Further Exploration of a Laguerre Polynomial Based Sphere Decoding Algorithm for MPC of Inverters

Adrián Parra Lafuente and Rudy Cepeda-Gomez *

August 26, 2026

Abstract

This paper explores the combination of Model Predictive Control (MPC) and the Sphere Decoding Algorithm (SDA) to control a three-phase, variable-speed induction motor drive through a three-level inverter. It builds upon previous work in which a framework to simplify the integer optimization problem using Laguerre Polynomials (LPs) was introduced. The objective is to obtain a much faster convergence of the algorithm through a reduction in the dimensionality of the optimization problem. Specifically, this work explores different parametric selections to better understand their effects on the performance of the algorithm.

1 Introduction

Model Predictive Control (MPC), a concept formulated almost half a century ago [1], has gained a lot of traction in the past couple decades, specially in the control of power electronics [2]. The increase in the number of theoretical and applied publications, especially after 2011 [3], is a testament of the rising popularity of this method. Particularly for the direct control of electrical drives, without a pulse width modulator, the Finite-Control-Set MPC offers good performance, combining the small steady-state distortions of classic optimized pulse patterns, with the large bandwidth of techniques based on pulse width modulation. The main drawback of this method is the need to solve an optimization problem at each iteration of the algorithm. When controlling electrical machines, this difficulty is increased by the integer nature of the constraints imposed by the inverter [4]. Reducing the elevated computational burden becomes a crucial aspect if this technique is to be used in this type of systems [5].

The Sphere Decoding Algorithm (SDA) is a mathematical tool well suited to reduce the computational complexity of optimization problems with integer constraints. It operates by examining points within a hypersphere, thereby shortening the execution time by limiting the search space to a confined region. One of the earliest applications of SDA in power electronics was presented in [6]. In this context, the algorithm proved efficacy as a suitable tool for simplifying the cost optimization problem, which involves handling the integer constraints imposed by the switching states of inverters.

A further attempt to improve the convergence rate of the resulting optimization problem is the usage of Laguerre Polynomials (LPs) to transform the search space, as suggested in [7]. This class of polynomials was proposed to modify the structure of the SDA search tree such that convergence is achieved faster. This initial work, however, did not find any significant advantage brought in by the usage of LPs.

The fundamental distinction between the preceding work and the present one lies in the reduction of the dimensionality in the Laguerre space. This is accomplished through an improved distinct methodology to generate the subspace of solutions, enabling the utilization of a reduced number of LPs, less than the number of steps in the prediction horizon. The implementation of this modification provides additional degrees of freedom for the tuning of the algorithm, making a careful selection necessary to exploit its advantages. The aim of the present paper is to dive deeper into this method and to explore different scenarios, attempting to better understand which parametric setups lead to a reduced computational burden.

The next section of the paper presents the model of the plant being considered, and the MPC, SDA, and LPs framework. Section 3 provides a detailed examination of the parametric selection of LPs, and section 4 contains the results obtained after the parametric study. Conclusions are presented in Section 5.

*The authors are with the Institute of Electrified Aeroengines of the German Aerospace Center, Cottbus, Germany. Emails adrian.parralafuente@dlr.de, rudy.cepedagomez@dlr.de

2 Description of the problem

2.1 The Plant

The plant considered is an induction machine driven by a three-level inverter. The motor model in the $\alpha - \beta$ or stationary reference frame in continuous time is taken from [8] as

$$\dot{x}(t) = Fx(t) + Gu(t) \tag{1a}$$
$$y(t) = Cx(t), \tag{1b}$$

with matrices

$$F = \begin{bmatrix} -\frac{1}{\tau_s} & 0 & \frac{X_m}{\tau_r D} & \frac{\omega_r X_m}{D} \\ 0 & -\frac{1}{\tau_s} & -\frac{\omega_r X_m}{D} & \frac{X_m}{\tau_r D} \\ \frac{X_m}{\tau_r} & 0 & -\frac{1}{\tau_r} & -\omega_r \\ 0 & \frac{X_m}{\tau_r} & \omega_r & -\frac{1}{\tau_r} \end{bmatrix} \tag{2a}$$

$$G = \frac{v_{dc} X_r}{2D} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} \end{bmatrix} \tag{2b}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{2c}$$

The state vector $x = [i_{s\alpha}\ i_{s\beta}\ \psi_{r\alpha}\ \psi_{r\beta}]^T$ contains the stator current and the rotor magnetic flux, the output vector $y = [i_{s\alpha}\ i_{s\beta}]^T$ is the stator current, and the input vector $u \in \mathbb{R}^{3\times 1}$ controls the action of the switches of the three legs of the inverter.

Inverters are inherently discrete systems, in which each leg has a limited set of possible states. Since a three-level inverter is considered, each leg can only be connected to the positive voltage, to ground, or to the negative voltage. Representing these states by 1, 0, and -1, respectively, each element of the input vector u is constrained to adopt one of these three integer values.

After discretizing (1) using the sampling time T_S, the prediction model becomes

$$x(k+1) = Ax(k) + Bu(k) \tag{3a}$$
$$y(k+1) = Cx(k+1), \tag{3b}$$

with

$$A = e^{(FT_S)} \qquad B = \int_0^{T_S} e^{F\tau} d\tau\ G. \tag{4}$$

2.2 Deployment of MPC

Considering a prediction horizon of N_p samples, a system with n states, m inputs, and p outputs, the output trajectory $\mathbf{Y}$ and the input sequence $\mathbf{U}$ are defined by

$$\mathbf{Y}(k+1) = \begin{bmatrix} y^T(k+1) & \cdots & y^T(k+N_p) \end{bmatrix}^T \in \mathbb{R}^{pN_p \times 1} \tag{5}$$
$$\mathbf{U}(k) = \begin{bmatrix} u^T(k) & \cdots & u^T(k+N_p-1) \end{bmatrix}^T \in \mathbb{R}^{mN_p \times 1} \tag{6}$$

Introducing these variables in the discrete prediction model (3) results in

$$\mathbf{Y}(k+1) = \Gamma x(k) + \Upsilon \mathbf{U}(k), \tag{7}$$

where $\Gamma \in \mathbb{R}^{pN_p \times n}$ and $\Upsilon \in \mathbb{R}^{pN_p \times mN_p}$, both defined in [6].

With this representation the cost function at each time step is defined as

$$J(k) = \sum_{i=k}^{k+N_p-1} ||y^*(i+1) - y(i+1)||_Q^2 + \lambda_u ||\Delta u(i)||_2^2, \tag{8}$$

where y^* is the output reference vector, Q is the accuracy weight matrix, λ_u is the switching penalization weight, and $\Delta u_i = u_i - u_{i-1}$. Notice that two norms are used here for any vector m: the p-norm $\|m\|_p = \sqrt[p]{x_1^p + x_2^p + \cdots}$ and the weighted norm $\|m\|_Q = \sqrt{m^T Q m}$, where p is a scalar number and Q a positive definite matrix. After some algebraic manipulation and completing squares, the cost is expressed in terms of the input sequence $\mathbf{U}$ as

$$J = \left(\mathbf{U}(k) + H^{-1}\Theta(k))\right)^T H \left(\mathbf{U}(k) + H^{-1}\Theta(k))\right). \tag{9}$$

The matrix Θ is to be computed at each time step, whereas H is a constant, positive definite matrix; both defined in [6].

The optimization problem consists in finding the switching sequence $\mathbf{U}(k) \in \mathcal{U} = \{-1, 0, 1\}^{3N_p}$, which minimizes the cost (9) subjected to the constraints imposed by inverter dynamics $\|\Delta u(i)\|_\infty \leq 1$.

An exhaustive search could be used to find the optimal solution to this problem, but its computational burden makes it unfeasible. For example, with $N_p = 3$, the algorithm must check $3^9 = 19683$ possible solutions in each sample interval. The SDA is an attempt to reduce this burden.

2.3 Sphere Decoding Algorithm

To apply the SDA, a solution is first obtained without considering the integer restrictions imposed by the inverter:

$$\frac{\partial J}{\partial \mathbf{U}} = 0 \rightarrow \mathbf{U}_{unc}(k) = -H^{-1}\Theta(k). \tag{10}$$

Equation (10) is referred to as the *unconstrained solution*.

The problem now consists on finding the constrained solution which better resembles the unconstrained one. That is, the distance between the two solutions is to be minimized. For this, the cost function is first expressed in terms of the unconstrained and the actual integer solution to be applied. Using the Cholesky factorization of the matrix $H = V^T V$, the cost function becomes

$$J = \left(V\mathbf{U}(k) - V\mathbf{U}_{unc}(k)\right)^T \left(V\mathbf{U}(k) - V\mathbf{U}_{unc}(k)\right), \tag{11}$$

and the optimization problem is stated as

$$\mathbf{U}_{opt}(k) = \underset{\mathbf{U}(k)}{argmin} \left\|V\mathbf{U}(k) - \bar{\mathbf{U}}_{unc}(k)\right\|_2^2 \tag{12a}$$

$$s.t.\ \ \mathbf{U}(k) \in \mathcal{U} = \{-1, 0, 1\}^{3N_p} \tag{12b}$$

$$\|\Delta u(i)\|_\infty \leq 1,\ \forall i = k, \cdots, k + N_p - 1, \tag{12c}$$

where $\bar{\mathbf{U}}_{unc}(k) = V\mathbf{U}_{unc}(k)$.

To find the optimal solution, SDA performs a search through the constrained solution space computing the size of the spheres with center in the unconstrained solution, $\bar{\mathbf{U}}_{unc}$, and radii defined by the sequence of integer constrained solutions $\bar{\mathbf{U}}_j$. It begins the iteration with an educated guess and then iterates until the smallest sphere is found. This corresponds to the optimal constrained solution $\bar{\mathbf{U}}_{opt}$.

The adaptation of this algorithm to the branch and bound technique, as described in Algorithm 1 in [8], further reduces the computational burden, creating a decision tree diagram. The number of levels of the search tree is equivalent to the number of legs of the inverter multiplied by the size of prediction horizon, $3N_p$, and each node branches into the three possible switching states for each leg $\{-1,\ 0,\ 1\}$. This leads to a deep and narrow tree as shown in Fig. 1, where the solution is chosen sequentially. Moreover, the impossible switching (in red), counts as an explored branch, but not as a node, as it is shown in algorithm 1 of chapter 5 of [6].

The main drawback is that the upper bound in the number of branches

$$UB_{NB}^{SDA} = \sum_{i=1}^{3N_p} 3^i \tag{13}$$

grows exponentially with the prediction horizon, again making the solution of this problem almost impossible for large values of N_p.

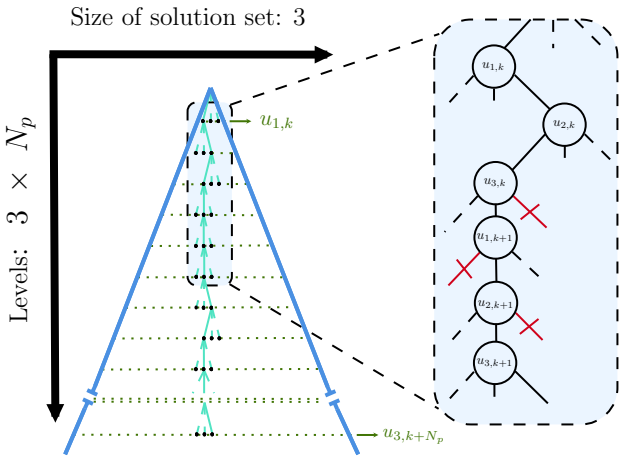

Figure 1: Decision tree of SDA.

2.4 Laguerre Polynomials (LPs)

The use of LPs as an approach to reduce the computational burden while satisfying the constraints of the model is the alternative method proposed in [7]. The goal is to transform the search space of the SDA, such that the structure of the search tree is reverted. This change implies a transition from a decision tree of $3 \times N_p$ levels to one of only 3 levels (one for each leg of the inverter), as seen in Fig. 2. As counterpart, the number of branches at each level increases, but the upper bound on the number of branches to explore is reduced. However, the solution found might be suboptimal if a dimensional reduction is considered.

For a given parameter $a \in [0,\ 1)$, called *pole* or *scaling factor*, the Z-transform of a sequence of LPs can be obtained recursively using

$$\Gamma_1(z) = \frac{\sqrt{1-a^2}}{1-az^{-1}}\ , \qquad \Gamma_i(z) = \Gamma_{i-1}(z)\frac{z^{-1}-a}{1-az^{-1}}. \tag{14}$$

The discrete time LPs are found using the inverse Z transform

$$l_i(k) = \mathcal{Z}^{-1}\left\{\Gamma_i(z)\right\}. \tag{15}$$

For a given inverter leg j at time step k, a sequence of N_L polynomials

$$L_j(k) = [l_1(k)\ l_2(k)\ \ldots\ l_{N_L}(k)], \tag{16}$$

is also found recursively using

$$L_j(k+1) = A_L L_j(k), \tag{17}$$

where A_L is computed as described in chapter 3 of [9].

This set of LPs is used to transform a switching sequence between the time domain and the so-called Laguerre domain. The transformation matrix from the solution in Laguerre space $\mathbf{E}_j$ into the time domain $\mathbf{U}_j$ for the j-th leg of the inverter is

$$\Lambda_j = \begin{bmatrix} L_j(0) & \cdots & L_j(N_p-1) \end{bmatrix}^T \in \mathbb{R}^{N_p \times N_L}, \tag{18}$$

such that

$$\mathbf{U}_j(k) = \Lambda_j \mathbf{E}_j(k). \tag{19}$$

The number of LPs N_L considered could be different for each leg of the inverter. This, however, would lead to an imbalance of the control actions on each leg, generating a non-uniform degradation of the switches and possibly leading to a malfunction of the inverter. Since this does not make sense from an implementation point of view, this work considers an uniform N_L for the three legs.

When all legs of the inverter are considered simultaneously, the transformation is

$$\mathbf{U}(k) = \Lambda \mathbf{E}(k), \tag{20}$$

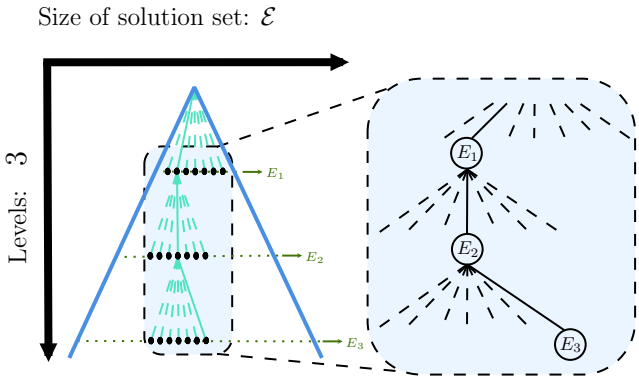

Figure 2: Decision tree of L-SDA.

with Λ obtained by reshaping the individual Λ_j into

$$\Lambda = \begin{bmatrix} L_1(0) & 0_{1\times N_L} & 0_{1\times N_L} \\ 0_{1\times N_L} & L_2(0) & 0_{1\times N_L} \\ 0_{1\times N_L} & 0_{1\times N_L} & L_3(0) \\ \vdots & \vdots & \vdots \\ L_1(N_p-1) & 0_{1\times N_L} & 0_{1\times N_L} \\ 0_{1\times N_L} & L_2(N_p-1) & 0_{1\times N_L} \\ 0_{1\times N_L} & 0_{1\times N_L} & L_3(N_p-1) \end{bmatrix}, \tag{21}$$

where $0_{1\times N_L}$ is a null matrix of size $1 \times N_L$.

Substituting (20) in (9):

$$J = \left(\Lambda\mathbf{E}(k) + H^{-1}\Theta(k)\right)^T H \left(\Lambda\mathbf{E}(k) + H^{-1}\Theta(k)\right). \tag{22}$$

The unconstrained solution is obtained similarly to (10):

$$\mathbf{E}_{unc}(k) = -\Omega^{-1}\Psi(k), \tag{23}$$

where $\Omega = \Lambda^T H \Lambda$ and $(\Psi(k))^T = (\Theta(k))^T \Lambda$. Then, applying Cholesky factorization to Ω

$$V_L^T V_L = \Omega. \tag{24}$$

The optimization problem for L-SDA is formulated as:

$$\mathbf{E}_{opt}(k) = \underset{\mathbf{E}(k)}{argmin} \left|\left|V_L\mathbf{E}(k) - \bar{\mathbf{E}}_{L,unc}(k)\right|\right|_2^2, \tag{25a}$$

$$s.t.\ \ \mathbf{E}_j(k) \in \mathcal{E} = (\Lambda_j^T\Lambda_j)^{-1}\Lambda_j^T\{-1,0,1\}^{N_L} \tag{25b}$$

where $\bar{\mathbf{E}}_{L,unc}(k) = V_L\mathbf{E}_{unc}(k)$. Finally, (20) is used to come back to the original space, $\mathbf{U}(k)$.

The LPs framework provides several parameters that impact the performance of the optimization algorithm, such as the prediction horizon N_p, the Laguerre pole a, and the number of LPs used, N_L. The following section describes in more detail whether and when modifications of these parameters are favorable.

3 Deployment of Laguerre polynomials (LPs)

The new set of integer solutions in Laguerre domain, $\mathcal{E}$, is the sequence of actions of each leg. The upper bound in the number of branches can be calculated as

$$UB_{NB}^{L-SDA} = \sum_{i=1}^{3} 3^{i\cdot N_L}. \tag{26}$$

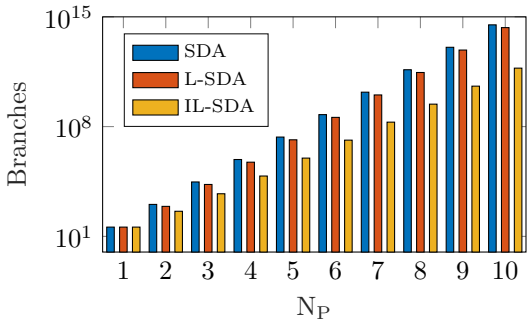

Figure 3: Upper bound on the number of branches for the three methods: SDA, L-SDA, and IL-SDA.

A further reduction in the number of branches to be explored can be achieved by considering the no-jumping restriction (12c). Sequences including this invalid switching can be computed and removed from the search tree offline, saving time due to the solution disposal of Laguerre's framework, where the input action of each leg are in order. This approach will now be referred to as the improved L-SDA (IL-SDA). The standard SDA method must compute this condition online, increasing the number of branches explored.

Figure 3 shows the upper bound on the number of branches for the three different methods considered: the original SDA method, the SDA with LPs (L-SDA) and the IL-SDA. While the improvement from SDA to L-SDA is not very large, a significant reduction is achieved with IL-SDA, even though dimension reduction has not yet taken place, $N_L = N_p$.

The control horizon must be no larger than the prediction horizon, i.e., $N_L \leq N_p$. In consequence, the solution obtained is

$$U = [U_{L-SDA}\ 0_{1\times 3(N_p-N_L)}]^T. \tag{27}$$

This leads to the removal of $N_p - N_L$ rows in each input, reducing the dimensionality of the problem, but leading to some information losses when $N_L < N_p$ resulting in a suboptimal solution. In equation (27), the missing values were filled with 0 to complete this vector. The option of staying in the last switching state has shown a slightly worse performance, therefore it was not considered for the study. Further discussion about this topic is beyond the scope of this work.

When the Laguerre pole is not zero, $a \neq 0$, the solution becomes suboptimal and the reduction in the dimensionality of the problem makes impossible the transformation of the solution from Laguerre domain back into time domain via (20). In this case, some considerations in the algorithm must be implemented, increasing the complexity and the computational burden. Moreover, it introduces a bias in the comparison because the computational cost per branch is higher than for $a = 0$, therefore this work considers only cases where the Laguerre pole is $a = 0$.

4 Parametric study

This study focuses on determining possible performance improvements of the algorithm through the addition of LPs, but does not attempt to demonstrate for which value of the prediction horizon the algorithm is computable. Therefore, the two performance indicators considered for the optimization algorithms are the value of the cost function and the computational burden. To have a fair basis for comparison between the different optimization methods, the cost function is normalized by the size of the prediction horizon $J^u = \frac{J}{N_p}$. The computational burden is evaluated by counting the branches explored in each case. For a fair comparison the SDA uses the same prediction and control horizons, $N_p = N_C$, and the IL-SDA use the number of LPs as control horizon, $N_L = N_C$.

Table 1 shows a quick evaluation for two specific values of N_p and one of N_L. It considers the average unitary cost obtained by each algorithm for an specific value of λ_u.

A more detailed performance comparison between SDA and IL-SDA considered more than 100 simulations for each method, using different prediction horizons and number of LPs. The sampling time was also changed, allowing it to take values of 5 µs, 10 µs and 25 µs. The effect of variations in parameters, like the switching penalty, the rotational speed, and varying slips, were also studied. The data analyzed in each corresponded to a fixed number of discrete time steps far from the initial state, to provide a comparison in the steady state.

Table 1: Average unitary cost function and decision rate for $N_{p1} = 1$ and $N_{p2} = 3$, ($\lambda_u = 10^{-3}$).

Algorithm	N_p	N_L	$\langle J^u \rangle$
SDA	1	n.a.	0.0048
	3	n.a.	0.0010
IL-SDA	3	1	0.0020

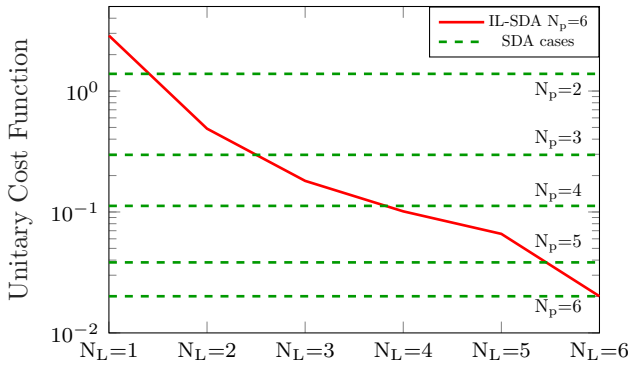

Figure 4: Average unitary cost function with IL-SDA depending on the number of LPs for $N_p = 6$.

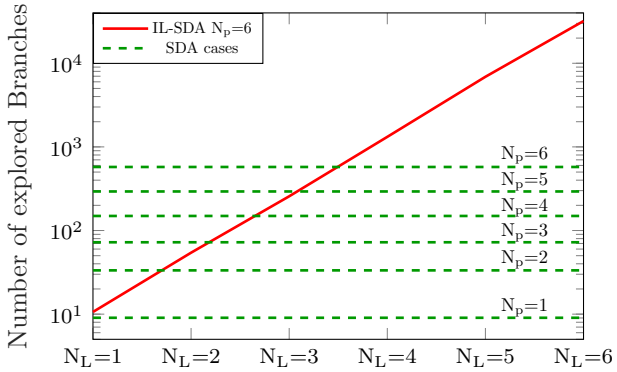

Figure 5: Average number of explored branches with IL-SDA depending on the number of LPs for $N_p = 6$.

4.1 Analysis of the Horizons

The first set of simulations considered prediction horizons of different length $N_p \in [1, 6]$ for the SDA method, represented in Fig. 4 and Fig. 5 as green dashed straight lines. This was compared to the IL-SDA method with prediction horizon of fixed length $N_p = 6$ and different values of the number of LPs $N_L \in [1, 6]$, represented in Fig. 4 and Fig. 5 as red lines.

Figure 4 illustrates an inverse relationship between the average unitary cost function $\langle J^u \rangle$ and the number of LPs N_L. On the other hand Fig. 5 shows the exponential increase in computational burden of IL-SDA method as the number of LPs N_L increases.

Another set of simulations compared the SDA and IL-SDA using different lengths for the prediction horizons N_p, while the number of LPs N_L of IL-SDA is kept constant. It can be seen in Fig. 6 how the case of IL-SDA with a constant $N_L = 2$ consistently reduces the value of the unitary cost function as the prediction horizon increases. With an increase of N_p beyond 4 the optimal value of the cost function falls in a quasi-logarithmic manner. The number of explored branches, shown in Fig. 7, shows that the computational burden in IL-SDA increases marginally with N_p. The average computational burden of the IL-SDA algorithm is slightly higher than the SDA for same N_p and LPs in an application case.

This set of simulations shows that IL-SDA with $N_L = 2$ and $N_p = 7$ can obtain a cost similar to that of SDA with $N_p = 4$ with a computational burden significantly smaller: the number of explored branches is about one order of magnitude less. The results of both sets of simulations show that the number of explored branches is directly

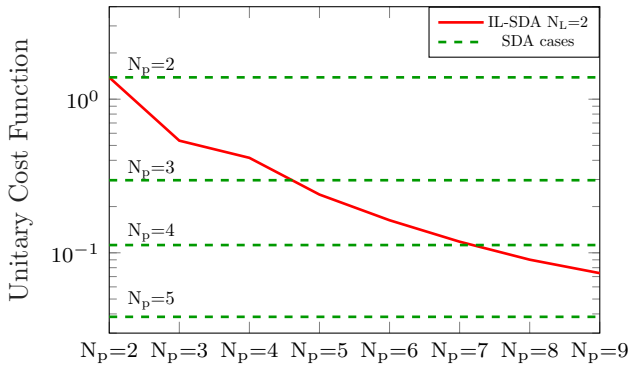

Figure 6: Average unitary cost function with IL-SDA depending on the prediction horizon for $N_L = 2$.

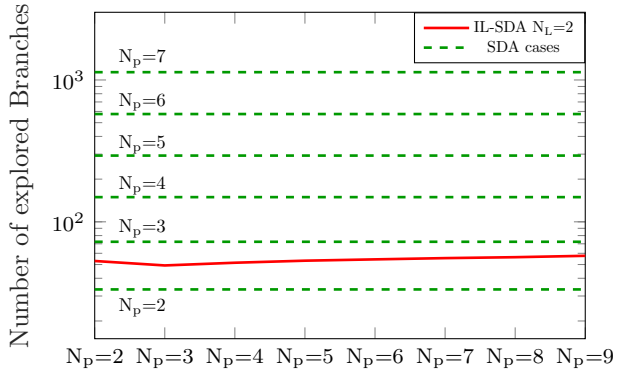

Figure 7: Average number of explored branches with IL-SDA depending on the prediction horizon for $N_L = 2$.

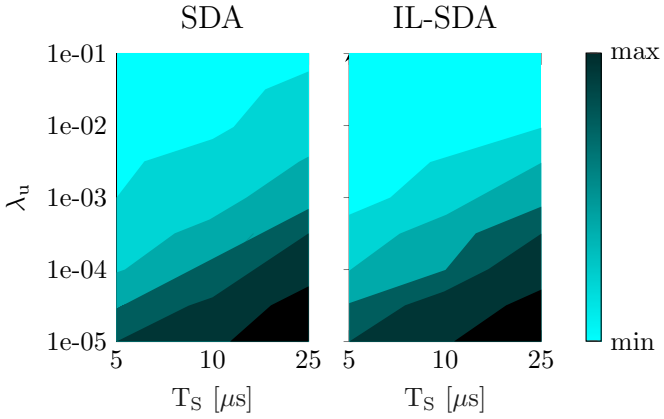

Figure 8: 2D contour map of number of explored branches for SDA and IL-SDA.

proportional to N_L, as it influences the number of solutions evaluated in the IL-SDA algorithm. N_p has a marginal influence on this. Nevertheless, the unitary cost function decreases with both of these parameters.

4.2 Further Parametric analysis

Data from the first set of simulations are now examined to understand the influence of the sampling time T_S and the penalty parameter λ_u. These parameters influence the effective switching frequency, whose optimization reduces thermal losses.

Figure 8 shows the number of explored branches for the SDA and IL-SDA methods as a function of the two parameters of interest. Looking at these results, it is notorious how the number of explored branches for both the SDA and IL-SDA algorithms increase with T_S and decrease with λ_u. Smaller sampling times enable faster executions

in both algorithms. As the penalization parameter increases, the system requires less frequent interventions, reducing the amount of branches explored in each iteration. This benefits, particularly, the IL-SDA algorithm. As stated in (27), the final $3(N_p - N_L)$ elements of the predicted input vector are zero, so the efficiency of that algorithm is maximized. That explains how the reduction of the solution set by the IL-SDA yields good computational performance along a wider operating range than for SDA, specially for higher values of λ_u or lower switching frequencies. For the rest of the contour map, both algorithms behave similarly.

5 Conclusion

A parametric study to compare the deployment of SDA and IL-SDA for MPC of inverters was executed. It has been observed that IL-SDA can provide a good performance in the application of MPC to a three-level inverter connected to a induction machine. Some general tendencies and the influence of some parameters, like the sampling time T_S and the penalty parameter λ_u, have been investigated.

The evaluation criteria employed in the present study are the number of explored branches, as a quantitative metric to assess computational burden, and the unitary cost function, to assess the accuracy. It is important to notice that a targeted reduction of the computational time has not been attempted. The actual execution time of this algorithms might have a strong dependency on the parameters of the induction machine, which may require a more involved case-by-case analysis.

The IL-SDA results show that the method converges to the optimal solution, although the average number of branches explored is slightly larger than with the SDA method. It is also seen that the reduction of the dimensionality produces a slight increase in the optimal value of the cost function and a small increase in the average computational burden. The convergence of the method can be assured even in the worst case scenario. It has also been seen that the computational burden of IL-SDA depends mainly on the number of Laguerre polynomials, N_L, used as control horizon, and that the accuracy can be improved as the prediction horizon N_p increases.

From the parametric analysis, it has been observed that the computational burden in both, the IL-SDA and the SDA algorithms, has a direct correlation to the penalty parameter λ_u and inverse correlation to the sampling time T_S. Moreover, keeping low switching frequencies, or high λ_u, improves the performance of the IL-SDA over the SDA algorithm.

Despite the advantages that MPC offers, its high computational burden poses a challenge for this application. Even in the worst case scenario, the mixed integer optimization problem has to be completed within a sampling period T_S, which is already very small. In this sense, the reduction in the maximum computational burden brought by the usage of IL-SDA with a dimension reduction may justify the risk of obtaining a practical slightly bigger average computation burden.

This study considered a maximum prediction horizon length of $N_p = 7$. Further studies for longer prediction horizons, as long as it is computationally tractable, could be undertaken as an extension of this work. The investigation of the maximum number of explored branches that can be computed during each sample time should be the focus of more studies. This would help to confirm whether the IL-SDA method can be implemented in real-time on drive systems and can provide improvements when compared against other, well-established control methods.

References

[1] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005109878900018

[2] R. Kennel, A. Linder, and M. Linke, "Generalized predictive control (gpc)-ready for use in drive applications?" in *2001 IEEE 32nd Annual Power Electronics Specialists Conference (IEEE Cat. No.01CH37230)*, vol. 4, 2001, pp. 1839–1844 vol. 4.

[3] T. Geyer, N. Oikonomou, G. Papafotiou, and F. D. Kieferndorf, "Model predictive pulse pattern control," *IEEE Transactions on Industry Applications*, vol. 48, no. 2, pp. 663–676, 2012.

[4] P. Karamanakos and T. Geyer, "Guidelines for the design of finite control set model predictive controllers," *IEEE Transactions on Power Electronics*, vol. 35, no. 7, pp. 7434–7450, 2020.

[5] J. Rodriguez, M. P. Kazmierkowski, J. R. Espinoza, P. Zanchetta, H. Abu-Rub, H. A. Young, and C. A. Rojas, "State of the art of finite control set model predictive control in power electronics," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 2, pp. 1003–1016, 2012.

[6] T. Geyer and D. E. Quevedo, "Multistep finite control set model predictive control for power electronics," *IEEE Transactions on Power Electronics*, vol. 29, no. 12, pp. 6836–6846, 2014.

[7] D. Struve, A. L. Pulzovan, and T. F. Geyer, "Assessment of a laguerre polynomial based sphere decoding algorithm for direct mpc of inverters," in *2024 European Control Conference (ECC)*, 2024, pp. 3337–3344.

[8] T. Geyer, *Model predictive control of high power converters and industrial drives*, 1st ed. Chichester West Sussex United Kingdom: Wiley, 2017.

[9] L. Wang, *Model predictive control system design and implementation using MATLAB*, ser. Advances in industrial control. London: Springer, 2009.