Distributed Convolutional Sparse Coding

Abstract

We consider the problem of building shift-invariant representations for long signals in the context of distributed processing. We propose an asynchronous algorithm based on coordinate descent called DICOD to efficiently solve the $\ell_1$-minimization problems involved in convolutional sparse coding. This algorithm leverages the weak temporal dependency of the convolution to reduce the inter-process communication to a few local messages. We prove that this algorithm converges to the optimal solution and that it scales with superlinear speedup, up to a certain limit. These properties are illustrated with numerical experiments and our algorithm is compared to the state-of-the-art methods used for convolutional sparse coding.

1 Introduction

Sparse coding aims at building sparse linear representations of a dataset based on a family of basic elements called atoms. It has proven to be useful in many different applications ranging from EEG to images and audio (Adler et al., 2013; Kavukcuoglu et al., 2010; Mairal et al., 2009; Grosse et al., 2007). Convolutional sparse coding is a specialization of this approach, focused on building sparse, shift-invariant representations of signals. Such representations present a major interest for applications like segmentation or classification as they separate the shape and the localization of patterns in a signal. Convolutional sparse coding can also be used to estimate the similarity of two signals which share similar local patterns and to find the correspondences between different temporal events. Depending on the context, the dictionary can either be fixed analytically (e.g. wavelets, see Mallat 2009), or learned from the data (Bristow et al., 2013; Mairal et al., 2009).

Several algorithms have been proposed to solve the convolutional sparse coding problem. In Kavukcuoglu et al. (2010), the authors extend to convolutional sparse coding the coordinate descent (CD) methods introduced by Friedman et al. (2007). This method greedily optimizes one coordinate at each iteration using fast local updates. The Fast Iterative Soft-Thresholding Algorithm (FISTA) was adapted for convolutional problems in Chalasani et al. (2013) and uses proximal gradient descent to compute the representation. The Feature Sign Search (FSS), introduced in Grosse et al. (2007), solves at each step a quadratic subproblem for an active set of the estimated nonzero coefficients and the Fast Convolutional Sparse Coding (FCSC) of Bristow et al. (2013) is based on Alternating Direction Method of Multipliers (ADMM).

Up to our knowledge, there is no scalable version of these algorithms for long signals. This is a typical situation, for instance, in physiological signal processing where sensor information can be collected for a few hours with sampling frequencies ranging from 100 to 1000Hz. In such a
context, there is a need for local and distributed algorithms which would address the problem of convolutional sparse coding. Recent studies have considered distributing coordinate descent algorithms for general ℓ1-regularized minimization (Scherrer et al., 2012a;b; Bradley et al., 2011; Yu et al., 2012). These papers derive general purpose synchronous algorithm using either locks or synchronization steps to ensure convergence in general case. In You et al. (2016), the authors derive an asynchronous distributed algorithm for the projected coordinate descent. However, this work relies on centralized communication and finely tuned step size to ensure the convergence of the method. By exploiting the structure of the convolutional problem, we design a novel distributed algorithm based on coordinate descent, named Distributed Convolution Coordinate Descent (DICOD). DICOD is asynchronous and each process can run independently without any locks or synchronization step. This algorithm uses a local communication scheme to reduce the number of inter-process message needed and does not rely on external learning rates. We prove in this paper that this algorithm scales super-linearly with the number of cores compared to the sequential CD, up to certain limitations.

In Section 2, we introduce the DICOD algorithm for the resolution of the convolutional sparse coding. Then, we prove in Section 3 that DICOD converges to the optimal solution for a wide range of settings and we analyze its complexity. Finally, Section 4 presents numerical experiments that illustrate the benefits of the DICOD algorithm with respect to other state-of-the-art algorithms and validate our theoretical analysis.

## 2 Distributed Convolutional Coordinate Descent (DICOD)

### Notations.

The space of multivariate signals of length $T$ in $\mathbb{R}^P$ is denoted by $\mathcal{X}_T^P$. We will denote with lower case univariate signals $x \in \mathcal{X}_T^1$, with upper case letters multivariate signals $X \in \mathcal{X}_T^P$ in $\mathbb{R}^P$. For these signals, their value at time $t \in [0..T-1]$ is denoted by $x[t] \in \mathbb{R}$ and $X[t] \in \mathbb{R}^P$ and for all $t \in [0..T-1]$, $x[t] = 0$ and $X[t] = 0_P$. The indicator function of $t_0$ is denoted $1_{t_0}$.

For any signal $X \in \mathcal{X}_T^P$, the reversed signal is defined as $\bar{X}[t] = X[T-t]$, the d-norm is defined as $\|X\|_d = \left(\sum_{t=0}^{T-1} \|X[t]\|_d^d\right)^{1/d}$ and the replacement operator as $\Phi_{t_0}(X)[t] = (1 - 1_{t_0}(t))X[t]$ , which replaces the value in $t$ by 0. Finally, for $L, W \in \mathbb{N}^*$, the convolution between $z \in \mathcal{X}_T^1$ and $D \in \mathcal{X}_W^P$ is the multivariate signal $X \in \mathcal{X}_T^P$ with $T=L+W-1$ such that

$$X[t] = (z * D)[t] \triangleq \sum_{\tau=0}^{W-1} z[t-\tau]D[\tau], \quad \forall t \in [0..T-1].$$

### 2.1 Coordinate descent for convolutional sparse coding

**Convolutional sparse coding.** Consider the multivariate signal $X \in \mathcal{X}_T^P$. Let $D = \{D_k\}_{k=1}^K \subset \mathcal{X}_W^P$ be a set of $K$ patterns with $W \ll T$ and $Z = \{z_k\}_{k=1}^K \subset \mathcal{X}_T^1$ be a set of $K$ activation signals

### Algorithm 1 Greedy Coordinate Descent

1: **Input:** $D, X$, parameter $\epsilon > 0$
2: $C = \{1..K\} \times \{0..L-1\}$
3: Initialization: $\forall (k, t) \in C$, $z_k[t] = 0$, $\beta_k[t] = (\bar{D}_k * X)[t]$
4: repeat
5: $\forall (k, t) \in C$, $z'_k[t] = \frac{1}{\|D_k\|_2^2} \text{Sh}(\beta_k[t], \lambda)$
6: Choose $(k_0, t_0) = \arg \max_{(k, t) \in C} \|\Delta z_k[t]\|_0$
7: Update $\beta$ using (6) and $z_{k_0}[t_0] \leftarrow z'_{k_0}[t_0]$
8: until $|\Delta z_{k_0}[t_0]| < \epsilon$

### Algorithm 2 DICODM

1: **Input:** $D, X$, parameter $\epsilon > 0$
2: **In parallel** for $m = 1 \cdots M$
3: For all $(k, t) \in C_m$, initialize $\beta_k[t]$ and $z_k[t]$
4: repeat
5: Receive messages and update $\beta$ with (6)
6: $\forall (k, t) \in C_m$, compute $z'_k[t]$ with (5)
7: Choose $(k_0, t_0) = \arg \max_{(k, t) \in C_m} |\Delta z_k[t]|$
8: Update $\beta$ with (6) and $z_{k_0}[t_0] \leftarrow z'_{k_0}[t_0]$
9: if $t_0 - mL_M < W$ then
10: Send $(k_0, t_0, \Delta z_{k_0}[t_0])$ to core $m - 1$
11: if $(m + 1)L_M - t_0 < W$ then
12: Send $(k_0, t_0, \Delta z_{k_0}[t_0])$ to core $m + 1$
13: until for all cores, $|\Delta z_{k_0}[t_0]| < \epsilon$
with \( L = T - W + 1 \). The convolutional sparse representation models a multivariate signal \( X \) as the sum of \( K \) convolutions between a local pattern \( D_k \) and an activation signal \( z_k \) such that:

\[
X[t] = \sum_{k=1}^{K} (z_k \ast D_k)[t] + \mathcal{E}[t], \quad \forall t \in [0..T - 1].
\]

with \( \mathcal{E} \in \mathbb{R}^T \) representing an additive noise term. This model also assumes that the coding signals \( z_k \) are sparse, in the sense that only few entries are nonzero in each signal. The sparsity property forces the representation to display localized patterns in the signal. Note that this model can be extended to higher order signals such as images by using the proper convolution operator. In this study, we focus on 1D-convolution for the sake of simplicity.

Given a dictionary of patterns \( D \), convolutional sparse coding aims to retrieve the sparse decomposition \( Z \) associated to the signal \( X \) by solving an \( \ell_1 \)-regularized optimization problem:

\[
Z^* = \arg \min_Z E(Z) \triangleq \frac{1}{2} \left\| X - \sum_{k=1}^{K} z_k \ast D_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \| z_k \|_1,
\]

for a given regularization parameter \( \lambda > 0 \). (2) can be interpreted as a special case of the LASSO problem with a band circulant matrix. Therefore, classical optimization techniques designed for LASSO can be applied to solve it with the same convergence guarantees.

**Convolutional coordinate descent.** The method proposed by Kavukcuoglu et al. (2010) iteratively updates at each iteration one coordinate \((k_0, t_0)\) of the sparse code. When the coefficient \((k, t)\) of \( Z^{(i)} \) is updated to a value \( u \in \mathbb{R} \) for \( Z^{(i+1)} \), a simple function of \( u \) gives the reduction of the cost obtained with this update and we denote its maxima

\[
\Delta E_{k}[t] = \max_{u \in \mathbb{R}} e_{k,t}(u) = E(Z^{(i)}) - E(Z^{(i+1)}).
\]

The greedy coordinate descent updates the chosen coordinate \((k_0, t_0)\) to the value \( z'_{k_0}[t_0] = \arg \max_{u \in \mathbb{R}} e_{k_0,t_0}(u) \), maximizing the cost reduction of the update. The coordinate is chosen as the one with the largest difference \( \max_{(k,t)} |\Delta z_k[t]| \) between its current value \( z_k[t] \) and the value \( z'_k[t] \) with

\[
\Delta z_k[t] = z_k[t] - z'_k[t].
\]

The updates are run until the \( \max_{k,t} |\Delta z_k[t]| \) become smaller than a specified tolerance parameter \( \epsilon \). We studied this update scheme as it aims to get the largest gain from each updates. Other coordinate update strategies were proposed such as cyclic updates (Friedman et al., 2007) or random updates (Shalev-Shwartz & Tewari, 2009) and our algorithm can easily be implemented with such update scheme. In this study, we focus on the greedy approach as it aims to get the largest gain from each updates. Moreover, as the updates in the greedy scheme are more complex to compute, distributing them provides a larger speedup compare to other strategy. We refer the reader to the work by Nutini et al. (2015) which discussed extensively the difference between these schemes.

A closed form solution exists to compute the optimal value \( z'_{k_0}[t_0] \) of \( e_{k_0,t_0} \)

\[
z'_{k_0}[t_0] = \frac{1}{\| D_{k_0} \|_2^2} \text{Sh}(\beta_{k_0}[t_0], \lambda) = \arg \max_{u \in \mathbb{R}} e_{k_0,t_0}(u),
\]

with \( \beta_k[t] = \left( \tilde{D}_k \ast \left( X - \sum_{k' \neq k} z_{k'} \ast D_{k'} - \Phi_1 (z_k) \ast D_k \right) \right)[t] \) and \( \text{Sh} \) the soft thresholding operator \( \text{Sh}(u, \lambda) = \text{sign}(u) \max(|u| - \lambda, 0) \). The success of this algorithm highly depends on the efficiency to compute the coordinate update. For problem (2), Kavukcuoglu et al. (2010) show that an update of the coefficient \((k_0, t_0)\) to the value \( z'_{k_0}[t_0] \) modifies \( \beta \) such that then it is possible to compute \( \beta^{(\text{new})} \) from \( \beta^{(\text{old})} \) using

\[
\beta_k^{(\text{new})}[t] = \beta_k^{(\text{old})}[t] - S_{k,k_0} [t - t_0] \Delta z_k^{(i)}[t_0], \quad \forall (k, t) \neq (k_0, t_0)
\]

with \( S_{k,i} [t] = (\tilde{D}_k \ast D_i)[t] \). For all \( t \notin [-W + 1..W - 1] \), \( S[t] \) is zero. Thus, only \( \mathcal{O}(KW) \) operations are needed to maintain \( \beta \) up to date with the current estimate \( Z \). Finally, the complexity of an iteration of CD is dominated by the \( \mathcal{O}(KT) \) operations needed to find the maximum of \( |\Delta z_k[t]| \).
2.2 Distributed Convolutional Coordinate Descent (DICOD)

**DICOD.** Algorithm 2 describes the steps of DICOD with $M$ workers. Each worker $m \in \{1..M\}$ is in charge of updating the coefficients of a segment $C_m$ of size $L_M = L/M$ defined by:

$$ C_m = \{(k,t) : k \in [1..K], \ t \in ((m-1)L_M, mL_M-1]\}. $$

The local updates are performed in parallel for all the cores using the greedy coordinate descent introduced in Section 2.1. When a core $m$ updates the coordinate $(k_0, t_0)$ such that $t_0 \in [(m-1)L_M + W, mL_M - W]$, the coefficients of $\beta$ that are updated are all contained in $C_m$ and there is no need to update $\beta$ on all the other cores. In these cases, the update is equivalent to a sequential update. When $t_0 \in [mL_M - W, mL_M]$ (resp. $t_0 \in [(m-1)L_M, (m-1)L_M + W]$), some of the coefficients of $\beta$ in core $m+1$ (resp. $m-1$) needs to be updated and the update is not local anymore. This can be done by sending the position of updated coordinate $(k_0, t_0)$, and the value of the update $\Delta \beta_{k_0}[t_0]$ to the neighboring core. Figure 1 illustrates this communication process. The inter-processes communication is very limited in DICOD. One node only communicates with its neighbors when it updates coefficients close to the extremity of its segment. When the size of the segment is reasonably large compared to the size of the patterns, only a small part of the iterations needs to send messages. We cannot apply the stopping criterion of CD in each worker of DICOD, as this criterion might not be reached globally. The updates in the neighbor cores can break this criterion. To avoid this issue, the convergence is considered to be reached once all the cores achieve this criterion simultaneously. Workers that reach this state locally are paused, waiting for incoming communication or for the global convergence to be reached.

The key point that allows to distribute the convolutional coordinate descent algorithm is that the solutions on time segments that are not overlapping are only weakly dependent. Equation (6) shows that a local change has impact on a segment of length $2W - 1$ centered around the updated coordinate. Thus, if two far enough coordinates are updated simultaneously, the resulting point $z$ is the same as if these two coordinates had been updated sequentially. By splitting the signal into continuous segment over multiple cores, coordinates can be updated independently on each core up to certain limits.

**Interferences.** When two coefficients $(k_0, t_0)$ and $(k_1, t_1)$ are updated by two neighboring cores before receiving the communications of the other update, the updates might not be independent and cannot be considered sequential. The local version of $\beta$ used for the second update does not account for the first update. We say that the updates are interfering. The cost reduction resulting from these two updates is denoted $\Delta E_{k_0,k_1}[t_0,t_1]$. Simple computations, detailed in Proposition A.2, show that

$$ \Delta E_{k_0,k_1}[t_0,t_1] = \Delta E_{k_0}[t_0] + \Delta E_{k_1}[t_1] - S_{k_0,k_1}[t_1 - t_0] \Delta z_{k_0}[t_0] \Delta z_{k_1}[t_1], \quad (7) $$

If $|t_1 - t_0| \geq W$, then $S_{k_0,k_1}[t_1 - t_0] = 0$ and the updates can be considered as sequential as the interference term is zero. When $|t_1 - t_0| < W$, the interference term does not vanish but Section 3 shows that under mild assumption, this term is controlled and does not break the convergence of DICOD.
Existing distributed coordinate descent algorithms. This algorithm differs from the existing paradigm to distribute CD as it does not rely on centralized communication. Indeed, other algorithms rely on a parameter server, which is an extra worker that holds the current value of $Z$. As the size of the problem and the number of nodes grow, the communication cost can rapidly become an issue with this kind of centralized communication. The natural splitting proposed with DICOD allows for more efficient interactions between the workers and reduces the need for inter-node communications. Moreover, to prevent the interferences breaking the convergence, existing algorithms rely either on synchronous updates (Bradley et al., 2011; Yu et al., 2012) or on reduced the step size in the updates You et al. (2016); Scherrer et al. (2012a). In both case, they are less efficient than our asynchronous greedy algorithm that can leverage both the large updates and the independence of the processes, without external parameters.

3 Properties of DICOD

Convergence of DICOD. The interference magnitude is related to the value of the cross-correlation between dictionary elements, as shown in Proposition 1. Thus, when the interferences have low probability and small magnitude, the distributed algorithm behaves as if the updates were applied sequentially, resulting in a large acceleration compared to the sequential CD algorithm.

Proposition 1. For concurrent updates for coefficients $(k_0, t_0)$ and $(k_1, t_1)$ of a sparse code $Z$, the cost update $\Delta E_{k_0k_1}[t_0, t_1]$ is lower bounded by

$$\Delta E_{k_0k_1}[t_0, t_1] \geq \Delta E_{k_0}[t_0] + \Delta E_{k_1}[t_1] - 2 \frac{S_{k_0,k_1}[t_0 - t_1]}{\|D_{k_0}\|_2\|D_{k_1}\|_2^2} \sqrt{\Delta E_{k_0}[t_0]\Delta E_{k_1}[t_1]}.$$  \hspace{1cm} (8)

The proof of this property is given in supplementary materials. It relies on the $||D_k||_2$-strong convexity of (5), which gives $|\Delta z_k[t]| \leq \sqrt{2\Delta E_k[t]/||Z||}$ for all $Z$. Using this inequality with (7) yields the expected result.

This property controls the interference magnitude using the cost reduction associated to a single update. When the correlations between the different elements of the dictionary are small enough, the interfering update does not increase the cost function. The updates are less efficient but do not degrade the current estimate. Using this control on the interferences, we can prove the convergence of DICOD.

Theorem 2. If the following hypotheses hold

H1. For all $(k_0,t_0),(k_1,t_1)$ such that $t_0 \neq t_1$, $\frac{|S_{k_0,k_1}[t_0 - t_1]|}{\|D_{k_0}\|_2\|D_{k_1}\|_2^2} < 1$.

H2. There exists $A \in \mathbb{N}^*$ such that all cores $m \in [1..M]$ are updated at least once between iteration $i$ and $i + A$ if the solution is not locally optimal.

H3. The delay in communication between the processes is inferior to the update time.

Then, the DICOD algorithm converges to the optimal solution $Z^*$ of (2).

Assumption H1 is satisfied as long as the dictionary elements are not replicated in shifted positions in the dictionary. It permits to make sure that at each step, the cost is updated in the right direction. This assumption can be linked to the shifted mutual coherence introduced in Papyan et al. (2016).

Hypothesis H2 ensures that all coefficients are updated regularly if they are not already optimal. This analysis is not valid when one of the core fails. As only one core is responsible for the update of a local segment, if a worker fails, this segment cannot be updated anymore and thus the algorithm will not converge to the optimal solution.

Finally, under H3, an interference only results from one update on each core. Multiple interferences occur when a core updates multiple coefficients in the border of its segment before receiving the communication from other processes border updates. When $T \gg W$, the probability of multiple interference is low and this hypothesis can be relaxed if the updates are not concentrated on the borders.
Proof sketch for Theorem 2. The full proof can be found in Appendix C. The key point in proving the convergence is to show that most of the updates can be considered sequentially and that the remaining updates do not increase the cost of the current point. By H3, for a given iteration, a core can interfere with at most one other core. Thus, without loss of generality, we can consider that at each step \( i \), the variation of the cost \( E \) is either \( \Delta E_{k_0}[t_0](Z^{(i)}) \) or \( \Delta E_{k_1}[t_1](Z^{(i)}) \), for some \( (k_0, t_0), (k_1, t_1) \in [1..K] \times [0..T − 1] \). Proposition 1 and H1 proves that \( \Delta E_{k_0}[t_0](Z^{(i)}) \geq 0 \). For a single update \( \Delta E_{k_0}[t_0](Z^{(i)}) \), the update is equivalent to a sequential update in CD, with the coordinate chosen randomly between the best in each segments. Thus, \( \Delta E_{k_0}[t_0](Z^{(i)}) > 0 \) and the convergence is eventually proved using results from Osher & Li (2009).

Speedup of DICOD. We denote \( S_{dicod}(M) \) the speedup of DICOD compared to the sequential CD. This quantify the number of iteration that can be run by DICOD during one iteration of CD.

**Theorem 3.** Let \( \alpha = \frac{1}{N} \) and \( M \in \mathbb{N}^* \). If \( \alpha M < \frac{1}{2} \) and if the non zero coefficients of the sparse code are distributed uniformly in time, the expected speedup \( \mathbb{E}[S_{dicod}(M)] \) is lower bounded by

\[
\mathbb{E}[S_{dicod}(M)] \geq M^2(1 - 2\alpha^2M^2 (1 + 2\alpha^2M^2)^{\frac{1}{2}}) .
\]

This result can be simplified when the interference probability \( \alpha M \) is small.

**Corollary 4.** The expected speedup \( \mathbb{E}[S_{dicod}(M)] \) when \( (\alpha M)^2 \to 0 \) is such that

\[
\mathbb{E}[S_{dicod}(M)] \gtrsim \frac{M^2(1 - 2\alpha^2M^2 + O(\alpha^4M^4))}{\alpha} .
\]

Proof sketch for Theorem 3. The full proof can be found in Appendix D. There are two aspects involved in DICOD speedup: the computational complexity and the acceleration due to the parallel updates. As stated in Section 2.1, the complexity of each iteration for CD is linear with the length of the input signal \( T \). In DICOD, each core runs on a segment of size \( \frac{T}{M} \), this speedup the execution of individual updates by a factor \( M \). Moreover, all the cores compute their update simultaneously. The updates without interference are equivalent to sequential updates. Interfering updates happens with probability \( (\alpha M)^2 \) and do not degrade the cost. Thus, one iteration of DICOD with \( N_i \) interference provides a cost variation equivalent to \( M - 2N_i \) iterations using sequential CD and, in expectation, it is equivalent to \( M - 2\mathbb{E}[N_i] \) iterations of DICOD. The probability of interference depends on the ratio between the length of the segments used for each cores and the size of the dictionary. If all the updates are spread uniformly on each segment, the probability of interference between 2 neighboring cores is \( \left( \frac{MN_i}{T} \right)^2 \). \( \mathbb{E}[N_i] \) can be upper bounded using this probability and this yields the desired result.

The overall speedup of DICOD is super-linear for the regime where \( (\alpha M)^2 \ll 1 \). It is almost quadratic for small \( M \) but as \( M \) grows, there is a sharp transition that significantly deteriorate the acceleration provided by DICOD. Section 4 empirically highlights this behavior. For a given \( \alpha \), it is possible to approximate the optimal number of cores \( M \) to solve convolutional sparse coding problems.

This super-linear speedup makes possible to have an improvement of performance by running DICOD sequentially. This algorithm, called SeqDICOD, randomly selects at each step one of the \( M \) contiguous sub-signals indexed by \( C_m \) extracted from \( x \) and performs a greedy coordinate update.

The number \( M \) of sub-signals considered balances the computational complexity of the iterations and the convergence rate. As \( M \) grows, the computational complexity reduces but the algorithm gets closer to randomized CD and thus the convergence rate is reduced.

### 4 Numerical Results

All the numerical experiments are run on five Linux machines with 16 to 24 Intel Xeon 2.70 GHz processors and at least 64 GB of RAM on local network. We use a combination of Python, C++ and the OpenMPI 1.6 for the algorithms implementation. The code to reproduce the figures is available online \(^1\). The runtime denotes the time for the system to run the full algorithm pipeline, from cold

\(^1\) see the supplementary materials
start and includes for instance the time to start the subprocesses. The convergence refers to the variation of the cost with the number of iterations and the speed to the variation of the cost relative to time.

Artificial signals. To further validate our algorithm, we generate signals and test the performances of DICOD compared to state-of-the-art methods proposed to solve the convolutional sparse coding. We generate a signal $X$ of length $T$ in $\mathbb{R}^P$ following the model described in (1). The $K$ dictionary atoms $D_k$ of length $W$ are drawn as generic dictionary. First, each entry is sampled from a gaussian distribution. Then, the pattern is normalized such that $\|D_k\|_2 = 1$. The sparse code entries are drawn from a Bernoulli-Gaussian distribution with Bernoulli parameter $\rho = 0.007$, mean 0 and standard variation $\sigma = 10$. The noise term $\mathcal{E}$ is chosen as a gaussian white noise with variance 1. The default values for the dimensions are set to $W = 200$, $K = 40$, $P = 7$, $T = 400 \cdot W$ and we used $\lambda = 0.1$.

Comparison of algorithms. DICOD is compared to the main state-of-the-art optimization algorithms for convolutional sparse coding: Fast Convolutional Sparse Coding (FCSC) from Bristow et al. (2013), Fast Iterative Soft Thresholding Algorithm (FISTA) using Fourier domain computation as described in Wohlberg (2016) and the convolutional coordinate descent (CD) (Kavukcuoglu et al., 2010). All the specific parameters for these algorithms are fixed based on the authors recommendations. DICOD$_M$ denotes the DICOD algorithm run using $M$ cores. We also include SeqDICOD$_{60}$, a sequential run of the DICOD algorithm using 60 segments.

Figure 2 presents the evolution of the cost function value relatively to the number of iterations and the time. To ensure reasonable computation, a timeout is set to 3600 seconds and it was only reached by the CD algorithm. The evolution of the performances of DICOD relative to the iteration are very close to the performances of CD, for both 30 and 60 cores. Also, SeqDICOD displays the same behaviors. The difference between the curve for SeqDICOD$_{60}$ and CD are only due to the updates of SeqDICOD being locally greedy, compared to the global greedy updates used in CD. The variations between the curve of DICOD$_{30}$, DICOD$_{15}$ and SeqDICOD$_{60}$ are due to the interfering steps that slows down the convergence. Thus, it requires more steps to reach the same accuracy with DICOD$_{60}$ than with CD. However, the number of extra steps required is quite low compared to the total number of steps and the performances are not highly degraded. As the probability of interference increases with the number of cores used, DICOD$_{60}$ needs to perform more iterations to obtain the same performance compare to DICOD$_{30}$. Another observation is that the performances of the global methods FCSC and Fista are much better than the methods based on local updates iteration wise. As each iteration can update all the coefficients for Fista, the number of iterations needed to reach the optimal solution is indeed smaller than for CD, as only one coordinate is updated at a time. The left curve displays the speed of the algorithms for a convolutional sparse coding problem with generated signals. The DICOD algorithm is faster compared to the state of the art algorithms and the speed up increases with the number of cores. DICOD has a shorter initialization period compared to the other algorithms. This shows that the overhead of starting the cores is balanced by the reduction of the computational burden for long signal. For shorter signals, we have observed that the initialization takes the same order of time as the other methods as the overhead of starting
the cores induces a constant initialization time. As the signal length grows, the computational cost to initialize $\beta$ increases and this overhead becomes small compared to the other non parallelized initializations. Note that FCSC suffers from a very long initialization as it requires pre-computing many constants but its performances match the one of FISTA close to convergence.

**Speedup evaluation.** The left part of Figure 3 displays the running time of each algorithm for different problem sizes, averaged over 10 repetitions. All methods were considered to have converged when the $\ell_\infty$-norm of the updates reached a certain threshold $\epsilon = 5e^{-2}$. This figure highlights the speedup obtained with the parallelization. The speed up ratio between DICOD$_{15}$ and DICOD$_{30}$ is on average of 4.99, and the ratio between DICOD$_{15}$ and CD is on average of 91.59. This corroborates the complexity analysis provided in Corollary 4 which states that the speedup obtained between DICOD and CD is quadratic in $M^2$. This plot also shows that DICOD is capable of solving problems with large size ($T = 400000$ samples) in around 3 minutes, whereas the running time of the other algorithms is above two hours. The right part of Figure 3 displays the speedup of DICOD as a function of the number of cores. We used 10 generated problems for 2 signal length $T = 150 \cdot W$ and $T = 750 \cdot W$ and we solved them using DICOD$_M$ with a number of core $M$ ranging from 1 to 75. The blue dots display the average running time for a given number of workers. For both setups, the speedup is superlinear up to the point where $M^\alpha \approx \frac{1}{2}$. For small $M$ the speedup is very close to quadratic and a sharp transition occurs as the number of cores grows. The vertical solid green line indicates the approximate position of the maximal speedup given in Corollary 4 and the dashed line is the expected theoretical runtime derived from the same expression. The transition after the maximum is very sharp. This approximation of the speedup for small values of $M^\alpha$ is close to the experimental speedup observed with DICOD. The computed optimal value of $M^*$ is close to the optimal number of cores in these two examples.

### 5 Conclusion

In this work, we introduced an asynchronous distributed algorithm that is able to speed up the resolution of the Convolutional Sparse Coding problem for long signals. This algorithm is guaranteed to converge to the optimal solution of (2) and scales superlinearly with the number of cores used to distribute it. These claims are supported by numerical experiments highlighting the performances of DICOD compared to other state-of-the-art methods. Our proofs rely extensively on the use of one dimensional convolutions. In this setting, a process $m$ only has two neighbors $m-1$ and $m+1$. This ensures that there is no high order interferences between the updates. Our analysis does not apply to distributed computation using square patches of images as the interferences are more complicated. A way to apply our algorithm with these guarantees to images is to only split the signals along one direction, to avoid higher order interferences. The extension of our results to this case is an interesting direction for future work.
References

Adler, A., Elad, M., Hel-Or, Y., and Rivlin, E. Sparse Coding with Anomaly Detection. In proceedings of IEEE International Workshop on Machine Learning for Signal Processing (MLSP), pp. 22 – 25, 2013.

Bradley, Joseph K., Kyrola, Aapo, Bickson, Danny, and Guestrin, Carlos. Parallel Coordinate Descent for L1-Regularized Loss Minimization. In proceedings of the International Conference on Machine Learning (ICML), pp. 321–328, 2011.

Bristow, Hilton, Eriksson, Anders, and Lucey, Simon. Fast convolutional sparse coding. In proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 391–398, 2013.

Chalasani, Rakesh, Principe, Jose C., and Ramakrishnan, Naveen. A fast proximal method for convolutional sparse coding. In proceedings of the International Joint Conference on Neural Networks (IJCNN), 2013.

Friedman, Jerome, Hastie, Trevor, Hölting, Holger, and Tibshirani, Robert. Pathwise coordinate optimization. The Annals of Applied Statistics, 1:302–332, 2007.

Grosse, Roger, Raina, Rajat, Kwong, Helen, Program, Symbolic Systems, and Ng, Andrew Y. Shift-Invariant Sparse Coding for Audio Classification. Cortex, 8:9, 2007.

Kavukcuoglu, Koray, Sermanet, Pierre, Boureau, Y-lan, Gregor, Karol, and Lecun, Yann. Learning Convolutional Feature Hierarchies for Visual Recognition. In proceedings of Advances in Neural Information Processing Systems (NIPS), pp. 1–9, 2010.

Mairal, Julien, Bach, Francis, Ponce, Jean, and Sapiro, Guillermo. Online Learning for Matrix Factorization and Sparse Coding. Journal of Machine Learning Research (JMLR), 11:19–60, 2009.

Mallat, Stéphane. A Wavelet Tour of Signal Processing. 2009. ISBN 9780123743701.

Nutini, Julie, Schmidt, Mark, Laradji, Issam H, Friedlander, Michael, and Koepke, Hoyt. Coordinate Descent Converges Faster with the Gauss-Southwell Rule Than Random Selection. In proceedings of the International Conference on Machine Learning (ICML), pp. 1–34, 2015.

Osher, Stanley and Li, Yingying. Coordinate descent optimization for 11 minimization with application to compressed sensing: a greedy algorithm. Inverse Problems and Imaging, 3:487–503, 2009.

Papyan, Vardan, Sulam, Jeremias, and Elad, Michael. Working Locally Thinking Globally - Part I: Theoretical Guarantees for Convolutional Sparse Coding. arXiv preprint, 1607(02005v1):1–13, 2016.

Scherrer, Chad, Halappanavar, Mahantesh, Tewari, Ambuj, and Haglin, David. Scaling Up Coordinate Descent Algorithms for Large 1 Regularization Problems. Technical report, 2012a.

Scherrer, Chad, Tewari, Ambuj, Halappanavar, Mahantesh, and Haglin, David J. Feature Clustering for Accelerating Parallel Coordinate Descent. In proceedings of Advances in Neural Information Processing Systems (NIPS), pp. 1–9, 2012b.

Shalev-Shwartz, Shai and Tewari, A. Stochastic Methods for ℓ1-regularized Loss Minimization. In proceedings of the International Conference on Machine Learning (ICML), pp. 929–936, 2009.

Wohlberg, Brendt. Efficient Algorithms for Convolutional Sparse Representations. IEEE Transactions on Image Processing, 25(1), 2016.

You, Yang, Lian, Xiangru, Liu, Ji, Yu, Hsiang-Fu, Dhillon, Inderjit S, Demmel, James, and Hsieh, Cho-Jui. Asynchronous parallel greedy coordinate descent. In proceedings of Advances in Neural Information Processing Systems (NIPS), pp. 4682–4690, 2016.

Yu, Hsiang Fu, Hsieh, Cho Jui, Si, Si, and Dhillon, Inderjit. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In proceedings of the IEEE International Conference on Data Mining (ICDM), pp. 765–774, 2012.
A Computation for the cost updates

When a coefficient $z_k[t]$ is updated to $u$, the cost update is a simple function of $z_k[t]$ and $u$.

**Proposition A.1.** The update of the weight in $(k_0, t_0)$ from $z_{k_0}[t_0]$ to $u \in \mathbb{R}$ gives a cost variation:

$$e_{k_0, t_0}(u) = \frac{\|D_{k_0}\|^2}{2} (z_{k_0}[t_0]^2 - u^2) - \beta_{k_0} [t_0] (z_{k_0}[t_0] - u) + \lambda (|z_{k_0}[t_0]| - |u|).$$

**Proof.** Let $x_{k_0}[t] = (X - \sum_{k=1}^K z_k \ast D_k)[t] + D_{k_0}[t - t_0]z_{k_0}[t_0]$ for all $t \in [0..T - 1]$ and $z_k^{(1)}[t] = \begin{cases} u, & \text{if } (k, t) = (k_0, t_0) \\ z_k[t], & \text{elsewhere} \end{cases}.$

$$e_{k_0, t_0}(u) = \frac{1}{2} \sum_{t=0}^{T-1} \left( X - \sum_{k=1}^K z_k \ast D_k \right)[t] + \lambda \sum_{k=1}^K ||z_k||_1 - \frac{1}{2} \sum_{t=0}^{T-1} \left( X - \sum_{k=1}^K z_k^{(1)} \ast D_k \right)[t] + \lambda \sum_{k=1}^K ||z_k^{(1)}||_1$$

$$= \frac{1}{2} \sum_{t=0}^{T-1} \left( \alpha_{k_0}[t] - D_{k_0}[t - t_0]z_{k_0}[t_0] \right)^2 - \frac{1}{2} \sum_{t=0}^{T-1} \left( \alpha_{k_0}[t] - D_{k_0}[t - t_0]u \right)^2 + \lambda (|z_{k_0}[t_0]| - |u|)$$

$$= \frac{1}{2} \sum_{t=0}^{T-1} D_{k_0}[t - t_0]^2 (z_{k_0}[t_0]^2 - u^2) - \sum_{t=0}^{T-1} \alpha_{k_0}[t] D_{k_0}[t - t_0] (z_{k_0}[t_0] - u) + \lambda (|z_{k_0}[t_0]| - |u|)$$

$$= \frac{\|D_{k_0}\|^2}{2} (z_{k_0}[t_0]^2 - u^2) - \left( \overbrace{\sum_{k=1}^K \alpha_{k_0}[t] \ast D_{k_0}}^{\beta_{k_0}[t_0]} \right) (z_{k_0}[t_0] - u) + \lambda (|z_{k_0}[t_0]| - |u|)$$

This conclude our proof. \(\square\)

Using this result, we can derive the optimal value $z'_{k_0}[t_0]$ to update the coefficient $(k_0, t_0)$ as the solution of the following optimization problem:

$$z'_{k_0}[t_0] = \arg \min_{z_{k_0}[t_0]} e_{k_0, t_0}(u) = \arg \min_{u \in \mathbb{R}} \frac{\|D_{k_0}\|^2}{2} (u - \beta_{k_0}[t_0])^2 + \lambda |u|. \quad (9)$$

In the case where two coefficients $(k_0, t_0), (k_1, t_1)$ are updated in the same iteration to values $u$ and $z'_{k_1}[t_1]$, we obtain the following cost variation.

**Proposition A.2.** The update of the weight $z_{k_0}[t_0]$ and $z_{k_1}[t_1]$ to values $z'_{k_0}[t_0]$ and $z'_{k_1}[t_1]$ with $
\Delta z_k[t] = z_k[t] - z'_{k_1}[t_1]$ gives an update of the cost:

$$\Delta E_{k_0, k_1}[t_0, t_1] = \Delta E_{k_0}[t_0] + \Delta E_{k_1}[t_1] - S_{k_0, k_1}[t_0 - t_1] \Delta z_{k_0}[t_0] \Delta z_{k_1}[t_1]$$

**Proof.** We define $z_k^{(1)}[t] = \begin{cases} z_{k_0}[t_0], & \text{if } (k, t) = (k_0, t_0) \\ z_{k_1}[t_1], & \text{if } (k, t) = (k_1, t_1) \\ z_k[t], & \text{otherwise} \end{cases}$

Let $\alpha[t] = (X - \sum_{k=1}^K z_k \ast D_k)[t] + D_{k_0}[t - t_0]z_{k_0}[t_0] + D_{k_1}[t - t_1]z_{k_1}[t_1]$. 

10
We have \( \alpha[t] = \alpha_{k_0}[t] + D_{k_1}[t - t_1]z_{k_1}[t_1] = \alpha_{k_1}[t] + D_{k_0}[t - t_0]z_{k_0}[t_0] \).

\[
\Delta E_{k_0 k_1}[t_0, t_1] = \frac{1}{2} \sum_{t=0}^{T-1} \left( X - \sum_{k=1}^{K} z_k * D_k \right)[t]^2 + \frac{1}{2} \sum_{k=1}^{K} \lambda \| z_k \|_1 - \sum_{t=0}^{T-1} \left( X - \sum_{k=1}^{K} z_k^{(1)} * D_k \right)[t] + \lambda \sum_{k=1}^{K} \| z_k^{(1)} \|_1
\]

\[
= \frac{1}{2} \sum_{t=0}^{T-1} (\alpha[t] - D_{k_1}[t - t_1]z_{k_1}[t_1])^2 + \lambda |z_{k_0}[t_0] - |z_{k_0}[t_0]\rangle
\]

\[
= \frac{1}{2} \sum_{t=0}^{T-1} \left[ D_{k_0}[t - t_0]^2(z_{k_0}[t_0]^2 - z_{k_0}[t_0]^2) + D_{k_1}[t - t_1]^2(z_{k_1}[t_1]^2 - z_{k_1}[t_1]^2) \right]
\]

\[
= \sum_{t=0}^{T-1} \left[ \alpha_{k_0}[t] D_{k_0}[t - t_0] \Delta z_{k_0}[t_0] + \alpha_{k_1}[t] D_{k_1}[t - t_1] \Delta z_{k_1}[t_1]
\]

\[
= \Delta E_{k_0}[t_0] + \Delta E_{k_1}[t_1]
\]

By definition of \( S_{k_0, k_1}[t] = \overline{D}_{k_0} * D_{k_1}[t] \). This conclude our proof.

\[ \Box \]

### B Intermediate results

Consider solving a convex problem of the form:

\[
\min E(Z) = F(Z) + \sum_{t=0}^{L-1} \sum_{k=1}^{K} g_k(z_k[t])
\]

where \( F \) is differentiable and convex, and \( g_k \) is convex. Let us first recall a theorem stated and proved in Osher & Li (2009).

**Theorem B.1.** Suppose \( F(z) \) is smooth and convex, with \( \left| \frac{\partial^2 F}{\partial z_i \partial z_j} \right|_\infty \leq M \), and \( E \) is strictly convex with respect to any one variable \( z_i \), then the statement that \( u = (u_1, u_2, \ldots, u_n) \) is an optimal solution of (10) is equivalent to the statement that every component \( u_i \) is an optimal solution of \( E \) with respect to the variable \( u_i \) for any \( i \).

In the convolutional sparse coding problem, the function \( F(Z) = \frac{1}{2} \| X - \sum_{k=1}^{K} z_k * D_k \| \) is smooth and convex and its Hessian is constant. The following Lemme B.2, can be used to show that the function \( E \) restricted to one of its variable is strictly convex and thus satisfies the condition of Theorem B.1.

**Lemme B.2.** The function \( f : \mathbb{R} \rightarrow \mathbb{R} \) defined for \( \alpha, \lambda > 0 \) and \( b \in \mathbb{R} \) by \( f(x) = \frac{\alpha}{2}(x - b)^2 + \lambda |x| \) is \( \alpha \)-strongly convex.

**Proof.** The property of monotone subdifferential states that a function \( f \) is \( \alpha \)-strongly convex if and only if

\[
\forall (x, x'), \quad (f(x) - f(x'), x - x') \geq \alpha \|x - x'\|_2^2
\]
Let us define the subdifferential of $f$:
\[
\partial f = \begin{cases} 
\alpha(x - b) + \lambda \text{sign}(x) & \text{if } x \neq 0 \\
-\alpha b + \lambda t, \text{ for } t \in [-1, 1] & \text{if } x = 0
\end{cases}
\]
The inequality is an equality for $x = x'$.
If $x' = 0$, we get for $|t| \leq 1$:
\[
\langle \alpha(x - b) + \lambda \text{sign}(x) + \alpha b - \lambda t, x \rangle = \alpha x^2 + \lambda (|x| - tx) \geq \alpha x^2 = \alpha(x - x')^2
\]
If $x' \neq 0$, we get:
\[
\langle \alpha(x - x') + \lambda(\text{sign}(x) - \text{sign}(x')), x - x' \rangle = \alpha(x - x')^2 + \lambda(|x| - \text{sign}(x)x' - \text{sign}(x')x) = \alpha(x - x')^2 + \lambda(1 - \text{sign}(x)\text{sign}(x'))(|x| + |x'|)
\]
\[
\geq \alpha(x - x')^2
\]
Thus $f$ is \(\alpha\)-strongly convex.

This can be applied to the function $e_k, t$ defined in (9), showing that the problem in one coordinate \((k, t)\) is \(\|D_k\|_2^2\)-strongly convex.

## C Proof of convergence for DICOD (Theorem 2)

We define
\[
C_{k_0, k_1}[t] = \frac{S_{k_0, k_1}[t]}{\|D_{k_0}\|_2 \|D_{k_1}\|_2}
\]

Let us first show how $C_{k_0, k_1}$ controls the interfering cost update.

**Proposition 1.** In case of concurrent update for coefficients \((k_0, t_0)\) and \((k_1, t_1)\), the cost update $\Delta E_{k_0, k_1}[t_0, t_1]$ is bounded as
\[
\Delta E_{k_0, k_1}[t_0, t_1] \geq \Delta E_{k_0}[t_0] + \Delta E_{k_1}[t_1] - 2C_{k_0, k_1}[t_0 - t_1] \sqrt{\Delta E_{k_0}[t_0] \Delta E_{k_1}[t_1]}.
\]

**Proof.** The problem in one coordinate \((k, t)\) given all the other can be reduced (9). Simple computations show that:
\[
\Delta E_k[t] = e_{k, t}(z_k[t]) - e_{k, t}(z_k'[t]).
\]
We have shown in Lemme B.2 that $e_{k, t}$ is $\|D_k\|_2^2$-Strong convex. Thus by definition of the strong convexity, and using the fact that $z_k'[t]$ is optimal for $e_{k, t}$
\[
|e_{k, t}(z_k[t]) - e_{k, t}(z_k'[t])| \geq \frac{\|D_k\|_2^2}{2} (z_k[t] - z_k'[t])^2
\]
i.e., $|\Delta z_k[t]| \leq \frac{\sqrt{\Delta E_k[t]}}{\|D_k\|_2^2}$, and the result is obtained using this inequality with Proposition A.2.

**Theorem 2.** If the following hypothesis are verified

**H1.** For all \((k_0, t_0)\), \((k_1, t_1)\) such that $t_0 \neq t_1$,
\[
|C_{k_0, k_1}[t_0 - t_1]| < 1.
\]

**H2.** There exists $A \in \mathbb{N}^*$ such that all cores $m \in [M]$ are updated at least once between iteration $q$ and $q + A$ if the solution is not locally optimal, i.e. $\Delta z_k[t] = 0$ for all \((k, t) \in C_m$.

**H3.** The delay in communication between the processes is inferior to the update time.

Then, the DICOD algorithm using the greedy updates $(k_0, t_0) = \arg \max_{(k, t) \in C_m} |\Delta z_k[t]|$ converges to the optimal solution $z^*$ of (2).
Proof. If several updates \((k_0, t_0), (k_1, t_1), \ldots (k_m, t_m)\) are updated in parallel without interference, then the update is equivalent to the sequential updates of each \((k_q, t_q)\). We thus consider that for each step \(i\), without loss of generality that

\[
\Delta E^{(i)} = \begin{cases} 
\Delta E_{k_0}^{(i)}[t_0], & \text{if there is no interference} \\
\Delta E_{k_0,k_1}^{(i)}[t_0, t_1], & \text{otherwise}
\end{cases}
\]

If \(\forall (k, t), \Delta z_k^{(i)}[t] = 0\), then \(z^{(i)}\) is coordinate wise optimal. Using the result from Theorem \(B.1\), \(z^{(i)}\) is optimal. Thus if \(z^{(i)}\) is not optimal, \(\Delta E_{k_0}^{(i)}[t_0] > 0\).

Using Proposition 1 and \(H1\)

\[
\Delta E_{k_0,k_1}^{(i)}[t_0, t_1] = \left( \sqrt{\Delta E_{k_0}^{(i)}[t_0]} - \sqrt{\Delta E_{k_1}^{(i)}[t_1]} \right)^2 \geq 0,
\]

so the update \(\Delta E^{(i)}\) is positive.

The sequence \((E(z^{(i)}))_n\) is decreasing and bounded by 0. It converges to \(E^*\) and \(\Delta E^{(i)} \xrightarrow{n \to \infty} 0\).

As \(\lim_{\|x\| \to \infty} E(z) = +\infty\), there exist \(M \geq 0, i_0 \geq 0\) such that \(\|z^{(i)}\|_\infty \leq M\) for all \(i > i_0\). Thus, there exist a subsequence \((z^{(i)})_q\) such that \(z^{(i)} \xrightarrow{q \to \infty} \bar{z}\). By continuity of \(E, E^* = E(\bar{z})\)

Then, we show that \(z^{(i)}\) converges to a point \(\bar{z}\) such that each coordinate is optimal for the one coordinate problem. By Proposition \(1\), the sequence \((z^{(i)})_i\) is \(\ell_\infty\)-bounded. It admits at least a limit point \(z^{(i)} \xrightarrow{q \to \infty} \bar{z}\). Moreover, the sequence \(z^{(i)}\) is a Cauchy sequence for the norm \(\ell_\infty\) as for \(p, q > 0\)

\[
\|z^{(p)} - z^{(q)}\|_\infty^2 \leq \frac{2}{\|D\|_{\ell_\infty,2}^2} \sum_{l > q} \Delta E^{(l)}
= \frac{2}{\|D\|_{\ell_\infty,2}^2} (E(z^q) - E^*) \xrightarrow{q \to \infty} 0
\]

Thus \(z^{(i)}\) converges to \(\bar{z}\).

Let \(m\) denote one of the \(M\) cores and \((k, t)\) be coordinates in \(C_m\). We consider the function \(h_{k,t} : \mathbb{R}^K \times L \to \mathbb{R}\) such that

\[
h(z) = z_k[t] = \frac{1}{\|D_k\|_2^2} \text{Sh}(\beta_k[t], \lambda).
\]

We recall that

\[
\beta_k[t](Z) = \left( \bar{D}_k \ast \left( X - \sum_{k' = 1, k' \neq k}^K z_{k'}[t] \ast D_{k'} - \Phi_t (z_k[t]) \ast D_k \right) \right)[t]
\]

The function \(\phi : Z \to \beta_k[t](Z)\) is linear. As \(\text{Sh}\) is continuous in its first coordinate and \(h(Z) = \text{Sh}(\phi(Z), \lambda)\), the function \(h_{k,t}\) is continuous. For \((k, t) \in C_m\), the gap between \(\bar{z}_k[t]\) and \(\bar{z}_k[t]\) is such that

\[
|\bar{z}_k[t] - \bar{z}_k[t]| = |\bar{z}_k[t] - h_{k,t}(\bar{z}_k[t])|
= \lim_{i \to \infty} |z_k^{(i)}[t] - h(\bar{z}_k^{(i)}[t])| \\
= \lim_{i \to \infty} |z_k^{(i)}[t] - y_k^{(i)}[t]| \quad (14)
\]

Using \(H2\), for all \(i \in \mathbb{N}\), if \(z_k^{(i)}[t]\) is not optimal, there exist \(q_i \in [i, i + A]\) such that the updated coefficient at iteration \(q_i\) is \((k_{q_i}, t_{q_i}) \in C_m\). As no update are done on coefficients between the
updates \( i \) and \( q_i \), \( z_k^{(i)}[t] = z_k^{(q_i)}[t] \). By definition of the update,

\[
|z_k^{(i)}[t] - y_k^{(i)}[t]| = |z_k^{(q_i)}[t] - y_k^{(q_i)}[t]|
\leq |z_k^{(q_i)}[t_{q_i}] - y_k^{(q_i)}[t_{q_i}]| \tag{greedy updates}
\]

Using this results with (14), \(|\tilde{y}_k[t] - \tilde{y}_k[t]| = 0\). This proves that \( \tilde{z} \) is optimal in each coordinate. By Theorem B.1, the limit point \( \tilde{z} \) is optimal for the problem (2).

\[\square\]

### D Proof of DICOD speedup (Theorem 3)

**Theorem 3.** Let \( \alpha = \frac{W}{T} \) and \( M \in \mathbb{N}^* \). If \( \alpha M < \frac{1}{4} \) and if the non zero coefficients of the sparse code are distributed uniformly in time, the expected speedup \( \mathbb{E}[S_{dicod}(M)] \) is lower bounded by

\[
\mathbb{E}[S_{dicod}(M)] \geq M^2(1 - 2\alpha^2M^2 \left(1 + 2\alpha^2M^2 \right)^{\frac{M}{2}} - 1).
\]

This result can be simplified when the interference probability \((\alpha M)^2\) is small.

**Corollary 4.** Under the same hypothesis, the expected speedup \( \mathbb{E}[S_{dicod}(M)] \) when \( (\alpha M)^2 \rightarrow 0 \) is

\[
\mathbb{E}[S_{dicod}(M)] \geq M^2(1 - 2\alpha^2M^2 + \mathcal{O}(\alpha^4M^4)).
\]

**Proof.** There are two aspects involved in DICOD speedup: the computational complexity and the acceleration due to the parallel updates.

As stated in Section 3, the complexity of each iteration for CD is linear with the length of the input signal \( T \). The dominant operation is the one that find the maximal coordinate. In DICOD, each core runs the same iterations on a segment of size \( \frac{T}{M} \). The hypothesis \( \alpha M < \frac{1}{4} \) ensures that the dominant operation is finding the maxima. Thus, when CD run one iteration, one core of DICOD can run \( M \) local iteration as the complexity of each iteration is divided by \( M \).

The other aspect of the acceleration is the parallel update of \( Z \). All the cores perform their update simultaneously and each update happening without interference can be considered as a sequential update. Interfering updates do not degrade the cost. Thus, one iteration of DICOD with \( N_i \) interference is equivalent to \( M - 2 \times N_{interf} \) iterations using CD and thus,

\[
\mathbb{E}[N_{dicod}] = M - 2 \times \mathbb{E}[N_{interf}] \tag{15}
\]

The probability of interference depends on the ratio between the length of the segments used for each core and the size of the dictionary. If all the updates are spread uniformly on each segment, the probability of interference between 2 neighboring cores is \((\frac{MW}{T})^2 = (\alpha M)^2\).

A process can only creates one interference with one of its neighbors. Thus, an upper bound on the probability to get exactly \( j \in [0, \frac{M}{2}] \) interferences is

\[
P(N_i = j) \leq \left( \frac{M}{2} \right)^j (2\alpha^2M^2)^j
\]

Using this result, we can upper bound the expected number of interferences for the algorithm

\[
\mathbb{E}[N_{interf}] = \sum_{j=1}^{\frac{M}{2}} jP(N_{interf} = j), \leq \sum_{j=1}^{\frac{M}{2}} j \left( \frac{M}{2} \right)^j (2\alpha^2M^2)^j, \leq \alpha^2M^3 \left(1 + 2\alpha^2M^2 \right)^{\frac{M}{2}} - 1.
\]
Pluggin this result in (15) gives us:

\[
\mathbb{E}[N_{\text{decod}}] \geq M(1 - 2\alpha^2 M^2 (1 + 2\alpha^2 M^2) \frac{M}{\theta} - 1),
\]


\[
\geq \alpha \rightarrow 0 \quad M(1 - 2\alpha^2 M^2 + O(\alpha^4 M^4)).
\]

Finally, by combining the two source of speedup, we obtain the desired result.

\[
\mathbb{E}[S_{\text{decod}}(M)] \geq M^2(1 - 2\alpha^2 M^2 (1 + 2\alpha^2 M^2) \frac{M}{\theta} - 1).
\]