle	[rad]
U_DC	constant DC voltage supplied to the power converter	[V]
σ_0	precommutation angle	[rad]
$\bar{\sigma}_i,\ i = 1 \dots l$	switching angle	[rad]
$\boldsymbol{\sigma}_\mathrm{sa} = [\sigma_1 \dots \sigma_l]$	switching angles relative to the precommutation angle	[rad]
l	number of switching angles in a quarter phase period	
U_init	voltage level after the first switching angle	[V]
$u_s(\varphi_\mathrm{el})$	generic pulse pattern	
$\boldsymbol{\sigma} \in \mathbb{R}^{D=l+1}$	vector containing the optimization variable	[rad]
D	number of optimization variables	
$F(\boldsymbol{\sigma})$	optimization cost function	
Q	weighing parameter for the multiobjective cost function	
σ_min	minimal pulse angle width	[rad]
$\boldsymbol{x}$	optimization variables	
$\mathcal{X}$	search space	
D	dimension of the search space	
$\mathcal{F}(\gamma)$	Lipschitz functions with constant γ	
f	cost function	
γ	Lipschitz constant of the cost	
g_s	s-th constraint function	

Variable	Description	Unit
ρ_s	Lipschitz constant of the s-th constraint	
$\boldsymbol{x}^{(n)}$	sample n	
$z^{(n)}$	value of the cost on sample n	
$\boldsymbol{c}^{(n)}$	array of constraints on sample n	
$\boldsymbol{x}^*$	optimal point	
z^*	optimum	
$\mathcal{G}$ $(\mathcal{G}_s)$	feasible set (for constraint s)	
$\mathcal{L}$	Lebesgue measure (D-volume)	
$\boldsymbol{X}^{\langle n \rangle}$	set of sampled point at iteration n	
$\tilde{\gamma}^{\langle n \rangle}$	estimate of γ at iteration n	
$\tilde{\rho}_s^{\langle n \rangle}$	estimate of ρ_s at iteration n	
$\underline{f}^{\langle n \rangle}$	lower bound on f at iteration n	
$\bar{f}^{\langle n \rangle}$	upper bound on f at iteration n	
$\tilde{f}^{\langle n \rangle}$	central approximation of f at iteration n	
$\lambda_f^{\langle n \rangle}$	uncertainty of f at iteration n	
$\boldsymbol{E}^{\langle n \rangle}$	set of candidate points at iteration n	
$N_{\max}$	maximum number of SMGO iterations	
$\boldsymbol{v}_i \in \boldsymbol{V}$	i-th vertex in the vertex set $\boldsymbol{V}$	
V	number of vertices	
$\hat{\boldsymbol{a}}_d$	coordinate direction	
$b_{\pm d}$	maximum length along $\pm\hat{\boldsymbol{a}}_d$	
$\bar{B}$	number of points in gridding	
$\boldsymbol{Y}$, $\boldsymbol{Y}_\circ^{(n)}$, $\boldsymbol{Y}_{+d}^{(n)}$, $\boldsymbol{Y}_{-d}^{(n)}$	new set of candidate points at iteration n	
$\delta^{\langle n \rangle}$	optimality gap	
$M \in \mathbb{R}^{(l-1)\times l}$	matrix to enforce linear constraint on the switching angles	
$\boldsymbol{x}^{*\langle n \rangle}$	best sample at iteration n	
$z^{*\langle n \rangle}$	optimum at iteration n	
$\boldsymbol{c}^{*\langle n \rangle}$	constraints of best sample at iteration n	
$\boldsymbol{x}_\theta$	exploitation candidate	
Δ	risk factor	
β	tuning weight for exploitation	
α	tuning parameter for exploration-exploitation trade-off	

Variable	Description	Unit
N_{cld}	tuning parameter: number of points cast in initialization and number of points of the trust region	
N_{cdpt}	tuning parameter: number of closest candidate points	
η	expected improvement threshold	
$\boldsymbol{x}_{\phi}$	exploration candidate	
$\mathcal{T}^{\langle n \rangle}$	trust region at iteration n	
$v^{\langle n \rangle}$	size of the trust region at iteration n	
$\underline{v}$	minimum size of the trust region	
$\bar{v}$	maximum size of the trust region	
κ	shrinking factor	
$\Phi^{\langle n \rangle}$	exploration merit function	
$\mathcal{P}_{\text{sa}}$	polytope describing switching angles feasible set according to linear constraints	
$\mathcal{P}$	polytope describing optimization variables feasible set according to linear constraints	
$\boldsymbol{\sigma}^{(n)} = \left[\sigma_0^{(n)}, ..., \sigma_l^{(n)}\right]$	point sampled by the algorithm	
$\boldsymbol{q} = [q_1, \ldots, q_V]$	set of V weights to describe a point in the search space as a combination of the vertex $\boldsymbol{v_i}$	
$\text{Dir}(\boldsymbol{q}, a)$	Dirichlet multivariate probability distribution with weighting parameter q and shaping parameter a	
$\Gamma(a)$	gamma multivariate probability distribution with shaping parameter a	
$\underline{d}$	minimum distance between each sampled point	
σ_s	sampling angular distance	[rad]
R	exploration-to-exploitation ratio	
R_{ref}	exploration-to-exploitation ratio reference	
$e(n)$	error between the R at iteration n and R_{ref}	
$\Sigma_e(n)$	integral error	
$\alpha_{\min}$	lower saturation of the controller control action α	
μ	step size for gradient computation	
$\mathcal{W}^{(n)}$	support set of endpoints at iteration n for Sunburst point generation	
$\boldsymbol{p}$	generic endpoint inside the support set $\mathcal{W}^{(n)}$	

List of Acronyms

Acronym	Description
OPP	Optimal Pulse Pattern
SM	Set Membership
SMGO	Set Membership Global Optimization
WLTC	Worldwide harmonized Light vehicles Test Cycles
DTC	Direct Torque Control
FOC	Field Oriented Control
PWM	Pulse Width Modulation
MP3C	Model Predictive Pulse Pattern Control
PMSM	Permanent Magnet Synchronous Machine
AC	Alternating Current
DC	Direct Current
CBPWM	Carrier-Based Pulse Width Modulation
SVPWM	Space-Vector Pulse Width Modulation
THD	Total Harmonic Distortion
PI	Proportional-Integral controller
SQP	Sequential Quadratic Programming
BFGS	Broyden – Fletcher – Goldfarb – Shanno quasi-newton method