Fast Matching Pursuit with Multi-Gabor Dictionaries

ZDENĚK PRŮŠA, NICKI HOLIGHAUS, and PETER BALAZS, Acoustics Research Institute, Austrian Academy of Sciences

Finding the best K-sparse approximation of a signal in a redundant dictionary is an NP-hard problem. Suboptimal greedy matching pursuit (MP) algorithms are generally used for this task. In this work, we present an acceleration technique and an implementation of the matching pursuit algorithm acting on a multi-Gabor dictionary, i.e., a concatenation of several Gabor-type time-frequency dictionaries, each of which consisting of translations and modulations of a possibly different window and time and frequency shift parameters. The technique is based on pre-computing and thresholding inner products between atoms and on updating the residual directly in the coefficient domain, i.e., without the round-trip to the signal domain. Since the proposed acceleration technique involves an approximate update step, we provide theoretical and experimental results illustrating the convergence of the resulting algorithm. The implementation is written in C (compatible with C99 and C++11) and we also provide Matlab and GNU Octave interfaces. For some settings, the implementation is up to 70 times faster than the standard Matching Pursuit Toolkit (MPTK).

CCS Concepts: • **Mathematics of computing** → **Solvers**; **Mathematical software performance**; *Nonconvex optimization*; Computation of transforms; • **Theory of computation** → **Numeric approximation algorithms**.

Additional Key Words and Phrases: greedy approximation, matching pursuit, time-frequency, short-time Fourier transform, Gabor dictionary

1 INTRODUCTION

The best K-sparse approximation of a signal $\mathbf{x} \in \mathbb{R}^L$ in an overcomplete dictionary of P normalized atoms (vectors) $\mathbf{D} = [\mathbf{d}_0|\mathbf{d}_1|\dots|\mathbf{d}_{P-1}] \in \mathbb{C}^{L \times P}$, $\|\mathbf{d}_p\|_2 = 1$ is an NP-hard problem [8]. Given the budget of K nonzero elements of the coefficient vector $\mathbf{c} \in \mathbb{C}^P$, the problem can be formally written as the minimization of the approximation error in the energy norm $\|\mathbf{x} - \mathbf{D}\mathbf{c}\|_2$ such that

$$\min\|\mathbf{x} - \mathbf{D}\mathbf{c}\|_2 \text{ subject to } \|\mathbf{c}\|_0 \leq K, \tag{1}$$

where the zero "norm" $\|\cdot\|_0$ returns the number of non-zero elements. A similar problem is the minimization of $\|\mathbf{c}\|_0$ given the approximation error tolerance E

$$\min\|\mathbf{c}\|_0 \text{ subject to } \|\mathbf{x} - \mathbf{D}\mathbf{c}\|_2 \leq E. \tag{2}$$

Both problems can be tackled by employing greedy *matching pursuit* (MP) algorithms. The only difference is the choice of the stopping criterion. However, greedy algorithms are known to be suboptimal in the sense that they are not guaranteed to choose the best combination of K atoms. Instead, an approximation rate i.e. the decrease of the approximation error with iterations has been studied. It has been shown that the basic version of MP [28] achieves an exponential approximation rate [8, 11, 18, 19]. To date, several variants of generic MP and its orthogonal version OMP [9, 29] were proposed e.g. complementary MP [33, 34], cyclic MP [41, 42], gradient pursuit [3, 4], local OMP [25, 26] and self projected MP [36]. In practice, without imposing any structure on the dictionary, the effectiveness of the algorithms quickly deteriorates when increasing the dimensionality of the problem; either by increasing the input signal length L or the size of the dictionary P. Even with structured dictionaries, which allow usage of fast algorithms in place of matrix operations, a naive implementation can still be prohibitively inefficient; e.g. processing even just a few seconds of an audio signal, which typically consist of tens of thousands of samples per second, can take hours.

An overview of greedy algorithms, a class of algorithms MP falls under, can be found in [37, 46] and in the context of audio and music processing in [30, 40, 47]. Notable applications of MP algorithms include audio analysis [17], [15], coding [5, 35, 44], time scaling/pitch shifting [10] [43], source separation [16], denoising [2], partial and harmonic detection and tracking [24] and EEG analysis [12].

In this contribution, we present a method for accelerating MP-based algorithms acting on a single overcomplete Gabor dictionary or on a concatenation of several Gabor dictionaries with possibly different windows and parameters (hence the term multi-Gabor dictionary). The main idea of the present acceleration technique is performing the residual update in the coefficient domain while exploiting the locality of the inner products between the atoms in the dictionaries and dismissing values below a user definable threshold. It is then feasible to store all significant inner products in a lookup table and avoid atom synthesis and the residual re-analysis in every iteration of MP as it is usually done in practice. The size of the lookup table as well as the cost of computing it are controlled by the threshold and depend on the parameters of the Gabor dictionaries but are independent of the signal length. An integral part of this contribution is the freely available implementation in C (compatible with C99 and C++11), which can be found in the backend library of the Matlab/GNU Octave Large Time-Frequency Analysis Toolbox (LTFAT, http://ltfat.github.io) [32, 39] available individually at http://ltfat.github.io/libltfat. The low level C language (or rather a subset of C99 and C++11 standards) was chosen for two reasons: First, the nature of the MP algorithm does not lend itself to an efficient implementation in a high level language due to its overhead. For example, a proof-of-concept Matlab implementation was about 50 times slower than the final C implementation. Second, a C-based shared (dynamic) library can be interfaced from most of the high and even low level languages. The programming interface (documentation available at http://ltfat.github.io/libltfat/group__multidgtrealmp.html) was designed with this use case in mind. Indeed, since version 2.3.0, LTFAT itself interfaces the library trough a MEX function, whose call is wrapped in a function `multidgtrealmp` (see http://ltfat.github.io/doc/gabor/multidgtrealmp.html).

To date, considering a vast body of literature dealing with MP, surprisingly few authors address effective (non-textbook) implementation of the algorithm let alone provide code. In the original paper, Mallat and Zhang [28, Appendix E] proposed to perform the residual update in the coefficient domain using inner products between the atoms. They present an analytic formula for evaluating the inner products between atoms of a multi-scale Gabor dictionary with a Gaussian window. An implementation by Ferrando et. al. [13] is tailored to the Gaussian window-based multiscale Gabor dictionary defined on an interval. The authors choose to trade updating the residual directly in the coefficient domain for the flexibility in choosing the dictionary parameters and in boundary handling. The de-facto standard implementation of several MP based algorithms is in

the Matching Pursuit Toolkit (MPTK) [21]. The toolbox is not restricted to Gabor dictionaries, and, therefore, the coefficient-domain update rule is not exploited. In comparison, the present method and implementation is applicable to general multi-Gabor dictionaries while being much faster than MPTK.

The paper is organized as follows. Section 2 summarizes the necessary theoretical background of the MP algorithm, and introduces the approximate residual update crucial to the proposed implementation, as well as a complementary convergence result. Section 4 introduces the main contribution of the paper: the method for accelerating MP iterations with multi-Gabor dictionaries. The rest of the paper consists of Section 5 which discusses some practical aspects of the proposed method and Section 6 containing timing and approximation quality comparisons with the reference implementation in MPTK.

2 PRELIMINARIES

Matrices will be denoted with bold capital upright letters, e.g., $\mathbf{M}$, column vectors with lowercase bold upright letter such as $\mathbf{x}$. Conjugate transpose will be denoted with a star superscript, ($\mathbf{x}^*, \mathbf{M}^*$), scalar variables with a capital or lowercase italics letter s, S and scalar constants as upright capital or lowercase letters like $\pi, \mathrm{e}, \mathrm{i}$. A single element of a matrix or a vector will be selected using round brackets $\mathbf{M}(m, n)$, $\mathbf{x}(l)$. The index is always assumed to be applied modulo vector length (or matrix size in the respective direction) such that $\mathbf{x}(l) = \mathbf{x}(l + kL)$ for $l = 0, \ldots, L - 1$ and $k \in \mathbb{Z}$. Moreover, we will use two indices and subscript for vectors such that $\mathbf{c}(m, n)_M = \mathbf{c}(m + nM)$ in order to transparently "matrixify" a vector. Sub-vectors and sub-matrices will be selected by an index set denoted by a caligraphic letter e.g. $\mathbf{x}(\mathcal{P})$ and the m-th row of a matrix $\mathbf{M}$ will be selected using the notation $\mathbf{M}(m, \bullet)$ and the n-th column by $\mathbf{M}(\bullet, n)$, respectively. We will omit brackets when indexing the outcome of the matrix-vector or matrix-matrix product i.e. we will use $\mathbf{M}\mathbf{x}(p)$ instead of $(\mathbf{M}\mathbf{x})(p)$. Scalar-domain functions used on matrices or vectors are applied element-wise e.g. $|\mathbf{x}|^2(l) = |\mathbf{x}(l)|^2$. The inner product of two vectors in $\mathbb{C}^L$ is given as $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{y}^* \mathbf{x} = \sum_{l=0}^{L-1} \mathbf{x}(l)\overline{\mathbf{y}(l)}$, where the overline denotes complex conjugation. Real and imaginary parts of a complex number will be denoted as $\mathrm{Re}(c)$ and $\mathrm{Im}(c)$ respectively and the phase as $\arg(c)$ such that $c = \mathrm{Re}(c) + \mathrm{i}\,\mathrm{Im}(c) = |c|\mathrm{e}^{\mathrm{i}\arg(c)}$. The 2−norm of a vector is defined as $\|\mathbf{x}\|_2 = (\sum_{l=0}^{L-1} |\mathbf{x}(l)|^2)^{1/2}$. In particular, the 2−norm relates to the inner product as $\|\mathbf{x}\|_2^2 = \langle \mathbf{x}, \mathbf{x} \rangle$. For a matrix $\mathbf{M}$, $\|\mathbf{M}\| = \|\mathbf{M}\|_2 = \max_{\|\mathbf{x}\|_2=1} \|\mathbf{M}\mathbf{x}\|_2$ is the matrix norm induced by the 2−norm.

2.1 Multi-Gabor Dictionaries

A Gabor dictionary $\mathbf{D}_{(\mathbf{g}, a, M)}$ generated from a window $\mathbf{g} \in \mathbb{R}^L$, $\|\mathbf{g}\|_2 = 1$, time shift a and a number of modulations M is given as

$$\mathbf{D}_{(\mathbf{g}, a, M)}(l, m + nM) = \mathbf{g}(l - na)\mathrm{e}^{\mathrm{i}2\pi m(l-na)/M} \tag{3}$$

for $l = 0, \ldots, L - 1$ and $m = 0, \ldots, M - 1$ for each $n = 0, \ldots, N - 1$, where $N = L/a$ is the number of window time shifts and the overall number of atoms is $P = MN$. The expression $(l - na)$ is assumed to be evaluated modulo L according to the circular indexing. The redundancy of a dictionary will be defined as $P/L = M/a$. A multi-Gabor dictionary consisting of W Gabor dictionaries is defined as

$$\left[\mathbf{D}_{(\mathbf{g}_1, a_1, M_1)} \big| \mathbf{D}_{(\mathbf{g}_2, a_2, M_2)} \big| \ldots \mathbf{D}_{(\mathbf{g}_W, a_W, M_W)} \right] \tag{4}$$

and we will also use a shortened notation $\mathbf{D}_w = \mathbf{D}_{(\mathbf{g}_w, a_w, M_w)}$. Generally, a_w, M_w need only be divisors of L. Due to technical reasons explained in Sec. 4.2, however, efficiency of the presented algorithm depends on the pairwise *compatibility* of a_u, a_v and M_u, M_v, implying some restrictions of the dictionary parameters. In the following, we focus on the optimal setting, i.e., parameters a_w chosen such that every pair a_u, a_v is divisible by $a_{\min} = \min\{a_u, a_v\}$ and, similarly, every pair of

M_u, M_v should divide $M_{\max} = \max\{M_u, M_v\}$ and each M_w/a_w should be a positive integer. While not strictly necessary, such setting is commonly used in practice and leads to the most efficient implementation.

2.2 Matching Pursuit – MP

In this section we recall the idea behind the MP algorithm, summarize its steps and explain an alternative way of performing the MP iterations exploiting the inner products between the atoms.

Recall that the main goal is to find the best k-term approximation of a given signal $\mathbf{x}$ by elements from the dictionary, i.e. $\mathbf{x} \approx \mathbf{x}_k = \sum_{l=1}^{k} c_l \mathbf{d}_l$. The MP algorithm iteratively decreases the approximation error (energy of the residual) $E_{k+1} = \|\mathbf{r}_{k+1}\|_2^2$ by considering orthogonal projections $\langle \mathbf{r}_k, \mathbf{d}_p \rangle \mathbf{d}_p$ of the residual $\mathbf{r}_k = \mathbf{x} - \mathbf{x}_k$ over the individual P elements of the normalized dictionary. The element, $p_{\max}$, which decreases the energy of the residual $\|\mathbf{r}_k - \langle \mathbf{r}_k, \mathbf{d}_p \rangle \mathbf{d}_p\|_2^2$ the most is selected and the residual is updated: $\mathbf{r}_{k+1} = \mathbf{r}_k - \langle \mathbf{r}_k, \mathbf{d}_{p_{\max}} \rangle \mathbf{d}_{p_{\max}}$. Since the energy of the new potential residual can be written as

$$\|\mathbf{r}_k - \langle \mathbf{r}_k, \mathbf{d}_p \rangle \mathbf{d}_p\|_2^2 = \|\mathbf{r}_k\|_2^2 - |\langle \mathbf{r}_k, \mathbf{d}_p \rangle|^2, \tag{5}$$

the best atom to choose is the one with the highest inner product with the residual i.e.

$$p_{\max} = \underset{p}{\operatorname{argmax}} |\langle \mathbf{r}_k, \mathbf{d}_p \rangle|.$$

The procedure is repeated until the desired approximation error is achieved or alternatively some other stopping criterion is met e.g. a sparsity or a selected inner product magnitude limits are reached. The error is usually normalized and converted to decibels by $10 \log_{10} E_{k+1}/\|\mathbf{x}\|_2^2$. It is known that the matching pursuit (MP) algorithm and its derivatives can benefit from pre-computing inner products between the atoms in the dictionary $\mathbf{G}(k, j) = \langle \mathbf{d}_j, \mathbf{d}_k \rangle$ i.e. from pre-computing the Gram matrix $\mathbf{G} = \mathbf{D}^*\mathbf{D} \in \mathbb{C}^{P \times P}$. With $\boldsymbol{\varrho}_k = \mathbf{D}^*\mathbf{r}_k$ denoting the coefficient-domain residual, the residual update step can be written as ([27, Ch. 12])

$$\boldsymbol{\varrho}_{k+1} = \boldsymbol{\varrho}_k - \mathbf{c}(p_{\max})\mathbf{G}(\bullet, p_{\max}). \tag{6}$$

Formally, the coefficient-domain matching pursuit algorithm is summarized in Alg. 1. The stopping criterion may contain several conditions, and the algorithm terminates if any of these conditions is met. Typical stopping conditions include reaching a certain error, a maximum number of atom selections or the largest entry in $\boldsymbol{\varrho}_k$ falling below some value.

Input: Input signal $\mathbf{x}$, dictionary Gram matrix $\mathbf{G} = \mathbf{D}^*\mathbf{D}$
Output: Solution vector $\mathbf{c}$
Initialization: $\mathbf{c} = \mathbf{0}$, $\boldsymbol{\varrho}_0 = \mathbf{D}^*\mathbf{x}$, $E_0 = \|\mathbf{x}\|_2^2$, $k = 0$
while *Stopping criterion not met* **do**

    (1) **Selection:** $p_{\max} \leftarrow \underset{p}{\operatorname{argmax}} |\boldsymbol{\varrho}_k(p)|$

    (2) **Update:**
      (a) **Solution:** $\mathbf{c}(p_{\max}) \leftarrow \mathbf{c}(p_{\max}) + \boldsymbol{\varrho}_k(p_{\max})$
      (b) **Error:** $E_{k+1} \leftarrow E_k - |\boldsymbol{\varrho}_k(p_{\max})|^2$
      (c) **Residual:** $\boldsymbol{\varrho}_{k+1} \leftarrow \boldsymbol{\varrho}_k - \boldsymbol{\varrho}_k(p_{\max})\mathbf{G}(\bullet, p_{\max})$

   $k \leftarrow k + 1$
end

Algorithm 1: Coefficient-Domain Matching Pursuit

This modification has the advantage of removing the necessity of synthesizing the residual and recomputing the inner product in the selection step. On the other hand, such approach is usually dismissed as impractical in the literature due to the high memory requirements for storing the Gram matrix. This is however not the case for a well behaved multi-Gabor dictionary, for which the Gram matrix can be precomputed and significant values stored compactly, as detailed in the next Sections.

3 APPROXIMATE UPDATE BY A TRUNCATED GRAM MATRIX

As already mentioned, our acceleration technique works with the coefficient domain update formula (6). Whenever the elements of the dictionary $\mathbf{D}$ are localized, most of the entries of the Gram matrix are close to zero. Discarding these entries is the first step towards reducing the memory requirements of coefficient-domain matching pursuit, at the cost of introducing a small approximation error. Before considering the additional structure imposed by a multi-Gabor dictionary, we discuss the implications of using a truncated Gram matrix in Step 2c in Alg. 1 and provide a basic worst-case error estimate.

For this purpose, denote by $\mathbf{G}_\epsilon$, $\epsilon > 0$, the hard-thresholded Gram matrix, i.e.,

$$\mathbf{G}_\epsilon(k, j) = \begin{cases} \mathbf{G}(k, j) & \text{if } |\mathbf{G}(k, j)| > \epsilon, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Similar to ϱ_k, we further denote by p_k the index selected in the k-th selection step. Using the full Gram matrix $\mathbf{G}$ in Alg. 1, we always have

$$\varrho_k = \mathbf{D}^* \mathbf{r}_k, \quad \text{i.e.,} \quad \varrho_k(p) = \langle \mathbf{r}_k, \mathbf{d}_p \rangle, \text{ for all } p, \tag{8}$$

where $\mathbf{r}_k = \mathbf{x} - \mathbf{D}\mathbf{c}_k$ is the true residual, i.e., the difference between the signal $\mathbf{x}$ and the proposed solution after the k-th step $\mathbf{D}\mathbf{c}_k$. If we use the truncated Gram matrix $\mathbf{G}_\epsilon$ instead, both the error E_k and the coefficient-domain residual ϱ_k is Alg. 1 are merely estimates of the true quantities. Thus, (8) does not hold anymore. Further, the selection of $p_{\max}$ is based directly on the coefficient-domain residual (estimate) ϱ_k, such that the sequence $(p_k)_k$ of selected positions must be expected to differ between matching pursuit and this approximate variant.

Unless noted otherwise, we will use the notation $\mathbf{c}_0 = \mathbf{0}$, $\varrho_0 = \mathbf{D}^* \mathbf{x}$ and, for $k \geq 0$,

$$p_{k+1} = \underset{p}{\operatorname{argmax}} |\varrho_k(p)|, \quad \varrho_{k+1} = \varrho_k - \varrho_k(p_{k+1})\mathbf{G}_\epsilon(\bullet, p_{k+1}), \quad \mathbf{c}_{k+1} = \mathbf{c}_k + \varrho_k(p_{k+1})\mathbf{e}_{p_{k+1}}, \tag{9}$$

where $\mathbf{e}_j \in \mathbb{C}^P$ is the j-th standard unit vector.

In the following, we show that despite these differences to accurate matching pursuit, the proposed approximate coefficient-domain matching pursuit algorithm reduces the approximation error $\|\mathbf{r}_k\|_2^2$, unless $\max_p |\varrho_k(p)|$ is too small. Although arbitrarily small approximation error cannot be guaranteed when this approximate scheme is used directly, we show that the approximate matching pursuit can be nested inside a simple reset procedure to ensure $\|\mathbf{r}_k\|_2^2 \to 0$, see Alg. 2. Similar to Alg. 1, both the reset and stopping criterion may consist of any number of conditions, terminating the loop if any condition is met. For this nested execution of approximate matching pursuit, we further provide a decay estimate on $\|\mathbf{r}_k\|_2^2$. The proofs of the following results can be found in the Appendix.

Theorem 1. *Fix some positive Gramian threshold $\epsilon > 0$ and some $0 < \delta < 1/2$. Let p_k and ϱ_k, $k \geq 0$, be as in* (9). *If*

$$|\varrho_k(p_{k+1})| \geq \frac{\epsilon}{\delta} \sum_{l=1}^{k} |\varrho_{l-1}(p_l)| \quad \text{and} \quad |\varrho_k(p_{k+1})| > \frac{2\delta}{1 - 2\delta} \|\mathbf{r}_k\|_2, \tag{10}$$

Input: Input signal $\mathbf{x}$, truncated dictionary Gram matrix $\mathbf{G}_\epsilon$, dictionary $\mathbf{D}$
Output: Solution vector $\mathbf{c}^{\mathrm{out}}$
Initialization: $\mathbf{c}^{\mathrm{out}} = \mathbf{0}$, $\mathbf{r}^{\mathrm{out}} = \mathbf{x}$, $E^{\mathrm{out}} = \|\mathbf{x}\|_2^2$, $l = 0$, $k = 0$, $k^{\mathrm{out}} = 0$
while *Stopping criterion not met* **do**

> (1) **Approximate Matching Pursuit:**
> **Input:** $\mathbf{r}^{\mathrm{out}}$, $\mathbf{G}_\epsilon$
> **Output:** Solution vector $\mathbf{c}$
> **Initialization:** $\mathbf{c} = \mathbf{0}$, $\varrho_k = \mathbf{D}^*\mathbf{r}^{\mathrm{out}}$,
> $E_k = E^{\mathrm{out}}$
> **while** *Reset criterion not met* **do**
>> (a) **Selection:** $p_{\max} \leftarrow \underset{p}{\operatorname{argmax}} \, |\varrho_k(p)|$
>> (b) **Update:**
>>> (i) $\mathbf{c}(p_{\max}) \leftarrow \mathbf{c}(p_{\max}) + \varrho_k(p_{\max})$
>>> (ii) $E_{k+1} \leftarrow E_k - |\varrho_k(p_{\max})|^2$
>>> (iii) $\varrho_{k+1} \leftarrow \varrho_k - \varrho_k(p_{\max})\mathbf{G}_\epsilon(\bullet, p_{\max})$
>>
>> $k \leftarrow k + 1$
>
> **end**
> (2) **Update:**
>> (a) **Selections:** $k^{\mathrm{out}} \leftarrow k$
>> (b) **Solution:** $\mathbf{c}^{\mathrm{out}} \leftarrow \mathbf{c}^{\mathrm{out}} + \mathbf{c}$
>> (c) **Residual:** $\mathbf{r}^{\mathrm{out}} \leftarrow \mathbf{r}^{\mathrm{out}} - \mathbf{D}\mathbf{c}$
>> (d) **Error:** $E^{\mathrm{out}} \leftarrow \|\mathbf{r}^{\mathrm{out}}\|_2^2$
>
> $l \leftarrow l + 1$

end

Algorithm 2: Approximate Coefficient-Domain Matching Pursuit with Reset

then $\|\mathbf{r}_{k+1}\|_2^2 < \|\mathbf{r}_k\|_2^2$.

The conditions (10) are mostly of theoretical interest for two reasons: Firstly, the estimates made in proving Theorem 1 are highly pessimistic in the sense that they assume the worst-case error. The actual error is highly likely to be significantly smaller. Secondly, verifying the second condition in (10) is expensive, as it requires the computation of the true time-domain residual. Therefore, while possible, we will not use (10) as conditions for the stopping criterion.

In the following theorem, ϱ_k is used as in Algorithm 2, i.e., when the reset criterion is trigered in the $(k + 1)$-th selection step, then the inner loop is restarted with $\varrho_k := \mathbf{D}^*(\mathbf{x} - \mathbf{D}\mathbf{c}_k)$. Otherwise, the defintion of ϱ_k coincides with the one given above.

Theorem 2. *Assume that* $\mathbf{D}$ *is finite and spans* $\mathbb{C}^L$ *with* $\lambda_{min} := \inf_{\substack{\mathbf{x} \in \mathbb{C}^L \\ \|\mathbf{x}\|_2 = 1}} \max_p |\langle \mathbf{x}, \mathbf{d}_p \rangle|^2 > 0$. *Fix some* $0 < \delta < 1/2$ *and* $0 < \varepsilon < 1 - 2\delta$, *such that* $\frac{2\delta}{1 - (2\delta + \varepsilon)} < \sqrt{\lambda_{min}}$. *If the Alg. 2 is initialized with* $\mathbf{G}_\epsilon$, $0 < \epsilon < 1$, *and the* reset criterion *includes the conditions*

$$|\varrho_k(p_{k+1})| < \frac{\varepsilon}{\delta} \sum_{l=k^{out}+1}^{k} |\varrho_{l-1}(p_l)| \quad and \quad |\varrho_k(p_{k+1})| < \frac{2\delta(1 - \varepsilon(1+\delta)^{-2}\lambda_{min})^{\frac{k^{out}-k}{2}}}{1 - (2\delta + \varepsilon)}\|\mathbf{r}^{out}\|_2, \quad (11)$$

then the residual decreases exponentially:

$$\|\mathbf{r}_{k+1}\|_2^2 < (1 - \varepsilon(1+\delta)^{-2}\lambda_{min})\|\mathbf{r}_k\|_2^2, \text{ for all } k \geq 0. \quad (12)$$

Corollary 3. *Fix some positive $\epsilon > 0$ and δ, ε as in Theorem 2. If Algorithm 2 is initialized $\mathbf{G}_\epsilon$, the* stopping criterion *includes the condition $E^{\text{out}} \leq E\|x\|_2^2$, for some $0 < E < 1$, and the* reset criterion *includes the conditions* (11) *and*

$$k > \frac{\log(E)}{\log(1 - \varepsilon(1+\delta)^{-2}\lambda_{min})}, \tag{13}$$

then the algorithm terminates after a finite number of total selection steps, achieving the desired approximation error $E^{\text{out}} \leq E\|x\|_2^2$.

Once more, the conditions given in Theorem 2 and Corollary 3 are very conservative worst-case conditions and it is not advisable to use them in practice. This is easy to see from the proofs presented in the Appendix. Nonetheless, the results serve as justification for the introduction of resets to ensure convergence. Although the results may suggest that resets are required rather regularly, it is in practice rarely required. In the case of the multi-Gabor dictionaries considered in this paper, excellent approximation quality is achieved without resets for $\epsilon = 10^{-4}$. Heuristically, we observed that it is sufficient to reset only after a large number of selection steps, except when the truncation threshold ϵ is large. The selection of reset criteria is also discussed shortly in Section 6.1.

4 FASTER APPROXIMATE COEFFICIENT-DOMAIN RESIDUAL UPDATE

In this section, we will show that, for multi-Gabor dictionaries, the significant values of the Gram matrix $\mathbf{G}$ can be precomputed at a cost independent of the entire signal length L, truncated and stored efficiently for a single as well as for a multi-Gabor dictionary. We exploit the fact that the Gram matrix $\mathbf{G}$ of a single Gabor dictionary $\mathbf{D} = \mathbf{D}_w$ is highly structured. Using this structure, we obtain a feasible, and in fact highly efficient, implementation of approximate coefficient-domain matching pursuit. In fact, $\mathbf{G}$ takes the form of a *twisted convolution* [20] matrix with a fixed kernel $\mathbf{h} = \mathbf{G}(\bullet, 0) = \mathbf{D}^*\mathbf{g} \in \mathbb{C}^{MN}$; a coefficient vector consisting of inner products of the window with all its possible time and frequency shifts. The p-th column of the Gram matrix $\mathbf{G}(\bullet, p)$ i.e., the inner products of the atom at the time-frequency position (k, j) (such that $p = k + jM$) with atoms at all time-frequency positions (m, n) are constructed by shifting and modulating the kernel such that

$$\mathbf{G}(m + nM, k + jM) = \mathbf{h}(m - k, n - j)_M e^{i2\pi k \frac{a}{M}(n-j)} \tag{14}$$

for $m = 0, \ldots, M - 1$ for each $n = 0, \ldots, N - 1$. Crucially, $\mathbf{h}(\bullet, \bullet)_M$ is essentially supported around the origin for localized, low-pass windows, such that it can be truncated and stored efficiently. By considering the concentration of the kernel relative to the time-frequency index (k, j), i.e. shifting $\mathbf{h}(\bullet, \bullet)_M$ from the origin to (k, j), we see that the modulation factor in (14) is in fact independent of the time index j. We can define

$$\mathbf{h}^{(k)}(m, n) := \mathbf{h}(m, n)_M e^{i2\pi k \frac{a}{M}n}. \tag{15}$$

Therefore, after selecting atom $p_{\max} = k_{\max} + j_{\max}M$ in the MP algorithm, the coefficient-domain residual update in the style of (6) reduces to a subtraction of a truncated, modulated and weighted kernel from the neighborhood of the time-frequency position $(k_{\max}, j_{\max})$. With $\mathbf{h}^{(k)}$ as in (15), the residual update can be written as

$$\varrho_{k+1}(\mathcal{K}, \mathcal{J})_M = \varrho_k(\mathcal{K}, \mathcal{J})_M - \varrho(p_{\max})\mathbf{h}^{(k_{\max})} \tag{16}$$

assuming the $\mathbf{h}^{(k)}$ have already been truncated and $\mathcal{K}$ and $\mathcal{J}$ denote index sets of the appropriate neighborhood encompassing the overlay with the truncated kernel. The neighborhood $(\mathcal{K}, \mathcal{J})_M$ is of fixed size and shape, independent of $p_{\max}$, and centered on $p_{\max}$. From Equations (16) and (15), it is easy to see that the computational cost of the residual update is directly related to the size of the
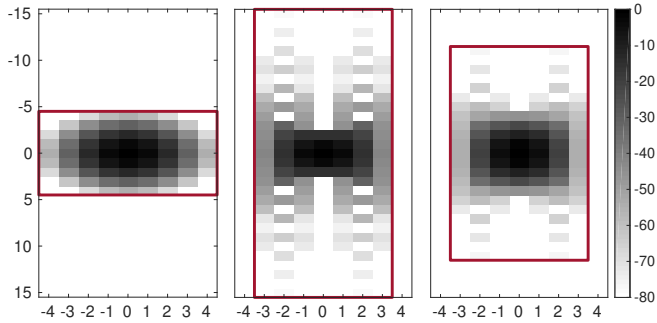
Fig. 1. Examples of abs. values of truncated kernels for (left) Gaussian (9×9), (middle) Hann (31×7) and (right) Blackman (23×7) windows.

neighborhood $(\mathcal{K}, \mathcal{J})_M$, .i.e., the size of the truncated kernel $\mathbf{h}$. The latter is equal to the number of nonzero elements per column of $\mathbf{G}_\epsilon$, which depends, for a fixed Gabor dictionary, only on the selected truncation threshold ϵ.

4.1 Pre-computing the Kernel for a Single Dictionary

The inner products between the window and its translations which are sufficiently far away are obviously zero or at least negligible. Therefore after determining the length of the window's effective support L_w, one can compute the minimum admissible $L_{\min}$ for the computation of values of the kernel $\mathbf{h}$ as being the next integer multiple of a bigger than twice the length of the effective support, i.e., $L_{\min} = \lfloor 2L_w/a \rfloor a + 1$. The kernel is further truncated also in the frequency direction such that values below a certain threshold are dismissed. The resulting size of the truncated kernel depends on the shape of the window and on the length of the time and frequency steps. The kernel size directly determines the number of complex multiplications and additions required to perform the entire residual update step and, obviously, also the memory requirements to store the kernel. Examples of abs. values of kernels for several windows using time shift $a = 512$ and $M = 2048$ frequency bins are depicted in Fig. 1. The values are in dB relative to the maximum element with 0 dB. Values below -80 dB were cropped and are not used in the residual update. The threshold selection is a trade-off between faster updates (higher threshold) or less requirement for resets (lower threshold). The idea of truncating the kernel $\mathbf{h}$ originates from Le Roux et al. [23] who used it for replacing the operation of the (cross) Gram matrix in an iterative scheme dealing with magnitude-only reconstruction. The authors of the aforementioned paper also noticed that the kernel is conjugate symmetric about both the horizontal (time) and the vertical (frequency) axes, which could be exploited for reducing the number of multiplications further. When inspecting formula (15), it is obvious that for a fixed frequency position k the modulation by $2\pi ka/M$ radians is performed on all rows of the kernel independently. Moreover, the modulation frequencies are $\operatorname{lcm}(a, M)/a$ periodic in k and, therefore, all unique complex exponentials can be tabulated and stored. In the best case when M is integer divisible by a, the memory requirements are equal to storing M/a additional rows of the kernel. The cost of applying the modulation during the residual update step is one complex multiplication per kernel column.

4.2 Pre-computing Cross-Kernels Between Dictionaries

The Gram matrix of a multi-Gabor dictionary consists of Gram matrices of individual dictionaries $\mathbf{D}_w$ and cross-Gram matrices [1] between the dictionaries. Denoting a cross-Gram matrix as

$\mathbf{G}_{w,v} = \mathbf{D}_w^* \mathbf{D}_v$ the overall Gram matrix is a block matrix with the following structure

$$
\begin{bmatrix}
\mathbf{G}_{1,1} & \mathbf{G}_{1,2} & \dots & \mathbf{G}_{1,W} \\
\mathbf{G}_{2,1} & \mathbf{G}_{2,2} & \dots & \mathbf{G}_{2,W} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{G}_{W,1} & \mathbf{G}_{W,2} & \dots & \mathbf{G}_{W,W}
\end{bmatrix}.
\tag{17}
$$

A cross-Gram matrix $\mathbf{G}_{w,v}$ shares the same twisted convolution structure with the regular Gram matrix with kernel $\mathbf{h}_{w,v} = \mathbf{D}_w^* \mathbf{g}_v$ only if the time-frequency shifts are equal i.e. $a_w = a_v$ and $M_w = M_v$. In the case the parameters differ, the direct twisted convolution structure is lost. The structure can be recovered on a finer "common" time-frequency grid given by the time step $\gcd(a_w, a_v)$ and the number of frequency bins $\mathrm{lcm}(M_w, M_v)$. The most efficient case is achieved when a_w and a_v are divisible by $a_{\min} = \min\{a_w, a_v\}$ and M_w and M_v both divide $M_{\max} = \max\{M_w, M_v\}$ resulting to a common grid given by $a_{\min}$ and $M_{\max}$. In the residual update step of the inner products of the residual with the w-th dictionary, the modulated kernel is subsampled by ratios $a_w/a_{\min}$ and $M_{\max}/M_w$ in horizontal and vertical directions respectively. To illustrate, consider a multi-Gabor dictionary consisting of two Gabor dictionaries $\mathbf{D}_1 = \mathbf{D}_{(g_1, a_1, M_1)}$ and $\mathbf{D}_2 = \mathbf{D}_{(g_2, a_2, M_2)}$ with $a_1 = 4a_{\min}, M_1 = 8a_{\min}$ and $a_2 = a_{\min}, M_2 = 2a_{\min}$. Both cross-kernels $\mathbf{h}_{1,2}$ and $\mathbf{h}_{2,1}$ are computed with $a_{\min} = a_2$ and $M_{\max} = M_1$. Similar to the kernel $\mathbf{h}$, the size of the stored cross-kernels directly determines the computational cost of the update step and depends only on the Gramian threshold ϵ. The example in Fig. 2 depicts an update of inner products of the residual with both dictionaries on the common grid.
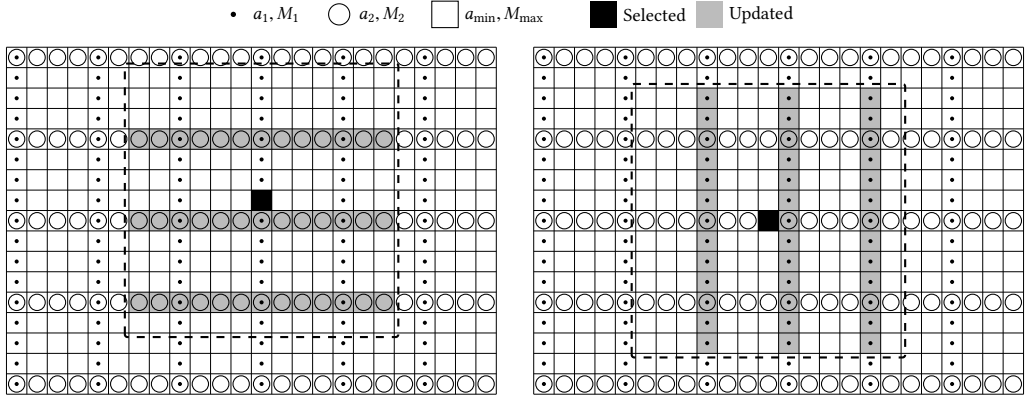

Fig. 2. Illustration of the residual update between dictionaries using cross kernels. The left figure shows the case when a coefficient from the dictionary 1 was selected in the selection step of MP and inner products with dictionary 2 are being updated and vice versa right. The dashed line square depicts the area covered by a cross-kernel with respect to the common grid $a_{\min}, M_{\max}$.

5 PRACTICAL CONSIDERATIONS

Since MP is a simple algorithm, the main focus will be on a detailed description of the accelerated residual update step from Sec. 4. Other steps will be mentioned in less detail.

The presented description is adapted to the setting of real signals $\mathbf{x} \in \mathbb{R}^L$. Therefore, only the first $M_{w,\mathcal{R}} = \lfloor M_w/2 \rfloor + 1$ frequency bins form each Gabor dictionary w will be considered and the

reduced dictionary will be denoted as $\mathbf{D}_{w,\mathcal{R}}$. A real signal can be recovered from reduced coefficient vectors $\mathbf{c}_{w,\mathcal{R}}$ as

$$\mathbf{x} = \sum_{w=1}^{W} \mathbf{D}_{w,\mathcal{R}} \mathbf{c}_{w,\mathcal{R}} + \overline{\mathbf{D}_{w,\mathcal{R}}} \overline{\mathbf{c}_{w,\mathcal{R}}} \tag{18}$$

where elements of $\overline{\mathbf{c}_{w,\mathcal{R}}}$ are set to zero whenever the conjugated partner is missing i.e. for frequency index $m = 0$ and $m = M_w/2$ if M_w is even. Obviously, in practice the matrix operations are replaced by an efficient FFT-based algorithm (see e.g. [31, 38]) or by memory efficient atom-by-atom synthesis as it is done in MPTK.

As discussed in [15, Appendix B], dealing with real signals in this way requires that conjugated pairs of complex atoms $\mathbf{d}, \overline{\mathbf{d}}$ are considered as real atoms, such that the signal approximation and the residual are real at any stage of the algorithm and the positive-negative frequency conjugate symmetry of the coefficients is preserved. Consequently, all inner products must be adjusted as

$$\widetilde{c} = \frac{\langle \mathbf{r}_k, \mathbf{d} \rangle - \langle \mathbf{d}, \overline{\mathbf{d}} \rangle \overline{\langle \mathbf{r}_k, \mathbf{d} \rangle}}{1 - |\langle \mathbf{d}, \overline{\mathbf{d}} \rangle|^2} \tag{19}$$

and subtracting the pair of atoms from the residual decreases its energy by

$$\widetilde{E} = \left[\operatorname{Re}(\widetilde{c})^2 + \operatorname{Im}(\widetilde{c})^2 + \operatorname{Re}(\langle \mathbf{d}, \overline{\mathbf{d}} \rangle) \left(\operatorname{Re}(\widetilde{c})^2 - \operatorname{Im}(\widetilde{c})^2 \right) - 2 \operatorname{Im}(\langle \mathbf{d}, \overline{\mathbf{d}} \rangle) \operatorname{Re}(\widetilde{c}) \operatorname{Im}(\widetilde{c}) \right] \cdot 2. \tag{20}$$

Obviously, as long as $\langle \mathbf{d}, \overline{\mathbf{d}} \rangle \approx 0$, we can simplify the equations such that it is enough to consider only a single atom $\mathbf{d}$. In the following, we address the issue of an efficient search for the maximum inner product.

As discussed above, theory suggests that (20) should be used to determine atoms for selection. In practice, however, discarding the effect of the nonzero inner product between the conjugated atoms in the selection step (setting $\langle \mathbf{d}, \overline{\mathbf{d}} \rangle = 0$) does not have a significant impact on the sparsity of the result achieving a specified approximation error and leads to about 20% overall speedup. Our implementation supports both options and the technically correct one will be referred to as *pedantic*. On the other hand, it is crucial that the selected coefficient is adjusted using (19) prior to the residual update step. Finally, due to the conjugate symmetry, the residual update step now involves a pair of atoms each of which can however be treated separately. In the description of the implementation, we will work exclusively with the reduced dictionary, therefore we will drop the $\mathcal{R}$ subscript in order to lighten the notation.

Given W sets of Gabor dictionary parameters $(\mathbf{g}_w, a_w, M_w)$ for $w = 1, \ldots, W$, the initialization involves pre-computing (or loading) cross-kernels and the complex exponentials according to the rules described in Sec. 4. The algorithm itself then starts by computing inner products of the input signal $\mathbf{r}_0 = \mathbf{x}$ (the initial residual) with atoms from all reduced dictionaries $\mathbf{c}_w = \mathbf{D}_w^* \mathbf{x}$ for all $w = 1, \ldots, W$. Whenever the algorithm is reset, this procedure is repeated, with the current residual $\mathbf{r}^{\text{out}}$ substituting for $\mathbf{x}$.

5.1 Keeping Track Of The Maximum

Performing the full search for the maximum inner product in each selection step is highly inefficient. The authors of MPTK [21] proposed to store positions of maxima for each window time position and organize them in a partial hierarchical tournament-style tree. Such tree contains at each level maxima from pairs from one level below. Since the residual update affects only a limited number of neighboring time positions, a bottom-up tree update becomes more efficient than a full search. To quantify the reduction of the number of comparisons required to find the maximum, consider an array of length L and a tree of depth d, where $d = 0$ means a fallback to a full array search. After the tree has been initialized, the search can be performed at the top level of the tree which requires only

Tree for n-th time frame
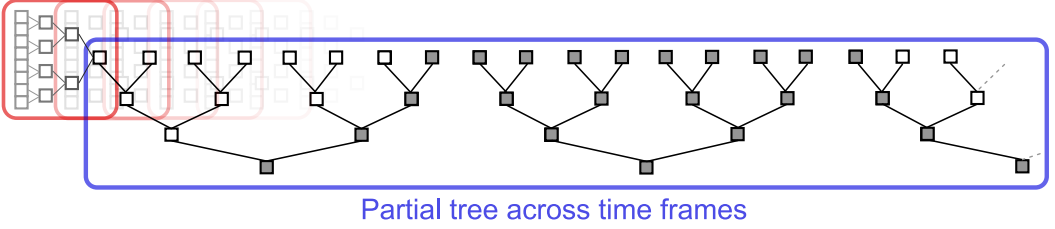

Partial tree across time frames

Fig. 3. Schematic of the tournament tree structure for keeping track of maxima. A red border indicates individual tournament trees across frequency bins for each time frame. The blue border indicates the partial tournament tree across time frames. An example of the worst-case 10 element bottom-up update of the tree across frames is shown in gray.

$\lceil L/2^d \rceil - 1$ comparisons. When Q consecutive elements from the array are modified, the bottom up tree update requires additional $Q + d$ comparisons in the worst case. Since Q is expected to be much smaller than L, the tree-search is more efficient than a simple search in the whole array which obviously requires $L - 1$ comparisons. The worst-case bottom-up update of a 3–level tree is depicted in Fig. 3. Moreover, in the case of the present method, the kernel and therefore the residual coefficient update is localized in frequency as well. Therefore, tournament-style trees are used for keeping track of maxima for individual window time positions (across frequency bins) in a similar manner.

The trees provide the maximal coefficient $\varrho_k(p_{\max})$ of the coefficient-domain residual, where $p_{\max}$ is given in terms of the dictionary $w_{\max}$ and the position $m_{\max}, n_{\max}$.

5.2 Fast Update Step With Real Atoms From Complex Multi-Gabor Dictionary

When dealing with real atoms, pairs of conjugated atoms are involved. The inner product of the currently selected atom with the conjugated partner $\langle \mathbf{d}_{\max}, \overline{\mathbf{d}}_{\max} \rangle$ can actually be extracted from the kernel $\mathbf{h}_{w_{\max}, w_{\max}}$. Note that since the kernel has been truncated, we consider the inner product to be zero if the conjugated atom is not in the range of the kernel update. We consider the inner product to be zero also if the conjugated atom is missing.

Before the update step, $c_{\max}$ is adjusted using (19). The solution update step is performed as in Alg. 1 step 2a while the error update step 2b uses (20) in place of the squared magnitude of the coefficient if the conjugated partner of $\mathbf{d}_{\max}$ is present. The steps of the coefficient residual update in the coefficient domain are summarized in Alg. 3. Note the substraction of the kernel is performed for the conjugated atom as well, if necessary.

6 COMPARISON WITH MPTK

In order to showcase the efficiency of the proposed algorithm and its implementation, in this section, we present a comparison with MPTK (version 0.7.0), which is considered to be the fastest implementation available. To our knowledge there is no previous implementation of coefficient-domain MP that is able to decompose signals of the size considered here. We measure the duration of the matching pursuit decomposition only. From MPTK, we used the modified mpd utility tool. The testing signal was the first channel from the file no. 39 from the SQAM database [45], which is a 137 seconds long piano recording sampled at 44.1 kHz totaling $6 \cdot 10^6$ samples. Both implementations link the same FFTW library [14] version 3.3 and were compiled using the GCC (g++) compiler (version 7.2.0) with the -Ofast optimization flag enabled. The creation of the FFTW plans and the

Input: $c_{\max}, w_{\max}, m_{\max}, n_{\max}, \langle \mathbf{d}_{\max}, \overline{\mathbf{d}}_{\max} \rangle, a_1, \ldots, a_W, M_1, \ldots, M_W, \mathbf{h}_{w,v}$ $(w, v = 1, \ldots, W)$
Output: Inner products to be updated $\mathbf{c}_1^{\mathbf{r}_k}, \ldots, \mathbf{c}_W^{\mathbf{r}_k}$
for $w = 1, \ldots, W$ **do**
> $\mathbf{c}_w^{\mathbf{r}_k} \leftarrow \text{singleDictionaryResidualUpdate}(m_{\max}, w, \mathbf{c}_w^{\mathbf{r}_k})$
> **if** $|\langle \mathbf{d}_{\max}, \overline{\mathbf{d}}_{\max} \rangle| > 0$ **then**
> > $m_{\text{conj}} = M_{w_{\max}} - m_{\max}$
> > $\mathbf{c}_w^{\mathbf{r}_k} \leftarrow \text{singleDictionaryResidualUpdate}(m_{\text{conj}}, w, \mathbf{c}_w^{\mathbf{r}_k})$
> **end**
end

Function $\mathbf{c}_w \leftarrow \text{singleDictionaryResidualUpdate}(m_{\max}, w, \mathbf{c}_w)$:
> $a_{\text{rat}} \leftarrow a_w / a_{w_{\max}}, M_{\text{rat}} \leftarrow M_{w_{\max}} / M_w$
> $a_{\text{step}} = a_{\text{rat}}$ or 1 if $a_{\text{rat}} < 1$, $M_{\text{step}} = M_{\text{rat}}$ or 1 if $M_{\text{rat}} < 1$
> /* Determine index sets   */
> Define $\mathcal{M}$, horizontal index set with stride M_{step} in cross kernel $\mathbf{h}_{w_{\max}, w}$ taking into the account the misalignment of the grids.
> Define $\mathcal{N}$, vertical index set in a similar way using stride a_{step}.
> Define $\mathcal{I}$, residual coefficient vector index set covered by the kernel.
> /* Update the residual (as in (16)) using truncated, subsampled and modulated cross-kernel
> (15):   */
> $\mathbf{c}_w(\mathcal{I}) = \mathbf{c}_w(\mathcal{I}) - c_{\max} \mathbf{h}_{w_{\max}, w}^{(m_{\max})}(\mathcal{M}, \mathcal{N})$
return

Algorithm 3: Approximate coefficient-domain MP residual update

computation of the kernels was excluded from the measurements. The specification of the PC the timing was performed on was Intel® Core™ i5-4570 3.20 GHz, 16 GB RAM running Ubuntu 16.04. The timing was done on an idle machine using the high-precision timer from the C++11 standard library chrono. The data type was double precision floating point. We used single and multi Gabor dictionaries with the Blackman window and various redundancies M_w / a_w. The length of the window was always equal to M (required by MPTK). In the decomposition we performed $1.8 \cdot 10^5$ iterations. By fixing the number of iterations instead of a desired approximation estimate, we ensure that execution time is roughly independent of the considered signal. Table 1 shows a comparison of execution times, in seconds, for a single Gabor dictionary, numbers of bins $M_w = 512, \ldots, 8192$ (additionally also 16384 for the proposed implementation) and various hop sizes a (and therefore redundancies). Additionally, a comparison of execution times using two multi-dictionaries is shown, each of which consists of five Gabor dictionaries (at redundancies 4 and 8 per dictionary).

In the tested setting, the proposed implementation clearly outperforms MPTK in terms of computational speed. The memory requirements are however notably higher since the residual is stored in the coefficient domain and, additionally, the pre-computed kernels and the modulation factors must be stored as well. Hence, the proposed method requires additional memory in an amount roughly proportional to the redundancy of the dictionary $\mathbf{D}$. Note, however, that storage of the kernels and the modulation factors is independent of the signal length L and thus increasingly insignificant.

Bins (M)	512	1024	2048	4096	8192	16384
			$a = M/4$			
MPTK	3.96	8.40	17.0	36.4	75.7	–
Proposed	0.92	1.00	1.02	1.03	1.03	1.08
			$a = M/8$			
MPTK	6.73	15.1	30.2	61.3	147	–
Proposed	1.70	1.90	1.95	2.10	2.20	2.21
			$a = M/16$			
MPTK	13.2	28.3	56.8	119	274	–
Proposed	3.20	3.50	4.00	4.50	4.60	5.08
			$a = M/32$			
MPTK	23.2	52.6	110	233	530	–
Proposed	6.35	7.40	7.90	8.20	9.60	10.7
Multi-Gabor	$a = M/4$			$a = M/8$		
MPTK	142			285		
Proposed	10.6			20.9		

Table 1. Execution time in seconds for MPTK and the proposed method on Gabor and Multi-Gabor dictionaries (180k selection steps). The Multi-Gabor dictionaries are a concatenation of the dictionaries with $M = 512, \ldots, 8192$ at redundancies $a = M/4$ and $a = M/8$, respectively.

6.1 Convergence in practice

The results in Section 3 suggest that the proposed approximate coefficient-domain update will not achieve arbitrarily small approximation error in general. Arbitrary approximation quality can only be achieved if resets are performed. To test the necessity of resets in practical applications, we compare the residual norm achieved after k selection steps by MPTK and the proposed method. We do so for various values of k and the truncation threshold ϵ. The experiment was performed using a concatenation of 3 Gabor dictionaries with Blackman window, $M_w = 512, 1024, 2048$ and $a_w = M_w/4$, as used in Section 6. In addition to the audio test signal used previously, we also consider a pseudo-random Gaussian noise of equal length, generated in Matlab using the function randn. While the considered audio signal is approximately sparse and can be approximated with a moderate number of atoms, the noise signal is not sparse at all. Generally, our implementation of the proposed method is set to terminate when $\sum_{l=1}^{k} \varrho_{l-1}(p_l)^2 > \|\mathbf{x}\|_2^2$. This condition simply means that the error estimate E_k is negative and serves as a cheap indicator that further selection steps are expected to harm the approximation quality. Although this condition is not sufficient to prevent unproductive selection steps altogether, it was usually sufficient to prevent notable divergence of the matching pursuit estimate for various signals and threshold values, in particular for all presented experiments. The number of selection steps performed until this stopping condition is met depends, however, heavily on the considered signal, see Figure 4. Instead, we observed similar final approximation quality across different signals when this is the only stopping condition. We also state the final approximation error and number k of selection steps performed before $\sum_{l=1}^{k} \varrho_{l-1}(p_l)^2 > \|\mathbf{x}\|_2^2$. In all experiments, approximation quality of the proposed method follows MPTK closely, as long as $\sum_{l=1}^{k} \varrho_{l-1}(p_l)^2$ does not approach $\|\mathbf{x}\|_2^2$. Similar results have been observed when testing other signals, which indicate that, for a fixed threshold ϵ and without resets, the final approximation quality achieved by the proposed method depends mainly on the sparsity of the considered signal's Gabor coefficients. In practice, memory usage of MPTK grows linearly in the

number of selection steps, such that we could only test MPTK approximation quality up to 10^6 selection steps.
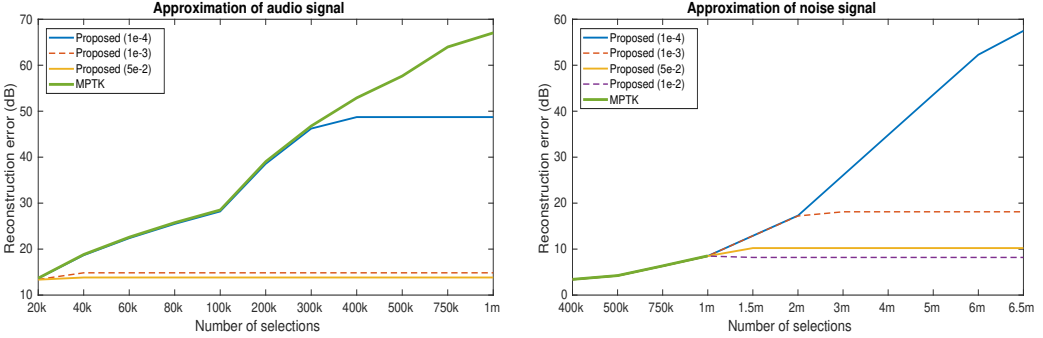

Fig. 4. Approximation errors achieved by MPTK and the proposed method. The proposed method was evaluated for various threshold values ϵ. For the audio signal (left) the proposed method terminated after 368235 ($\epsilon = 10^{-4}$), 35115 ($\epsilon = 10^{-3}$) and 29138 ($\epsilon = 5 \cdot 10^{-3}$) steps, achieving an approximation quality of 48.72, 14.84 and 13.83 dB, respectively. For the noise signal (right) the proposed method terminated after 6623689 ($\epsilon = 10^{-4}$), 2160138 ($\epsilon = 10^{-3}$), 1320287 ($\epsilon = 5 \cdot 10^{-3}$) and 1240012 ($\epsilon = 10^{-3}$) steps, achieving an approximation quality of 57.48, 18.12, 10.22 and 8.19 dB, respectively.

When resets are activated, our Matlab interface relies on a reset criterion combining a maximum number of selections per reset and some relative error tolerance by default, see Section 7. The default reset criterion depends on the kernel threshold and was conservatively selected to achieve desired approximation quality in reasonable time for audio signals and kernel thresholds in the range $\epsilon \in [10^{-2}, 10^{-4}]$. However, the optimal selection of reset criteria, accounting for accuracy, runtime and the number of selection steps, remains an open problem. In the following experiment, we determined the maximum number of selections per reset conservatively from the point of stagnation found without resets in the previous experiment.

On the other hand, the number of steps before termination obtained in the test above, see Figure 4, provides us with a better idea of how many *good* selections we can perform before a reset is required. For the two tested signals, we can choose a less conservative reset condition. For a second experiment, we repeated the previous test, setting the reset criterion to a fixed number of iterations somewhat below the numbers obtained in the first test. With these values, we attempted once more to approximate both test signals. Specifically, resets were performed after $32 \cdot 10^4$, $3 \cdot 10^4$ and $2 \cdot 10^4$ selection steps for the kernel thresholds 10^{-4}, 10^{-3} and $5 \cdot 10^{-3}$, respectively, on the audio test signal. For the noise signal, resets were performed after $7 \cdot 10^6$, $2 \cdot 10^6$, 10^6 and $8 \cdot 10^5$ selection steps for thresholds 10^{-4}, 10^{-3}, $5 \cdot 10^{-3}$ and 10^{-3}, respectively. In all tested cases, the error of the MP approximation follows the same curve, up to deviations of ± 1 dB for the audio signal and ± 0.1 dB for the noise signal (below $6 \cdot 10^6$ selections). Up to 10^6 selection steps (the tested range), approximation quality achieved by MPTK follows the same curve. Note that both signals are $6.04 \cdot 10^6$ samples long, such that performing more than $6 \cdot 10^6$ selection steps is unlikely to be practically relevant. With appropriate reset criteria, the same trend was observed when testing other signals.

7 USING THE C MODULE

Using the Large Time-Frequency Analysis Toolbox (LTFAT, http://ltfat.github.io), we provide the interface `multidgtrealmp.m` for using the proposed matching pursuit implementation from Matlab/GNU Octave.

```
c = multidgtrealmp(f,dicts,errdb,maxit)
```

Computes the MP decomposition `c` of input vector `f` with respect to multi-Gabor dictionary `dicts` given as cell array of triplets `{g,a,M,...}` specifying the Gabor dictionary using the supported window type `g`, hop size `a` and `M` frequency channels. The window length will be set to `M`. The optional parameters `errdb` and `maxit` are used to specify the stopping criterion in terms of the targeted residual energy in dB and the maximum number of selection steps.

The key-value pair `'kernthr'`,ϵ can be used to set the kernel threshold ϵ. The default value is 10^{-4}.

The flag `'reset'` activates resets. By default, the inner stopping criterion is derived from the kernel threshold ϵ and given by a maximum number of selections per reset and a tolerance for the relative error estimate E_k/E^{out}. Reset conditions can be set manually using the key-value pairs `'resetit'`, `it` and `'reseterrdb'`, `err`, where `it` is the maximum number of iterations per reset and the latter triggers a reset when `err` $> 10\log_{10}(E_k/E^{\mathrm{out}})$.

Further options are detailed in the documentation of `multidgtrealmp.m`.

Additionally, Matlab/GNU Octave functions for performing the experiments in Section 6.1 are supplied at https://ltfat.github.io/notes/ltfatnote052_convergence.zip. The results obtained by MPTK are precomputed, supplied as a numerical vector and do not need to be recomputed.

To demonstrate the usage of the `dgtrealmp` module of LIBLTFAT in C or C++ directly, we provide the example implementations `multigabormpd.cpp` at https://github.com/ltfat/libltfat/tree/master/examples/multigabormp. An overview of the module can be found at http://ltfat.github.io/libltfat/group__multidgtrealmp. The general workflow is as follows.

1) Initialize the parameter setup structure:
```
int dgtrealmp_parbuf_init(dgtrealmp_parbuf **p)
```

2) Add dictionaries to the setup structure using
```
int dgtrealmp_parbuf_add_firwin(dgtrealmp_parbuf *parbuf, LTFAT_FIRWIN win, ltfat_int gl, ltfat_int
a, ltfat_int M)
```
Here, `win` is a supported window type, `gl` is the window length and `a`, `M` are the hop size and number of frequency channels of the Gabor dictionary, respectively.

3) Optionally change MP parameters using the functions
```
int dgtrealmp_setparbuf_...(dgtrealmp_parbuf *parbuf, ...)
```
By default, the algorithm targets an error estimate $10\log_{10}(E_k/\|\mathbf{x}\|^2) \leq -40$ with at most $L/5$ selections at relative kernel threshold 10^{-4}. Examples:
```
int dgtrealmp_setparbuf_maxit(dgtrealmp_parbuf *parbuf, size_t maxit)
int dgtrealmp_setparbuf_errtoldb(dgtrealmp_parbuf *parbuf, double errtoldb)
int dgtrealmp_setparbuf_kernrelthr (dgtrealmp_parbuf *parbuf, double thr)
```
See http://ltfat.github.io/libltfat/group__multidgtrealmp for more options.

4) Initialize MP state for fixed input length L:
```
int dgtrealmp_init(dgtrealmp_parbuf *pb, ltfat_int L, dgtrealmp_state **p)
```
Precomputes the truncated kernels and cross-kernels, as well as all other data necessary to perform the MP decomposition.

5) Compute MP approximation for input vector f:

```
int dgtrealmp_execute(dgtrealmp_state *p, const LTFAT_REAL f[], LTFAT_COMPLEX *cout[], LTFAT_REAL
fout[])
```

The outputs `cout` and `fout` represent the final MP approximation in the coefficient and signal domains, respectively. This step may be repeated at will for different input vectors of the same length L.

6) Clean up:

```
int dgtrealmp_done(dgtrealmp_state **p)
```

The function `dgtrealmp_execute` computes the MP decomposition `cout` and synthesizes the signal domain approximation `fout`. The current implementation only supports resets via the Matlab/GNU Octave interface. In a future update, the reset procedure will be implemented directly into the C module.

8 CONCLUSION AND OUTLOOK

We have presented an accelerated MP algorithm alongside a reference implementation suitable for multi-Gabor dictionaries. Due to the structure of the Gram matrix of the multi-Gabor dictionary, the coefficient domain residual update step becomes very fast while the memory requirements for storing the inner products between the atoms remain constant with increasing signal length. Moreover, the time and frequency locality of the residual update in turn allows faster search for the maximum in the next iteration. We have shown that the proposed method converges to the true solution, if a simple reset procedure is occasionally performed. Benchmarks show that, depending on the dictionary, our implementation is 3.5–70 times faster than the standard MPTK implementation. In the single dictionary case, the most notable feature is that the execution time is virtually independent of the number of bins M when the redundancy M/a is fixed. Moreover, as it turned out, MPTK could not handle dictionaries with the number of bins higher than 8192. In our code, no explicit optimization techniques like exploiting the SIMD operations or parallelization of the code were used, therefore it is possible that there is still room for improvement.

Since the presented acceleration technique applies only to the update step of the algorithm, it is also applicable to various extensions of MP like the molecular MP [6], perceptual MP [5, 22], and guided MP [47] etc. Extensions to the Local OMP [25, 26], cyclic MP [41, 42], self-projected MP [36] and to the complementary MP [33, 34] seem to be possible as well. Future work may investigate the suitability of the proposed implementation to sliding local MP [7] and real-time MP mentioned in [47]. We will also investigate the structure of the Gram matrix of a dictionary consisting of an ensemble of real-valued windowed discrete cosine bases used by Ravelli et. al. [35] to determine whether similar acceleration technique is feasible.

ACKNOWLEDGMENT

REFERENCES

[1] P. Balazs. 2008. Frames and finite dimensionality: Frame transformation, classification and algorithms. *Applied Mathematical Sciences* 2, 41–44 (2008), 2131–2144.

[2] G. Bhattacharya and P. Depalle. 2014. Sparse denoising of audio by greedy time-frequency shrinkage. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. 2898–2902.

[3] T. Blumensath and M. E. Davies. 2008. Gradient pursuit for non-linear sparse signal modelling. In *Proc. European Signal Processing Conference (EUSIPCO)*. IEEE, 1–5.

[4] T. Blumensath and M. E. Davies. 2008. Gradient pursuits. *IEEE Tran. Signal Processing* 56, 6 (2008), 2370–2382.

[5] G. Chardon, T. Necciari, and P. Balazs. 2014. Perceptual matching pursuit with Gabor dictionaries and time-frequency masking. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. 3102–3106.

[6] L. Daudet. 2006. Sparse and structured decompositions of signals with the molecular matching pursuit. *IEEE Tran. Audio, Speech, and Language Processing* 14, 5 (Sept 2006), 1808–1816.

[7] L. Daudet. 2010. Audio sparse decompositions in parallel: Let the greed be shared! *IEEE Signal Processing Magazine* 27, 2 (March 2010), 90–96.

[8] G. Davis, S. Mallat, and M. Avellaneda. 1997. Adaptive greedy approximations. *Constructive Approximation* 13, 1 (1997), 57–98.

[9] G. M. Davis, S. G. Mallat, and Z. Zhang. 1994. Adaptive time-frequency decompositions. *Optical Engineering* 33, 7 (1994), 2183–2191.

[10] O. Derrien. 2007. Time-scaling of audio signals with muti-scale Gabor analysis. In *Proc. Int. Conf. Digital Audio Effects (DAFx–07)*. Bordeaux, France.

[11] R. A. DeVore and V. N. Temlyakov. 1996. Some remarks on greedy algorithms. *Advances in Computational Mathematics* 5, 1 (01 Dec 1996), 173–187.

[12] P. Durka. 2007. *Matching pursuit and unification in EEG analysis.* Artech House, Inc. 184 pages.

[13] S. E. Ferrando, L. A. Kolasa, and N. Kovačević. 2002. Algorithm 820: A flexible implementation of matching pursuit for Gabor functions on the interval. *ACM Trans. Math. Softw.* 28, 3 (Sept. 2002), 337–353.

[14] M. Frigo and S. G. Johnson. 2005. The design and implementation of FFTW3. *Proc. of the IEEE* 93, 2 (2005), 216–231. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[15] R. Gribonval. 2001. Fast matching pursuit with a multiscale dictionary of Gaussian chirps. *IEEE Tran. Signal Processing* 49, 5 (May 2001), 994–1001.

[16] R. Gribonval. 2002. Sparse decomposition of stereo signals with matching pursuit and application to blind separation of more than two sources from a stereo mixture. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Vol. 3. III–3057–III–3060.

[17] R. Gribonval, P. Depalle, X. Rodet, E. Bacry, and S. Mallat. 1996. Sound signals decomposition using a high resolution matching pursuit. In *Proc. Int. Computer Music Conf. (ICMC'96)*. 293–296.

[18] R. Gribonval, R. M. Figueras i Ventura, and P. Vandergheynst. 2006. A simple test to check the optimality of a sparse signal approximation. *Signal Processing* 86, 3 (2006), 496 – 510.

[19] R. Gribonval and P. Vandergheynst. 2006. On the exponential convergence of matching pursuits in quasi-incoherent dictionaries. *IEEE Tran. Information Theory* 52, 1 (Jan 2006), 255–261.

[20] K. Gröchenig. 2001. *Foundations of time-frequency analysis.* Birkhäuser, Boston, Basel, Berlin.

[21] S. Krstulović and R. Gribonval. 2006. MPTK: Matching pursuit made tractable. In *Proc. Int. Conf. on Acoustics Speech and Signal Processing ICASSP 2006*, Vol. 3. III–496–III–499.

[22] H. Lahdili, H. Najaf-Zadeh, R. Pichevar, and L. Thibault. 2008. Perceptual matching pursuit for audio coding. In *Audio Engineering Society Convention 124*.

[23] J. Le Roux, H. Kameoka, N. Ono, and S. Sagayama. 2010. Fast signal reconstruction from magnitude STFT spectrogram based on spectrogram consistency. In *Proc. 13th Int. Conf. on Digital Audio Effects (DAFx-10)*. 397–403.

[24] P. Leveau and L. Daudet. 2006. Multi-resolution partial tracking with modified matching pursuit. In *Proc. 14th European Signal Processing Conference*. 1–4.

[25] B. Mailhé, R. Gribonval, F. Bimbot, and P. Vandergheynst. 2009. A low complexity orthogonal matching pursuit for sparse signal approximation with shift-invariant dictionaries. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICAASP)*. 3445–3448.

[26] B. Mailhé, R. Gribonval, P. Vandergheynst, and F. Bimbot. 2011. Fast orthogonal sparse approximation algorithms over local dictionaries. *Signal Processing* 91, 12 (2011), 2822 – 2835.

[27] Stphane Mallat. 2008. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way* (3rd ed.). Academic Press.

[28] S. G. Mallat and Z. Zhang. 1993. Matching pursuits with time-frequency dictionaries. *IEEE Tran. Signal Processing* 41, 12 (Dec 1993), 3397–3415.

[29] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. 1993. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proc. 27th Asilomar Conference on Signals, Systems and Computers*. 40–44 vol.1.

[30] M. D. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. E. Davies. 2010. Sparse representations in audio and music: From coding to source separation. *Proc. IEEE* 98, 6 (June 2010), 995–1005.

[31] M. R. Portnoff. 1976. Implementation of the digital phase vocoder using the fast Fourier transform. *IEEE Tran. Acoustics, Speech, and Signal Processing* 24, 3 (Jun 1976), 243–248.

[32] Z. Průša, P. L. Søndergaard, N. Holighaus, Ch. Wiesmeyr, and P. Balazs. 2014. The large time-frequency analysis toolbox 2.0. In *Sound, Music, and Motion*. Springer International Publishing, 419–442.

[33] G. Rath and C. Guillemot. 2008. A complementary matching pursuit algorithm for sparse approximation. In *Proc. 16th European Signal Processing Conference (EUSIPCO)*. 1–5.

[34] G. Rath and Ch. Guillemot. 2010. On a simple derivation of the complementary matching pursuit. *Signal Processing* 90, 2 (2010), 702 – 706.

[35] E. Ravelli, G. Richard, and L. Daudet. 2008. Union of MDCT bases for audio coding. *IEEE Tran. Audio, Speech, and Language Processing* 16, 8 (Nov 2008), 1361–1372.

[36] L. Rebollo-Neira, M. Rozložník, and P. Sasmal. 2017. Analysis of a low memory implementation of the orthogonal matching pursuit greedy strategy. *CoRR* abs/1609.00053v2 (2017).

[37] I. Rish and G. Grabarnik. 2015. *Sparse modeling: Theory, algorithms, and applications*. CRC Press. 253 pages.

[38] P. L. Søndergaard. 2012. Efficient algorithms for the discrete Gabor transform with a long FIR window. *J. Fourier Anal. Appl.* 18, 3 (2012), 456–470.

[39] P. L. Søndergaard, B. Torrésani, and P. Balazs. 2012. The linear time frequency analysis toolbox. *International Journal of Wavelets, Multiresolution Analysis and Information Processing* 10, 4 (2012), 1–27.

[40] B. L. Sturm. 2009. *Sparse approximation and atomic decomposition: Considering atom Interactions in evaluating and building signal representations*. Ph.D. Dissertation. University of California.

[41] B. L. Sturm and M. G. Christensen. 2010. Cyclic matching pursuits with multiscale time-frequency dictionaries. In *Conf. Record of the 44th Asilomar Conference on Signals, Systems and Computers*. 581–585.

[42] B. L. Sturm, M. G. Christensen, and R. Gribonval. 2011. Cyclic pure greedy algorithms for recovering compressively sampled sparse signals. In *Conference Record of the 45th Asilomar Conference on Signals, Systems and Computers*. IEEE, 1143–1147.

[43] B. L. Sturm, L. Daudet, and C. Roads. 2006. Pitch-shifting audio signals using sparse atomic approximations. In *Proc. 1st ACM Workshop on Audio and Music Computing Multimedia* (Santa Barbara, California, USA) *(AMCMM '06)*. ACM, New York, NY, USA, 45–52.

[44] B. L. Sturm and J. D. Gibson. 2006. Matching pursuit decompositions of non-noisy speech signals using several dictionaries. In *Proc. IEEE Int. Conf. on Acoustics Speech and Signal Processing*, Vol. 3. III–III.

[45] The European Broadcasting Union. 2008. *Tech 3253: Sound quality assessment material recordings for subjective tests*. Technical Report. Geneva. https://tech.ebu.ch/docs/tech/tech3253.pdf

[46] M. Yaghoobi and M. E. Davies. 2009. *Fast and scalable: A survey on sparse approximation methods*. Technical Report. The University of Edinburgh.

[47] D. Zantalis. 2016. *Guided matching pursuit and its application to sound source separation*. Ph.D. Dissertation. University of York.

A PROOFS

A.1 Proof of Theorem 1

We begin with some preparation. Recall that $\mathbf{r}_k = \mathbf{x} - \mathbf{D}\mathbf{c}_k$. By linearity, $\mathbf{r}_{k+1} = \mathbf{r}_k - \mathbf{D}(\mathbf{c}_{k+1} - \mathbf{c}_k) = \mathbf{r}_k - \varrho_k(p_{k+1})\mathbf{d}_{p_{k+1}}$.

Further, we have $\mathbf{D}^*\mathbf{r}_k = \varrho_0 - \mathbf{G}\mathbf{c}_k$ and $\varrho_k = \varrho_0 - \mathbf{G}_\epsilon\mathbf{c}_k$. With $\rho_k := \sum_{l=1}^{k} |\varrho_{l-1}(p_l)|$, it is easily verified that $|(\mathbf{G} - \mathbf{G}_\epsilon)\mathbf{c}_k|(p) \leq \epsilon\rho_k$, for all p, and consequently, $|\langle\mathbf{r}_k, \mathbf{d}_p\rangle - \varrho_k(p)| \leq \epsilon\rho_k$ as well.

By assumption, $\epsilon\rho_k \leq \delta|\varrho_k(p_{k+1})|$, with $0 < \delta < 1/2$. We have

$$\|\mathbf{r}_k\|_2^2 - |\langle\mathbf{r}_k, \mathbf{d}_{p_{k+1}}\rangle|^2 \leq \|\mathbf{r}_k\|_2^2 - (1-\delta)^2|\varrho_k(p_{k+1})|^2 \leq \|\mathbf{r}_k\|_2^2. \tag{21}$$

Using the triangle inequality and (21), we obtain

$$\begin{aligned}
\|\mathbf{r}_k - \varrho_k(p_{k+1})\mathbf{d}_{p_{k+1}}\|_2^2 &\leq (\|\mathbf{r}_k - \langle\mathbf{r}_k, \mathbf{d}_{p_{k+1}}\rangle\mathbf{d}_{p_{k+1}}\|_2 + \epsilon\rho_k)^2 \\
&= \|\mathbf{r}_k - \langle\mathbf{r}_k, \mathbf{d}_{p_{k+1}}\rangle\mathbf{d}_{p_{k+1}}\|_2^2 + 2\epsilon\rho_k\|\mathbf{r}_k - \langle\mathbf{r}_k, \mathbf{d}_{p_{k+1}}\rangle\mathbf{d}_{p_{k+1}}\|_2 + \epsilon^2\rho_k^2 \\
&\leq \|\mathbf{r}_k\|_2^2\left(1 - \frac{(1-\delta)^2|\varrho_k(p_{k+1})|^2}{\|\mathbf{r}_k\|_2^2} + \frac{\epsilon^2\rho_k^2}{\|\mathbf{r}_k\|_2^2} + \frac{2\epsilon\rho_k}{\|\mathbf{r}_k\|_2}\right).
\end{aligned}$$

Hence,

$$\frac{\epsilon^2 \rho_k^2 + 2\epsilon \rho_k \|\mathbf{r}_k\|_2}{(1-\delta)^2} \leq \frac{\delta^2 |\varrho_k(p_{k+1})|^2 + 2\delta |\varrho_k(p_{k+1})| \|\mathbf{r}_k\|_2}{(1-\delta)^2} < |\varrho_k(p_{k+1})|^2$$

ensures that $\|\mathbf{r}_{k+1}\|_2 < \|\mathbf{r}_k\|_2$. The latter inequality is equivalent to

$$\frac{2\delta}{1-2\delta} \|\mathbf{r}_k\|_2 < |\varrho_k(p_{k+1})|.$$

□

A.2 Proof of Theorem 2

It is clear that there exist δ and ε such that $\frac{2\delta}{1-(2\delta+\varepsilon)} < \sqrt{\lambda_{\min}}$. With such a choice, we have that

$$\max_p |\langle \mathbf{y}, \mathbf{d}_p \rangle|^2 \geq \lambda_{\min} \|\mathbf{y}\|_2^2 > \left(\frac{2\delta}{1-(2\delta+\varepsilon)}\right)^2 \|\mathbf{y}\|_2^2. \tag{22}$$

First, assume that $k = 0$ or either of the stopping conditions in (11) was met in the $(k+1)$-th selection step and ϱ_k is re-initialized with $\varrho_k := \mathbf{D}^*(\mathbf{x} - \mathbf{Dc}_k) = \mathbf{D}^* \mathbf{r}^{\mathrm{out}} = \mathbf{D}^* \mathbf{r}_k$. In other words, $\varrho_k(p) = \langle \mathbf{r}_k, \mathbf{d}_p \rangle$, for all p and we have $k^{\mathrm{out}} = k$. Furthermore, $p_{k+1} = \operatorname{argmax}_p |\varrho_k(p)| = \operatorname{argmax}_p |\langle \mathbf{r}_k, \mathbf{d}_p \rangle|$. With this, the right-hand side of the first inequality in (11) equals 0, such that it is trivially not satisfied. On the other hand, the second inequality in (11) reduces to

$$\max_p |\langle \mathbf{r}_k, \mathbf{d}_p \rangle| < \frac{2\delta(1 - \varepsilon(1+\delta)^{-2}\lambda_{\min})^{\frac{k^{\mathrm{out}}-k}{2}}}{1-(2\delta+\varepsilon)} \|\mathbf{r}^{\mathrm{out}}\|_2 = \frac{2\delta}{1-(2\delta+\varepsilon)} \|\mathbf{r}_k\|_2 < \lambda_{\min} \|\mathbf{r}_k\|_2,$$

contradicting the definition of $\lambda_{\min}$.

Hence, we can assume at every selection, that neither condition in (11) is satisfied. Therefore, for every $k \geq 0$, and with $\rho_k := \sum_{l=k^{\mathrm{out}}+1}^{k} |\varrho_{l-1}(p_l)|$ (note that k^{out} depends on k), we have

$$(1+\delta)|\varrho_k(p_{k+1})| \geq |\varrho_k(p_{k+1})| + \epsilon \rho_k \geq \max_p |\langle \mathbf{r}_k, \mathbf{d}_p \rangle| \geq \sqrt{\lambda_{\min}} \|\mathbf{r}_k\|_2.$$

Although k^{out} depends on k, the estimate $(1+\delta)|\varrho_k(p_{k+1})| \geq \sqrt{\lambda_{\min}} \|\mathbf{r}_k\|_2$ does not. Clearly, for $l = 0$, the inequality $|\varrho_l(p_{l+1})| \geq \frac{2\delta}{1-(2\delta+\varepsilon)} \|\mathbf{r}_l\|_2$ holds by (22). Assume that it holds for for all $0 \leq l < k$, for some fixed $k \geq 0$. Then, similar to the proof of Theorem 1

$$\epsilon^2 \rho_l^2 + 2\epsilon \rho_l \|\mathbf{r}_l\|_2 - (1-\delta)^2 |\varrho_l(p_{l+1})|^2 \leq \delta^2 |\varrho_l(p_{l+1})|^2 + 2\delta |\varrho_l(p_{l+1})| \|\mathbf{r}_l\|_2 - (1-\delta)^2 |\varrho_l(p_{l+1})|^2$$
$$\leq -\varepsilon |\varrho_l(p_{l+1})|^2,$$

such that

$$\|\mathbf{r}_{l+1}\|_2^2 = \|\mathbf{r}_l - \varrho_l(p_{l+1}) \mathbf{d}_{p_{l+1}}\|_2^2 \leq \left(1 - \varepsilon \frac{|\varrho_l(p_{l+1})|^2}{\|\mathbf{r}_l\|_2^2}\right) \|\mathbf{r}_l\|_2^2 \leq \left(1 - \varepsilon \frac{\lambda_{\min}}{(1+\delta)^2}\right) \|\mathbf{r}_l\|_2^2.$$

With $\mathbf{r}_0 = \mathbf{x}$, we obtain

$$\|\mathbf{r}_k\|_2 \leq \left(1 - \varepsilon(1+\delta)^{-2}\lambda_{\min}\right)^{\frac{k^{\mathrm{out}}-k}{2}} \|\mathbf{r}^{\mathrm{out}}\|_2 \leq \left(1 - \varepsilon(1+\delta)^{-2}\lambda_{\min}\right)^{k/2} \|\mathbf{x}\|_2. \tag{23}$$

Since the second inequality in (11) is not satisfied, the (23) implies

$$|\varrho_k(p_{k+1})| \geq \frac{2\delta \|\mathbf{r}_k\|_2}{1-(2\delta+\varepsilon)},$$

completing the induction step and thus the proof.

□

A.3   Proof of Corollary 3

Under the conditions of the corollary, we can invoke Theorem 2. The result easily follows by considering (23) in the proof of Theorem 2 and noting that $k > \frac{\log(E)}{\log(1-\varepsilon(1+\delta)^{-2}\lambda_{\min})}$ implies $(1 - \varepsilon(1 + \delta)^{-2}\lambda_{\min})^k < E$. $\qquad\square$