Fast Active-Set Thresholding Method for Nonnegative Least Squares

Benjamin Cobb*, Ramakrishnan Kannan†, Konstantin Pieper†, Piyush Sao†,
Yongseok Soh‡, Jee W. Choi‡, Richard Vuduc*, Haesun Park*
*Georgia Tech, Atlanta, GA, USA
†Oak Ridge National Laboratory, Oak Ridge, TN, USA
‡University of Oregon, Eugene, OR, USA

***Abstract*—Nonnegative Least Squares (NNLS) is a fundamental constrained optimization problem encountered in many applications such as image deblurring, signal processing, nonnegative matrix factorization, magnetic microscopy, and hyperspectral imaging. Active-set based methods are a common class of algorithms for solving NNLS which identify the optimal variable set of the NNLS solution. They do so by iteratively solving a series of unconstrained least squares problems, identifying which variables violate the nonnegativity constraints, and then swapping variables in/out of consideration until the optimal set of variables is found. Several variations improving upon this method exist in the literature. In this work, we propose an active-set swap heuristic which further improves upon existing active-set based methods for NNLS. Our optimizations are based upon adding multiple variables to the passive set within a threshold of the smallest gradient value and removing variables within a similar threshold of the closest boundary constraint. We leverage these optimizations to yield a Fast Active-Set Thresholding NNLS (FAST-NNLS) algorithm which significantly outperforms the existing state-of-the-art NNLS algorithms for a wide range of problems. Rigorous convergence guarantees are proven for the proposed method. We demonstrate the effectiveness of our proposed method on multiple synthetic datasets and two real-world text analysis applications. In doing so, we present the most comprehensive NNLS solver comparison in the literature to date.**

***Index Terms*—Nonnegative Least Squares, Active-Set Methods**

I. Introduction

Nonnegative Least Squares (NNLS) is a fundamental problem in data modeling that arises when solving linear systems $\mathbf{Ax} = \mathbf{b}$ with the additional constraint that all elements of $\mathbf{x}$ must be nonnegative. This constraint is crucial in many real-world applications where the unknown parameters $\mathbf{x}$ represent quantities that are inherently nonnegative, such as physical measurements, probabilities, or material properties. The NNLS formulation ensures feasible and interpretable solutions while preserving the model's linear structure, making it applicable across various domains.

In signal processing and imaging, it addresses tasks such as image deblurring [1], hyperspectral image analysis [2], [3], and magnetic microscopy [4], [5], ensuring interpretable, physically meaningful solutions. In acoustics, it supports source mapping and beamforming to localize and reconstruct sound fields [6], [7]. For sparse recovery and optimization, NNLS proves effective in sparse signal recovery [8], radiation dosage planning for cancer treatment [9], and other sparse optimization problems [10]–[12]. In control and engineering, it aids in model predictive control for system optimization under nonnegativity constraints [13], as well as battery balancing in electric vehicles [14]. NNLS is integral to numerical analysis and a range of theoretical applications where nonnegative solutions are required [15] such as Nonnegative Matrix and Tensor Factorization (NMF/NTF) [16]–[22].

The NNLS problem can be formulated as:

$$\min_{\mathbf{x} \geqslant 0} \|\mathbf{Ax} - \mathbf{b}\|_2^2, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^{n \times 1}, \mathbf{x} \in \mathbb{R}^{n \times 1}$ and $m \geqslant n$.

Active-set methods are common for solving NNLS problems due to their guaranteed convergence to the optimal solution in a finite number of iterations, practical real-world performance, and ease of implementation. These methods iteratively classify variables as either *active* (violating nonnegativity constraints) or *passive* (satisfying constraints) to identify the non-zero variables of the optimal solution. One of the primary differentiator between active-set methods is the rule by which they determine what variables to switch between the active and passive sets. These rules vary from conservative single-swap rules [23] to variants that evaluate objective function values for determining the optimal number of variables to swap [24] to greedy variants that attempt to swap all infeasible variables [25]. The choice of swap rule impacts factors such as passive set size, number of function evaluations, and iteration count all of which heavily influence performance.

Published in Proc. IEEE BigData 2025. DOI: https://doi.org/10.1109/BigData66926.2025.11402623

Corresponding author: Benjamin Cobb (bcobb33@gatech.edu)

We propose Fast Active-Set Thresholding NNLS (FAST-NNLS), an adaptive threshold-based swapping rule which maintains small passive set sizes, avoids expensive function evaluations, and empirically converges in few iterations. We prove that our proposed FAST-NNLS method is guaranteed to converge in a finite number of iterations to the optimal solution. We show that FAST-NNLS excels at NNLS problems with small optimal passive set to column count ratios. Empirical tests on synthetic and real-world datasets demonstrate FAST-NNLS consistently outperforms existing state-of-the-art methods in this problem class in terms of both runtime and solution quality. Our NNLS experiments constitute the most comprehensive comparison of NNLS methods in the literature to date.

II. Preliminaries and Related Work

We now cover a brief survey of the NNLS literature and introduce the notation used throughout this paper.

A. NNLS Algorithms

Due to the ubiquitousness of the NNLS problem many approaches in the literature have been developed for solving it. In this work we extend the class of active-set based methods derived from [23] for solving NNLS and compare our proposed algorithms against a wide variety of NNLS optimization methods.

NNLS is a subset of the box-constrained optimization problem. Thus many box-constrained optimization methods can be adapted to solve NNLS, as is done in [26]. [26] develops a Projected Quasi-Newton Limited-memory Broyden-Fletcher-Goldfarb-Shanno (PQN-LBFGS) method for solving box-constrained optimization problems which they apply to NNLS as well as Nonnegative Kullback-Leibler (NNKL) problems. In doing so, they compare against the popular TRON [27] and LBFGS-B [28] bound constrained optimization methods on both NNLS and NNKL problems. In this work, we compare our proposed methods to both the PQN-LBFGS and LBFGS-B [29], [30] implementations.

Additional quasi-Newton and projected gradient NNLS methods can be found in [31]–[35]. An ADMM method for the NNLS problem is proposed in [36].

Interior point methods are another class of optimization methods which have been applied to the NNLS problem. An interior point method capable of converging even in the case of solution degeneracy is proposed in [37]. An approach for converting the NNLS problem to the linear complementarily problem and then solving it via a feasible interior point algorithm is presented in [38]. An interior point gradient method for totally nonnegative least squares is presented in [39]. Another interior point method for large-scale totally nonnegative least squares which establishes global convergence is presented in [40]. The aforementioned interior-point NNLS methods do not provide implementations and we thus do not directly compare our proposed methods to them. We compare against the MatLab [41] built-in functions *solve* and *lsqlin* for bound constrained optimization using the default interior-point solvers backends.

Various other optimization methods have been proposed for the NNLS problem. An index-search method for NNLS is proposed in [42]. A random projection based approach is proposed in [43]. [44] presents the Subspace Barzilai-Borwein (SBB) NNLS method based upon the unconstrained Barzilai-Borwein optimization. They demonstrate competitive performance on large-scale, sparse problems. In our experiments, we compare against the implementation used in [44] and confirmed the speed of the SBB method on several large, sparse problems.

Active-set-based methods [16], [23]–[25], [45]–[50] are one the most predominant classes of optimization methods used for solving NNLS. They are founded on the observation that, if the nonzero elements of the final solution are known beforehand, the problem can be reduced to an unconstrained least squares problem over the nonzero elements, while the remaining elements are fixed at zero. *Working sets* are employed to track the active and passive sets throughout the optimization process, until the optimal solution is reached. We refer the reader to [51] for an in-depth comparison of traditional active-set, predictor-corrector, and interior-point methods applied to NNLS. We also compare the implementations presented in [51] against our proposed methods.

While there are a wide variety of approaches to solving NNLS, including various aforementioned constrained optimization techniques and hybrid methods, this paper will primarily focus on extending active-set based methods for solving the NNLS problem.

B. Notation

We begin by outlining the key notations and terminology used throughout this work. Matrices are denoted by bold capital letters, such as $\mathbf{A} \in \mathbb{R}^{I \times J}$, while vectors are represented by bold lowercase letters, such as $\mathbf{a} \in \mathbb{R}^{I}$. Scalars are written as regular lowercase letters (e.g., a). All matrices and vectors are assumed to consist of real numbers. We use the following notation for the Euclidean inner product $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$ and norm $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}}$. $\mathbf{A} \geqslant 0$ denotes a nonnegative matrix, where for every element a_{ij} of $\mathbf{A} = (a_{ij})$, we have $a_{ij} \geqslant 0$ for all i and j . In the context of active-set methods, we use $\mathcal{P}$ and $\mathcal{A}$ to represent passive and active sets. Matrix $\mathbf{A}^{\mathcal{P}}$ is a sub-matrix of $\mathbf{A}$ that contains only the columns of $\mathbf{A}$ corresponding to the indices in the passive set $\mathcal{P}$. We define $\mathbf{x}_{\mathcal{P}}$ similarly for the vector case. We use MatLab notation to index into matrices as well as elementary MatLab built-in functions such as min and max.

III. Active-Set Based Methods

In this section we provide a detailed discussion of active-set based methods for NNLS. We describe the criteria for building the optimal passive set employed by the Lawson-Hanson single-swap [23] and Block Principal Pivoting (BPP) [25] methods before introducing the passive set criteria of our proposed FAST-NNLS method in the subsequent section.

A. Active-Set (Act): Lawson and Hanson [23]

The foundational active-set method for NNLS was introduced by [23]. This method starts with an empty passive set and an active set containing all variables. At each iteration the value with the smallest gradient value is moved from the active to passive set. An unconstrained least squares is then computed for all variables and columns of $\mathbf{A}$ corresponding to the passive set. In the instance that this unconstrained solution is infeasible, the variable with the closest proximity to the nearest boundary condition is removed and the unconstrained least squares is evaluated again. This process is repeated until the solution is feasible and the KKT conditions are met, at which point the optimal solution is found. Details of the method can be found in the original paper [23]. Note that in this algorithm, a single variable is exchanged between the passive and active-set per unconstrained solve.

B. Block Principal Pivoting (BPP): Júdice and Pires [25]

Block Principal Pivoting (BPP) NNLS methods aim to improve upon the single exchange rule of the [23] method by exchanging multiple variables per unconstrained solve [16], [25], [51]. Here we introduce several of the details of the BPP method's active-set swap heuristic which will be useful in laying the foundation for our proposed approach.

Th BPP NNLS method starts by initializing the passive and active sets, denoted respectively as $\mathcal{P}$ and $\mathcal{A}$. It then computes $\mathbf{x}_{\mathcal{P}}$ by solving an unconstrained least squares problem and $\mathbf{y}_{\mathcal{A}}$ using Eq. 2, where the $\mathcal{P}$ and $\mathcal{A}$ subscripts indicate indices belonging to the corresponding passive/active sets. The pair $(\mathbf{x}_{\mathcal{P}}, \mathbf{y}_{\mathcal{A}})$ is a *complementary basic solution*, deemed *feasible* if $\mathbf{x}_{\mathcal{P}} \geqslant 0$ and $\mathbf{y}_{\mathcal{A}} \geqslant 0$ in which case the optimal solution has been found , and *infeasible* otherwise.

$$\mathbf{x}_{\mathcal{P}} = \operatorname{argmin}_{\mathbf{x}_{\mathcal{P}}} \|\mathbf{A}_{\mathcal{P}}\mathbf{x}_{\mathcal{P}} - \mathbf{b}\|_2^2 \tag{2}$$

$$\mathbf{y}_{\mathcal{A}} = \mathbf{A}_{\mathcal{A}}^T(\mathbf{A}_{\mathcal{P}}\mathbf{x}_{\mathcal{P}} - \mathbf{b}) \tag{3}$$

The index set of infeasible variables $\mathcal{V}$ is defined as:

$$\mathcal{V} = \{i \in \mathcal{P} : x_i < 0\} \cup \{i \in \mathcal{A} : y_i < 0\} \tag{4}$$

Working sets $\mathcal{P}$ and $\mathcal{A}$ are updated by swapping indices:

$$\mathcal{P} = (\mathcal{P} - \hat{\mathcal{V}}) \cup (\hat{\mathcal{V}} \cap \mathcal{A}) \tag{5}$$

$$\mathcal{A} = (\mathcal{A} - \hat{\mathcal{V}}) \cup (\hat{\mathcal{V}} \cap \mathcal{P}) \tag{6}$$

where $\hat{\mathcal{V}}$ is a subset of $\mathcal{V}$ which represents the set of variables that are swapped per iteration. BPP typically uses a full exchange rule ($\mathcal{V} = \hat{\mathcal{V}}$), swapping all variables in one iteration to accelerate the process of finding the optimal passive set. If this fails consecutively, it falls back to a single exchange rule of exchanging the largest index of $\mathcal{V}$ [25] until a new minimum number of infeasible variables is reached:

$$\hat{\mathcal{V}} = \{i : i = \max \mathcal{V}\} \tag{7}$$

C. Additional Active-set NNLS Methods

TNT-NN [24] is an active-set NNLS solver used in rock magnetism analysis. It evaluates potential active/passive set swaps based on their ability to minimize the 2-norm of the residual. This evaluation process is more computationally expensive than single-swap and BPP heuristics.

Lawson-Hanson with Deviation Maximization [50] is a block active-set method for NNLS that extends the Lawson-Hanson method designed to leverage arithmetically intense linear algebra operations. Finite convergence of the method is proved. A high performance C implementation with MatLab wrappers is provided which we compare against in our experiments.

Luo et al. [49] present a QR-based active-set NNLS method using the single exchange rule from the Lawson-Hanson active-set method. To remove an element from the passive set, they propose using modified Gram-Schmidt or Givens rotations to update the QR factorization.

IV. Fast Active Set Thresholding Method

We propose a heuristic for swapping variables between active and passive sets to reduce the computational cost of Equation 33 whilst preserving guarantees of convergence to the optimal solution. Our proposed FAST-NNLS approach aims to decrease outer iterations (K) without excessively enlarging passive sets ($\mathcal{P}_i$), based on swapping variables with gradient values within a certain percentage of the smallest gradient value. This heuristic combines elements from [16], [24], [25], [45]. While BPP [25] reduces iterations and Lawson-Hanson [23], [45] limits passive set size, TNT-NN [24] achieves both but at a higher swap heuristic computational cost. The experimental results show that FAST-NNLS attains all three of small passive set size, low iteration count, and inexpensive variable swap selection.

A. FAST-NNLS Setup

We define the objective function of Equation (1):

$$\begin{aligned} f(\mathbf{x}) =& \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 \\ =& \frac{1}{2}\langle \mathbf{x}, \mathbf{A}^T\mathbf{A}\mathbf{x}\rangle - \langle \mathbf{A}^T\mathbf{b}, \mathbf{x}\rangle + \frac{1}{2}\|\mathbf{b}\|^2 \end{aligned} \tag{8}$$

We generate a sequence of iterates $\mathbf{x}^k \geqslant 0$ in iteration $k \in \mathbb{N}_0$, starting at $\mathbf{x}^0 \geqslant 0$. Let $\mathbf{x} \in \mathbb{R}^n$ and denote by $\mathcal{A} = \{i \mid \mathbf{x}_i = 0\}$ the active set associated to $\mathbf{x}$ and by $\{1, 2, \ldots n\} \setminus \mathcal{A} = \mathcal{A}^c = \mathcal{P}$ the passive set.

In each iteration, we are going to perform one unconstrained least squares problem solve, and we are then either going to add new variables to the passive set $\mathcal{P}$, or remove from it (and vice-versa for the active set $\mathcal{A}$, its complement).

B. Adding Points to Passive Set

If $\mathbf{x}^0 = 0$ we start here, otherwise in the removal stage. Let $\mathbf{x}^{k-1}$ be the old iterate. Adding variables to $\mathcal{P}$ has a precondition, which is that for $\mathbf{x}^{k-1}$ and the associated passive set $\mathcal{P}^{k-1} = \{i \mid \mathbf{x}_i^{k-1} > 0\}$ the solution to the problem:

$$\mathbf{z}^{k-1} = \mathrm{argmin}_{\mathbf{z}\in\mathbb{R}^n, \mathbf{z}|\mathcal{A}^{k-1}=0} \frac{1}{2}\|\mathbf{A}\mathbf{z} - \mathbf{b}\|^2 \tag{9}$$

is equal to $\mathbf{x}^{k-1}$. Here $\mathcal{A}^{k-1}$ is the active set of $\mathbf{x}^{k-1}$ and equal to $(\mathcal{P}^k)^c$. We also define the gradient:

$$\mathbf{g}^{k-1} = \nabla f(\mathbf{x}^{k-1}) = \mathbf{A}^T(\mathbf{A}\mathbf{x}^{k-1} - \mathbf{b}) \tag{10}$$

Due to $\mathbf{x}^{k-1} = \mathbf{z}^{k-1}$ minimizing the least squares loss over the passive set $\mathcal{P}^{k-1}$, we have:

$$\mathbf{g}_i^{k-1} = 0 \quad \text{for all } i \in \mathcal{P}^{k-1}. \tag{11}$$

We note that in this situation, the set of infeasible variables contains only indices where the dual variable is infeasible:

$$\mathcal{V}^{k-1} = \{\, i \mid \mathbf{g}_i^{k-1} < 0 \,\} \tag{12}$$

We now select some candidate set $\hat{\mathcal{V}} \in \mathcal{V}^{k-1}$ to be added to the passive set and define the resulting passive set as:

$$\mathcal{P}' = \mathcal{P}^{k-1} \cup \hat{\mathcal{V}}. \tag{13}$$

The Lawson-Hanson single-swap method uses the following exchange rule:

$$\hat{\mathcal{V}} = \{\hat{\imath} \mid \hat{\imath} = \mathrm{argmin}\, \mathbf{g}_i^{k-1}\} \tag{14}$$

We propose a new criteria for selecting $\hat{\mathcal{V}}$ as follows:

$$\hat{\mathcal{V}} = \{i \mid \mathbf{g}_i^{k-1} \leqslant \mathbf{g}_{\hat{\imath}}^{k-1}(1-\gamma)\} \tag{15}$$

Where $\mathbf{g}_{\hat{\imath}}^{k-1} = \min \mathbf{g}_i^{k-1}$ and γ is the passive set addition thresholding parameter. This can be interpreted as selecting all variables within a certain threshold to the smallest gradient value to add to the passive set.

After adding variables to the passive set, eq. (13), we check whether a new number of infeasible variables has been attained. We define the number of infeasible variables at iteration k as:

$$\upsilon_k = |\mathcal{V}^k|, \tag{16}$$

where $\mathcal{V}$ is as defined in eq. (4). We define the current minimum number of infeasible variables attained as:

$$\hat{\upsilon}_k = \min\{\, |\mathcal{V}^i| \mid i = 0, 1, \ldots k \,\} \tag{17}$$

If $\upsilon_k < \hat{\upsilon}_{k-1}$, we increase γ by

$$\gamma' = \gamma + \gamma^+, \tag{18}$$

where γ^+ the constant used to increase γ. Otherwise, $\upsilon_k \geqslant \hat{\upsilon}_{k-1}$, and we decrease γ by

$$\gamma' = \max(\gamma - \gamma^-, 0), \tag{19}$$

where γ^- the constant used to decrease γ. Note that if $\hat{\upsilon}^k$ does not decrease after a certain number of iterations, then $\gamma = 0$ and the addition phase becomes identical to eq. (14).

Once the variable set $\hat{\mathcal{V}}$ has been selected, they are added to the passive set as in eq. (13) and new iteration begins with an unconstrained least squares solve corresponding to the variables in $\mathcal{P}'$ described next.

C. Removing Points from Passive Set

In the case that the solution from the unconstrained least squares contains negative entries, we proceed to remove variables from the passive set. We solve the subproblem

$$\mathbf{z}' = \mathrm{argmin}_{\mathbf{z}\in\mathbb{R}^n, \mathbf{z}|\mathcal{A}'=0} \frac{1}{2}\|\mathbf{A}\mathbf{z} - \mathbf{b}\|^2, \tag{20}$$

where $\mathcal{A}' = (\mathcal{P}')^c$. If $\mathbf{z}' \geqslant 0$, then we set:

$$\mathbf{x}^k = \mathbf{z}', \quad \mathcal{A}^k = \{\mathbf{z}' = 0\}$$

and move back to adding variables as described in section IV-B. Otherwise, we perform a backtracking line-search:

$$\mathbf{x}_\tau = P(\mathbf{x}^{k-1} + \tau(\mathbf{z}' - \mathbf{x}^{k-1})) \tag{21}$$

We select a $\tau' \in (0, 1]$ in the breakpoint set:

$$\boldsymbol{\tau} = \left\{\tau_i = \min\left\{\tfrac{\mathbf{x}_i^{k-1}}{\mathbf{z}_i' - \mathbf{x}_i^{k-1}}, 1\right\} \mid i \in \mathcal{P}'\right\}. \tag{22}$$

The Lawson-Hanson single-swap method [23] selects $\tau' = \min \boldsymbol{\tau}$. If only at most a single variable was added in the previous step (according to eq. (14)) it is guaranteed that:

$$f(\mathbf{x}_{\tau'}) < f(\mathbf{x}^{k-1}) \tag{23}$$

and

$$\mathcal{P}_{\tau'} = \{\mathbf{x}_{\tau'} > 0\} \subsetneq \mathcal{P}', \tag{24}$$

and $\mathbf{x}^k = \mathbf{x}_{\tau'}$ and $\mathcal{P}^k = \mathcal{P}_{\tau'}$ is accepted. It then proceeds recompute eq. (20) and repeats the removal from $\mathcal{P}$ one variable at a time, if necessary.

We propose a new criteria for removing multiple variables from the passive set per evaluation of eq. (20). We do so by selecting multiple variables associated to τ_i's within a certain threshold. This is based upon the observation that the backtracking line-searches are computationally inexpensive relative to computing eq. (20). The τ values are selected as follows:

$$\mathcal{T} = \{\, i \mid \tau_i < \tau'(1+\rho) \,\}, \tag{25}$$

where ρ is the passive set removal thresholding parameter. For each $\tau \in \mathcal{T}$ in ascending order, we update $\mathbf{x}_\tau$ as in eq. (21). The variables corresponding to the entries of $\mathcal{T}$ are then removed from $\mathcal{P}$:

$$\mathcal{P}^k = \mathcal{P} \setminus \mathcal{T}. \tag{26}$$

After removing variables from the passive set in this manner and conversely adding them to the active set ($\mathcal{A}^k = \mathcal{A} \cup \mathcal{T}$), eq. (20) is recomputed. We then check whether a new minimum number of infeasible variables, previously defined as $\hat{\upsilon}^k$, has been attained. Then, similar to before, if $\upsilon_k < \hat{\upsilon}_{k-1}$, we increase ρ by

$$\rho' = \rho + \rho^+, \tag{27}$$

where ρ^+ the constant used to increase ρ. Otherwise, $\upsilon_k \geqslant \hat{\upsilon}_{k-1}$, and we decrease ρ by

$$\rho' = \max(\rho - \rho^-, 0), \tag{28}$$

where ρ^- the constant used to decrease ρ. Again note that if a new $\hat{\upsilon}_k$ does not decrease after a certain number of iterations, then $\rho = 0$ and the removal phase becomes identical to [23].

The pseudocode for the resulting FAST-NNLS method based upon these addition and removal rules can be seen in Algorithm 1.

Algorithm 1: FAST-NNLS

```
   Input: $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m, \rho, \rho^+, \rho^-, \gamma, \gamma^+, \gamma^-$
   Output: $\mathbf{x} \geqslant 0$ such that $\mathbf{x} = \text{argmin}\,\|\mathbf{Ax} - \mathbf{b}\|^2$
1  Initialize $\mathcal{P} = \emptyset$, $\mathcal{A} = \{1, \dots, n\}$, $\mathbf{z} = 0$;
   // Compute normal equations. We precompute
      these for all benchmarked methods.
2  $\mathbf{ATA} = \mathbf{A}^T\mathbf{A}$; $\mathbf{atb} = \mathbf{A}^T\mathbf{b}$;
3  while $0 < \hat{\upsilon}_k$ do
4      if $\min(\mathbf{z}) < 0$ then
5          Remove variables from $\mathcal{P}$ per eqs. (25) and (26);
6          Update $\mathbf{x}_\tau$ per eq. (21);
7      else
8          Add variables to $\mathcal{P}$ per eq. (15);
9          $\mathbf{x} = \mathbf{z}$;
10     Compute $\mathbf{z}$ per eq. (20);
11     Compute $\upsilon_k$ by eq. (16);
12     if $\upsilon_k < \hat{\upsilon}_{k-1}$ then
13         $\hat{\upsilon}_k = \upsilon_k$;
14     Update $\gamma$ (eqs. (18) and (19)) and $\rho$ (eqs. (27) and (28));
```

V. Proof of Guaranteed Finite Convergence to Optimal Solution

We now prove the convergence properties of our proposed FAST-NNLS method. To aid in the convergence proof, we start by proving that the Lawson-Hanson single-swap [23] strategy converges to the optimal solution in a finite number of iterations from any starting point.

Lemma 1 (Single-Swap Convergence). *The sequence $\mathbf{x}^k \geqslant 0$ generated by addition rule eq.* (14) *and removal rule $\tau' = \min \boldsymbol{\tau}$ from eq.* (22) *fulfills $f(\mathbf{x}^k) < f(\mathbf{x}^{k-1})$ and terminates after a finite number of iterates with the optimal solution.*

Proof. First we consider when a variable is swapped into $\mathcal{P}$. Since $\mathbf{z}'$ minimizes f over all $\mathbf{z} \in \mathbb{R}^n$ s.t. $\mathbf{z}|_{\mathcal{A}'} = 0$ and $\mathbf{x}^{k-1}|_{\mathcal{A}'} = 0$, we have:

$$f(\mathbf{z}') < f(\mathbf{x}^{k-1}). \tag{29}$$

To quantify the descent, we define an arbitrary stepsize $\tau \in [0, 1]$ and define the line connecting $\mathbf{x}^{k-1}$ to $\mathbf{z}'$ as $\mathbf{z}_\tau = \mathbf{x}^{k-1} + \tau\mathbf{d}'$ with $\mathbf{d}' = \mathbf{z}' - \mathbf{x}^{k-1}$ and calculate:

$$\begin{aligned} f(\mathbf{z}_\tau) &= \frac{1}{2}\|\mathbf{Az}' - \mathbf{b}\|^2 \\ &= \frac{1}{2}\left\|\mathbf{Ax}^{k-1} - \mathbf{b}\right\|^2 + \langle \mathbf{Ax}^{k-1} - \mathbf{b}, \mathbf{A}(\mathbf{z}_\tau - \mathbf{x}^{k-1})\rangle \\ &\quad + \frac{1}{2}\|\mathbf{A}(\mathbf{z}' - \mathbf{x}^{k-1})\|^2 \\ &= f(\mathbf{x}^{k-1}) + \tau\langle \mathbf{g}^{k-1}, \mathbf{d}'\rangle + \frac{\tau^2}{2}\|\mathbf{Ad}'\|^2. \end{aligned} \tag{30}$$

Thus, the scalar function $h(\tau) = f(\mathbf{z}_\tau)$ is a parabola. Since $\mathbf{z}'$ is the minimizer of f for all $\mathbf{z}|_{\mathcal{A}'} = 0$ and $\mathbf{z}_\tau|_{\mathcal{A}'} = 0$, we deduce that $\tau = 1$ is the minimum of the parabola h. By setting its derivative to zero, we obtain:

$$f(\mathbf{z}') - f(\mathbf{x}^{k-1}) = \frac{1}{2}\langle \mathbf{g}^{k-1}, \mathbf{d}'\rangle = -\frac{1}{2}\|\mathbf{Ad}'\|^2 < 0 \tag{31}$$

In the case that $\mathbf{z}' \geqslant 0$ we set $\tau = 1$ and we have found a better feasible solution with a lower objective function value than the previous $\mathbf{x}^{k-1}$. Following [23], we now show that this also holds for the case when entries of $\mathbf{z}'$ are infeasible and when infeasible variables are removed from $\mathcal{P}$. This involves backtracking along τ to where the first boundary constraint is encountered. This backtracking step can be viewed as taking a step along the parabola h until $\mathbf{z}_\tau$ starts to differ from $\mathbf{x}_\tau = P(\mathbf{z}_\tau)$, which is exactly the minimum breakpoint $\tau' = \min \boldsymbol{\tau}$. First, we argue that $\tau' > 0$. For this it suffices to show that the single newly added point $\hat{\imath}$ from the previous iterate has a positive entry. If no new point was added in the previous iterate, we have $\mathbf{x}^{k-1} > 0$ on $\mathcal{A}'$, which directly implies that $\tau' > 0$. In the other case this follows from $0 > \langle \mathbf{g}^{k-1}, \mathbf{d}'\rangle = \mathbf{g}^{k-1}_{\hat{\imath}}\mathbf{d}'_{\hat{\imath}}$ using eq. (11) and eq. (14). Since $\mathbf{g}^{k-1}_{\hat{\imath}} < 0$ with eq. (12), $\mathbf{d}'_{\hat{\imath}}$ must be positive.

Thus we can set $\mathbf{x}^k = \mathbf{x}_{\tau'} = \mathbf{z}_{\tau'} \geqslant 0$ with

$$f(\mathbf{x}^k) = f(\mathbf{z}_{\tau'}) = h(\tau') < h(0) = f(\mathbf{x}^{k-1}).$$

Thus, in each case the backtracking step to the closest boundary condition yields a new feasible solution that further decreases the objective function eq. (1).

If we insert at most one point in every iteration, in an amortized sense (averaged over all k), an iteration where no passive points are added can not happen more often than the number of iterations where a point was added (plus the number of entries in the initial passive set). For every iteration $\bar{k}$ where we have $\tau' = 1$ and $\mathbf{x}^{\bar{k}} = \mathbf{z}'$, we have an active set $\mathcal{A}^{\bar{k}}$ with:

$$f(\mathbf{x}^{\bar{k}}) = \text{argmin}_{\mathbf{x}=0 \text{ on } \mathcal{A}^{\bar{k}}} f(\mathbf{x}) < f(\mathbf{x}^{\bar{k}'}) \tag{32}$$

where $\bar{k}'$ is the prior iteration that $\tau' = 1$. This sequence of active sets is uniquely associated to this minimum value, and since it decreases monotonically a prior set can never be revisited, and thus the iteration must terminate, which only happens once the optimality conditions are fulfilled. □

Note that the above proof holds true for any starting passive set. We will now leverage this fact to prove the guaranteed convergence of our proposed FAST-NNLS method.

Theorem 1 (FAST-NNLS Convergence). *The sequence $\mathbf{x}^k \geqslant 0$ terminates with the optimal solution after a finite number of FAST-NNLS iterations as defined in Section IV and shown in Algorithm 1.*

Proof. We prove this by contradiction. Assume that the sequence $\mathbf{x}^k \geqslant 0$ does *not* terminate after a finite number of iterations. At each iteration, we add or remove variables as discussed in section IV. Only if a new value for $\hat{\upsilon}_k$ (eq. (17)) is attained in iteration k, i.e. $\hat{\upsilon}_k < \hat{\upsilon}_{k-1}$, we allow for the variable γ to be increased. Else, if a new value for $\hat{\upsilon}_k$ is not attained, γ will be decreased eq. (19) (and similarly for

ρ eq. (28)). This is repeated until a new $\hat{v}_k$ is attained or $\gamma = 0$ and $\rho = 0$. In the latter case, the FAST-NNLS method becomes identical to the Lawson-Hanson [23] single-swap method (Lemma 1). Once this occurs, γ and ρ cannot be incremented unless $\hat{v}_k$ is decreased.

As $\hat{v}_k$ can only decrease $n+1$ times, there is only a finite number of times γ or ρ can be increased. Assuming that the algorithm does not terminate, there exists a $\bar{k} > 0$ such that $v_k = \hat{v}_{\bar{k}}$ for all $k \geqslant \bar{k}$. However, since for $k > \bar{k}$ it performs an infinite number of removal and addition steps while decreasing γ and ρ, respectively, $\gamma = 0$ and $\rho = 0$ must hold for large enough k. This directly contradicts Lemma 1, and we thus conclude that Algorithm 1 does terminate after a finite number of iterations. □

VI. Cost Model

For most active-set based NNLS methods, evaluating the unconstrained least squares solutions is the most computationally expensive step. Our analysis and proposed heuristics aim to reduce this cost. Assuming each unconstrained least squares is solved via Cholesky decomposition, we propose the following approximate cost model:

$$\frac{1}{3}\sum_{k=1}^{K} |\mathcal{P}^k|^3 \tag{33}$$

Here, K is the total number of outer iterations (unconstrained least squares computed via Cholesky), and $|\mathcal{P}^k|$ is the passive set size at iteration k.

The computational cost can be reduced by reducing K or keeping $\mathcal{P}^k$ small. Several existing active-set methods implicitly achieve this. For instance, FNNLS [45]starts with an empty passive set and builds the passive set via the single-exchange rule, generally resulting in large K but small $\mathcal{P}^k$ values which are always less than or equal to the optimal passive set size.

The greedy full swap heuristic of the BPP method aims to reduce K by swapping all infeasible variables in each outer iteration which typically results in small K values. However, it often results in large passive sets, $\mathcal{P}^k$, especially in the first few iterations. We empirically observed that when starting with an empty passive set, the second outer iteration generally moves many variables into the passive set, potentially leading to unnecessarily large Cholesky factorizations. This can become computationally expensive for large n ($\mathbf{A} \in \mathbb{R}^{m\times n}$).

A. Single-Swap Active-Set Method Lower Asymptoptic Bound

Here we will derive a lower asymptoptic bound for active-set methods which add or remove a single variable at each iteration such as the Lawson-Hanson (Act) [23] and FNNLS [45] NNLS methods. Under the assumption that the starting passive set, $\mathcal{P}^0$, is initially empty and that a new optimal passive set variable is identified at each iteration, i.e. $\mathcal{P}^{k+1} = \mathcal{P}^k + 1$, we can apply Faulhaber's formula [52], [53] to Equation (33) to derive the following asymptotic lower bound:

$$\Omega(Act) = \frac{1}{12}\sum_{r=0}^{3} \binom{4}{r} \mathcal{B}_r K^{4-r} \tag{34}$$

Where K is as defined in Equation (33), $\binom{4}{r}$ is 4 *choose* r, and $\mathcal{B}_r$ is the r'th Bernoulli number (with $\mathcal{B}_1 = \frac{1}{2}$). Note that in this case as we assume a single optimal variable is added in each iteration to the passive set until the entire optimal passive set has been identified, K is both the number of iterations and the size of the optimal passive set.

In the experiments section, we empirically observed that the Lawson-Hanson single-swap active-set (Act) NNLS method closely followed this lower bound for well-conditioned problems.

VII. Experiments

All experiments were run on an Ubuntu 24.04 server with an i9-14900F processor and 192GB DDR5 RAM using MatLab R2024b with a single thread. Unless otherwise stated the FAST-NNLS hyperparameters were: γ=1, γ^+=.05, γ^-=.1, ρ=0, ρ^+=.05, ρ^-=.1. These hyperparameter values were selected based upon tuning experiments, an example of which can be seen in Figure 1. All methods, except the *lsqnonneg* MatLab built-in function, were constrained to a max runtime of 2 minutes with the recommended default hyperparameters for each method. Unless otherwise stated, all times are the average over 5 runs. Additionally, all open source MatLab approximation methods had the convergence criteria of $|f(\mathbf{x}^k) - f(\mathbf{x}^{k-1})| <$ 1e-6. Unless otherwise stated, all experiments are for a single right hand side (RHS), i.e $b \in \mathbb{R}^{m\times 1}$ to simplify analysis. In this work we are primarily interested in analyzing and optimizing the determination of the optimal passive set. We intend to provide rigorous experiments and analysis for the multiple RHS case in future work.

We do not directly include the time to form the normal equations in the timing comparisons in Figures 4 to 7. We do this as in practice the time to compute the normal equations is amortized over multiple RHS. When amortized over multiple RHS, the time to compute the normal equations is often relatively inconsequential, particularly for the problem configurations that we consider in this work. We intend to provide additional analysis into the cost of the normal equations relative to NNLS in future work. The normal equations were precomputed and used as input for all methods to account for their not being included in the timings. We empirically observed that this significantly improved the runtime of all methods except lsqlin without compromising solution quality.

Unless otherwise stated, when determining the passive set assignments in the Act, BPP, and FAST-NNLS active-set methods, values whose magnitude is less than 1e-12 are set to 0. In other words, when determining whether a value is nonnegative for the purposes of determining whether the variable is violating the nonnegativity constraints, any variable whose absolute value is less than 1e-12 is set to 0. This value cutoff has implications for both the numerical accuracy and stability of active-set NNLS solvers. We empirically observed

DW1	dense $(2^{12} \times 2^{11})$ well-conditioned
DI1	dense $(2^{12} \times 2^{11})$ ill-conditioned
DW2	dense $(2^{17} \times 2^{5})$ well-conditioned
DI2	dense $(2^{17} \times 2^{5})$ ill-conditioned
SS1	sparse $(2^{12} \times 2^{11})$ structured
SU1	sparse $(2^{12} \times 2^{11})$ unstructured
SS2	sparse $(2^{17} \times 2^{5})$ structured
SU2	sparse $(2^{17} \times 2^{5})$ unstructured
Aminer	Aminer (31890×4216), 1384270 NNZ
Patents	PatentsView (37268×3198), 1131186 NNZ

TABLE I: Dataset Abbreviations

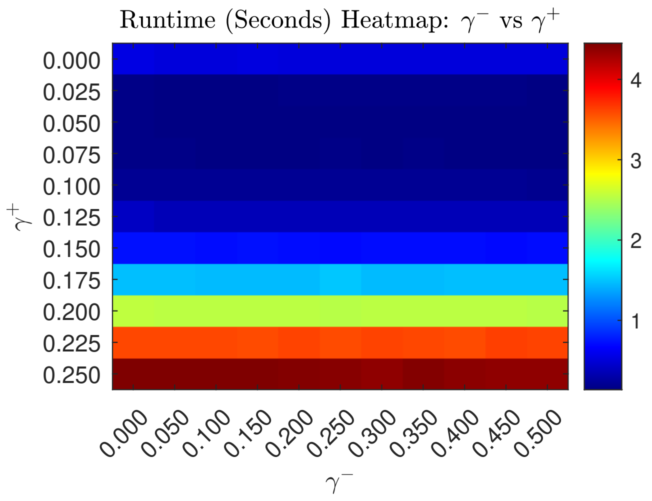


Fig. 1: Hyperparameter tuning experiments for γ^- and γ^+ on well-conditioned dense problem, $\mathbf{A} \in \mathbb{R}^{2048\times1024}, \mathbf{B} \in \mathbb{R}^{2048\times64}$. Note that the runtime performance was more impacted by γ^+ than γ^-. Similar experiments were performed for all hyperparameters when determining default values.

that higher cutoff values resulted in less accurate solutions but increased stability. Conversely, lower cutoff values resulted in more accurate solutions which were more prone to failure due to entering cycles where values close to the boundary conditions oscillated across the boundary condition for numerical reasons. We intend to further explore this phenomena in future work. In this work, we generally leave the cutoff value to the previously established [16] default value of 1e-12 unless otherwise stated.

A. Datasets

1) Synthetic: Dense well-conditioned synthetic datasets were randomly generated, denoted respectively as DW1 and DW2 in Table I. Similarly, dense ill-conditioned synthetic datasets were generated by randomly generating $\mathbf{A}$, computing the SVD of $\mathbf{A}$, multiplying the upper tertile of singular values by 64, and dividing the lower tertile of singular values by 64. They are denoted respectively as DI1 and DI2 in Table I. Thus we present results on a total of four dense synthetic datasets. $\mathbf{b}$ was randomly generated for each of these datasets in Figure 5. In Figures 3 and 4 we introduce sparsity into the solution by setting $\mathbf{A}\mathbf{x}_t = \mathbf{b}$, where a set number of entries of $\mathbf{x}_t$ are zero and the rest positive. In Figures 3 and 4, "Solution Sparsity" refers to the fraction of positive elements in the optimal solution, $\mathbf{x}_t$, used to form the problem. For example, a "Solution Sparsity" of .2 means that 20% of the entries in the optimal solution are positive whilst the rest are zero.

Sparse synthetic datasets were generated by randomly generating $\mathbf{A}$ and $\mathbf{b}$ and then setting a set percent of entries of $\mathbf{A}$ to zero. They are denoted respectively as SU1 and SU2 in Table I. In Figure 6, 99.9% of the entries of $\mathbf{A}$ where set to zero. In Figures 3a and 4a, 90% of the entries of $\mathbf{A}$ were set to zero. Two additional sparse synthetic datasets were generated by forming $\mathbf{b}$ as a nonnegative linear combination of a set percent of the columns of $\mathbf{A}$ as was done for the dense synthetic matrices in Figures 3 and 4. These sparse structured matrices are denoted as SS1 and SS2 in Table I. In Figure 6, we expect the optimal passive set to consist of 10% of the variables, which is representative of problems with low-rank structure that commonly occur in real world applications. The identity matrix was added to all synthetic sparse problems to avoid rank deficiency.

2) Aminer: Real-world experiments were run on subsets of the ArnetMiner (AMiner) [54] academic graph. For each dataset, a set of keywords such as ['mechanical engineer', 'economy', 'astrophysics'] were selected. We then selected a subset of 4216 documents with the keywords based upon criteria such as citation relationship count, performed standard preprocessing, and used the resulting corpus to populate the $(document \times word)$ matrix. We selected a random column of this matrix to be our $\mathbf{b}$. Denoted as Aminer in Table I.

3) PatentsView: Real-world experiments were run on subsets of the PatentsView dataset [55] which consists of over 12 million patents. We selected a subset of 3198 based upon category and citation relationship count, performed standard preprocessing, and used the resulting corpus to populate the $(document \times word)$ matrix. We selected a random column of this matrix to be our $\mathbf{b}$. Denoted as Patents in Table I.

B. Compared Methods

Act: Single-swap active-set based method based upon [23], [47]. MatLab implementation provided in [16] which was used as a starting point for our FAST-NNLS implementation.

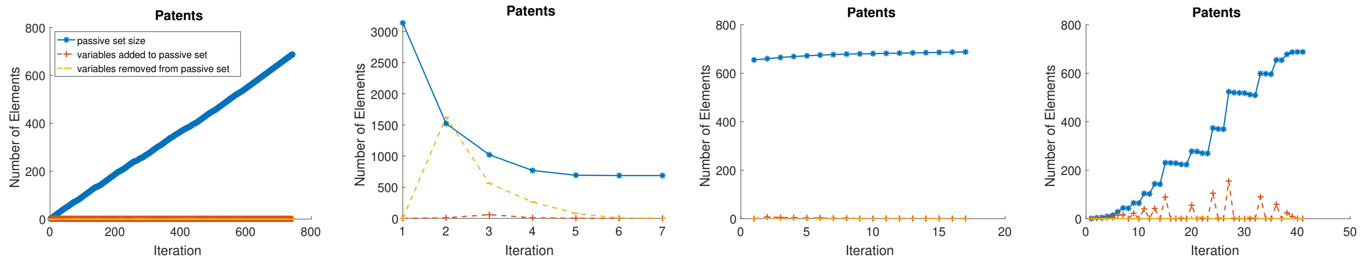
FAST-NNLS: Proposed. Pseudocode in Algorithm 1.

BPP: Block Principal Pivoting algorithm [25] as described in Section III-B. Implementation provided in [16].

FNNLS: Single-swap active-set method [45] with several novel optimizations improving upon [23].

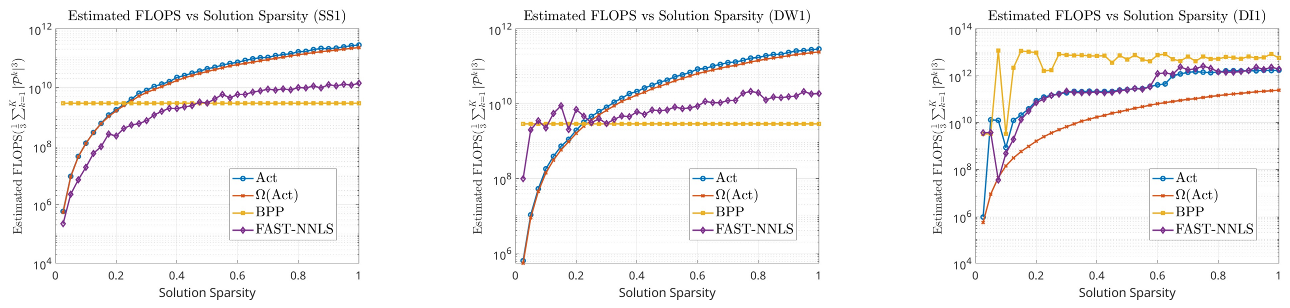
TNTNN: Active-set method developed for use in rock magnetism applications proposed in [24].

LBFGSBC-mex: Popular LBFG-B-C bound constrained optimization algorithm [28]. Ported to C from FORTRAN [29] by Becker et al. [30] and bound out to MatLab with mex wrapper.



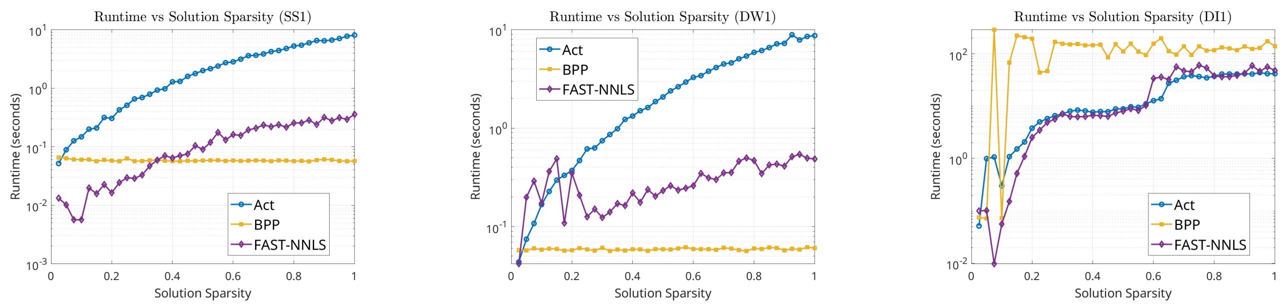
(a) Act (b) BPP (c) TNTNN (d) FAST-NNLS

Fig. 2: Sample Patents real-world problem passive set analysis plots. FAST-NNLS finds the solution in significantly fewer iterations than Act whilst maintaining a significantly smaller maximum intermediate passive set than BPP using a heuristic which is significantly less computationally expensive than TNTNN's. See Figure 7 for resulting timing comparison.



(a) SS1 (b) DW1 (c) DI1

Fig. 3: Estimated FLOPS for Act, BPP, and FAST-NNLS based upon Equation (33). Note that Act closely follows the lower bound derived in Equation (34) for well-conditioned problems as seen in (a) and (b). For well-conditioned problems, the computational cost of BPP is dominated by the largest unconstrained solve when the majority of variables are swapped into the passive set. For the ill-conditioned problem (c), all methods required increased rates of active and passive set swaps with BPP being the most impacted. The zero cutoff was raised from the default value of 1e-12 to 1e-8 to prevent BPP from cycling in the ill-conditioned case.



(a) SS1 (b) DW1 (c) DI1

Fig. 4: Runtime comparison between Act, BPP, and FAST-NNLS active-set methods. Note that runtime performance aligns with the estimated FLOPS for each method in Figure 3. This validates the cost model proposed in Equation (33). As expected, FAST-NNLS consistently outperforms Act for nearly all problem configurations. FAST-NNLS generally outperforms BPP for sparse solutions as seen in (a). We observed that the surveyed real-world problems tended to yield sparse solutions which leads to FAST-NNLS outperforming BPP as seen in Figure 7. FAST-NNLS was also observed to be more robust to ill-conditioning than BPP as seen in (c). The zero cutoff was raised from the default value of 1e-12 to 1e-8 to prevent BPP from cycling in the ill-conditioned case.

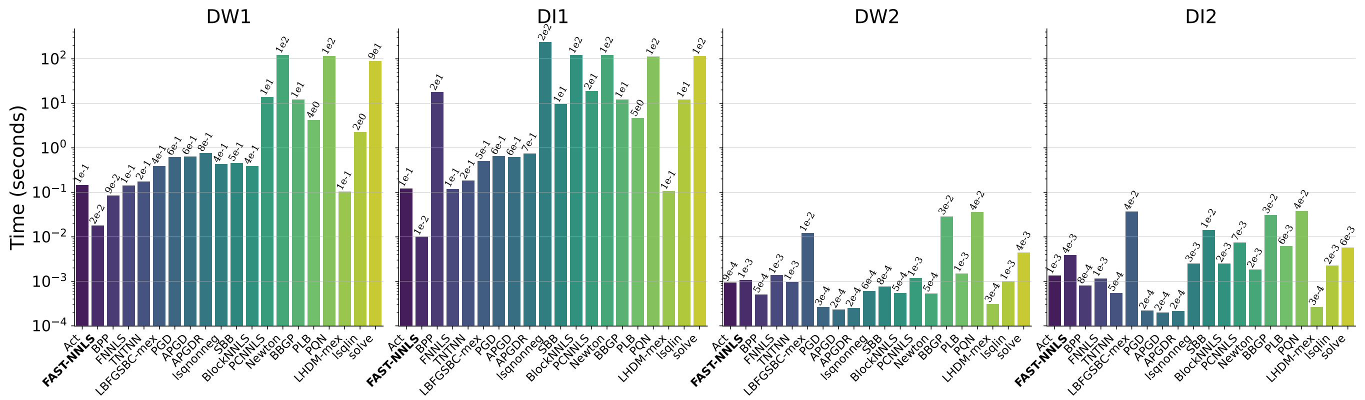


Fig. 5: Mean execution time comparison of all surveyed algorithms. FAST-NNLS outperforms all baselines on DW1 and DI1 problems. BPP, LBFGSBC, BlockNNLS, PCNNLS and PQN methods in some cases did not converge on DI1 datasets, as seen in Figure 8.

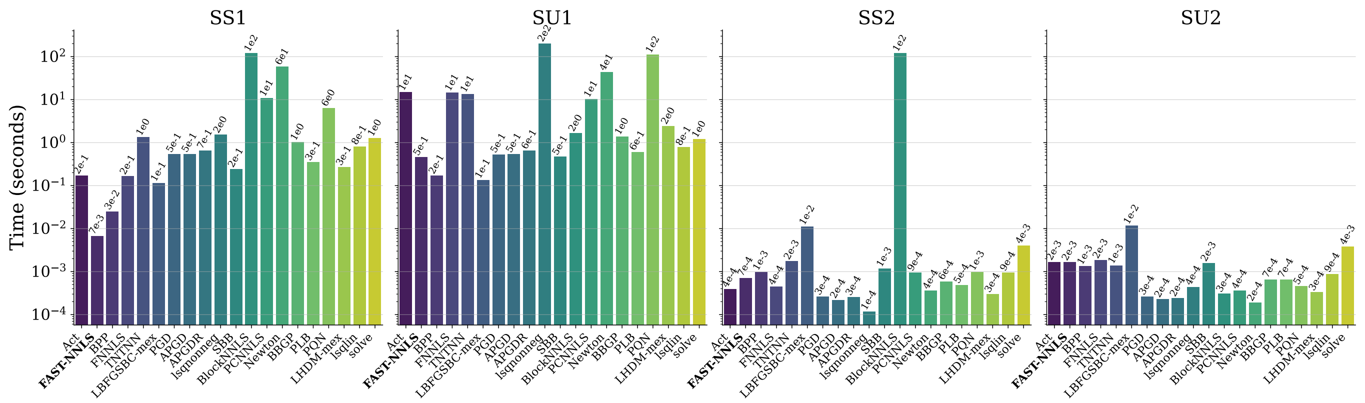


Fig. 6: Mean execution time comparison of all surveyed algorithms. Note that FAST-NNLS outperforms all baselines on SS1. For the SS1 and SU1 problems, the LBFGSBC-mex and SBB methods performed competitively. This is to be expected as the LBFGCBS method is know to perform well on sparse probelms and the SBB method was specfically designed for the sparse case. For SS2 and SU2 the active-set based methods perform well relative to the other baselines. This is to be expected as the size of the optimal passive set in this case is constrained to be small, i.e. to be less than or equal to the number of columns of SS2 and SU2.

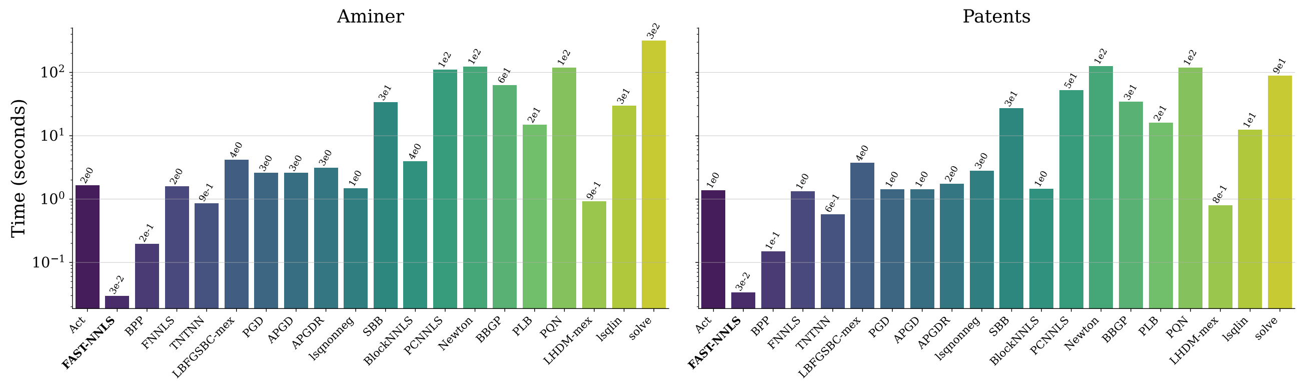
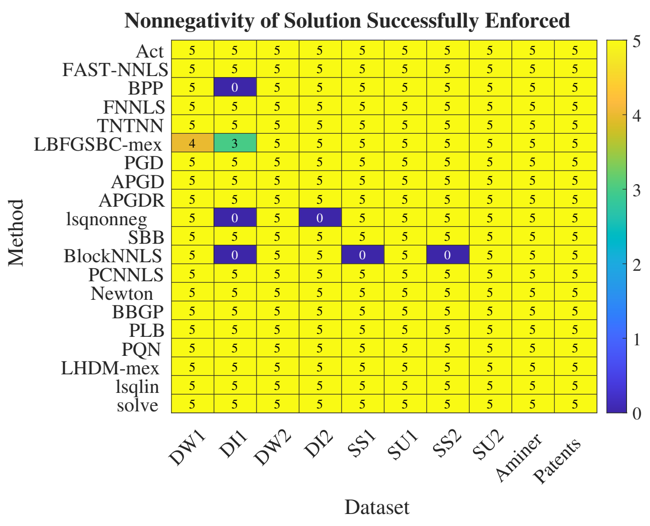


Fig. 7: Real-world runtime bar charts. Note that FAST-NNLS outperforms all baselines in both cases.



	DW1	DI1	DW2	DI2	SS1	SU1	SS2	SU2	Aminer	Patents
Act	5	5	5	5	5	5	5	5	5	5
FAST-NNLS	5	5	5	5	5	5	5	5	5	5
BPP	5	0	5	5	5	5	5	5	5	5
FNNLS	5	5	5	5	5	5	5	5	5	5
TNTNN	5	5	5	5	5	5	5	5	5	5
LBFGSBC-mex	4	3	5	5	5	5	5	5	5	5
PGD	5	5	5	5	5	5	5	5	5	5
APGD	5	5	5	5	5	5	5	5	5	5
APGDR	5	5	5	5	5	5	5	5	5	5
lsqnonneg	5	0	5	0	5	5	5	5	5	5
SBB	5	5	5	5	5	5	5	5	5	5
BlockNNLS	5	0	5	5	0	5	0	5	5	5
PCNNLS	5	5	5	5	5	5	5	5	5	5
Newton	5	5	5	5	5	5	5	5	5	5
BBGP	5	5	5	5	5	5	5	5	5	5
PLB	5	5	5	5	5	5	5	5	5	5
PQN	5	5	5	5	5	5	5	5	5	5
LHDM-mex	5	5	5	5	5	5	5	5	5	5
lsqlin	5	5	5	5	5	5	5	5	5	5
solve	5	5	5	5	5	5	5	5	5	5

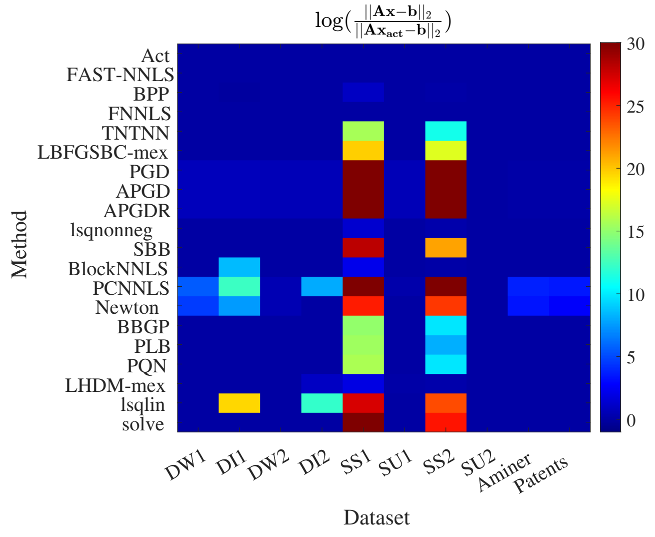


Fig. 8: Solution nonnegativity and quality relative to Act baseline on log scale out of 5 runs. Note that FAST-NNLS successfully enforces nonnegativity and obtains high solution quality in all cases. FAST-NNLS is guaranteed to find the optimal solution.

PGD: Projected Gradient Descent method MatLab implementation provided by [34].

APGD: Accelerated Projected Gradient Descent method MatLab implementation provided by [34].

APGDR: Accelerated Projected Qradient Descent with Restart MatLab implementation provided by [34].

lsqnonneg: MatLab built-in NNLS function which implements the active-set method by [23].

SBB: Subspace Barzilai-Borwein method proposed in [44].

BlockNNLS: Original block principal pivoting algorithm proposed by Júdice and Pires [25] and surveyed in [51]. MatLab implementation provided by [56].

PCNNLS: Predictor corrector method originally proposed in [57], surveyed in [51], and provided by [56].

Newton: Newton optimization method for NNLS and surveyed in [51]. MatLab implementation provided by [56].

BBGP: Barzilai-Borwein Gradient Projection [26]. Similar in several regards to SBB method [44].

PLB: Projected L-BFGS [26].

PQN: Projected Quasi Newton BFGS [26].

LHDM-mex: Lawson-Hanson algorithm with Deviation Maximization (LHDM) [50]. Implemented in C/C++ and bound out to MatLab with mex wrapper.

lsqlin: MatLab *lsqlin* function for bound constrained optimization using the default interior-point solver backend.

solve: MatLab *solve* function for bound constrained optimization using the default interior-point solver backend.

C. *Passive Set Analysis and Runtime Comparison*

Our proposed active-set thresholding swap heuristic performs well when the ratio of the optimal passive set size to number of columns of $\mathbf{A}$ is small. This can be seen in the DW1, DI1, SS1, Aminer, and PatentsView results respectively in Figures 3 to 7. Our proposed method outperforms all surveyed baselines in 5/10 tested problem configurations in Figures 5 to 7.

The benefit of the baseline active-set method's single swap rule is that it performs predictably, taking a number of swap iterations close to the size of the optimal passive set size. This performs decently well when the optimal passive set size is small, but can become prohibitively slow when the optimal passive set size is large as seen in Figure 4. Another benefit of the single exchange rule is that it maintains passive sets which are almost always smaller than the optimal passive set, which in turn results in less expensive Cholesky computations. This active-set analysis also holds true for the other algorithms that employ the single exchange rule such as FNNLS and MatLab's lsqnonneg (which implements the Lawson-Hanson active-set algorithm). We expect these single-swap active-set to closely follow the lower bound derived in Equation (34) for well-conditioned problems as seen in Figure 3.

A benefit of the surveyed BPP method's greedy swap heuristic is that it often requires few outer iterations to converge. A disadvantage of the aggressive greedy swap heuristic is that it can lead to passive sets which consist of almost all variables and which are significantly larger than the optimal passive set size. This results in evaluating larger than necessary Cholesky factorizations. Both behaviours can be seen in Figure 2. Additionally, for some ill-conditioned problems BPP enters a cycle and takes prohibitively long to converge as seen in experiments on the DI1 data in Figures 4 and 5. In these ill-conditioned cases, BPP often reached the maximum number of iterations (twice the number of variables) before obtaining the optimal solution causing it to return an infeasible solution as seen in Figure 8. BPP performs best when the optimal passive set size to number of columns of $\mathbf{A}$ ratio is high, such as the DW2, SS2 and SU2 synthetic problems in Figure 5 and 6. Note that these problem sizes are similar to those encountered in Nonnegative Matrix Factorization (NMF) where why BPP is commonly used. [16] pioneered the application of BPP [25] to NMF. Figures 3 and 4 demonstrate that as the ratio of optimal passive set size to number of columns of $\mathbf{A}$ increases, i.e. the solution vector becomes denser, that BPP performs better relative to the other active-set methods.

For TNTNN, we observed that the employed heuristic for determining the passive set successfully decreased the number of outer iterations, but was significantly more computationally expensive than the heuristics of the other surveyed active-set based methods. This resulted in decreased runtime performance relative to the other methods.

VIII. Conclusion

The proposed FAST-NNLS method is grounded in novel observations regarding the variables added to the optimal variable set in active-set methods. FAST-NNLS leverages these novel insights to significantly outperform current state-of-the-art NNLS methods in terms of both runtime and solution quality across a wide range of problem instances. We prove that FAST-NNLS is guaranteed to converge to the optimal solution in a finite number of iterations. We demonstrate the effectiveness of FAST-NNLS on multiple synthetic datasets as well as two real-world text analysis applications.

Future avenues of research include exploring novel techniques based on QR factorization to avoid the numerical instabilities associated with the normal equations, Givens rotation updating methods applied incrementally as variables enter/exit the passive set to efficiently compute the unconstrained least squares at each iteration, exploring the case of FAST-NNLS with multiple RHS, and parallel FAST-NNLS variants. Finally, we are developing a novel model that is parameterized by the matrix size, condition number, and the number of RHS. This model will automatically select the most suitable algorithm for solving the given problem. Similar models have been explored for various linear algebra software packages such as BLAS and LAPACK. We believe that incorporating such a model into an NNLS library will significantly enhance its impact across a broad range of real-world applications.

IX. Acknowledgements

The authors would like to thank Stephen Becker for sharing the code to several of the baselines. Additionally, we would like to thank Gudrun Cobb and Cassey Waldo for their extensive proof-reading and editing as well as the anonymous reviewers for their invaluable comments which helped improve this work. This research is based upon work supported by the National Science Foundation (NSF) under award number 2316176. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the supporting agencies or employers.

References

[1] F. Benvenuto, R. Zanella, L. Zanni, and M. Bertero, “Nonnegative least-squares image deblurring: improved gradient projection approaches,” *Inverse Problems*, vol. 26, no. 2, p. 025004, Dec. 2009. [Online]. Available: https://dx.doi.org/10.1088/0266-5611/26/2/025004

[2] N. Nadisic, A. Vandaele, N. Gillis, and J. E. Cohen, “Exact Sparse Nonnegative Least Squares,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 5395–5399, iSSN: 2379-190X. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9053295

[3] V. Menon, Q. Du, and J. E. Fowler, “Random-projection-based nonnegative least squares for hyperspectral image unmixing,” in *2016 8th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, 2016, pp. 1–5.

[4] J. M. Myre, I. Lascu, E. A. Lima, J. M. Feinberg, M. O. Saar, and B. P. Weiss, “Using TNT-NN to unlock the fast full spatial inversion of large magnetic microscopy data sets,” *Earth, Planets and Space*, vol. 71, no. 1, p. 14, Feb. 2019. [Online]. Available: https://doi.org/10.1186/s40623-019-0988-8

[5] B. P. Weiss, E. A. Lima, L. E. Fong, and F. J. Baudenbacher, “Paleomagnetic analysis using squid microscopy,” *Journal of Geophysical Research: Solid Earth*, vol. 112, no. B9, 2007. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2007JB004940

[6] K. Ehrenfried and L. Koop, “Comparison of Iterative Deconvolution Algorithms for the Mapping of Acoustic Sources,” *AIAA Journal*, vol. 45, no. 7, pp. 1584–1595, 2007, publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.26320. [Online]. Available: https://doi.org/10.2514/1.26320

[7] H. Liang, “HauLiang/Acoustic-Beamforming-Methods,” Aug. 2024, original-date: 2021-09-05T09:12:58Z. [Online]. Available: https://github.com/HauLiang/Acoustic-Beamforming-Methods

[8] M. Slawski and M. Hein, “Sparse recovery by thresholded non-negative least squares,” in *Advances in Neural Information Processing Systems*, vol. 24. Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2011/hash/d6723e7cd6735df68d1ce4c704c29a04-Abstract.html

[9] V. K. Potluru, S. M. Plis, S. Luan, V. D. Calhoun, and T. P. Hayes, “Sparseness and a reduction from Totally Nonnegative Least Squares to SVM,” in *The 2011 International Joint Conference on Neural Networks*. San Jose, CA, USA: IEEE, Jul. 2011, pp. 1922–1929. [Online]. Available: http://ieeexplore.ieee.org/document/6033459/

[10] D. Wang, F. Cong, and T. Ristaniemi, “Sparse Nonnegative CANDECOMP/PARAFAC Decomposition in Block Coordinate Descent Framework: A Comparison Study,” *ArXiv*, Dec. 2018. [Online]. Available: https://www.semanticscholar.org/paper/b2559e38b73e0a28fb1209ae2ffb168de01be25c

[11] N. Alger, T. Hartland, N. Petra, and O. Ghattas, “Point Spread Function Approximation of High-Rank Hessians with Locally Supported Nonnegative Integral Kernels,” *SIAM Journal on Scientific Computing*, vol. 46, no. 3, pp. A1658–A1689, Jun. 2024. [Online]. Available: https://epubs.siam.org/doi/10.1137/23M1584745

[12] E. Esser, Y. Lou, and J. Xin, “A method for finding structured sparse solutions to nonnegative least squares problems with applications,” *SIAM Journal on Imaging Sciences*, vol. 6, no. 4, pp. 2010–2046, 2013. [Online]. Available: https://doi.org/10.1137/13090540X

[13] A. Bemporad, “A Quadratic Programming Algorithm Based on Nonnegative Least Squares With Applications to Embedded Model Predictive Control,” pp. 1111–1116, 2016. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7163541

[14] A. Turksoy and A. Teke, “A fast and energy-efficient nonnegative least square-based optimal active battery balancing control strategy for electric vehicle applications,” *Energy*, vol. 262, p. 125409, Jan. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544222022915

[15] D. Chen and R. J. Plemmons, “Nonnegativity constraints in numerical analysis,” in *The Birth of Numerical Analysis*. WORLD SCIENTIFIC, Nov. 2009, pp. 109–139. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/9789812836267_0008

[16] J. Kim and H. Park, “Fast nonnegative matrix factorization: An active-set-like method and comparisons,” *SIAM J. Sci. Comput.*, vol. 33, no. 6, p. 32613281, nov 2011. [Online]. Available: https://doi.org/10.1137/110821172

[17] J. Kim, Y. He, and H. Park, “Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework,” *J. of Global Optimization*, vol. 58, no. 2, p. 285319, feb 2014. [Online]. Available: https://doi.org/10.1007/s10898-013-0035-4

[18] R. Du, B. Drake, and H. Park, “Hybrid clustering based on content and connection structure using joint nonnegative matrix factorization,” *Journal of Global Optimization*, vol. 74, pp. 1 – 17, 08 2019.

[19] D. Choi, A. Xiang, O. Ozturk, D. Shrestha, B. L. Drake, H. Haidarian, F. Javed, and H. Park, "Wellfactor: Patient profiling using integrative embedding of healthcare data," in *IEEE International Conference on Big Data, BigData 2023, Sorrento, Italy, December 15-18, 2023*. IEEE, 2023, pp. 616–625. [Online]. Available: https://doi.org/10.1109/BigData59044.2023.10386138

[20] S. Eswar, B. Cobb, K. Hayashi, R. Kannan, G. Ballard, R. Vuduc, and H. Park, "Distributed-memory parallel jointnmf," in *Proceedings of the 37th International Conference on Supercomputing*, ser. ICS '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 301312. [Online]. Available: https://doi.org/10.1145/3577193.3593733

[21] S. Eswar, K. Hayashi, B. Cobb, R. Kannan, G. Ballard, R. Vuduc, and H. Park, "On rank selection for nonnegative matrix factorization," in *2024 IEEE International Conference on Big Data (BigData)*, 2024, pp. 1294–1301.

[22] B. Cobb, R. Velasquez, R. Vuduc, and H. Park, "Clustering and topic discovery of multiway data via joint-ncmtf," in *2024 IEEE International Conference on Big Data (BigData)*, 2024, pp. 1268–1275.

[23] C. Lawson and R. Hanson, "Solving Least Squares Problems | SIAM Publications Library," 1995. [Online]. Available: https://epubs.siam.org/doi/book/10.1137/1.9781611971217

[24] J. M. Myre, E. Frahm, D. J. Lilja, and M. O. Saar, "TNT-NN: A Fast Active Set Method for Solving Large Non-Negative Least Squares Problems," *Procedia Computer Science*, vol. 108, pp. 755–764, Jan. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050917307858

[25] J. J. Jdice and F. M. Pires, "A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems," *Computers & Operations Research*, vol. 21, no. 5, pp. 587–596, 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0305054894901066

[26] D. Kim, S. Sra, and I. S. Dhillon, "Tackling box-constrained optimization via a new projected quasi-newton approach," *SIAM Journal on Scientific Computing*, vol. 32, no. 6, pp. 3548–3563, 2010. [Online]. Available: https://doi.org/10.1137/08073812X

[27] C.-J. Lin and J. J. Moré, "Newton's method for large bound-constrained optimization problems," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 1100–1127, 1999. [Online]. Available: https://doi.org/10.1137/S1052623498345075

[28] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995. [Online]. Available: https://doi.org/10.1137/0916069

[29] C. Zhu, R. Byrd, J. Nocedal, and J. L. Morales. [Online]. Available: https://users.iems.northwestern.edu/~nocedal/lbfgsb.html

[30] S. Becker. [Online]. Available: https://github.com/stephenbeckr/L-BFGS-B-C?tab=readme-ov-file

[31] H.-H. Chou, J. Maly, and C. M. Verdun, "Non-negative Least Squares via Overparametrization," Oct. 2022, arXiv:2207.08437 [cs, math]. [Online]. Available: http://arxiv.org/abs/2207.08437

[32] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Comput.*, vol. 19, no. 10, p. 27562779, Oct. 2007. [Online]. Available: https://doi.org/10.1162/neco.2007.19.10.2756

[33] D. Kim, S. Sra, and I. S. Dhillon, *Fast Newton-type Methods for the Least Squares Nonnegative Matrix Approximation Problem*, pp. 343–354. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611972771.31

[34] A. Ang, "Nonnegative least squares pgd, accelerated pgd and with restarts." [Online]. Available: https://angms.science/doc/NMF/nnls_pgd.pdf

[35] R. Polyak, "Projected gradient method for non-negative least square roman," 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:195834980

[36] C. Zheng, M. Yu, J. Shan, A. Wang, and H. Chen, "Fast sparse non-negative least squares via admm for high resolution doa estimation," *IEEE Sensors Journal*, vol. 23, no. 4, pp. 3901–3910, 2023.

[37] S. Bellavia, M. Macconi, and B. Morini, "An interior point newtonlike method for nonnegative leastsquares problems with degenerate solution," *Numerical Linear Algebra With Applications*, vol. 13, no. 10, pp. 825–846, 2006.

[38] L. Yong, "A feasible interior point algorithm for a class of nonnegative least squares problems," in *2009 ETP International Conference on Future Computer and Communication*, 2009, pp. 157–159.

[39] M. Salahi and A. Taati, "Interior point gradient algorithm for totally nonnegative least-squares problems in inequality sense," *International Journal of Computer Mathematics*, vol. 91, no. 7, pp. 1593–1600, 2014. [Online]. Available: https://doi.org/10.1080/00207160.2013.854883

[40] M. Merritt and Y. Zhang, "Interior-point gradient method for large-scale totally nonnegative least squares problems," *J. Optim. Theory Appl.*, vol. 126, no. 1, p. 191202, Jul. 2005. [Online]. Available: https://doi.org/10.1007/s10957-005-2668-z

[41] T. M. Inc., "Matlab (r2024a)," Natick, Massachusetts, United States, 2024. [Online]. Available: https://www.mathworks.com

[42] Y.-C. Kuo and C.-S. Liu, "An index search method based inner-outer iterative algorithm for solving nonnegative least squares problems," *Journal of Computational and Applied Mathematics*, vol. 424, p. 114954, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377042722005520

[43] C. Boutsidis and P. Drineas, "Random projections for the nonnegative least-squares problem," *ArXiv*, vol. abs/0812.4547, 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:6826978

[44] D. Kim, S. Sra, and I. S. Dhillon, "A non-monotonic method for large-scale non-negative least squares," *Optimization Methods and Software*, vol. 28, no. 5, pp. 1012–1039, Oct. 2013. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/10556788.2012.656368

[45] R. Bro and S. Jong, "A Fast Non-negativity-constrained Least Squares Algorithm," *Journal of Chemometrics*, vol. 11, pp. 393–401, Sep. 1997.

[46] J. Cantarella and M. Piatek, "tsnnls: A solver for large sparse least squares problems with non-negative variables."

[47] M. H. Van Benthem and M. R. Keenan, "Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems." [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cem.889

[48] J. Diakonikolas, S. Padmanabhan, C. Li, and C. Song, "A Fast Scale-Invariant Algorithm for Non-negative Least Squares with Non-negative Data."

[49] Y. Luo and R. Duraiswami, "Efficient Parallel Nonnegative Least Squares on Multicore Architectures," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2848–2863, Jan. 2011, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://epubs.siam.org/doi/10.1137/100799083

[50] M. Dessole, M. DellOrto, and F. Marcuzzi, "The lawsonhanson algorithm with deviation maximization: Finite convergence and sparse recovery," *Numerical Linear Algebra with Applications*, vol. 30, no. 5, Jan. 2023. [Online]. Available: http://dx.doi.org/10.1002/nla.2490

[51] L. F. Portugal, J. J. Júdice, and L. N. Vicente, "A comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables," 1994. [Online]. Available: https://doi.org/10.2307/2153286

[52] J. Faulhaber, *Academia algebrae, darinnen die miraculosische Inventiones zu den höchsten Cossen weiters continuirt und profitiert werden*. Augspurg: Johann Ulrich Schönigk, 1631, in Verlag Johann Remmelins.

[53] J. Bernoulli, *Ars conjectandi, opus posthumum. Accedit Tractatus de seriebus infinitis, et epistola gallicè scripta de ludo pilae reticularis*. Basileæ: Impensis Thurnisiorum, Fratrum, 1713, edited and published posthumously by Nicolaus I Bernoulli.

[54] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: Extraction and mining of academic social networks," ser. KDD '08, 2008. [Online]. Available: https://doi.org/10.1145/1401890.1402008

[55] USPTO. [Online]. Available: https://patentsview.org/download/data-download-tables

[56] U. Roque. [Online]. Available: https://www.mathworks.com/matlabcentral/profile/authors/339586

[57] I. J. Lustig, R. E. Marsten, and D. F. Shanno, "Computational experience with a primal-dual interior point method for linear programming," 1991. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0024379591902752