

1 **SPARSE MATRIX FACTORIZATIONS FOR FAST LINEAR SOLVERS**  
2 **WITH APPLICATION TO LAPLACIAN SYSTEMS\***

3 MICHAEL T. SCHAUB <sup>†</sup>, MAGUY TREFOIS <sup>‡</sup>, PAUL VAN DOOREN <sup>§</sup>, AND  
4 JEAN-CHARLES DELVENNE <sup>¶</sup>

5 **Abstract.** In solving a linear system with iterative methods, one is usually confronted with the  
6 dilemma of having to choose between cheap, inefficient iterates over sparse search directions (e.g.,  
7 coordinate descent), or expensive iterates in well-chosen search directions (e.g., conjugate gradients).  
8 In this paper, we propose to interpolate between these two extremes, and show how to perform  
9 cheap iterations along non-sparse search directions, provided that these directions can be extracted  
10 from a new kind of sparse factorization. For example, if the search directions are the columns of a  
11 hierarchical matrix, then the cost of each iteration is typically logarithmic in the number of variables.  
12 Using some graph-theoretical results on low-stretch spanning trees, we deduce as a special case a  
13 nearly-linear time algorithm to approximate the minimal norm solution of a linear system  $Bx = b$   
14 where  $B$  is the incidence matrix of a graph. We thereby can connect our results to recently proposed  
15 nearly-linear time solvers for Laplacian systems, which emerge here as a particular application of our  
16 sparse matrix factorization.

17 **Key word.** matrix factorization, linear system, Laplacian matrix, iterative algorithms, sparsity,  
18 hierarchical matrices

19 **AMS subject classifications.** 15A06, 15A23, 15A24

20 **1. Introduction.** Finding solutions of large linear systems of equations is a  
21 fundamental issue, underpinning most areas of mathematical sciences and quantitative  
22 research. For instance, consider partial differential equations arising in various areas  
23 of physics, mechanics and electro-magnetics. These have commonly to be solved  
24 numerically, and a spatial discretization of such a problem naturally leads to solving  
25 a large sparse or structured linear system [29].

26 In principle, two strategies to solve linear systems exist. First, there are *direct*  
27 *methods* [7] like Cholesky factorization or Gaussian elimination. Those methods pro-  
28 vide a (numerically) exact solution of the system by performing a finite number of  
29 computations. However, these algorithms can be computationally expensive, in par-  
30 ticular as the full set of computations has always to be performed to obtain a problem  
31 solution, even if a coarser approximation thereof would be sufficient.

32 A second strategy is to use *iterative methods* [8, 22, 29], such as the Jacobi method  
33 or gradient descent. Unlike for direct methods, the result after every step of an  
34 iterative algorithm may be interpreted as an approximate solution to the problem,  
35 which keeps getting improved until a desired stopping criterion, e.g., a predefined  
36 precision, is reached. As in practice the specification of the system to be solved

---

\*We thank Raf Vandebril and Francois Glineur for fruitful discussions and comments. This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimisation), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, and the ARC (Action de Recherche Concertée) on Mining and Optimization of Big Data Models funded by the Wallonia-Brussels Federation.

<sup>†</sup>ICTEAM, Université catholique de Louvain & naXys, Université de Namur, Belgium ([mschaub@mit.edu](mailto:mschaub@mit.edu)). Equally contributing first author; present address: Institute for Data, Systems, and Society, Massachusetts Institute of Technology.

<sup>‡</sup>ICTEAM, Université catholique de Louvain, Belgium ([maguy.trefois@uclouvain.be](mailto:maguy.trefois@uclouvain.be)). Equally contributing first author

<sup>§</sup>ICTEAM, Université catholique de Louvain, Belgium ([paul.vandooren@uclouvain.be](mailto:paul.vandooren@uclouvain.be)).

<sup>¶</sup>ICTEAM and CORE, Université catholique de Louvain, Belgium ([jean-charles.delvenne@uclouvain.be](mailto:jean-charles.delvenne@uclouvain.be)).

37 is hardly ever exact, this ability to stop at suitable approximate solutions renders  
 38 iterative methods generally less costly in terms of running time. For instance, the  
 39 complexity of direct Gaussian elimination for a system of size  $n$  is  $\mathcal{O}(n^3)$ . In contrast,  
 40 the iterative Jacobi method takes only  $\mathcal{O}(Nn^2)$  time. Here,  $N$  is the number of  
 41 iterations needed, which can usually be kept small.

42 However, when the system size  $n$  is very large, effectively all classical direct and  
 43 iterative methods become computationally prohibitive, unless the matrix is known  
 44 to have a special structure (banded, Toeplitz, semiseparable, etc.). Methods which  
 45 provide faster means for solving linear systems are thus highly demanded.

46 **1.1. Background and Related work.** The success of any iterative update  
 47 scheme in solving a linear system depends on two intertwined factors. On the one  
 48 hand, we would like to design our iterations such that each update brings us as close  
 49 as possible to the true solution. On the other hand, we would like to make each  
 50 iteration computationally as cheap as possible.

51 Let us initially consider the first of these two objectives here. Trivially, the update  
 52 that would bring us closest to the true solution entails finding the correct solution  
 53 directly, and thus requires only one iteration. However, this is clearly not feasible, if  
 54 our initial problem evaded direct solution methods. A more realistic scheme, aiming  
 55 to bring us as close as possible to the desired solution would be conjugate gradient  
 56 descent, which tries to find good search directions at each step using gradient infor-  
 57 mation. The downside of an approach like gradient descent is that each step can be  
 58 computationally very costly, e.g., as in general all coordinates have to be updated at  
 59 each step.

60 This brings us back to the second objective mentioned above: making each itera-  
 61 tion as computationally cheap as possible. On this end of the methodological spectrum  
 62 there are approaches like (canonical) coordinate descent. Here the idea is to keep the  
 63 updates very sparse and only update one (or a small number of  $k$ ) coordinates at  
 64 a time, thereby facilitating cheap iterations. However, as this imposes quite strong  
 65 restrictions on the allowed search directions, this results in general in a large number  
 66 of iterations needed, possibly outweighing the gain in computational complexity for  
 67 each iteration.

68 Recently, Spielman and Teng [23] provided a seminal contribution and showed  
 69 that one can construct iterative algorithms to solve symmetric, diagonally dominant  
 70 (SDD) systems in nearly-linear running time. Here, *nearly-linear* refers to a com-  
 71 plexity of the form  $\mathcal{O}(\ell \log^c \ell \log(\varepsilon^{-1}))$ , where  $\ell$  is the number of nonzero entries in  
 72 the system matrix,  $c$  is an arbitrary positive constant, and  $\varepsilon$  is a desired accuracy  
 73 to be reached. These results have been further improved and simplified in the last  
 74 decade [6, 15–18, 20], and there is now a substantial literature on solving SDD systems  
 75 effectively in nearly-linear time. Interestingly, all these algorithms follow essentially  
 76 the same paradigm. The problem is first reduced to solving a system of the form  
 77  $Lx = b$ , where  $L$  is the Laplacian matrix of an undirected graph. The Laplacian  
 78 system is then solved efficiently using graph theoretic techniques.

79 **1.2. Main contributions.** We provide a sparse matrix factorization that en-  
 80 ables the construction of fast iterative algorithms. Namely, using our  $k$ -sparse matrix  
 81 factorization allows for cheap iterative updates in efficient directions.

82 The key question we address is in how far cheap, coordinate descent like updates  
 83 can also be performed in more flexible search directions. As we show in the following  
 84 the answer is indeed affirmative. If the iterative updates are performed along direc-  
 85 tions  $q_i$  that can be assembled into a  $k$ -sparse decomposable matrix  $Q = [q_1, \dots, q_n]$ ,

86 then we can always perform cheap iterative updates, despite the fact that the search  
 87 direction may not have sparse support, i.e.,  $Q$  might be a dense matrix. This signifi-  
 88 cantly enlarges the array of possible search directions and paves the way for efficient  
 89 algorithms that can benefit from both cheap updates and well-chosen search direc-  
 90 tions.

91 Remarkably our  $k$ -sparse factorization is applicable for a variety of matrices with  
 92 seemingly disparate structures. In particular, we can design iterative algorithms for  
 93 sparse, hierarchical, semiseparable, or Laplacian matrices, with a complexity similar  
 94 to specially tailored algorithm for those respective classes. In the case of Laplacian  
 95 systems (and therefore all SDD systems through the usual reduction), our approach  
 96 differs from previous work in that we take a different, matrix-theoretic approach,  
 97 rather than relying purely on graph-theoretic machinery to achieve a nearly-linear  
 98 complexity. Finally, we show that this algorithm can be applied to solve Laplacian  
 99 systems in nearly linear time, thereby establishing a connection to the previous lit-  
 100 erature. Rather than emphasizing one particular application and providing detailed  
 101 simulations for our algorithms, the focus of the present paper is on the theoretical  
 102 development of a new sparse matrix factorization and its algebraic properties, which  
 103 may then be used in different contexts.

104 Note that both sparse and dense systems are in principle amenable for a  $k$ -sparse  
 105 decomposition. Therefore, in principle, the target systems for our  $k$ -sparse matrix  
 106 factorization and the associated iterative solution strategy may be dense or sparse.  
 107 For instance, Laplacian systems, which serve as our final application example in this  
 108 paper, are typically sparse systems. Nevertheless, the theory developed is equally  
 109 applicable to dense systems as will become apparent when discussing hierarchical ma-  
 110 trices. Of course, in the case of very large dense systems, one may have to find efficient  
 111 representations or approximations for storing such data (e.g., using hierarchical ma-  
 112 trices [12, 13], or semiseparable matrices [27, 28]). This is a challenge in its own right,  
 113 not addressed in the present manuscript.

114 **1.3. Outline of the paper.** In Section 2, we first review some preliminaries  
 115 for iteratively solving linear systems and set up some notation. In Section 3, we then  
 116 motivate and define our  $k$ -sparse matrix factorization. We highlight some properties of  
 117 this factorization and show how it enables an iteration of the form (2) to be computed  
 118 in  $\mathcal{O}(k)$  time. We then discuss, how these cheap iterations can be utilized to construct  
 119 fast iterative solvers for linear systems. In Section 4, we review several examples of  $k$ -  
 120 sparsely factorizable matrices, including some sparse matrices, hierarchical matrices,  
 121 semi-separable matrices, as well as the incidence matrices of trees. Of particular  
 122 interest here are hierarchical matrices [4, 11, 12], which are an example of  $k$ -sparse  
 123 factorizable matrices for which  $k$  does not depend on the size of the matrix. In  
 124 Section 5, we present fast iterative solvers for systems of hierarchical matrices, based  
 125 on  $k$ -sparse decompositions. In Section 6 we then show how similar techniques can be  
 126 applied if the system matrix is the incidence matrix of a graph, and how this naturally  
 127 leads to an algorithm for solving a Laplacian system in nearly-linear time. Section  
 128 7 concludes the paper and discusses possible avenues for future work. To improve  
 129 readability, some technical proofs are reported in the appendix.

130 **2. Preliminaries.** For simplicity of notation we will consider only real vectors  
 131 and matrices, although generalizations to the complex case are straightforward. In  
 132 the sequel, the index variable  $t$  will be reserved to denote the  $t$ -th iterate of a vector  
 133 ( $x$ , or  $y$  respectively). Otherwise, an indexed vector  $v_i$  is to be interpreted as the  
 134  $i$ th column vector of a set of column vectors (usually associated with a corresponding

135 matrix  $V = [v_1, v_2, \dots]$ .

136 From an abstract point of view, we consider the problem of finding the minimal  
 137 norm vector  $x$  within an affine space  $\mathcal{X}$ . Let  $v \in \mathcal{X}$  be any point in our affine space.  
 138 Then by updating  $x$  within this search space along a set  $\{q_i\}$  of chosen search directions  
 139 spanning  $\mathcal{X} - v$ , one can find the minimal norm solution of  $x$ . More precisely, starting  
 140 from an  $x_0 \in \mathcal{X}$  we iteratively solve:

$$141 \quad (1) \quad \min \|x\|$$

$$142 \quad \text{s.t. } x - v \in \text{span}(\{q_i\}).$$

144 As we review in next section, this problem is closely connected to iteratively solving  
 145 a linear system, and the natural updates are of the form:

$$146 \quad (2) \quad x_{t+1} = x_t - \frac{x_t^T q_i}{q_i^T q_i} q_i.$$

147 The goal of this work is to show that if the search directions for problem (1)  
 148 are such that they correspond to the columns  $q_i$  of a matrix  $Q$  that is  $k$ -sparsely  
 149 factorizable, then all iterative updates of the form (2) can be performed in  $\mathcal{O}(k)$   
 150 time. Here  $k$  is usually much smaller than the dimension of the search space, thereby  
 151 facilitating fast iterative updates schemes, as we will see in the subsequent sections.

152 **2.1. Underdetermined systems.** Given a compatible linear system  $Ax = b$ ,  
 153 we are looking for the optimal solution of the following optimization problem:

$$154 \quad (3) \quad \min \|x\|$$

$$155 \quad \text{s.t. } Ax = b,$$

157 where  $\|x\| := \sqrt{x^T x}$ . We denote this optimal solution by  $x^*$ :

$$158 \quad (4) \quad x^* := \arg \min_{s.t. Ax=b} \|x\|,$$

159 This problem can be readily solved as follows. Suppose we are given a matrix  $Q$   
 160 where the columns  $q_i$  form a basis of the null space,  $\text{null}(A)$ , of  $A$ . If  $x_0$  is a feasible  
 161 solution to  $Ax = b$ , we can write (4) as

$$162 \quad (5) \quad x^* := \arg \min_{s.t. x=x_0+Qy} \|x\|,$$

163 for some unknown vector  $y$ . Consequently, we may compute increasingly accurate  
 164 approximations of  $x^*$  by iteratively updating  $x$  according to:

$$165 \quad (6) \quad x_{t+1} = x_t + \alpha_t^* q_i \quad \text{with} \quad \alpha_t^* = \arg \min_{\alpha_t \in \mathbb{R}} \|x_t + \alpha_t q_i\| = -\frac{x_t^T q_i}{q_i^T q_i}.$$

166 Thus each iteration is of the form (2). We remark that these updates may be inter-  
 167 preted in the context of a (randomized) Kaczmarz scheme as discussed in the Appendix.  
 168 If we start with a feasible solution  $x_0$ , each iterate  $x_t$  is an exact solution of  $Ax = b$ ,  
 169 since all updates added to  $x_0$  are in the null space of  $A$ . Therefore, the above iterative  
 170 method converges to the optimal  $x^*$ .

171 **2.2. Overdetermined and square systems.** Iteration (2) also appears natu-  
 172 rally when iteratively solving an overdetermined system:

$$173 \quad (7) \quad \arg \min_y \|Ay - b\|.$$

174 By simply making the substitution  $x = Ay - b$ , we can transform the above into the  
 175 equivalent problem:

$$176 \quad (8) \quad \min \|x\|$$

$$177 \quad \text{s.t. } x + b \in \text{Im}(A),$$

i.e., we are again trying to find the minimum norm solution of  $x$  within an affine space. Now an arbitrary  $y_0$  will provide a starting point  $x_0 = Ay_0 - b$  for an iterative update procedure, and the search directions can be set to  $Q = A$ . Let  $e_i$  denote the  $i$ -th unit coordinate vector. It is now easy to see that our update rule (2) for  $x$  amounts to dual updates in  $y$  in coordinate descent form:

$$y_{t+1} = y_t - \frac{(Ay_t - b)^T q_i}{\|q_i\|^2} e_i = y_t + \alpha_t^* e_i.$$

179 Hence, we can iteratively construct the solutions in  $y$  and  $x$  by keeping track of the  
 180 stepsizes  $\alpha_t^*$  in the directions of  $Q$ . One may of course alternatively choose  $Q = AS$ ,  
 181 for any full-row-rank matrix  $S$ . The case of a square invertible system corresponds to  
 182 the overdetermined scenario in which the minimum-norm solution  $x$  is zero. Most of  
 183 our results for the underdetermined case can thus be simply recast, *mutatis mutandis*,  
 184 to the overdetermined or square invertible setting, and vice versa.

### 185 3. A new sparse matrix factorization for fast iterative updates.

**3.1. A  $k$ -sparse matrix factorization enabling efficient updates for iterative algorithms.** We are now prepared to introduce the notion of  $k$ -sparse matrix factorization. Our motivation for this factorization is that it should enable fast iterative updates of the form (2), i.e., we want to compute *any* iteration

$$x_{t+1} = x_t - \frac{x_t^T q_i}{q_i^T q_i} q_i,$$

186 in  $\mathcal{O}(k)$  time, if  $q_i$  is a column of the  $k$ -sparsely factorizable matrix  $Q = [q_1, q_2, \dots]$ .

187 The underlying idea here is akin to the case where  $q_i$  is a sparse vector with only  
 188  $k$  non-zero entries. Then just  $k$  non-zero products need to be computed. Hence, the  
 189 computational cost of the update is  $\mathcal{O}(k)$ . However, in order to solve a generic linear  
 190 system efficiently, we need to ensure that we can find a set of vectors  $\{q_1, \dots, q_n\}$  such  
 191 that *all* necessary iterative updates can be performed with this complexity. This will  
 192 be the key ingredient of our results on linear solvers presented in Section 5.

DEFINITION 1 (Support and sparsity of vectors and matrices). *The support of a vector  $v = (v^1, \dots, v^m)^T \in \mathbb{R}^m$  is the set of indices of the nonzero entries of  $v$ :*

$$\text{supp}(v) = \{i \in \{1, \dots, m\} : v^i \neq 0\}.$$

193 A vector  $v \in \mathbb{R}^m$  is said to be  $k$ -sparse, if the size of its support,  $|\text{supp}(v)|$ , is less  
 194 than or equal to  $k$ . Similarly, a matrix is said to be  $k$ -column ( $k$ -row) sparse if each  
 195 of its columns (rows) is  $k$ -sparse.

196 Suppose that  $x_t$  is not stored in the canonical basis, but in a different set of  
 197 coordinates encoded by a matrix  $C$ . That is, instead of performing iterations (2) on  
 198  $x_t$ , we keep track of a vector  $y_t$  such that  $x_t = Cy_t$ . To yield a sparse update, we  
 199 may choose  $C$  such that  $q_i$  is sparse in this representation, i.e.,  $q_i = Cd_i$ , where  $d_i$  is  
 200 a  $l$ -sparse vector. This leads to an iteration of the form:

$$201 \quad Cy_{t+1} = Cy_t - \frac{x_t^T q_i}{q_i^T q_i} Cd_i.$$

202 Using this representation, every update would be sparse in that it would only effect  $l$   
 203 components of  $y$ . However, this is not enough to perform each iteration (2) fast, as  
 204 one also needs to compute the scalar product  $x_t^T q_i$ , which in the new basis becomes  
 205  $y_t^T C^T C d_i$ , i.e., the iteration in terms of  $y_t$  is of the form:

$$206 \quad y_{t+1} = y_t - \frac{y_t^T C^T C d_i}{d_i^T C^T C d_i} d_i$$

207 To bound the complexity of this operation, one must understand the sparsity pattern  
 208 of  $C^T C$ , which is dictated by how the supports of the columns of  $C$  overlap. Observe  
 209 that the entry  $[C^T C]_{ij}$  contains the scalar products between the  $i^{\text{th}}$  and the  $j^{\text{th}}$   
 210 column of  $C$ . Whence, if every column of  $C$  overlaps in support with at most  $c$  other  
 211 columns, then every column of  $C^T C$  contains at most  $c$  non-zero entries. If we can  
 212 find a matrix for which this is true, then  $C^T C d_i$  is a  $k = cl$  sparse vector, since  $d_i$   
 213 is  $l$ -sparse, and  $y_t C^T C d_i$  is computed in time  $\mathcal{O}(k)$ . If we compile all such vectors  $q_i$   
 214 into a matrix  $Q$ , then we say that  $Q = CD$  is a  $k$ -sparse factorization.

215 While this reasoning provides us with some intuition, this definition must in fact  
 216 be improved to reach tighter complexity bounds. First, we can exploit the symmetry  
 217 of  $C^T C$ , by noting that it can be decomposed as  $C^T C = U^T + U$ , where  $U$  is an  
 218 upper-triangular matrix. Observe that the number of non-zero entries in the  $i$ th  
 219 column of  $U^T$  (or  $i$ th row of  $U$ ) is bounded by the number of columns  $c_j$  that overlap  
 220 with  $c_i$  for  $j \geq i$ . Second, two columns of  $U^T$  may have their non-zero entries at the  
 221 same positions. Therefore, the support of the sum of two columns does not necessarily  
 222 increase. To bound the complexity we need to look at the size of the union of supports  
 223 of all columns  $u_j$  of  $U^T$ , for which  $j$  belongs to the support of  $d_i$ . This number can  
 224 indeed be much lower than the approximate estimate  $cl$  above. This justifies the  
 225 following definition.

226 **DEFINITION 2.** *Suppose a matrix  $Q \in \mathbb{R}^{m \times n}$  has a factorization  $Q = CD$ . Let us*  
 227 *denote the columns of  $C \in \mathbb{R}^{m \times p}$  and  $D \in \mathbb{R}^{p \times n}$  by  $c_i$  and  $d_j$ , respectively. We define*  
 228 *the forward-overlap  $FO(c_i)$  of a column  $c_i$  to be the list of columns  $c_j$ , with  $j \geq i$ , that*  
 229 *have a support overlapping with the support of  $c_i$ . We call the factorization  $Q = CD$*   
 230  *$k$ -sparse if  $|\cup_{i \in \text{supp}(d_j)} FO(c_i)| \leq k$  for all  $j$  (see Figure 1 for an illustration). Without*  
 231 *loss of generality each column of  $C$  and each row of  $D$  is supposed to be nonzero.*

The example in Figure 1 shows an 8-sparse factorization of the given matrix  
 $Q$ . For instance, one can easily check that the forward overlap of column  $c_{12}$  is  
 $FO(c_{12}) = \{c_{12}, c_{13}, c_{16}, c_{20}\}$ , and e.g.

$$|\cup_{i \in \text{supp}(d_5)} FO(c_i)| = |\{c_{11}, c_{12}, c_{13}, c_{15}, c_{16}, c_{19}, c_{20}\}| = 7 \leq k = 8.$$

232 To gain some further intuition, let us consider an alternative definition of a sparse  
 233 factorization. We define a partial order on the columns of  $C$  with the following

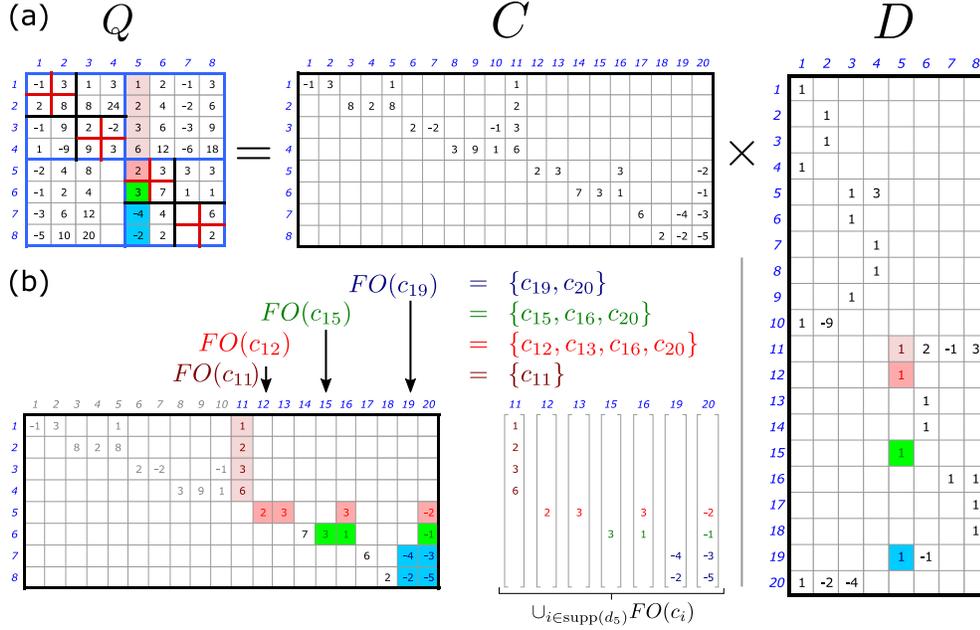


FIG. 1. (a) An example of 8-sparse factorization. (b) Illustration of the forward overlap of all columns  $i \in \text{supp}(d_5)$

234 properties. First, only columns  $c_i$  with overlapping support are comparable. Second,  
 235 every subset  $T_i = \{c_i, \dots\}$  spanning a column  $q_i$  has an upper set of at most  $k$   
 236 elements. The upper set is here defined as the union of  $T_i$  and all columns of  $C$   
 237 larger than any element of  $T_i$  in the partial order. Indeed the factorization  $Q = CD$   
 238 expresses nothing but the fact that every column  $q_i$  is a linear combination of a set  
 239 of columns of  $C$  with coefficients given by entries of  $i$ th column of  $D$ .

240 The following properties of a  $k$ -sparse factorization are worth noting.

- 241 1. Any  $m$ -by- $n$  matrix  $Q$  is  $\min(m, n)$ -sparsely factorizable with either  $Q = QI$   
 242 or  $Q = IQ$ . Similarly, it is easy to see from an SVD that every rank  $k$  matrix  
 243 is  $k$ -sparse factorizable.
- 244 2. If  $Q = CD$  is a  $k$ -sparse factorization, then for every column  $c_i$  of  $C$ ,  
 245  $|FO(c_i)| \leq k$ ,  $C$  is  $k$ -row sparse and each column of  $D$  is  $k$ -sparse.
- 246 3. Conversely, a matrix  $C$  such that  $|FO(c_i)| \leq k$  for all columns  $c_i$  is trivially  $k$ -  
 247 sparsely factorizable. A  $k$ -column sparse matrix  $D$  is also trivially  $k$ -sparsely  
 248 factorizable.
- 249 4. If  $Q = CD$  is a  $k$ -sparse factorization and  $F$  is  $f$ -column sparse, then  $QF =$   
 250  $C(DF)$  is a  $kf$ -sparse factorization of  $QF$ .
5. If  $Q_1 = C_1D_1$  is a  $k_1$ -sparse factorization and  $Q_2 = C_2D_2$  is a  $k_2$ -sparse  
 factorization, then the matrix  $(Q_1^T Q_2^T)^T$  is  $(k_1 + k_2)$ -sparsely factorizable. In  
 order to see this, we write

$$\begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} = \begin{pmatrix} C_1 & \\ & C_2 \end{pmatrix} \begin{pmatrix} D_1 \\ D_2 \end{pmatrix}.$$

251 In particular, if  $Q_2$  is the identity, the compound matrix is  $(k_1 + 1)$ -sparsely  
 252 factorizable.

253 The following theorem establishes the running time of  $N$  iterations of the form  
 254 (2), when the vectors  $q_i$  are the columns of a  $k$ -sparsely factorizable matrix. The  
 255 proof of the theorem is given in the appendix.

THEOREM 3. Let  $Q \in \mathbb{R}^{m \times n}$ ,  $C \in \mathbb{R}^{m \times p}$  and  $D \in \mathbb{R}^{p \times n}$  be matrices such that  
 $Q = CD$  is a  $k$ -sparse factorization of  $Q$ , and consider iterations of the form (2) that  
 start from an arbitrary vector  $x_0 \in \mathbb{R}^m$ . If every  $q_i$  in (2) is a column of  $Q$ , then the  
 computational complexity of running  $N$  iterations of (2) is:

$$\mathcal{O}(Nk + (m + n)k^2).$$

256 With the same complexity, we can compute a  $y_N$  such that  $x_N = x_0 + Qy_N$ , where  
 257  $x_N$  denotes the vector resulting from the  $N$  first iterations. By applying sufficiently  
 258 many iterations of form (2) we thus obtain both the solution to the primal problem in  
 259  $x$ , as well as the solution to the dual problem in  $y$ .

260 The remarkable point about Theorem 3 is that the running time of each iteration  
 261 is merely  $\mathcal{O}(k)$ , even if some columns of  $Q$  are full. Hence, if  $k \ll m$ , then the cost  
 262 per iteration can be largely reduced through the use of a  $k$ -sparse factorization, and  
 263 the overhead term  $(m + n)k^2$  is more than compensated.

264 **3.2. Ensuring fast convergence by randomized updates.** From our dis-  
 265 cussion above, we know that after sufficiently many iterations (2) over all columns of  
 266  $Q$ ,  $x_t$  converges to:

$$267 \quad (9) \quad x^* = \underset{x \in x_0 + \text{Im } Q}{\text{arg min}} \|x\|_2.$$

268 However, to ensure that we can construct an efficient algorithm based on such cheap  
 269 updates, we need to guarantee that the required number of updates is not too large,  
 270 as this would undermine the purpose of the fast updates. Stated differently, we need  
 271 the convergence rate of our iterations to be not too slow.

Remarkably, one can indeed ensure a sufficient convergence rate using a random  
 sampling of the columns of  $Q$ . To this end, at each iteration randomly select a column  
 $q_i$  with probability proportional to  $\|q_i\|$ . This guarantees a convergence rate of the  
 form

$$\mathbb{E}\|x_t - x^*\|_2^2 = \left(1 - \frac{\sigma_{\min}^2(Q)}{\|Q\|_{\text{Frob}}^2}\right)^t \|x_0 - x^*\|_2^2,$$

272 where  $\|Q\|_{\text{Frob}} = \sqrt{\text{Tr } Q^T Q}$  is the Frobenius norm and  $\sigma_{\min}^2(Q) = \lambda_{\min}(Q^T Q)$  is the  
 273 smallest nonzero squared singular value [9, 25]. The proof of this result is provided in  
 274 the appendix. There we also discuss interpretations of the here presented scheme in  
 275 terms of a randomized Kaczmarz or randomized coordinate descent method – with a  
 276 particular choice of update directions.

277 The above results state that the expected error in computing  $x^*$  is decreased by  
 278 an order of magnitude, e.g., by a factor of  $\delta^{-1} = 10$  after a number of iterations given  
 279 by

$$280 \quad (10) \quad N_1 = \frac{-\log(\delta^{-1})}{\log(1 - \sigma_{\min}^2(Q)/\|Q\|_{\text{Frob}}^2)} \approx \mathcal{O}(\|Q\|_{\text{Frob}}^2/\sigma_{\min}^2(Q)).$$

281 The main challenge for the construction of a fast algorithm is thus to find a ma-  
 282 trix  $Q$  spanning the desired search space, with efficient  $k$ -sparse factorization and  
 283 low ‘condition number’  $\|Q\|_{\text{Frob}}/\sigma_{\min}(Q)$ . Note that scaling each column of  $Q$  by a

284 different scalar will not change whether or not the updates will converge. Neither,  
 285 will it change the complexity of each update (as columns of  $Q$  only matter for their  
 286 directions). However, scaling the column may change the ‘condition number’ of  $Q$ ,  
 287 and hence the bound on the convergence time.

288 **3.2.1. The underdetermined case.** Let us develop the above reasoning some-  
 289 what further for the underdetermined case. One seeks the minimum-norm solution  
 290  $x^*$  to  $Ax = b$ , where  $A$  is an  $n$ -by- $m$  matrix with full-row-rank. Therefore it can be  
 291 decomposed as  $A = \begin{pmatrix} E & F \end{pmatrix}$ , where  $E$  is an invertible  $n \times n$  submatrix of  $A$ .

292 A matrix  $Q$  whose columns span the null space of  $A$  can then be constructed as:

$$293 \quad (11) \quad Q = \begin{pmatrix} E^{-1}F \\ -I_{m-n} \end{pmatrix},$$

294 where  $I_{m-n}$  is the identity matrix of dimension  $m - n$ . We clearly have  $AQ = 0$ , and  
 295 thus the columns of  $Q$  belong to the null space of  $A$ . The rank of  $Q$  is  $m - n$ , which  
 296 is the dimension of  $\text{null}(A)$ .

297 Moreover, we have that  $\sigma_{\min}^2(Q) = \lambda_{\min}(F^T E^{-T} E^{-1} F + I_{m-n}) \geq 1$ . The number  
 298 of steps to decrease the error by one order of magnitude is therefore at most of the  
 299 order of:

$$300 \quad (12) \quad N_1 = \mathcal{O}\left(\frac{\|Q\|_{\text{Frob}}^2}{\sigma_{\min}^2(Q)}\right) = \mathcal{O}(\|E^{-1}F\|_{\text{Frob}}^2 + m).$$

301 It follows from the elementary properties of sparse factorization that if  $E^{-1} = CD$   
 302 is  $k_0$ -sparsely factorizable and  $F$  is  $f$ -column sparse, then  $E^{-1}F$  is  $k_0 f$ -sparsely-  
 303 factorizable, and  $Q$  is  $k = (k_0 f + 1)$ -sparsely-factorizable:

$$304 \quad (13) \quad Q = \tilde{C}\tilde{D} = \begin{pmatrix} C & 0 \\ 0 & I_{m-n} \end{pmatrix} \begin{pmatrix} DF \\ -I_{m-n} \end{pmatrix}.$$

305 Hence, we have a good complexity if we can find an invertible square submatrix  
 306  $E$  such that  $\|E^{-1}F\|_{\text{Frob}}$  is small, and the resulting  $Q$  is  $k$ -sparsely factorizable, for  
 307 low  $k$ .

308 We still have to find a fairly good initial guess, however. A simple initial solution  
 309 is given by  $x_0 = \begin{pmatrix} E^{-1}b \\ 0 \end{pmatrix}$ , which can be shown to fulfill the following error bound:

$$310 \quad \begin{aligned} \|x_0\|^2 &= \|E^{-1}b\|^2 = \|E^{-1}Ax^*\|^2 = \|(I \quad E^{-1}F) x^*\|^2 \\ 311 \quad &\leq \|(I \quad E^{-1}F)\|_{\text{Frob}}^2 \|x^*\|^2 = \mathcal{O}(n + \|E^{-1}F\|_{\text{Frob}}^2) \|x^*\|^2. \end{aligned}$$

Overall, reducing the initial relative error

$$\epsilon_0 = \|x_0 - x^*\|/\|x^*\| \leq 1 + \|x_0\|/\|x^*\| = \mathcal{O}\left(\sqrt{n + \|E^{-1}F\|_{\text{Frob}}^2}\right)$$

313 to a prescribed value  $\epsilon$ , requires thus a reduction by  $\mathcal{O}(\log(n + \|E^{-1}F\|_{\text{Frob}}^2)/2 +$   
 314  $\log \epsilon^{-1})$  orders of magnitude, which is also in  $\mathcal{O}(\log(m + \|E^{-1}F\|_{\text{Frob}}^2) + \log \epsilon^{-1})$  given  
 315 that  $n \leq m$ .

316 In summary, denoting  $\kappa = m + \|E^{-1}F\|_{\text{Frob}}^2$ , we find that it takes  $N_1 = \mathcal{O}(\kappa)$  iter-  
 317 ations to decrease the error by an order of magnitude. Further, it takes  $\mathcal{O}(\log(\kappa \epsilon^{-1}))$   
 318 orders of magnitude to achieve relative accuracy  $\epsilon$ . Following Theorem 3, the total  
 319 complexity is thus  $\mathcal{O}(\kappa \log(\kappa \epsilon^{-1})k + mk^2)$ .

TABLE 1

Complexity of solving (compatible) structured linear systems with a  $k$ -sparse matrix factorization approach compared to known results in the literature.

Structure	$k$ -sparse factorization	Literature
$k$ row/column sparse	$\mathcal{O}(Nk)$	$\mathcal{O}(Nk)$ (randomized Kacmarz [25])
Hierarchical	$\mathcal{O}(N \log(n) + n \log^2(n))$	$\mathcal{O}(n \log^2(n))$ (direct method [2])
semiseparable	$\mathcal{O}(N \log(n) + n \log^2(n))$	$\mathcal{O}(n)$ [27, 28]
Laplacian	$\mathcal{O}(m \log^2 n \log \log n \log(m\epsilon^{-1}))$ (Thm. 16)	[15] (similar to this paper) [6] (fastest algorithm)

320 **4. Classes of sparsely factorizable matrices.** Many modern and classical  
 321 methods aim at exploiting particular structure in the system matrix for fast algo-  
 322 rithms. Table 1 provides an overview of results known from the literature and the  
 323  $k$ -sparse factorization approach presented in this paper. Interestingly, our  $k$ -sparse  
 324 matrix factorization approach provides good complexity results for a range of different  
 325 matrix types, and might thus be seen as a general framework for seemingly different  
 326 matrix structures. We will now discuss some classes in more detail.

327 Let us start with some intuitive examples first. A simple case is the overdeter-  
 328 mined system  $Ay = b$  where  $A$  is  $k$ -column-sparse. In this case, taking  $Q = A = IA$   
 329 as a trivial  $k$ -sparse factorization, and our algorithm can be seen as a randomized  
 330 Kacmarz scheme for the normal equation  $A^T x = A^T b$ , which keeps track of the up-  
 331 dates in the  $x$  coordinates but also in the  $y$  coordinates. In the space of  $y$ , this is  
 332 simply coordinate descent with a cost  $\mathcal{O}(k)$ , as discussed in Section 2.2. The total cost  
 333 amounts to  $\mathcal{O}(Nk)$  as the overhead cost becomes irrelevant when  $C$  in the  $Q = CD$   
 334 decomposition is the identity.

335 If  $A$  is  $k$ -row-sparse and invertible then  $Q = IA^T$  is a  $k$ -sparse factorization. In  
 336 this case a trivial modification of the algorithm in the proof of Theorem 3 simply  
 337 coincides again with a randomized Kacmarz scheme [25] (see Appendix).

338 **4.1. Hierarchical matrices.** In the following, we will discuss hierarchical  $\mathcal{H}_r$ -  
 339 matrices [13], originally introduced by Hackbusch [12], and show that they are  $k$ -  
 340 sparsely factorizable. Importantly, in this case  $k$  depends only on the height and the  
 341 degree of the hierarchical structure.

342 **4.1.1. Definition of an  $\mathcal{H}_r$ -matrix.** As the name suggests,  $\mathcal{H}_r$ -matrices are  
 343 intimately related to hierarchical structures. As a hierarchy may be aptly represented  
 344 as a tree we introduce these matrices here with the help of (tree-)graphs. As we will  
 345 see this also enables us to establish a connection to graph-theoretic algorithms for  
 346 solving Laplacian systems in subsequent sections.

347 **DEFINITION 4 (Dendrogram).** A dendrogram is a hierarchical partitioning  $\mathcal{P}$  of  
 348 the set  $\{1, \dots, n\}$ . Every dendrogram comprises a sequence of increasingly finer par-  
 349 titions  $P_h, \dots, P_0$  starting from the coarsest (global) partition  $P_h$  given by the whole  
 350 set, up to the finest (singleton) partition  $P_0$  into  $n$  sets. A dendrogram is conveniently  
 351 represented by a rooted directed tree. The nodes of this tree at height  $i$  are the subsets  
 352 of partition  $P_i$ . Thus the root ( $i = h$ ) is the full set while the leaves ( $i = 0$ ) are the  $n$   
 353 single-element subsets. The children (out-neighbours) of a node at height  $i$  correspond  
 354 to the subsets of this node as specified by the next lower partition  $P_{i-1}$ . We call  $h$  the  
 355 height of the dendrogram, and the maximum number of children of a node in the tree  
 356 is denoted as maximum degree  $d$ .

357 Figure 2a shows an example of a dendrogram with height 3 and maximum degree

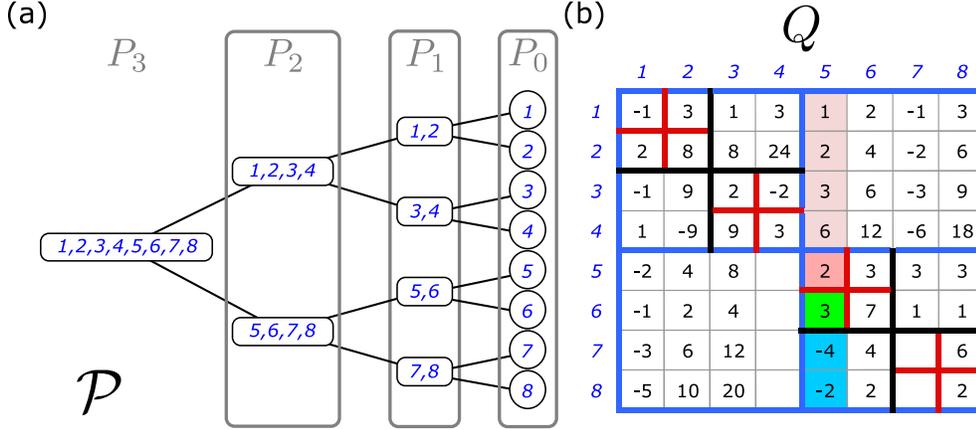


FIG. 2. (a) A dendrogram of  $I = \{1, \dots, 8\}$  of height  $h = 3$  and degree  $d = 2$ . (b) The  $P$ -partitioning of an  $8 \times 8$  matrix where  $\mathcal{P}$  is the dendrogram of  $\{1, \dots, 8\}$  in (a).

2. For simplicity of notation and without loss of generality, we suppose throughout the paper that every node of a dendrogram has consecutive elements.

A dendrogram  $\mathcal{P}$  induces a hierarchical block segmentation of a matrix  $E \in \mathbb{R}^{n \times n}$  as follows. Let us denote the degree of the root node by  $t \leq d$ . The rows and columns of  $E$  are first block-partitioned according to the partition  $P_{h-1}$ :

$$(14) \quad E = \begin{pmatrix} E_{I_1 \times I_1} & E_{I_1 \times I_2} & \cdots & E_{I_1 \times I_t} \\ E_{I_2 \times I_1} & E_{I_2 \times I_2} & \cdots & E_{I_2 \times I_t} \\ \vdots & \vdots & \vdots & \vdots \\ E_{I_t \times I_1} & E_{I_t \times I_2} & \cdots & E_{I_t \times I_t} \end{pmatrix},$$

where  $I_1, \dots, I_t$  are the elements of partition  $P_{h-1}$ . The diagonal blocks  $E_{I_i \times I_i}$ , are recursively sub-partitioned according to  $P_{h-2}$ , etc. This partitioning of  $E$  is called  $\mathcal{P}$ -partitioning. See Figure 2(b) for an illustration.

**DEFINITION 5. (Elementary block)** We use the term elementary block to refer to a sub-matrix of  $E$  generated by the  $\mathcal{P}$ -partitioning that is not further subdivided. In other words it is a block of the form  $E_{I_i \times I_j}$  where  $I_i$  and  $I_j$  are either two different sets in the same partition  $P_k$ , or two single-element sets of the finest partition  $P_0$ .

**DEFINITION 6. (Hierarchical Matrix)** An  $\mathcal{H}_r(\mathcal{P})$ -matrix is a square matrix, structured according to the dendrogram  $\mathcal{P}$ , for which the elementary blocks have rank at most  $r \in \mathbb{N}$ . We use the shorthand  $\mathcal{H}_r$  when the dendrogram is clear from the context.

Note that a sub-matrix  $E_{I_i \times I_i}$  of an  $\mathcal{H}_r(\mathcal{P})$ -matrix  $E$ , where  $I_i$  is a set of some partition  $P_k$ , is an  $\mathcal{H}_r$ -matrix as well.

**4.1.2. Sparse factorization property.** In the following, we prove that  $\mathcal{H}_r(\mathcal{P})$ -matrices are  $k$ -sparsely factorizable, and express  $k$  in terms of the rank  $r$ , maximum degree  $d$  and height  $h$ .

Recall that an  $\mathcal{H}_r(\mathcal{P})$ -matrix  $E$  is of the form (14). Every non-elementary block  $E_{I_i \times I_i}$  on the diagonal is recursively of the same form until the diagonal block is just a scalar. Hence, every diagonal non-elementary block is a hierarchical matrix, too. Further, note that every column of the full matrix  $E$  is built by concatenating the

383 corresponding columns of the  $E_{I_i \times I_j}$  blocks. For example, the first column of  $E$  can  
 384 be built by stacking up the first columns of  $E_{I_1 \times I_1}, E_{I_2 \times I_1}, \dots, E_{I_t \times I_1}$ .

385 We can thus build a  $k$ -sparse factorization  $E = CD$  as follows. As every off-  
 386 diagonal elementary block  $E_{I_i \times I_j}$  has a rank of at most  $r$ , there is a matrix  $D_{ij}$   
 387 such that the elementary block can be decomposed as  $E_{I_i \times I_j} = C_{ij}D_{ij}$ , where  $C_{ij}$   
 388 has at most  $r$  columns. Thus, we know how to express all the elements in the off-  
 389 diagonal blocks using this factorization. Hence, if we knew a sparse decomposition  
 390 of the diagonal blocks  $E_{I_i \times I_i} = C_{ii}D_{ii}$ , we could assemble the whole matrix  $E$  by  
 391 appropriate concatenation of the matrices  $C_{ij}$ .

392 To factorize the diagonal blocks we apply this construction recursively. To make  
 393 the recursion well defined, if the diagonal block  $E$  is a scalar (a  $1 \times 1$  matrix), we  
 394 define  $E = CD$ , where  $C$  is an arbitrary nonzero scalar, for instance we take  $C = E$   
 395 and take  $D = 1$ . Decomposing the columns of  $E$  in this recursive way, we obtain a  
 396 sparse factorization  $E = CD$ .

397 We illustrate this for the case  $t = 3$ , hereafter. For each  $i \in \{1, 2, 3\}$ , let each  
 398 diagonal block  $E_{I_i \times I_i} = C_{ii}D_{ii}$  be a  $k_i$ -sparse decomposition (recursively), and recall  
 399 that each elementary block  $E_{I_i \times I_j}$  ( $i \neq j$ ) can be factorized as  $E_{I_i \times I_j} = C_{ij}D_{ij}$ . Then  
 400 a  $k$ -sparse factorization of  $E$  is given by:

(15)

$$401 \quad E = \left( \underbrace{\begin{pmatrix} C_{11} & C_{12} & C_{13} & & & \\ & C_{22} & C_{21} & C_{23} & & \\ & & & & C_{33} & C_{31} & C_{32} \end{pmatrix}}_C \right) \underbrace{\begin{pmatrix} D_{11} & 0 & 0 \\ 0 & D_{12} & 0 \\ 0 & 0 & D_{13} \\ 0 & D_{22} & 0 \\ D_{21} & 0 & 0 \\ 0 & 0 & D_{23} \\ 0 & 0 & D_{33} \\ D_{31} & 0 & 0 \\ 0 & D_{32} & 0 \end{pmatrix}}_D, \quad 402$$

403 where  $C_{11}, C_{22}, C_{33}$  are recursively defined according to the diagonal blocks of  $E$ .

404 Having thus found a possible factorization, the question remains what sparsity,  
 405  $k$ , it affords. To answer this question, let us first consider the columns of  $C$  necessary  
 406 to build the first columns of  $E$ , and the union of their forward overlaps. There are  
 407 two types of columns in  $C$  needed to build up the first block of columns in  $E$ .

- 408 1. the columns in the  $(C_{11} \ 0 \ 0)^T$  block. Their forward-overlap is  $k_1 + r(l-1)$ ,  
 409 where  $k_1$  is the sparsity of the factorization of  $C_1$ , and the  $r(l-1)$  term  
 410 accounts for the overlap with the  $(l-1)$   $r$ -column matrices  $C_{12}$  and  $C_{13}$ .
- 411 2. The columns in the blocks  $(0 \ C_{21} \ 0)^T$  and  $(0 \ 0 \ C_{31})^T$ . Their forward  
 412 overlap is  $r(l-1)$  at most.

413 As this argument holds for any column of  $E$ , the factorization  $E = CD$  is  $k$ -sparse  
 414 for  $k = \max_i k_i + r(l-1)$ , where  $k_i$  is determined recursively from the decomposition  
 415 of the diagonal block  $E_{I_i \times I_i}$ . Unravelling the recursion over all  $h$  levels, we find that  
 416  $k = rd(d-1)(h+1)$ , where  $d$  is the maximal degree of the dendrogram, as before.

417 Throughout the paper, in a  $k$ -sparse factorization  $E = CD$  of an  $\mathcal{H}_r(\mathcal{P})$ -matrix,  
 418 the matrix  $C$  is supposed to be of the generic form (15), for an accordingly determined  
 419 degree  $d$ . We will call this type of matrix a  $C$ -matrix. In the Appendix we prove that  
 420 the number  $p$  of columns of  $C$  in the recursive construction in (15) is bounded by  
 421  $p \leq rd^2n$ .

422 We formalize the above findings in the following theorem.

423 **THEOREM 7.** *Let  $E \in \mathbb{R}^{n \times n}$  be an  $\mathcal{H}_r(\mathcal{P})$ -matrix with a dendrogram  $\mathcal{P}$  of height*  
 424  *$h$  and maximum degree  $d$ . Then, there are matrices  $C \in \mathbb{R}^{n \times p}$  and  $D \in \mathbb{R}^{p \times n}$  such*  
 425 *that  $p \leq rd^2n$  and the factorization  $E = CD$  is  $k$ -sparse for  $k = rd(d-1)(h+1)$ .*

426 **4.2. Semiseparable matrices.** Another important matrix class which has re-  
 427 ceived much attention in the literature are semi-separable matrices, whose inverses  
 428 are given by tridiagonal matrices [27, 28] and thus can be solved in linear time.

DEFINITION 8. [26] *An  $n \times n$  matrix  $E$  is called  $(p, q)$ -semiseparable if the fol-  
 lowing relations are satisfied:*

$$\text{rank}(E(1 : i + q - 1, i : n)) \leq q \text{ and } \text{rank}(E(i : n, 1 : i + p - 1)) \leq p$$

429 *for all feasible  $1 \leq i \leq n$ .*

430 **THEOREM 9.** *An  $n \times n$  matrix that is  $(p, q)$ -semiseparable is an  $\mathcal{H}_r(\mathcal{P})$ -matrix*  
 431 *where  $r = \max\{p, q\}$  and  $\mathcal{P}$  is a binary dendrogram.*

432 The proof is given in the appendix. It follows directly that semi-separable matrices  
 433 are  $k$ -sparsely factorizable, too. We note that, due to their remarkable structural  
 434 properties, algorithms solving semiseparable systems in linear time are well known in  
 435 the literature [27, 28].

436 **4.3. Reduced incidence matrices of trees and their inverse.** In what fol-  
 437 lows, we define a reduced incidence matrix of a tree, and show that it is  $k$ -sparsely  
 438 factorizable as it is an  $\mathcal{H}_1(\mathcal{P})$ -matrix where  $\mathcal{P}$  is a binary dendrogram ( $d = 2$ ). We  
 439 remark that, to the best of our knowledge, this connection between hierarchical matrices  
 440 and incidence matrices of trees has not been reported in the literature so far. The  
 441 importance of this observation arises in the context of Laplacian systems, as we will  
 442 see in a later section.

443 We first give the definitions of an incidence matrix of a graph and of a reduced  
 444 incidence matrix of a tree.

445 **DEFINITION 10** (Incidence matrix, reduced incidence matrix). *Let  $G$  be a posi-*  
 446 *tively weighted undirected graph on  $n$  nodes and  $m$  edges with an arbitrary direction*  
 447 *chosen for each edge. An incidence matrix  $B \in \mathbb{R}^{n \times m}$  of  $G$  is a node-by-edge matrix*  
 448 *such that given an edge  $e_i$  of  $G$  from node  $i_1$  to node  $i_2$  with weight  $w_i$ , the  $i$ th column*  
 449 *of  $B$  takes value  $-\sqrt{w_i}$  at the source node  $i_1$ , value  $\sqrt{w_i}$  at the target node  $i_2$  and*  
 450 *value 0 at any other node.*

451 *A reduced incidence matrix of a graph  $G$  is an incidence matrix of  $G$  from which*  
 452 *one row has been removed.*

453 To reveal the hierarchical structure in the reduced incidence matrix of a tree, one  
 454 has to recursively split the nodes of the tree in a balanced way. A classic way to do  
 455 so is provided by the tree-vertex-separator lemma.

456 **LEMMA 11** (Tree Vertex Separator Lemma, [5, 14]). *For any forest  $T$  with  $n \geq 2$*   
 457 *nodes, one can divide  $T$  into two forests both of at most  $2n/3$  nodes, by removing at*  
 458 *most one node  $d$ , which can be computed in  $\mathcal{O}(n)$  time.*

459 **PROPOSITION 12.** *A reduced incidence matrix of an  $n$ -edge tree is, for some or-*  
 460 *dering of the nodes and edges, an upper-triangular  $\mathcal{H}_1(\mathcal{P})$ -matrix for a binary den-*  
 461 *drogram  $\mathcal{P}$  with height  $\mathcal{O}(\log n)$ . The inverse of the reduced incidence matrix is, for*  
 462 *the same ordering of nodes and edges, also an upper-triangular  $\mathcal{H}_1(\mathcal{P})$ -matrix. The*  
 463 *dendrogram  $\mathcal{P}$  and both hierarchical matrices can be computed in time  $\mathcal{O}(n \log n)$ .*

464 Thus, a  $\mathcal{O}(\log n)$ -sparse factorization of (the inverse of) such a hierarchical matrix is  
 465 computable in time  $\mathcal{O}(n \log^2 n)$ .

466 *Proof.* Note that in this proof we consider  $T$  as an undirected tree with root  $v$ . A  
 467 tree  $T$  of  $n$  nodes has  $n - 1$  edges, and hence is described by an  $n$ -by- $(n - 1)$  incidence  
 468 matrix. By convention we assign an arbitrary direction to each edge, encoded by the  
 469 signs of the entries in the incidence matrix. However, the chosen direction does not  
 470 play any role for the results in the following. By removing a row from the incidence  
 471 matrix, we obtain a square reduced incidence matrix of dimension  $n - 1$ .

472 We now split the tree  $T$  into two forests  $T_1$  and  $T_2$  following the procedure of the  
 473 Tree Vertex Separator Lemma. Each of  $T_1, T_2$  will accordingly have no more than  
 474  $2n/3$  nodes. We assign the separator node  $d$  (if any) to  $T_2$ . We now order the nodes  
 475 in our reduced incidence matrix in two blocks according to this split:

$$476 \quad E = \begin{pmatrix} E_{I_1 \times I_1} & E_{I_1 \times I_2} \\ 0 & E_{I_2 \times I_2} \end{pmatrix},$$

477 where  $E_{I_i \times I_i}$  (for  $i = 1, 2$ ) is the reduced incidence matrix of  $T_i$  and  $E_{I_1 \times I_2}$  is a  
 478 rank-1 matrix with at most one non-zero entry corresponding to the edge linking  $d$   
 479 to its father. Here, the indices of the edges have been assigned as follows: an edge  
 480 connecting node  $i$  and  $j$  is indexed by  $j$ , if  $j$  is one step further away from the root  
 481 than  $i$  (i.e.  $j$  is the ‘child’ of  $i$ ).

We repeat this argument recursively and thereby create a dendrogram  $P$  on the  
 nodes of  $T$  of height  $\mathcal{O}(\log n)$ , and a corresponding upper triangular  $\mathcal{H}_1(\mathcal{P})$ -matrix  
 structure for  $E$ . From the ordering of edges, we see that the  $i$ th node is always incident  
 to the  $i$ th edge, thus the diagonal entry of  $E$  is  $\pm\sqrt{w_i}$ , making it easily invertible.  
 Indeed, the inverse of  $E$  can be computed recursively as

$$E^{-1} = \begin{pmatrix} E_{I_1 \times I_1}^{-1} & F \\ 0 & E_{I_2 \times I_2}^{-1} \end{pmatrix},$$

482 with  $F = -E_{I_1 \times I_1}^{-1} E_{I_1 \times I_2} E_{I_2 \times I_2}^{-1}$ . Note that we may write  $F = uv^T$  as it is clearly  
 483 of rank one at most, thus leading to an upper-triangular  $\mathcal{H}_1(\mathcal{P})$ -matrix for  $E^{-1}$  as  
 484 well. Both for  $E$  and  $E^{-1}$ , every of the  $\mathcal{O}(\log n)$  steps of the recursion takes  $\mathcal{O}(n)$ ,  
 485 required to finding the tree vertex separators and (in case of  $E^{-1}$ ) computing  $u$  and  
 486  $v$ , solutions of triangular systems. Therefore we get a total cost of  $\mathcal{O}(n \log n)$ .

487 Finally, using the procedure outlined above we can decompose  $E^{-1} = CD$ . Using  
 488  $E_{I_1 \times I_1}^{-1} = C_{11}D_{11}$  and  $E_{I_2 \times I_2}^{-1} = C_{22}D_{22}$ , we recursively construct:

$$489 \quad E^{-1} = \begin{pmatrix} C_{11} & u \\ & C_{22} \end{pmatrix} \begin{pmatrix} D_{11} \\ & D_{22} \\ & & v^T \end{pmatrix}.$$

490 By unfolding this recursion we can see that this leads to a forward-overlap of size  
 491  $\mathcal{O}(\log(n))$  in  $C$ , and an  $\mathcal{O}(\log(n))$  column-sparse matrix  $D$ . Similarly, a  $\mathcal{O}(\log n)$ -  
 492 sparse factorization can be obtained for  $E$ .  $\square$

493 **5. Fast iterative linear solvers on hierarchical systems.** To illustrate the  
 494 usefulness of our results, in the following we showcase two concrete application sce-  
 495 narios in which the above developed theory can be employed.

496 **5.1. A strategy for solving underdetermined systems.** In the following, we  
 497 focus again on the case of an underdetermined system  $Ax = b$ . We devise a strategy

498 that assumes a decomposition of the  $n$ -by- $m$  full-rank matrix  $A$  (with  $n < m$ ) of the  
 499 form  $A = \begin{pmatrix} E & F \end{pmatrix}$ , where  $E$  is an invertible  $n \times n$  submatrix of  $A$ . In particular, let  
 500 us consider the case where  $E^{-1}$  is hierarchical. We can then combine Theorem 3 and  
 501 the subsequent discussion, and Theorem 7 to obtain the following result.

THEOREM 13. *Let  $A = \begin{pmatrix} E & F \end{pmatrix}$  be an  $n \times m$  matrix with  $n < m$ , where  $E \in \mathbb{R}^{n \times n}$  is invertible and  $E^{-1}$  is an  $\mathcal{H}_r(\mathcal{P})$ -matrix with an associated dendrogram  $\mathcal{P}$  of maximum degree  $d$  and height  $h$ . Further, let  $F \in \mathbb{R}^{n \times (m-n)}$  be  $f$ -column sparse. Then, we can compute an approximation of  $x^* := \arg \min_{s.t. Ax=b} \|x\|$  by applying  $N$  iterations of the form (2), in time*

$$\mathcal{O}(Nfrd^2h + mf^2r^2d^4h^2) + \text{Cost}(CD),$$

502 where  $\text{Cost}(CD)$  is the cost of computing a  $(rd^2(h+1))$ -sparse factorization of  $E^{-1}$ .  
 503 The number of iterations to gain one order of magnitude on the error is at most  
 504  $N_1 = \mathcal{O}(\|E^{-1}F\|_{\text{Frob}}^2 + m)$ .

505 *Proof.* Following Theorem 7, let  $E^{-1} = CD$  be a  $k$ -sparse factorization with  
 506  $k = rd(d-1)(h+1) = \mathcal{O}(rd^2h)$ . By the second elementary property of the sparse  
 507 factorization (see Property 2 on page 7), we know that  $C$  is  $k$ -row sparse and that each  
 508 column of  $D$  is  $k$ -sparse. A feasible solution to  $Ax = b$  is then given by  $x_0 = \begin{pmatrix} E^{-1}b \\ 0 \end{pmatrix}$   
 509 where  $E^{-1}b = CDb$  is computed in  $\mathcal{O}(kn)$  time.

Now, consider the matrix  $Q$  given in (11). From our discussion above we know that the columns of  $Q$  are a basis of  $\text{null}(A)$  and that the matrix  $Q$  is  $(kf+1)$ -sparsely factorizable. Let  $Q = \tilde{C}\tilde{D}$  be the  $(kf+1)$ -sparse factorization given in (13). We start from the vector  $x_0$  and iteratively pick a column  $q$  of  $Q$  and perform an iteration of the form (2). Theorem 3 with  $Q, \tilde{C}$  and  $\tilde{D}$  then shows that the running time is given by

$$\mathcal{O}(Nfk + mf^2k^2) + \text{Cost}(CD).$$

510 **5.2. Square hierarchical systems.** The present technique can be also applied  
 511 to solve square systems  $Ax = b$ , where  $A$  is hierarchical and invertible.

THEOREM 14. *The system  $Ay = b$ , where  $A$  is an invertible  $n$ -by- $n$   $\mathcal{H}_r(\mathcal{P})$ -matrix with  $\mathcal{P}$  a dendrogram of degree  $d$  and height  $h$ , can be solved iteratively in time*

$$\mathcal{O}(Nrd^2h + nr^2d^4h^2) + \text{Cost}(CD),$$

512 where  $N$  is the number of iterations and  $\text{Cost}(CD)$  is the running time needed to  
 513 compute a  $k$ -sparse factorization of  $A$  with  $k = rd(d-1)(h+1)$ .

*Proof.* In section 2.2, page 5, we explain how to solve an overdetermined system using iterations (2). Trivially, we can use the presented method for the square system  $Ay = b$ . Following the notations of Theorem 3, here  $Q = A$ ,  $m = n$  and the running time is

$$\mathcal{O}(Nk + nk^2) + \text{Cost}(CD).$$

We moreover use Theorem 7 which states  $k = rd(d-1)(h+1)$  to deduce that the running time is

$$\mathcal{O}(Nrd^2h + nr^2d^4h^2) + \text{Cost}(CD).$$

In particular, if  $A$  is an  $\mathcal{H}_1(\mathcal{P})$ -matrix (rank  $r = 1$ ) with a binary ( $d = 2$ ) dendrogram  $\mathcal{P}$  of height  $h = \mathcal{O}(\log n)$  (e.g.,  $A$  could be the reduced incidence matrix of a tree), then this running time becomes

$$\mathcal{O}(N \log n + n \log^2 n),$$

514 where we have used Proposition 12 which states that a sparse factorization of  $A$  is  
 515 computed in time  $\mathcal{O}(n \log^2 n)$ .

516 As far as we know, this is the best iterative method in terms of cost per iteration  
 517 ( $\log n$ ). Most standard method would exhibit a cost of  $\mathcal{O}(n)$  per iteration, the cost of  
 518 a matrix-vector product. However, for solving squared hierarchical systems a direct  
 519 method exists that solves such a problem in  $\mathcal{O}(n \log^2 n)$  [2].

520 **6. Solving Laplacian systems in nearly linear time.** In the following we  
 521 demonstrate how the approach outlined above can be used to solve Laplacian systems.

522 **6.1. Minimum norm solution for a system with reduced incidence ma-**  
 523 **trix.**

524 **COROLLARY 15.** *Let  $A$  be a reduced incidence matrix of a connected undirected*  
 525 *graph on  $n$  nodes and  $m$  edges. Then, the minimal norm solution  $x^*$  of a compat-*  
 526 *ible system  $Ax = b$  can be computed with relative accuracy  $\epsilon = \|x_t - x^*\|/\|x^*\|$  in*  
 527  *$\mathcal{O}(m \log^2(n) \log \log(n) \log(m\epsilon^{-1}))$  time.*

528 *Proof.* Note that every edge in the graph corresponds to one column of  $A$ , and  
 529 thus every spanning tree is associated with a submatrix  $E$  which is invertible by  
 530 construction [24]. Choosing an invertible (sub-)matrix  $E$  such that  $A = \begin{pmatrix} E & F \end{pmatrix}$   
 531 is therefore equivalent to selecting a spanning tree of  $G$ . We now claim that we  
 532 can choose  $E$ , i.e., choose an appropriate spanning tree, such that  $\|E^{-1}F\|_{\text{Frob}}^2 =$   
 533  $\mathcal{O}(m \log n \log \log n)$ .

534 For any choice of spanning tree, we define the root as the node whose row has  
 535 been removed from the incidence matrix  $A$  to obtain a reduced incidence matrix. We  
 536 choose the (arbitrary) orientation on the edges so as to go from root to leaves. We  
 537 also order the nodes from root to leaves (topological order) and edges so that any edge  
 538 has the same index as its destination. Let us call the *unweighted, directed* adjacency  
 539 matrix of this spanning tree  $T_E$ . With the choices made above  $T_E$  is upper triangular.  
 540 Then we can write  $E = (I - T_E)\sqrt{W_{T_E}}$  where  $W_{T_E}$  is the diagonal matrix weights  
 541 on the edges.

542 Using a Neumann series expansion we can see that  $E^{-1} = W_{T_E}^{-1/2}(I + T_E + T_E^2 +$   
 543  $T_E^3 + \dots + T_E^h)$  where  $h$  is the height of the tree. The columns of  $E^{-1}$  encode the  
 544 paths between root and leaves, with entries given by the (positive) inverse square root  
 545 of the edge-weights.

546 Since  $F$  is a (reduced) incidence matrix, each column  $i$  of  $E^{-1}F$  is the (weighted)  
 547 difference between two columns of  $E^{-1}$ . In fact, each column  $i$  of  $E^{-1}F$  describes  
 548 the (signed) path in the tree between the extremities of edge  $i$ , on which each edge  
 549  $e$  has weight  $\sqrt{w_i/w_e}$ . Therefore the squared Frobenius norm of  $E^{-1}F$  is the so-  
 550 called stretch of the tree in the graph with *inverse* weights, i.e. weight  $w_e^{-1}$  on  
 551 each edge  $e$  of the graph, as already noticed in [15]. Using the algorithm in Ref. [1]  
 552 we can therefore find a spanning tree with reduced incidence matrix  $E$  such that  
 553  $\|E^{-1}F\|_{\text{Frob}}^2 = \mathcal{O}(m \log n \log \log n)$ , where  $m$  is the number of edges in the graph.  
 554 The incurred computational cost for is  $\mathcal{O}(m \log n \log \log n)$  [1].

555 From Proposition 12, it follows that  $E^{-1}$  is an  $\mathcal{H}_1(\mathcal{P})$ -matrix, with is a binary  
 556 dendrogram  $\mathcal{P}$  of height  $h = \mathcal{O}(\log n)$ , and parameters  $r = 1$ ,  $d = 2$ . A sparse  
 557 decomposition of  $E^{-1}$  can thus be computed in time  $\mathcal{O}(n \log^2 n)$ . Using Theorem 13,  
 558 we can thus compute the minimal norm solution  $x^*$  of  $Ax = b$  in nearly linear time.

559 More precisely, following Section 3.2.1 we define  $\kappa = \|E^{-1}F\|_{\text{Frob}}^2 + m$ , which  
 560 is  $\mathcal{O}(m \log n \log \log n)$ . We then find that  $N_1 = \mathcal{O}(\kappa)$  iterations, each of which  
 561 costs  $k = \mathcal{O}(\log n)$ , suffice to gain one order of magnitude, and the overall cost to

562 achieve a relative accuracy  $\epsilon$  is  $\mathcal{O}(\kappa \log(\kappa \epsilon^{-1}) + mk^2)$ , which in this case reduces to  
 563  $\mathcal{O}(m \log^2 n \log \log n \log(m \epsilon^{-1}))$ .

564 **6.2. Solving Laplacian systems.** The above corollary provides the critical step  
 565 in solving a compatible Laplacian system  $L\chi = c$ , where  $L$  is the Laplacian of the  
 566 same graph, as we show now. For a given incidence matrix  $B$  the Laplacian is defined  
 567 as  $L = BB^T$ , or equivalently as the node-by-node matrix with entries  $L_{ij} = -w_{ij}$  for  
 568 every edge  $ij$  of weight  $w_{ij}$ ,  $L_{ij} = 0$  if  $i$  is not adjacent to  $j$ , and the weighted degree  
 569  $L_{ii} = \sum_k w_{ik}$  on diagonal entries. Such a system  $L\chi = c$  can be solved in two steps:

- 570 1. solve  $Bx = c$  so that  $x$  is in the image of  $B^T$ ;
- 571 2. solve the compatible, overdetermined system  $B^T\chi = x$ .

572 This strategy of splitting the problem of solving a Laplacian system into 2 parts is in  
 573 line with the approach followed by Kelner et al. [15]. However, their algorithm relies  
 574 on graph-theoretic notions and a specific data structure construction, rather than a  
 575 matrix decomposition.

576 Note that the first step in the procedure above is equivalent to finding the  
 577 minimum-norm solution of  $Bx = c$ . Any solution of  $Bx = c$  is of the form  $x = B^T\chi + v$ ,  
 578 for some  $v$  such that  $Bv = 0$ . This implies that  $v$  is orthogonal to  $B^T\chi$ , and thus  
 579  $B^T\chi + v$  has a norm larger than  $B^T\chi$ , with the minimum norm solution given by  
 580  $v = 0$ . The goal is therefore to solve  $Bx = c$  in the minimum norm sense. Since the  
 581 columns of  $B$  sum to zero, we can remove an arbitrary row without affecting the solu-  
 582 tion, i.e., we can ‘ground’ the system. Let us call  $A$  the so-obtained reduced incidence  
 583 matrix of the graph, and  $b$  the vector obtained from  $c$  by removing one entry. Now  
 584 we have to solve  $Ax = b$ , which can be done efficiently as discussed above.

585 The second step outlined above then requires finding the solution of a compatible  
 586 overdetermined system. This can be found by solving the square invertible triangular  
 587 subsystem  $E^T y = x_E$  where  $E$  is the reduced incidence matrix of the spanning tree  
 588 used to solve  $Ax = b$  (see the proof of Corollary 15) and  $x_E$  is the corresponding part  
 589 of vector  $x$ . Solving this triangular system takes  $\mathcal{O}(n)$  time, from leaves to root.

590 We remark that when solving a semi-definite positive system  $L\chi = c$ , the  $L$ -  
 591 pseudo-norm  $\|\chi\|_L^2 = \chi^T L\chi$  is often used as the error norm. Note that all  $\|\chi\|_L^2$   
 592 vanishes only if vector  $\chi$  has identical entries. The relative accuracy of the solution  
 593  $\chi$  is accordingly defined as  $\epsilon = \|\chi - \chi^*\|_L / \|\chi^*\|_L$ .

594 Putting these pieces together, we obtain the following theorem:

595 **THEOREM 16.** *Given a Laplacian matrix  $L$  of a connected graph with  $m$  edges  
 596 and a zero-sum vector  $c$ , the (compatible) system  $L\chi = c$  can be solved within time  
 597  $\mathcal{O}(m \log^2 n \log \log n \log(m \epsilon^{-1}))$  with relative accuracy  $\epsilon$ , as measured in the  $L$ -pseudo-  
 598 norm.*

599 *Proof.* From Corollary 15 we find an approximate solution  $x^* + \Delta x$  to the problem  
 600  $Bx = c$ , with  $\|\Delta x\| / \|x^*\| \leq \delta$ , in time  $\mathcal{O}(m \log^2 n \log \log n \log(n \delta^{-1}))$ .

601 We then find the approximate solution  $\chi^* + \Delta\chi$  as  $E^{-T}(x_E^* + \Delta x_E)$ , where  $x_E$   
 602 denotes the restriction of the  $m$ -dimensional vector  $x$  to the  $n$  entries corresponding  
 603 to  $E$ . The incurred error  $\Delta\chi$  can be bounded, using  $L = BB^T$  and  $B = (E \ F)$ :

$$604 \quad (16) \quad \|\Delta\chi\|_L^2 = \|E^{-T}\Delta x_E\|_L^2 = \|(I \ E^{-1}F)^T \Delta x_E\|^2 \leq \mathcal{O}(m \log n \log \log n) \|\Delta x\|^2.$$

Moreover the exact solution fulfills  $\|\chi^*\|_L^2 = \|x^*\|^2$  by definition of  $x = B^T\chi$ .  
 Thus, we see that the relative accuracy on  $x$  in terms of  $\|\cdot\|_L$  is

$$\frac{\|\Delta\chi\|_L^2}{\|\chi^*\|_L^2} = \mathcal{O}(m \log n \log \log n) \frac{\|\Delta x\|^2}{\|x^*\|^2}.$$

606 Therefore we can choose  $\delta^{-1} = \mathcal{O}(\sqrt{m \log n \log \log n})\epsilon^{-1}$ , for any required accuracy  
 607 level  $\epsilon$ . The proof is concluded by Corollary 15.  $\square$

608 We remark that the computational complexity of our final algorithm could be  
 609 reduced further, by using some of the computational techniques discussed in [15–17],  
 610 which are beyond the scope of this paper, however. For instance, one could employ  
 611 a preconditioning to change the norm of  $\|E^{-1}F\|$  and thereby obtain a better initial  
 612 estimate for  $\hat{x}_0$ . Indeed using such a preconditioning recursively, Kelner et al. are  
 613 able to obtain an algorithm with a total complexity of  $\mathcal{O}(m \log^2 n \log \log n \log \epsilon^{-1})$  [15].  
 614 Note, however, that Kelner et al. [15] employ quite different means to establish this  
 615 result. Instead of a matrix factorization, the core tool invoked is an efficient data-  
 616 structure which enables fast updates. Our  $k$ -sparse matrix factorization approach  
 617 may thus be seen as an alternative perspective on the problem of solving Laplacian  
 618 systems.

619 **7. Conclusion.** In this paper we have considered the problem of finding the  
 620 minimum norm vector  $x$  within an affine space, which arises naturally when solving  
 621 an under- or overdetermined linear system. We have shown that this problem can  
 622 be solved very efficiently in an iterative manner by choosing the matrix of search  
 623 directions  $Q = [q_1, \dots, q_m]$  in an appropriate way. Specifically, if there exists a  $k$ -  
 624 sparse matrix factorization of  $Q$ , each iterative update of the form  $x_{t+1} = x_t - \frac{x_t^T q_i}{q_i^T q_i} q_i$   
 625 can be computed in  $\mathcal{O}(k)$  time, enabling us to construct fast algorithm for solving  
 626 linear systems. The notion of a  $k$ -sparse matrix factorization is indeed central to these  
 627 findings, as it ensures the existence of a computationally efficient update scheme  
 628 despite the fact that  $Q$  might be full, i.e., the search directions are not formed by  
 629 sparse vectors.

630 We have shown that some important classes of matrices are  $k$ -sparsely factoriz-  
 631 able, and in particular that in the case of hierarchical matrices  $k$  does not depend  
 632 on the size of the matrix, but merely on the depth of the hierarchy. From this, we  
 633 have deduced an iterative method with fast iterations that approximates the minimal  
 634 norm solution of underdetermined linear systems. In particular, this approach can  
 635 be applied when the coefficient matrix is the incidence matrix of a connected graph.  
 636 This leads naturally to a method to solve Laplacian systems in nearly-linear time. In  
 637 this context, our work provides a complementary algebraic perspective to the problem  
 638 of solving Laplacian system, and connects combinatorial and graph-theoretic notions  
 639 with the problem of finding a  $k$ -sparse matrix factorization.

640 An important direction for future work is to characterise the general class of  
 641 matrices that can be sparsely factorized in more detail, and see how it can be extended  
 642 beyond the matrices discussed within the present manuscript. For instance, solvers  
 643 based on tensor decompositions [3, 19, 21] have been presented in the literature, which  
 644 assume that the linear system under study has an inherent Kronecker-product [3, 19]  
 645 or tensor-train [21] representation (or at least can be well approximated by such a  
 646 structure). It would be interesting to investigate in how far these matrix structures  
 647 are also amenable to a  $k$ -sparse factorization.

648 Other avenues for future work include investigating possible parallelization of the  
 649 here presented techniques, or combining them with other randomized update schemes  
 650 [9, 10] than the here considered randomized Kaczmarz updates [25]. For instance, it  
 651 would be interesting to see in how far block updates (instead of single coordinate  
 652 updates), could lead to more efficient iterative algorithms.

653

## REFERENCES

- 654 [1] I. ABRAHAM AND O. NEIMAN, *Using petal-decompositions to build a low stretch spanning tree*,  
655 in Proceedings of the forty-fourth annual ACM symposium on Theory of computing, ACM,  
656 2012, pp. 395–406.
- 657 [2] S. AMBIKASARAN AND E. DARVE, *An  $\mathcal{O}(N \log N)$  Fast Direct Solver for Partial Hierarchically*  
658 *Semi-Separable Matrices*, Journal of Scientific Computing, 57 (2013), pp. 477–501.
- 659 [3] J. BALLANI AND L. GRASEDYCK, *A projection method to solve linear systems in tensor format*,  
660 Numerical Linear Algebra with Applications, 20 (2013), pp. 27–43.
- 661 [4] S. BÖRM, L. GRASEDYCK, AND W. HACKBUSCH, *Introduction to hierarchical matrices with*  
662 *applications*, Engineering Analysis with Boundary Elements, 27 (2003), pp. 405–422.
- 663 [5] F. R. K. CHUNG, *Separator theorems and their applications*, in Algorithms and Combinatorics  
664 9, B. Korte, L. Lovász, and H. J. Prömel, eds., Springer, 1990.
- 665 [6] M. B. COHEN, R. KYNG, J. W. PACHOCKI, R. PENG, AND A. RAO, *Preconditioning in expect-*  
666 *ation*, arXiv:1401.6236, (2014).
- 667 [7] T. A. DAVIS, *Direct methods for sparse linear systems*, vol. 2, Siam, 2006.
- 668 [8] H. C. ELMAN, *Iterative methods for linear systems*, Large-scale matrix problems and the nu-  
669 merical solution of partial differential equations, 3 (1994), pp. 69–177.
- 670 [9] R. M. GOWER AND P. RICHTÁRIK, *Randomized iterative methods for linear systems*, SIAM  
671 Journal on Matrix Analysis and Applications, 36 (2015), pp. 1660–1690.
- 672 [10] R. M. GOWER AND P. RICHTÁRIK, *Stochastic dual ascent for solving linear systems*,  
673 arXiv:1512.06890, (2015).
- 674 [11] L. GRASEDYCK AND W. HACKBUSCH, *Construction and arithmetics of  $H$ -matrices*, Computing,  
675 70 (2003), pp. 295–334.
- 676 [12] W. HACKBUSCH, *A sparse matrix arithmetic based on  $\mathcal{H}$ -matrices. part i: Introduction to  $\mathcal{H}$ -*  
677 *matrices*, Computing, 62 (1999), pp. 89–108.
- 678 [13] W. HACKBUSCH, *Hierarchical matrices: Algorithms and analysis*, vol. 49, Springer, 2015.
- 679 [14] C. JORDAN, *Sur les assemblages de lignes*, J. Reine Angew. Math, 70 (1869), p. 81.
- 680 [15] J. A. KELNER, L. ORECCHIA, A. SIDFORD, AND Z. A. ZHU, *A simple, combinatorial algorithm*  
681 *for solving SDD systems in nearly-linear time*, in Proceedings of the forty-fifth annual  
682 ACM symposium on Theory of computing, ACM, 2013, pp. 911–920.
- 683 [16] I. KOUTIS, G. L. MILLER, AND R. PENG, *Approaching optimality for solving SDD linear sys-*  
684 *tems*, in Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium  
685 on, IEEE, 2010, pp. 235–244.
- 686 [17] I. KOUTIS, G. L. MILLER, AND R. PENG, *A nearly- $m \log n$  time solver for sdd linear systems*, in  
687 Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on, IEEE,  
688 2011, pp. 590–598.
- 689 [18] I. KOUTIS, G. L. MILLER, AND R. PENG, *A fast solver for a class of linear systems*, Commu-  
690 nications of the ACM, 55 (2012), pp. 99–107.
- 691 [19] D. KRESSNER AND C. TOBLER, *Krylov subspace methods for linear systems with tensor product*  
692 *structure*, SIAM journal on matrix analysis and applications, 31 (2010), pp. 1688–1714.
- 693 [20] Y. T. LEE AND A. SIDFORD, *Efficient accelerated coordinate descent methods and faster algo-*  
694 *rithms for solving linear systems*, in Foundations of Computer Science (FOCS), 2013 IEEE  
695 54th Annual Symposium on, IEEE, 2013, pp. 147–156.
- 696 [21] I. V. OSELEDETS AND S. DOLGOV, *Solution of linear systems and matrix inversion in the*  
697 *TT-format*, SIAM Journal on Scientific Computing, 34 (2012), pp. A2718–A2739.
- 698 [22] Y. SAAD, *Iterative methods for sparse linear systems*, Siam, 2003.
- 699 [23] D. A. SPIELMAN AND S.-H. TENG, *Nearly-linear time algorithms for graph partitioning, graph*  
700 *sparsification, and solving linear systems*, in Proceedings of the thirty-sixth annual ACM  
701 symposium on Theory of computing, ACM, 2004, pp. 81–90.
- 702 [24] G. STRANG, *Introduction to applied mathematics*, Wellesley-Cambridge Press, Wellesley, MA,  
703 1986.
- 704 [25] T. STROHMER AND R. VERSHYNIN, *A randomized Kaczmarz algorithm with exponential con-*  
705 *vergence*, Journal of Fourier Analysis and Applications, 15 (2009), pp. 262–278.
- 706 [26] R. VANDEBRIL, M. VAN BAREL, G. GOLUB, AND N. MASTRONARDI, *A bibliography on semisep-*  
707 *arable matrices*, Calcolo, 42 (2005), pp. 249–270.
- 708 [27] R. VANDEBRIL, M. VAN BAREL, AND N. MASTRONARDI, *Matrix computations and semiseparable*  
709 *matrices: linear systems*, vol. 1, Johns Hopkins University Press, 2007.
- 710 [28] R. VANDEBRIL, M. VAN BAREL, AND N. MASTRONARDI, *Matrix Computations and Semisep-*  
711 *arable Matrices: Eigenvalue and Singular Value Methods*, vol. 2, Johns Hopkins University  
712 Press, 2008.
- 713 [29] D. M. YOUNG, *Iterative solution of large linear systems*, Elsevier, 2014.

714 **Appendix A. Proof of Theorem 3.**

THEOREM 17 (Theorem 3). *Let  $Q \in \mathbb{R}^{m \times n}$ ,  $C \in \mathbb{R}^{m \times p}$  and  $D \in \mathbb{R}^{p \times n}$  be matrices such that  $Q = CD$  is a  $k$ -sparse factorization of  $Q$ , and consider iterations of the form (2) that start from an arbitrary vector  $x_0 \in \mathbb{R}^m$ . If every  $q_i$  in (2) is a column of  $Q$ , then the computational complexity of running  $N$  iterations of (2) is:*

$$\mathcal{O}(Nk + (m + n)k^2).$$

715 *With the same complexity, we can compute a  $y_N$  such that  $x_N = x_0 + Qy_N$ , where*  
 716  *$x_N$  denotes the vector resulting from the  $N$  first iterations.*

717 *Proof.* Let us first comment on the general strategy for computing fast iterations.  
 718 Given  $x_t \in \mathbb{R}^m$  and a column  $q_j = Cd_j$  of  $Q$ , recall that the next iteration we aim to  
 719 compute is of the form

$$720 \quad (17) \quad x_{t+1} = x_t - \frac{x_t^T q_j}{q_j^T q_j} q_j.$$

721 In order to get a running time for each iteration not depending on the system  
 722 size  $m$ , we make use of two generating sets of  $\mathbb{R}^m$ . The sets are given by the columns  
 723 of  $C$ , as well as the columns of  $CU^{-T}$ , where  $U$  is the  $p \times p$  upper triangular matrix  
 724 such that  $C^T C = U^T + U$ . Each column of  $Q$  has a decomposition in terms of these  
 725 generating sets with a sparsity governed by  $k$ ; indeed a column  $q_j$  is expressed as  
 726  $q_j = Cd_j = CU^{-T}U^T d_j$ , where  $d_j$ , a column of  $D$ , is  $k$ -sparse and  $e_j := U^T d_j$  is  
 727  $k$ -sparse by definition of the  $k$ -sparse factorisation. Using these sets we can thus  
 728 express  $x_t$ , with the coefficient vectors  $h_t, g_t$ , defined via the relationships  $x_t = Ch_t$   
 729 and  $x_t = CU^{-T}g_t$ . Note that such a vector  $g_t$  is given by  $g_t = U^T h_t$ . Now at each  
 730 iteration, we only use the vectors  $h_i, g_i, d_j$  and  $e_j$ , and do not need to store the full  
 731 vector  $x_t$ . In particular the inner-product can be computed as:

$$732 \quad x_t^T q_j = h_t^T (C^T C) d_j = h_t^T (U + U^T) d_j$$

$$733 \quad = (U^T h_t)^T d_j + h_t^T (U^T d_j) = g_t^T d_j + h_t^T e_j.$$

735 This can be done in  $\mathcal{O}(k)$  time, as we will show in the following.

736 In order to establish this key result about the complexity of the inner product,  
 737 which leads directly to an efficient algorithm for performing our iterative updates, we  
 738 will prove the following facts.

739 **Fact 1** We can compute the matrix  $U$  in  $\mathcal{O}(mk^2)$  (which is also the cost of computing  
 740  $C^T C$ )

741 **Fact 2** We can compute an  $m$ -sparse vector  $h_0 \in \mathbb{R}^p$  such that  $x_0 = Ch_0$  in time  
 742  $\mathcal{O}(m)$

743 **Fact 3** We can compute  $g_0 := U^T h_0$  in time  $\mathcal{O}(mk)$ .

744 **Fact 4** The matrix  $U^T D$  can be computed in time  $\mathcal{O}(nk^2)$

745 **Fact 5** All the scalar products  $q_i^T q_i$ , where  $q_i$  is a column of  $Q$  are computable in  
 746 time  $\mathcal{O}(nk)$

747 **Proof of Fact 1** The cost of computing  $C^T C$  can be estimated by the number  
 748 of scalar additions and multiplications involved. In fact the number of additions is  
 749 the same as the number of multiplications, so we need only track the number of  
 750 scalar multiplications. From the elementary properties of the  $k$ -sparse factorization,  
 751 it follows that there are at most  $k$  entries per row. In the course of computing the  
 752 entries of  $C^T C$  all the scalar products between the  $p$  columns of  $C$  will be computed.

753 Thus we find that every entry of the first row of  $C$  will be multiplied with every of the  
 754  $k$  (or less) entries of first row, which gives  $\mathcal{O}(k^2/2)$  scalar multiplications associated  
 755 to the entries of the first row. Since every row can be treated similarly, the cost of  
 756 computing  $C^T C$  is at most  $\mathcal{O}(mk^2)$ .

757 **Proof of Fact 2** We can assume without loss of generality that the columns of  $C$   
 758 contain the canonical basis of  $\mathbb{R}^m$ . To see this, one can set  $\tilde{C} := (I_m \ C) \in \mathbb{R}^{m \times (p+m)}$   
 759 and  $\tilde{D} := (0 \ D^T)^T \in \mathbb{R}^{(p+m) \times n}$ . It then follows that for each column  $\tilde{c}_i$  of  $\tilde{C}$ ,  
 760  $|FO(\tilde{c}_i)| \leq k + 1$ , that  $\tilde{D}$  is  $(k + 1)$ -column sparse and that for each column  $\tilde{d}_j$ ,  
 761  $|\cup_{i \in \text{supp}(\tilde{d}_j)} FO(\tilde{c}_i)| \leq k + 1$ . Consequently, even though  $\tilde{D}$  has some zero rows, the  
 762 factorization  $\tilde{C}\tilde{D}$  has all the properties of a  $(k + 1)$ -sparse factorization and we say  
 763 that  $\tilde{C}\tilde{D}$  is  $(k + 1)$ -sparse. As a consequence, the running time does asymptotically  
 764 not depend on the choice of the decomposition  $CD$  or  $\tilde{C}\tilde{D}$ . Hence, we can assume  
 765 without loss of generality that a vector  $h_0 \in \mathbb{R}^p$ , such that  $x_0 = Ch_0$ , can be computed  
 766 in  $\mathcal{O}(m)$  time.

767 **Proof of Fact 3** Denote by  $U$  the  $p \times p$  upper triangular matrix such that  
 768  $C^T C = U^T + U$ . Notice that the  $i^{\text{th}}$  row of  $U$  is  $|FO(c_i)|$ -sparse. Since  $|FO(c_i)| \leq k$ ,  
 769 this implies that the matrix  $U$  is  $k$ -row sparse. Moreover, as each column of  $C$  is  
 770 assumed to be nonzero, we can deduce that  $U$  is invertible. Hence, given  $h_0$ , since  
 771  $U^T$  is  $k$ -column sparse, we compute the vector  $g_0 := U^T h_0$  in time  $\mathcal{O}(mk)$ .

772 **Proof of Fact 4** Let  $d_j$  be a column of  $D$ , which is  $k$ -sparse. Then, since  $Q = CD$   
 773 is a  $k$ -sparse factorization, the vector  $e_j := U^T d_j$  is  $k$ -sparse and is computed in time  
 774  $\mathcal{O}(k^2)$ . Consequently, we can compute the matrix product  $U^T D$ , i.e., all vectors  $e_i$  in  
 775  $\mathcal{O}(nk^2)$ .

776 **Proof of Fact 5** We compute any product  $q_i^T q_i$  as:

$$777 \quad q_i^T q_i = d_i^T (C^T C) d_i = d_i^T (U + U^T) d_i = (U^T d_i)^T d_i + d_i^T (U^T d_i) = e_i^T d_i + d_i^T e_i.$$

779 Since  $e_i$  and  $d_i$  are  $k$ -sparse, it takes  $\mathcal{O}(k)$  time to compute  $q_i^T q_i$ , and thus  $\mathcal{O}(nk)$  to  
 780 compute all the products.

781 **Appendix B. Fast iterative algorithms.** Following the analogous reasoning  
 782 as in the proof of fact 5, we see that

$$783 \quad x_i^T q_j = g_i^T d_j + h_i^T e_j$$

785 is also computable in  $\mathcal{O}(k)$  time. Hence, we can compute a first iteration of (17)  
 786 efficiently.

787 In order to make this computational gain available at every iteration, we have to  
 788 find a way to update  $h_t$  and  $g_t$  in a fast manner, too. Given  $h_t, g_t \in \mathbb{R}^p$  such that  
 789  $x_t = Ch_t$  and  $g_t = U^T h_t$  and given  $e_j = U^T d_j$ , the vectors

$$790 \quad h_{t+1} := h_t - \frac{x_t^T q_j}{q_j^T q_j} d_j$$

$$791 \quad g_{t+1} = g_t - \frac{x_t^T q_j}{q_j^T q_j} e_j$$

793 are such that  $x_{t+1} = Ch_{t+1}$  and  $g_{t+1} = U^T h_{t+1}$ . Moreover, from fact 2 and 3, and the  
 794 sparsity of  $d_j$ , it follows that the vectors  $h_{t+1}$  and  $g_{t+1}$  are computed in time  $\mathcal{O}(k)$ .

795 Consequently, at each iteration, we only need the vectors  $h_t, g_t, d_j$  and  $e_j$  in order  
 796 to compute  $h_{t+1}$  and  $g_{t+1}$ . Note that both  $h_{t+1}$  and  $g_{t+1}$  are required to compute

797 the scalar product  $x_{t+1}^T q_j$  (needed in the next iteration) in time  $\mathcal{O}(k)$ . Finally, the  
 798 approximate solution after  $N$  steps,  $x_N$ , is computed from the relation  $x_N = Ch_N$ .  
 799 This can be done in  $\mathcal{O}(mk)$  time due to the sparsity of  $C$ .

Combining these results, it follows that  $N$  iterative updates can be performed in time

$$\mathcal{O}(Nk + pk + (m + n)k^2).$$

Finally, the computation of  $y_N$  such that  $x_N = x_0 + Qy_N$  can be performed while computing  $x_N$  with the above described method without additional costs. Indeed, start with  $y_0 = 0$ . If the  $(t + 1)^{th}$  iteration is

$$x_{t+1} = x_t - \frac{x_t^T q_j}{q_j^T q_j} q_j,$$

800 then  $y_{t+1}$  corresponds to updating the  $j^{th}$  entry of  $y_t$  by adding  $-\frac{x_t^T q_j}{q_j^T q_j}$ . As the  
 801 required scalar products are computed for  $x_{t+1}$ , no additional cost is incurred.  $\square$

802 **B.1. Relationships to randomized Kaczmarz and randomized coordinate descent.** In the following we discuss how the iterative updates we discuss in  
 803 Section 2 can be interpreted from the lens of (randomized) Kaczmarz and (randomized)  
 804 coordinate descent methods.  
 805

806 **B.1.1. The underdetermined case.** We consider finding the minimum norm  
 807 solution for a consistent linear system  $Ax = B$  where  $A$  is an  $n \times m$  matrix with  
 808  $m > n$ . As discussed in Section 2, given any initial solution  $x_0$ , this can be achieved  
 809 by iteratively updating  $x$ , by projecting it onto the hyperplane orthogonal to the  
 810 vectors  $q_i$ :

$$811 \quad (18) \quad x_{t+1} = \left[ I - \frac{q_i q_i^T}{q_i^T q_i} \right] x_t = x_t - \frac{x_t^T q_i}{\|q_i\|^2} q_i,$$

812 where the update directions  $q_i$  lie within the null-space of  $A$ . Stated differently, the  
 813 matrix  $Q = [q_1, q_2, \dots]$  fulfils  $AQ = 0$ .

814 Now we can relate the above to the Kaczmarz scheme as follows: Let us denote  
 815 the row vectors of  $A$  by  $a_i^T$  ( $i \in 1, \dots, n$ ). One update step according to the Kaczmarz  
 816 scheme is defined as:

$$817 \quad (19) \quad x_{t+1} = x_t + \frac{b^i - a_i^T x_t}{a_i^T a_i} a_i,$$

818 where  $b^i$  is the  $i$ -th component of the right hand side.

819 To see that finding this minimal norm solution via the update (18) can indeed  
 820 be interpreted as Kaczmarz update scheme, let us define the augmented  $m \times m$  linear  
 821 system:

$$822 \quad (20) \quad A'x = \begin{pmatrix} A \\ Q^T \end{pmatrix} x = \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

823 Note the (unique) solution to this system is indeed the minimum norm solution of  
 824  $Ax = b$ .

825 Let us now consider iteratively solving (20) according to the Kaczmarz scheme.  
 826 Since we assumed that we start with an initial condition  $x_0$  that fulfils  $Ax_0 = b$ , the  
 827 first  $m$  equations are automatically fulfilled. Given that the right hand side has to be  
 828 zero for the  $m - n$  equations for the solution to be of minimum norm, we can easily  
 829 see that all the updates are indeed of the desired form.

830 *Finding a feasible solution  $x_0$ .* Let us briefly discuss the scenario that we cannot  
 831 obtain a feasible solution  $x_0$  in a simple manner, but the matrix  $(A')^T$  in (20) is  
 832 sparsely factorizable. As using our  $k$ -sparse factorization, all inner-products are cheap  
 833 to compute, we can also compute iterations of the form (19) efficiently. In particular,  
 834 for a compatible square system of the form  $Ax = b$ , where  $A^T$  is sparsely factorizable  
 835 (say  $Q = IA^T$  is  $k$ -column sparse), we can employ our  $k$ -sparse matrix factorization  
 836 to compute any iteration of the form (19) in  $\mathcal{O}(k)$  time.

837 **B.1.2. The overdetermined case.** In this case we have a system of the form  
 838  $Ay = b$  where  $a$  is an  $n \times m$  matrix with  $m < n$ . Let us define  $x = Ay - b$  as discussed  
 839 in Section 2. From the analytical solution to the normal equations  $A^T Ay = A^T b$   
 840 we know that we must have  $A^T x = 0$ . Whence, if we choose  $Q = A$  in our update  
 841 rule (18), this is exactly equivalent to an update of the form (19), and can be solved  
 842 efficiently using our  $k$ -sparse matrix factorization.

As discussed by Gower and Richtarik [9, 10] the dual update in  $y$  simply corresponds to coordinate descent:

$$y_{t+1} = y_t - \frac{(Qy - b)^T q_i}{\|q_i\|^2} e_i,$$

843 where  $e_i$  is the  $i$ th unit vector. Indeed by keeping track of the step sizes  $\alpha_t^* =$   
 844  $(q_i^T x_t / q_i^T q_i) q_i$  we effectively construct  $y^*$  such that  $Qy^* + x_0 = x^*$  in (5).

#### 845 Appendix C. Semiseparable and hierarchical matrices.

846 LEMMA 18. *The number of columns in  $C$  in the recursive construction in (15) in*  
 847 *Section 4.1.1 is given by  $p \leq rd^2 n$ .*

848 *Proof.* By induction on  $n$ , we prove that  $p \leq rd(d-1) \left( \frac{d}{d-1} n - \frac{1}{d-1} \right)$ .

- 849 1. If  $n = 2$ , then  $d = 2$  and  $p \leq 4 \leq rd(d-1) \left( \frac{d}{d-1} n - \frac{1}{d-1} \right) \leq rd^2 n$ .
- 850 2. If  $n > 2$ , then  $E$  is of the form (14). Let us denote the size of a diagonal  
 851 block  $E_{I_i \times I_i}$  by  $n_i$ , so we have  $n = \sum_{i=1}^d n_i$ . Now, from the construction of  
 852  $C$  we know that  $p \leq r(d-1)d + \sum_{i=1}^d p_i$ , where  $p_i$  is the maximum number of  
 853 columns in the matrix  $C_i$  of  $E_{I_i \times I_i}$  ( $1 \leq i \leq d$ ). Consequently, by induction  
 854 we have

$$855 \quad p \leq r(d-1)d + r(d-1)d \sum_{i=1}^d \left( \frac{d}{d-1} n_i - \frac{1}{d-1} \right)$$

$$856 \quad = r(d-1)d \left( \frac{d}{d-1} n - \frac{1}{d-1} \right) \leq rd^2 n. \quad \square$$

858 THEOREM 19 (Theorem 9). *An  $n \times n$  matrix that is  $(p, q)$ -semiseparable is an*  
 859  *$\mathcal{H}_r(\mathcal{P})$ -matrix where  $r = \max\{p, q\}$  and  $\mathcal{P}$  is a binary dendrogram.*

860 *Proof.* Following the definition, we have  $1 \leq p, q \leq n$  and we assume without loss  
 861 of generality that  $n \geq 2$ . Now, let  $E$  be an  $n \times n$  matrix which is  $(p, q)$ -semiseparable  
 862 and let  $\{I_1, I_2\}$  be a partition of  $\mathcal{I} = \{1, \dots, n\}$  with  $I_1 = \{1, \dots, \lfloor \frac{n}{2} \rfloor\}$  and  $I_2 = \mathcal{I} \setminus I_1$ .  
 863 Consider an integer  $i_1 \in \mathcal{I}$  such that  $\lfloor \frac{n}{2} \rfloor - q + 1 \leq i_1 \leq \min\{\lfloor \frac{n}{2} \rfloor, n - q + 1\}$ . Then,  
 864 the submatrix  $E(1 : i_1 + q - 1, i_1 : n)$  is of rank  $\ell_1 \leq q$  and contains  $E_{I_1 \times I_2}$ .

865 Similarly, if  $i_2 \in \mathcal{I}$  such that  $\lfloor \frac{n}{2} \rfloor - p + 1 \leq i_2 \leq \min\{\lfloor \frac{n}{2} \rfloor, n - p + 1\}$ , then the  
 866 submatrix  $E(i_2 : n, 1 : i_2 + p - 1)$  is of rank  $\ell_2 \leq p$  and contains  $E_{I_2 \times I_1}$ . Therefore,

867 we have shown that the off-diagonal blocks  $E_{I_1 \times I_2}$  and  $E_{I_2 \times I_1}$  are of a rank smaller  
868 of equal to  $r$ .

869 From the definition of semiseparable matrix, it follows that the diagonal blocks  
870  $E_{I_1 \times I_1}$  and  $E_{I_2 \times I_2}$  of  $E$  are also  $(p, q)$ -semiseparable matrices. Repeating the previous  
871 argument recursively on  $E_{I_1 \times I_1}$  and  $E_{I_2 \times I_2}$  shows that  $E$  is an  $\mathcal{H}_r(\mathcal{P})$ -matrix with  $\mathcal{P}$   
872 being a binary dendrogram, i.e  $d = 2$ .  $\square$

873 **Appendix D. Convergence rate and required number of iterations for**  
874 **randomly sampled search directions.** The proof is due to Strohmer and Ver-  
875 shyinin [25] and has originally been given in the context of a randomized Kaczmarz's  
876 method for solving linear systems. The version we give here is adapted to the context  
877 of this paper.

We want to establish the speed of convergence of iterations (2), when each column  $q_i$  of the matrix  $Q$  is chosen with probability proportional to  $\|q_i\|^2$ . In order to do so, for any  $x$  we first consider the auxiliary quantity

$$\sum_i \langle x, q_i \rangle^2 = x^T Q Q^T x \geq \sigma_{\min}^2(Q) \|x\|^2.$$

Here  $\langle x, q_i \rangle$  denotes the usual scalar product  $x^T q_i$ . If each direction  $q_i$  is selected with probability  $p_i = \|q_i\|^2 / \sum_j \|q_j\|^2 = \langle q_i, q_i \rangle / \|Q\|_{\text{Frob}}^2$ , we can rewrite this inequality as

$$\sum_i p_i \langle x, q_i \rangle^2 / \langle q_i, q_i \rangle \geq \frac{\sigma_{\min}^2(Q)}{\|Q\|_{\text{Frob}}^2} \|x\|^2.$$

878 Now, we know that  $x^*$ , the minimum norm point in the affine space  $x_0 + \text{span}\{q_i\}$ ,  
879 must be orthogonal to all directions  $q_i$  in the space. It thus follows that  $\langle x, q_i \rangle =$   
880  $\langle x - x^*, q_i \rangle$ . Therefore, we can write:

$$881 \quad (21) \quad \sum_i p_i \langle x, q_i \rangle^2 / \langle q_i, q_i \rangle \geq \frac{\sigma_{\min}^2(Q)}{\|Q\|_{\text{Frob}}^2} \|x - x^*\|^2.$$

Furthermore, we have

$$\|x_t - x^*\|^2 = \|(x_{t+1} - x^*) - (x_{t+1} - x_t)\|^2 = \|x_{t+1} - x^*\|^2 + \|x_{t+1} - x_t\|^2.$$

The second equality is due to orthogonality

$$\langle x_{t+1} - x^*, x_{t+1} - x_t \rangle = 0 = \langle x_{t+1} - x^*, \text{const} \cdot q_i \rangle.$$

882 This can be checked from the two following observations. First,  $x^*$  is orthogonal to all  
883 directions  $q_i$  in the affine space, as it is the point with the minimal norm in our affine  
884 space. Second,  $x_{t+1}$  is computed as the minimum norm point on the line  $x_t + \alpha_t q_i$ ,  
885 and is therefore also orthogonal to the current search direction  $q_i$ . Thus the error  
886  $x_{t+1} - x^*$  is also orthogonal to search direction  $q_i$ .

887 Finally we combine the results and observe that the expected value of the error  
888 norm

$$889 \quad (22) \quad \sum_i p_i \|x_{t+1} - x^*\|^2 = \sum_i p_i \|x_t - x^*\|^2 - \sum_i p_i \|x_{t+1} - x_t\|^2$$

$$890 \quad (23) \quad = \|x_t - x^*\|^2 - \sum_i p_i \frac{\langle x_t, q_i \rangle^2}{\langle q_i, q_i \rangle^2} \|q_i\|^2 \leq \left(1 - \frac{\sigma_{\min}^2(Q)}{\|Q\|_{\text{Frob}}^2}\right) \|x_t - x^*\|^2,$$

891

892 which is the desired result.