ITERATIVE SLICING AS SOLUTION TO THE MODEL PREDICTIVE CONTROL PROBLEM OF A THREE-LEVEL INVERTER

J. Raath [*] **and H. du Toit Mouton** [†] **and T. Geyer** [‡]

[*] *Department of Electrical, Electronic and Computer Engineering, Central University of Technology, Bloemfontein, South Africa, E-mail: jraath@cut.ac.za*

[†] *Department of Electrical and Electronic Engineering, University of Stellenbosch, Stellenbosch, 7602, South Africa, E-mail: dtmouton@sun.ac.za*

[‡] *ABB Corporate Research, ABB Switzerland Ltd, Power Electronic Systems, Switzerland, E-mail: t.geyer@ieee.org*

Abstract: This paper considers a model predictive controller with reference tracking that manipulates the integer switch positions of a power converter. It can be shown that the optimal switch position can be computed in a new coordinate system by solving the closest vector problem in a lattice by iterative slicing. A list of Voronoi relevant vectors defining the basic Voronoi cell of a lattice is used to find the Voronoi cell containing the unconstrained optimum in an iterative manner. This concept is exemplified for a three-level single-phase converter with an RL load.

Key words: Model Predictive Control, Voronoi diagram, Voronoi relevant vectors, Closest vector problem, Iterative slicing.

1. INTRODUCTION

Model predictive control (MPC), or receding horizon control, uses a model of the system to predict over a finite horizon the future evolution of the system's controlled variables. Using MPC over extended horizons has the potential to improve the performance significantly [1, 2]. Unfortunately, this leads to an exponential increase in the number of possible control sequences. In practice this implies more computations to be performed during each sampling interval. In power electronic systems with short sampling intervals, this computational burden is a challenging factor when considering an extension of the MPC horizon.

The main purpose of our research is to reduce the online computational burden of the controller so as to practically implement extended horizon MPC for a multilevel inverter. To achieve this, it is necessary to find the solution to the optimization problem, i.e. the predicted control vector of minimum Euclidean distance to the unconstrained optimum in a transformed solution space. The work presented here is a continuation of our previous work done in addressing the same challenge [3]. Instead of utilizing a binary search tree for solving the *nearest neighbor problem* we now address the challenge via lattice theory and the classic mathematical problem of *closest vector problem* (CVP) as noted in the study of *geometry of numbers* [4]. The general closest vector problem, as a function of the dimension, has been shown to be non-deterministic polynomial-time hard (NP-hard) [5], implying that all *exact* solutions have exponential complexity. Approximate solutions are faster and terminate in higher dimensions but have also proven to be NP-hard for a certain degree of optimality/exactness [6]. NP is formally referred to as the set of decision problems solvable in polynomial time by a theoretical non-deterministic Turing machine [7].

Three main approaches exist for solving the CVP, namely enumeration based algorithms, space saturation algorithms and Voronoi based algorithms. Enumeration based algorithms following the Pohst [8] strategy have traditionally been used as a practical tool [9]. This research is aimed at implementing an alternative Voronoi based method with pre-processing to find the *exact* solution to CVP and subsequently optimizing the MPC problem. The work applies the lattice slicing algorithm to the unique lattice structure generated by the MPC optimization problem in an attempt to evaluate performance and practicality of the off- and on-line computational burden.

Section 2 introduces the mathematical background of the MPC problem with the resulting state-space and geometry laid out in section 3. In section 4 the lattice approach and CVP solver are presented. The inverter implementation and pre-processing requirements are stipulated in section 5. Section 6 concludes the paper.

2. MODEL PREDICTIVE CONTROL

The control action is determined by solving a finite horizon open-loop optimal control problem at each sampling instant, using the current state of the system, searching for an optimal control sequence over the horizon and then applying the first control action in the sequence to the system. MPC applied to a single-phase Neutral Point Clamped (NPC) inverter controls the inverter switches in such a manner so as to generate an output current i in the load that tracks a reference current i_r as closely as possible with minimal switching losses in the power switches. The inverter topology is shown in Figure 1 with the neutral point voltage assumed constant.

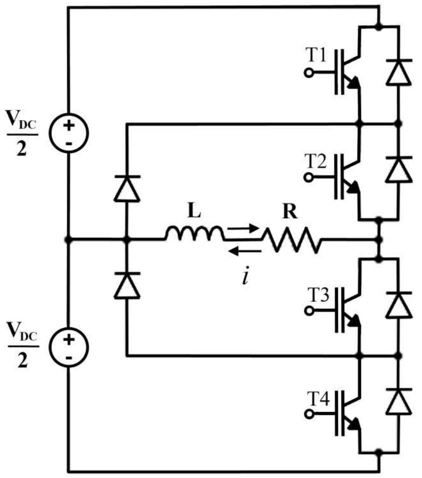

Figure 1: Single-phase Neutral Point Clamped inverter

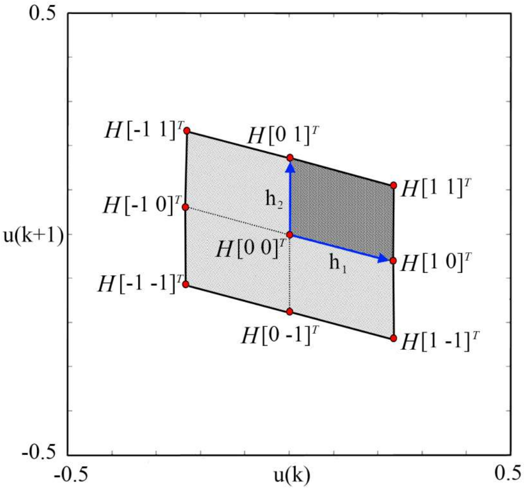

Figure 2: HU input vectors in the H-coordinate solution space for horizon $N = 2$.

2.1 *Mathematical modeling*

The NPC inverter leg can deliver three voltage levels of $-0.5V_{DC}$, $0V_{DC}$ and $+0.5V_{DC}$ across the load which can be represented by the integer values $u \in \{-1, 0, +1\}$ defining the state of the switch positions. The voltage applied to the *RL* load is $v(t) = 0.5V_{DC} \cdot u(t)$. The differential equation describing the *RL* load in the continues time domain is:

$$v(t) = Ri(t) + L\frac{di(t)}{dt}. \tag{1}$$

Converting (1) to the discrete time domain with sample period T_s results in the predictive load current model $i(k+1)$ with input $u(k)$ and state $i(k)$ as noted in

$$i(k+1) = Ai(k) + Bu(k) \tag{2}$$

with $A = e^{-T_s/\tau}$, $B = \frac{V_{DC}}{2R}(1-A)$ and $\tau = \frac{L}{R}$.

2.2 *Cost function*

To find the optimal control input to the inverter, we define over a finite horizon N the quadratic cost function

$$J = \sum_{l=k}^{k+N-1} (i_r(l+1) - i(l+1))^2 + \lambda_u (\Delta_u(l))^2. \tag{3}$$

This function is similar to the cost function defined in [1] and consist of two terms, $(i_r(l+1) - i(l+1))^2$ to quantify tracking error from the reference current i_r, and $\lambda_u (\Delta_u(l))^2$ the switching cost. A tuning factor λ_u is used to adjust the weight of the switching cost. J is a function of the switching sequence $U = [u(k)u(k+1)...u(k+N-1)]^T$ which leads to an exponential increase in possible switching sequences over the horizon to evaluate.

2.3 *Optimization problem*

The optimization problem for finding the optimum switching sequence U_{opt} can be stated formally as

$$U_{opt}(k) = \arg\min_{U(k)} J, \tag{4}$$

subject to

$$i(l+1) = Ai(l) + Bu(l) \tag{5}$$

$$u(l) \in \{-1, 0, +1\}$$

$$\forall l = k, ..., k+N-1.$$

A solution to the optimization problem (4) can be found by rewriting the cost function in terms of the unconstrained optimal solution $U_{unc}(k)$ as derived in [1] and [10]

$$J = \|HU(k) - HU_{unc}(k)\|_2^2. \tag{6}$$

H is an invertible lower triangular matrix that transforms the switching sequence $U(k)$ and the unconstrained optimal $U_{unc}(k)$ to the m-dimensional H-coordinate solution space with $m = N$.

3. SOLUTION SPACE PARTITIONING

3.1 *Solution space*

Solving the optimization problem (4) with cost function (6) translates into the nearest neighbor search of the m-dimensional vector $HU_{unc}(k)$ to the set of m-dimensional input vectors $HU(k)$ in $\mathbb{R}^m$ Euclidean space. Figure 2 depicts the spatial arrangement of the 3^N input vectors in the H-transformed coordinate space for the horizon $N = 2$ case. It can be observed that the arrangement constitutes an m-dimensional parallelotope with 2^m orthants. The convex hull of the parallelotope is defined by $2m$ hyperplanes of dimension $m-1$.

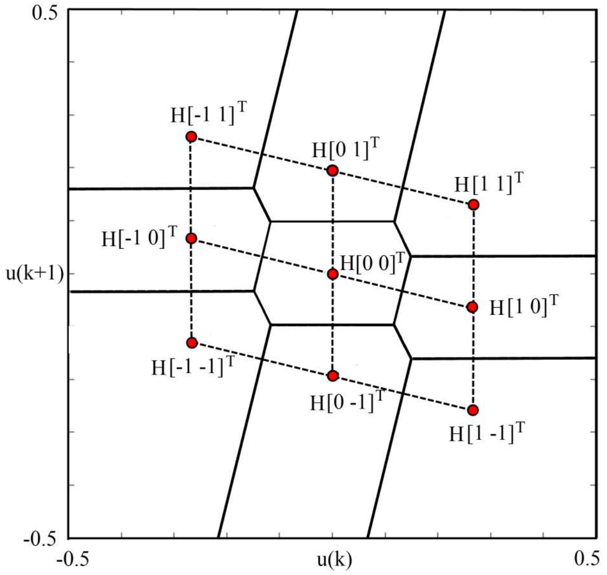

Figure 3: Voronoi diagram for the *HU* sites in the *H*-coordinate space for horizon $N = 2$.

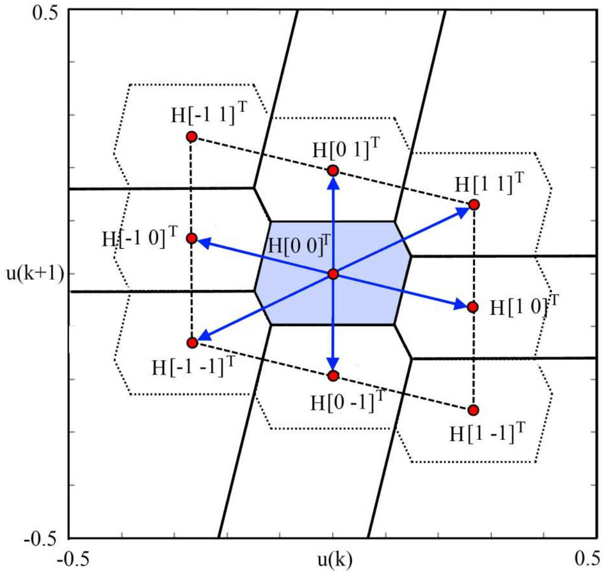

Figure 4: The *Voronoi relevant vectors* defining the *basic Voronoi cell* of a lattice and the translation thereof to all lattice points in the *H*-coordinate space ($N = 2$).

3.2 Voronoi partitioning

Polyhedral partitioning of the H-solution space into a Voronoi diagram results in a number of convex Voronoi cells, each consisting of all vectors in $\mathbb{R}^m$ nearest to their respective HU Voronoi site [11]. The Voronoi cell V of an individual site $s \in HU$ can be defined as

$$V(HU,s) = \{x \in \mathbb{R}^m : \|x-s\| \leq \|x-j\|\} \quad (7)$$
$$\forall j \in (HU), j \neq s.$$

The Voronoi diagram for the horizon $N = 2$ case is shown in Figure 3 with 16 hyperplanes partitioning the H-coordinate space and defining the borders of the subsequent Voronoi cells.

4. CVP SOLVER

4.1 Voronoi cell of a lattice

The spatial arrangement of the HU input vectors can also be defined in lattice theory as

$$\Lambda(H) = \Lambda(h_1,h_2,\ldots,h_n) = \{HU \,|U \in \mathbb{Z}^n\}. \quad (8)$$

The lattice Λ is generated from the *full-rank* base matrix H with linearly independent column vectors $h_1,h_2,\ldots,h_n \in \mathbb{R}^m$. The dimension and rank of the lattice are denoted by the number of rows m and columns n of the base matrix H, respectively. The two base vectors h_1,h_2 for the horizon $N = 2$ case are indicated in Figure 2 with the darkened orthant highlighting the span of the lattice base.

The Voronoi cells for a lattice structure are convex polyhedra and are symmetrical in reflection through its Voronoi site s. This can be observed in Figure 4 from the shaded Voronoi cell enclosing the origin $s = \underline{0}$ also known as the *basic Voronoi cell* of a lattice. The borders defining the m-dimensional basic Voronoi cell are $(m-1)$-dimensional hyperplanes derived from their respective *Voronoi relevant vectors* v. Voronoi relevant vectors are lattice points closest to a specific lattice point; in a m-dimensional lattice a maximum of $2^{m+1}-2$ Voronoi relevant vectors can exist [12]. Each Voronoi relevant vector is bisected orthogonally at its midpoint $\frac{1}{2}v$ by a hyperplane defining the border between the Voronoi site and its Voronoi relevant vector. A border hyperplane is also known as a facet of the respective Voronoi cell. In Figure 4 the Voronoi relevant vectors to the origin are indicated by the 6 blue arrows with their respective facets defining the basic Voronoi cell. The translative nature of the basic Voronoi cell to all other lattice points is illustrated by the dotted lines. The Voronoi relevant vectors can be described as a subset $S(\Lambda) \subseteq \Lambda$ of the lattice, defining the basic Voronoi cell as

$$V(\Lambda,\underline{0}) = \{x \in \mathbb{R}^m : \|x\| \leq \|x-v\|\} \quad (9)$$
$$\forall v \in S(\Lambda), v \neq \underline{0}.$$

Due to the symmetrical nature of the Voronoi cell $V(\Lambda,\underline{0})$ the Voronoi relevant vectors are also reflections and for every v the vector $-v$ is also a Voronoi relevant vector. The Voronoi cell can thus be defined in terms of Voronoi relevant vector pairs $\{v,-v\}$. This simplifies $S(\Lambda)$ to a minimal set $S'(\Lambda) \subseteq \Lambda$ where only one representative of each Voronoi relevant vector pair is used to define the set. The basic Voronoi cell can now be defined by

$$V(\Lambda,\underline{0}) = \{x \in \mathbb{R}^m : \|x\| \leq \|x-v\|, \|x\| \leq \|x+v\|\} \quad (10)$$
$$\forall v \in S'(\Lambda), v \neq \underline{0}.$$

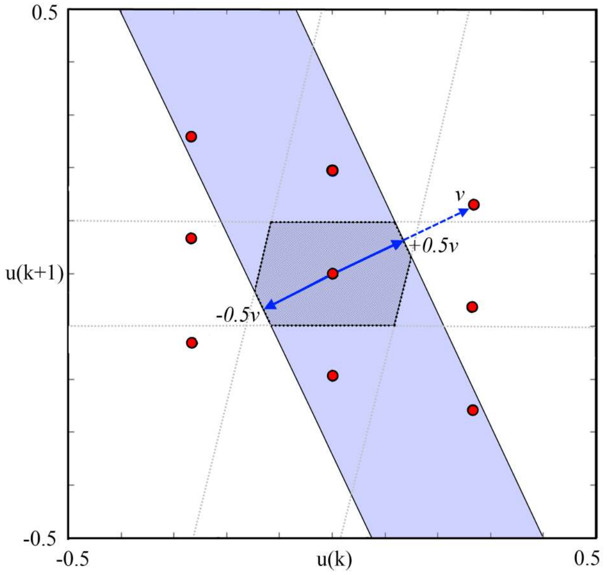

Figure 5: Slicing of the H-coordinate space ($N = 2$) with Voronoi border pairs located at $\pm\frac{1}{2}v$.

This clearly halves the number of Voronoi relevant vectors required to define the basic Voronoi cell of a lattice. The advantage obtained from the symmetry and the translation of the basic Voronoi cell in a m-dimensional lattice is that all the borders defining the *Voronoi diagram* of the solution space can be determined from $2^m - 1$ *single-sided Voronoi relevant vectors*. For the horizon $N = 2$ case this equals 3 single-sided Voronoi relevant vectors representing 3 hyperplanes or facets needed to describe the Voronoi diagram consisting of 16 hyperplanes in total.

4.2 *Iterative slicing algorithm*

In the quest to find the closest lattice point to a vector the algorithm developed by [13] achieves the objective by iteratively calculating which Voronoi cell of the lattice contains the given vector. Determining the closest lattice point s to a vector $x \in \mathbb{R}^m$ is done by finding the lattice point $s \in \Lambda$ with the error vector $e = s - x$ that resides inside the basic Voronoi cell $V(\Lambda, \underline{0})$. To verify the containment of x in $V(\Lambda, \underline{0})$ a number of *slicing* operations are required using the pre-calculated list of Voronoi relevant vectors. Instead of using distances, the basic Voronoi cell is re-defined in terms of inner products to give

$$V(\Lambda, \underline{0}) = \left\{ x \in \mathbb{R}^m : |x \cdot v| \leq \frac{1}{2} \|v\|^2 \right\} \tag{11}$$

$$\forall v \in S'(\Lambda), v \neq \underline{0}.$$

Geometrically $|x \cdot v|$ translates into the orthogonal projection of x onto v, where the Voronoi border pair defined by v is orthogonal and located at $\pm\frac{1}{2}v$. This *slice* through $\mathbb{R}^m$ is illustrated in Figure 5. The algorithm typically starts with the lattice point at the origin $s = \underline{0}$ and iteratively updates s until the resulting error e is found to be between all the slices defined by the set $v \in S'(\Lambda)$. At first glance this search process seems to be exhaustive in nature but the algorithm implements two principles for speeding up.

Firstly, the Voronoi relevant vectors $v \in S'(\Lambda)$ are sorted in an ascending order in terms of their Euclidean norm during the pre-calculation stage. This arrangement is then utilized during the *on-line* operation of the algorithm when the error vector is evaluated by $\|e\|^2 < \frac{1}{4}\|v\|^2$ for a specific Voronoi relevant vector v. Geometrically this verifies if e resides within a sphere of radius $\frac{1}{2}v$. If so then it is not necessary to verify containment of e by the rest of the Voronoi relevant vectors in the ranked set of increasing Euclidean norm thus eliminating unnecessary computations.

Secondly, a further reduction in computational complexity is obtained by not initializing the algorithm from the typical lattice point $s = \underline{0}$ but from an estimated lattice point closer to the given vector x. This will ensure $e = s - x$ to be smaller, resulting in less iterations and faster termination of the algorithm. This estimation can be achieved by various methods of which the Babai rounding technique is a good example. The method estimates a lattice point by

$$s = H\left\lfloor xH^{-1} \right\rceil, \tag{12}$$

with H the lattice basis of full rank. Although this rounding method is not guaranteed to solve CVP exactly, it is simple and has been proved by Babai that $\|s - x\|$ is within an exponential factor of the minimal value if the lattice basis is LLL-reduced [14]. Due to length constraints on this paper, any further details on the lattice slicing algorithm is deterred from but the reader is referred to [13] for a comprehensive description.

5. IMPLEMENTATION

5.1 *Off-line requirements*

The load model parameters are required to determine the H-transformation matrix for the horizon N. The lattice slicing algorithm requires the basic Voronoi cell of a lattice to solve the closest vector problem. This is a task that is also deemed to be NP-hard [15], but since this operation is only to be done once in the off-line process, we opted for using an altered sphere-decoder from [9] to calculate the $2^{m+1} - 2$ closest vectors to the origin. These vectors are in ± pairs and only one representative *one-sided Voronoi relevant vector* of each pair is stored and ranked in terms of their Euclidean norm. Another off-line computation required is defining the convex hull of the m-dimensional parallelotope by its $2m$, $(m-1)$-dimensional hyperplanes. This is done for reasons explained in the on-line procedure.

5.2 *On-line requirements*

Implementing MPC in an multi-level inverter equates to solving the closest vector problem (CVP) in the N-dimensional H-solution space as described in section

3.1. The lattice slicing algorithm solves CVP in a lattice and it is not valid for finding vectors located outside the lattice structure. In our three-level inverter application, the lattice of the HU sites is as defined in (7) but with the limitation that $U \in \{-1,0,+1\}$ (the lattice is truncated). The valid region of operation for the slicing algorithm for our lattice is the area enclosed by the basic Voronoi cell and all translations thereof to the surrounding lattice points limited to $\{-1,0,+1\}$ in m-dimensions. This area is graphically illustrated in Figure 4 by the bounding dotted line. The Voronoi diagram partitioning the solution space for the HU-sites are similar to the Voronoi cell translation of the HU-lattice only inside the described parallelotope. Outside the parallelotope the partitioning differs because our lattice is bounded by $\{-1,0,+1\}$ resulting in un-bounded Voronoi cells.

Therefore we must consider two possible scenarios that might occur. One, in which HU_{unc} is inside the convex hull of the parallelotope and another one in which HU_{unc} is located outside the parallelotope. The first situation is the norm under stable continuous operation of the inverter and the second usually occurs under transient conditions. In addressing the second scenario it can be observed that all Voronoi borders exiting the parallelotope are orthogonal to the hull of the parallelotope. Thus if HU_{unc} is located outside the parallelotope we project HU_{unc} orthogonally to a point with minimum distance on the hull of the parallelotope. HU_{unc} is then updated with this new hull-point which is valid for the lattice slicing algorithm to solve. To initialize the algorithm, an approximated lattice point is determined by utilizing (12)

$$HU_{est} = H\left\lfloor U_{unc}H^{-1}\right\rceil. \quad (13)$$

The algorithm finds the closest HU lattice point to HU_{unc} which relates to the optimal control sequence U_{opt} for the inverter over N-horizons. From this sequence only the first control action $u(k)$ is selected and applied to the inverter. A flowchart for the on-line process is shown in Figure 6.

5.3 Simulation

The proposed *off-* and *on-line* procedures were implemented in a *MATLAB*© simulation model of the single-phase NPC three-level inverter. The validity and performance of the control actions selected by the *iterative slicer* were compared to those of the benchmark *exhaustive search method* and an adapted version of the *sphere-decoder*. The sphere-decoder applied the standard top-down search method and was initialized with a sphere radius equal to the initial error vector as introduced in the lattice slicing algorithm

$$e = \|HU_{est} - HU_{unc}\|. \quad (14)$$

A typical example with the following parameters is considered. Sampling interval of $T_S = 25\mu$s, load-resistance of $R = 2\Omega$, and inductance $L = 2m$H. The rated r.m.s. output voltage of the inverter is $V_{AC} = 3.3k$V with an input dc-link voltage of $V_{DC} = 5.2k$V. Base quantities

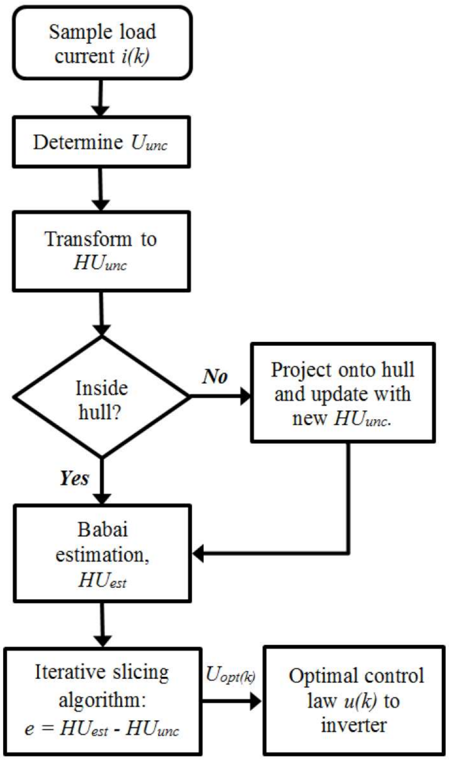
Sample load current i(k)
Determine Uunc
Transform to HUunc
Inside hull?
No
Project onto hull and update with new HUunc.
Yes
Babai estimation, HUest
Iterative slicing algorithm: e = HUest - HUunc
Uopt(k)
Optimal control law u(k) to inverter

Figure 6: Flowchart of the *on-line* process for attaining the optimal inverter control action during every sample period.

are used to establish a per unit system and the current reference is assumed to be 0.8pu amplitude. A tuning factor λ_u of 0.02 was selected. To determine the straight out performance of the respective algorithms in terms of their CVP solving capability we opted for a random set of input vectors $i(k)$ that limited HU_{unc} to stay within the borders of the HU parallelotope. Using random values instead of a reference sine-wave ensured that no sequence or function is followed that can possibly assist in the prediction of HU_{est}. Limiting the input values and thus U_{unc} to values between $\{-1,+1\}$ eliminated possible variations in performance due to the different approaches used in addressing transient conditions. The performance in terms of average processing time by the respective algorithms for 1000 samples over horizons 1 to 9 is plotted in Figure 7.

The exhaustive search method enumerates all switching sequences; the exponential increase in the computational burden when extending the prediction horizon is shown in Figure 7. This method is practical only in horizon 1 and 2 applications. We used it as benchmark and to verify the correctness of the control actions determined by the other two methods. Over the respective horizons the control sequence (U_{opt}) determined by both the iterative slicing and sphere decoding methods correlated 100 percent with the sequence obtained using the exhaustive search method. In terms of algorithm termination time the iterative slicing method outperformed the sphere decoder for horizons $N < 6$ or dimensions $m < 6$. In higher dimensions, its performance deteriorated due to the rapid increase in

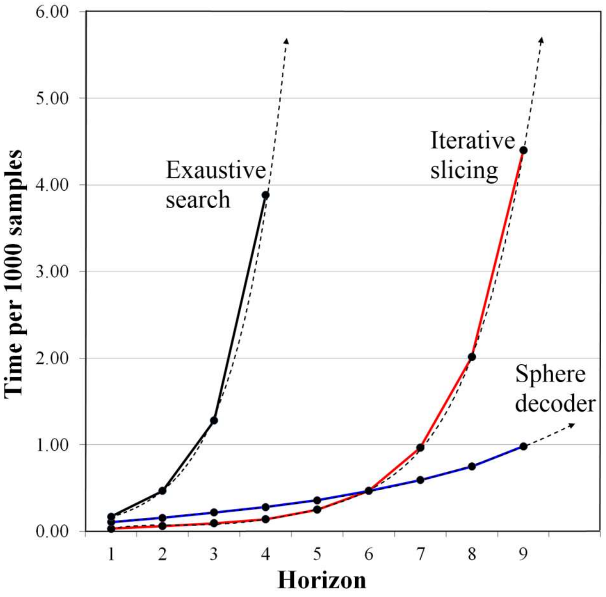

Figure 7: Algorithm termination times per 1000 random samples over *N*-horizons.

complexity of the basic Voronoi cell. Containment of a specific vector in a Voronoi cell is verified by the iterative evaluation of $2^m - 1$ single-sided Voronoi relevant vectors.

6. CONCLUSION

We presented the application of iterative slicing as alternative to the traditional enumeration method in solving the closest vector problem and thus the MPC problem for a single-phase NPC inverter. The method obtains the *exact* solution to the problem and performed well in dimensions below seven. Although the desired performance in higher dimension has not been achieved, some progress compared to our previous work was obtained in the pre-processing time. This is due to the calculation of the basic Voronoi cell of a lattice and storage thereof in terms of the single-sided Voronoi relevant vectors compared to storing all the Voronoi borders in the Voronoi diagram of the partitioned solution-space. Future work on reducing the computational burden will include the possible improvement in initial lattice point approximation and optimization of the search sequence through the lattice structure.

REFERENCES

[1] T. Geyer and D. E. Quevedo, "Multistep finite control set model predictive control for power electronics", *Power Electronics, IEEE Transactions on*, vol. 29, pp. 6836-6846, 2014.

[2] D. Quevedo, R. Aguilera, and T. Geyer, "Predictive Control in Power Electronics and Drives: Basic Concepts, Theory, and Methods", *in Advanced and Intelligent Control in Power Electronics and Drives*, Vol. 531, T. Orowska-Kowalska, F. Blaabjerg, and J. Rodrguez, Eds., ed: Springer International Publishing, pp. 181-226, 2014.

[3] Raath, J., Mouton, H. and Geyer, T., "Integration of inverter constraints in geometrical quantification of the optimal solution to an MPC controller", *In Control and Modeling for Power Electronics (COMPEL)*, 2016 IEEE 17th Workshop on, pp. 1-6, June 2016.

[4] J. W. S. Cassels, "An introduction to the geometry of numbers", *Springer-Verlag*, New York, 1971.

[5] D. Micciancio, "The hardness of the closest vector problem with preprocessing", *IEEE Trans. Inform. Theory*, vol. 47, pp. 1212-1215, March 2001.

[6] C. Lamy and J. Boutros, "On random rotations diversity and minimum MSE decoding of lattices", *IEEE Trans. Inform. Theory*, vol. 46, pp. 1584-1589, July 2000.

[7] Alsuwaiyel, M.H., "Design techniques and analysis", *Algorithms*, 10, pp. 283, 1999.

[8] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications", *ACM SIGSAM Bulletin*, vol. 15, pp. 37-44, Feb. 1981.

[9] Agrell, E. ; Eriksson, T. ; Vardy, A. et al., "Closest point search in lattices", *IEEE Transactions on Information Theory*, vol. 48,(8) pp. 2201-2214, 2002.

[10] D. E. Quevedo, G. C. Goodwin, and J. A. De Dona, "Finite constraint set receding horizon quadratic control," *International journal of robust and nonlinear control*, vol. 14, pp. 355-377, 2004.

[11] G. M. Voronoi, "Nouvelles applications des parametres continus a la theorie des formes quadratiques. deuxieme Memoire: Recherches sur les parallelloedres primitifs", *J. Reine Angew. Math.*, 134:198-287, 1908.

[12] Bonifas, N. and Dadush, D., "Short paths on the Voronoi graph and the closest vector problem with preprocessing", *arXiv preprint*, arXiv:1412.6168, Dec 2014.

[13] Sommer, N., Feder, M. and Shalvi, O., "Finding the closest lattice point by iterative slicing", *SIAM Journal on Discrete Mathematics*, vol. 23,(2) pp. 715-731, 2009.

[14] Babai, L., "On Lovszlattice reduction and the nearest lattice point problem", *Combinatorica*, 6(1), pp.1-13, 1986.

[15] M. D. Sikiric, A. Schurmann, and F. Vallentin., "Complexity and algorithms for computing Voronoi cells of lattices", *Mathematics of Computation*, 78(267):1713-1731, July 2009.