Mathematics

https://www.aimspress.com/journal/Math

AIMS Mathematics, 11(2): 4691–4704.
DOI: 10.3934/math.2026191
Received: 16 September 2025
Revised: 06 November 2025
Accepted: 18 November 2025
Published: 26 February 2026

*Research article*

# A numerical LP-Newton method for solving linear programming using separating hyperplanes

**Yuki Matsuno***

Graduate School of Management, Tokyo University of Science, 1-11-2 Fujimi, Chiyoda-ku, Tokyo 102-0071, Japan

* **Correspondence:** Email: yuhki908@gmail.com, 8623702@ed.tus.ac.jp.

**Abstract:** We propose an LP-Newton-type method for linear programming (LP) with box constraints. In the standard LP-Newton method, each iteration computes a separating hyperplane using the Wolfe algorithm to find the minimum-norm point in a convex polytope. Because this inner routine can be computationally expensive in the worst case (and the Wolfe algorithm may require exponential time), we replace it with a simpler procedure that separates the origin from the convex hull of projected points. This change retains the Newton-type iterative framework while avoiding a potentially intractable inner routine, because the separating hyperplane can be constructed much more efficiently. We also derive an upper bound on the number of iterations. In our experiments, the resulting Naive Separation Algorithm (NSA) ran faster in some settings than the simplex method, which is often cited as one of the Top 10 Algorithms of the 20th Century [1]. These results suggest that this direction may lead to efficient algorithms for box-constrained LP.

**Keywords:** LP-Newton method; linear programming; zonotope; box constraints; supporting hyperplane
**Mathematics Subject Classification:** 65K05, 90C05

## 1. Introduction

Linear Programming (LP) is one of the most fundamental problems in mathematical optimization, and has played a central role in a wide range of disciplines including operations research, economics, engineering, and computer science.

The simplex method for solving LP was proposed by Dantzig in 1947 [2], and owing to its remarkable computational efficiency in practice, it has been widely adopted as the standard method and is listed in Top 10 algorithms of the 20th century [1]. However, the example of Klee and Minty demonstrated that the simplex method requires exponential time in the worst case [3], thereby facing a fundamental challenge in terms of computational complexity. Subsequently, polynomial-time algorithms were established: the ellipsoid method by Khachiyan in 1979 [4], Karmarkar's interior point method in 1984 [5], and primal-dual interior point methods by Kojima, Mizuno, and Yoshise as well as by Megiddo [6–8].

From the perspective of industrial applications, LP has consistently expanded from classical to modern domains. A representative early application is Hitchcock's transportation problem [9], which formulated logistics connecting supply sources and demand destinations as an LP, enabling decision-making in freight minimization, resource allocation, and supply chain design involving factories, warehouses, and retailers. The seminal work of Dantzig, Fulkerson, and Johnson on the traveling salesman problem (TSP) [10] combined LP with the cutting-plane method to tackle large-scale combinatorial optimization, thereby demonstrating the practical utility of LP in complex planning problems. Furthermore, the maximum flow algorithm by Ford and Fulkerson [11] laid the foundation for infrastructure optimization, including capacity allocation and redundant routing in networks, with wide-ranging applications in electricity, communication, and transportation systems.

In recent years, LP has continued to evolve toward new industrial challenges. Robust LP, developed by Ben-Tal and Nemirovski [12], explicitly accounts for data and constraint uncertainty, providing solutions that remain reliable under demand fluctuations and supply risks. This approach has improved the robustness of planning across manufacturing, logistics, finance, and energy industries. Online LP, introduced by Agrawal, Wang, and Ye [13], addresses dynamic allocation of resources as data arrive sequentially, with applications to digital industries such as online advertising, revenue management, and real-time inventory allocation. In data science, Chen, Donoho, and Saunders proposed the Basis Pursuit formulation [14], which enables sparse signal recovery via $\ell_1$ minimization cast as an LP. This work established LP as a key framework in compressed sensing, feature selection, and sparse regression, thereby positioning LP as a foundation not only in optimization but also in machine learning and modern data analysis. In this manner, LP has maintained a central role from classical applications to contemporary industrial problems.

On the theoretical side, the existence of a strongly polynomial-time algorithm for LP remains unresolved. Ye demonstrated strong polynomiality for a special class of Markov decision problems [15], and Mizuno proposed a strongly polynomial simplex method for totally unimodular LPs [16]. Nevertheless, the problem remains unsolved for general LP, representing a fundamental open question in complexity theory and algorithm design.

Within this context, Fujishige et al. proposed the LP-Newton method for LP with box constraints, based on zonotopes [17]. This was later extended to standard form LP by Kitahara, Mizuno, and Shi [18] and extended the LP-Newton framework to conic programming problems by Tanaka and Okuno [19]. Kitahara and Sukegawa introduced a simple projection algorithm [20], while the author of this paper and Shi recently developed a finite-termination variant that circumvents the Wolfe algorithm [21] and preserves Newton-type structures.

The LP-Newton method is fundamentally based on the minimum norm point problem, which seeks a separating hyperplane by locating the point in the convex hull nearest to the origin. Wolfe proposed an efficient algorithm for this problem [22], and Fujishige and Zhan reformulated it as maximizing the distance between a convex hull and a supporting hyperplane [23]. Subsequent studies extended this line of work to intersections with hyperplanes [24,25] and to the minimum norm point problem between two convex hulls [26], and more recently, theoretical developments such as [27] have also been reported.

From an application perspective, the minimum norm point problem arises in numerous contexts: it is equivalent to minimizing over a base polyhedron in submodular function minimization [28–30], is employed in combinatorial optimization and regularization in machine learning [31], and the mixing problem [32] as well as large-scale portfolio optimization [33].

Nevertheless, the Wolfe algorithm is known to exhibit exponential complexity in the worst case [34]. Therefore, the LP-Newton method may inherit the same computational difficulties because the method relies on repeated calls to this subroutine to create a separating hyperplane.

In this study, we propose a new LP-Newton method that directly creates a separating hyperplane without using the Wolfe algorithm.

The contributions of this study are summarized as follows:

(1) We propose an alternative to the minimum norm point algorithm for computing separating hyperplanes, allowing such hyperplanes to be obtained more easily.
(2) Inspired by the fact that classical Newton methods generally exhibit quadratic convergence, we design the algorithm to preserve the Newton-type iterative structure.
(3) We theoretically establish that the proposed separating procedure exhibits asymptotic convergence.
(4) We show that the proposed method is easier to implement than conventional approaches that rely on solving the minimum norm point problem.
(5) In the preliminary experiments, the proposed method outperformed the simplex method in terms of computation time under certain conditions.

The remainder of this paper is organized as follows. Section 2 reviews the LP-Newton method of Fujishige and its connection with the minimum norm point problem, highlighting its computational bottlenecks. Section 3 introduces the proposed separating-hyperplane-based algorithm and describes its integration into Newton-type iterations. Section 4 reports numerical experiments on box-constrained LPs, comparing the performance of the proposed approach with existing methods. Finally, Section 5 concludes the paper and outlines future research directions.

## 2. LP-Newton method

In this section, we summarize Fujishige's LP-Newton algorithm proposed in [17]: Consider the following linear programming:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{maximize}} \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} \quad & A\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{x} \in B := \{\boldsymbol{x} \mid \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}\}. \end{aligned} \tag{2.1}$$

We consider the standard form with data $A \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, and $\boldsymbol{c}, \boldsymbol{\ell}, \boldsymbol{u} \in \mathbb{R}^n$. The decision variable is $\boldsymbol{x} \in \mathbb{R}^n$ subject to $\boldsymbol{\ell} \le \boldsymbol{x} \le \boldsymbol{u}$. The symbol "$\top$" is the transpose of a vector or matrix, ":=" denotes that the left-hand side is defined to be equal to the right-hand side. An algorithm in this context for the standard form of problem (2.1) can be found in [18].

Linear optimization is more manageable on a zonotope. Hereafter, a zonotope $\bar{Z}$ is defined as

$$\bar{Z} := \left\{ \bar{z} \in \mathbb{R}^{m+1} \,\middle|\, \bar{z} = \bar{A}\boldsymbol{x},\ \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u} \right\}, \tag{2.2}$$

where

$$\bar{A} := \begin{bmatrix} A \\ \boldsymbol{c}^\top \end{bmatrix} \in \mathbb{R}^{(m+1)\times n}.$$

Therefore, problem (2.1) can be recast as

$$\begin{aligned} &\text{maximize} && \gamma \\ &\text{subject to} && \begin{bmatrix} \boldsymbol{b} \\ \gamma \end{bmatrix} \in \bar{Z}. \end{aligned} \tag{2.3}$$

Let $\hat{\boldsymbol{x}}_0 := (\hat{x}_{10}, \dots, \hat{x}_{n0})^\top$ with

$$\hat{x}_{j0} := \begin{cases} u_j, & \text{if } c_j > 0, \\ l_j, & \text{otherwise,} \end{cases} \tag{2.4}$$

then $\gamma_0 := \boldsymbol{c}^\top \hat{\boldsymbol{x}}_0$ is an upper bound of the $(m+1)$st coordinate value on the zonotope $\bar{Z}$.

Fujishige's LP-Newton algorithm starts with $\bar{\boldsymbol{b}}_0 := (\boldsymbol{b}^\top, \gamma_0)^\top$. Suppose that $\bar{\boldsymbol{b}}_k$ is given at the $k$th iteration. The algorithm finds $\bar{z}_{k+1} := \arg\min\{\|\boldsymbol{y} - \bar{\boldsymbol{b}}_k\| \mid \boldsymbol{y} \in \bar{Z}\}$ by using the Wolfe algorithm. At the end of the iteration, a vertex subset $J_k$ of the box $B$, consisting of elements $\bar{\boldsymbol{x}}_i$, is obtained such that

$$\bar{z}_{k+1} = \sum_{\bar{\boldsymbol{x}}_i \in J_k} \lambda_i \left(\bar{A}\bar{\boldsymbol{x}}_i\right), \tag{2.5}$$

$$\sum_{\text{same } i \text{ as } \bar{\boldsymbol{x}}_i \in J_k} \lambda_i = 1, \quad \lambda_i > 0. \tag{2.6}$$

Then $\bar{\boldsymbol{b}}_{k+1}$ is updated by the intersection of the following hyperplane $H_k$ and the line $L$:

$$\bar{\boldsymbol{b}}_{k+1} := H_k(\bar{z}_{k+1}) \cap L, \tag{2.7}$$

where $H_k(\bar{z}_{k+1}) := \left\{\boldsymbol{x} \mid \bar{z}_{k+1}^\top \boldsymbol{x} = \bar{z}_{k+1}^\top \bar{z}_{k+1}\right\}$ and $L := \{(\boldsymbol{b}^\top, \gamma)^\top \mid \forall \gamma \in \mathbb{R}\}$. The LP-Newton algorithm is summarized below.

**Algorithm 1** LP-Newton method

```
Require: A linear programming problem of the form (2.1).
Ensure: An optimal solution $\boldsymbol{x}^*$ to problem (2.1), or declare infeasible.
1:  Define the set $J_0$ corresponding to the point $\hat{\boldsymbol{x}}_0$ in (2.4), set $\gamma_0 := \boldsymbol{c}^\top \hat{\boldsymbol{x}}_0$, and $k := 0$.
2:  while true do
3:      Set $\bar{\boldsymbol{b}}_k := (\boldsymbol{b}^\top, \gamma_k)^\top$.
4:      Compute $\bar{z}_{k+1} := \arg\min \left\{ \|\boldsymbol{y} - \bar{\boldsymbol{b}}_k\| \;\middle|\; \boldsymbol{y} \in \bar{Z} \right\}$.
5:      Compute $\bar{\boldsymbol{b}}_{k+1}$ by (2.7), and express $\bar{z}_{k+1} := \left(z_{k+1}^\top, \xi_{k+1}\right)^\top$.
6:      Update the set $J_k$ using (2.5) and (2.6).
7:      if $\left\|\bar{z}_{k+1} - \bar{\boldsymbol{b}}_k\right\| \leq \varepsilon$ then
8:          Set $\boldsymbol{x}^* := \sum_{\bar{\boldsymbol{x}}_i \in J_k} \lambda_i \bar{\boldsymbol{x}}_i$.
9:          return $\boldsymbol{x}^*$
10:     end if
11:     if $\xi_{k+1} > \gamma_k$ then
12:         return infeasible
13:     end if
14:     Update $\gamma_{k+1} := \xi_{k+1} - \frac{\|\boldsymbol{b} - z_{k+1}\|^2}{\gamma_k - \xi_{k+1}}$.
15:     Set $k := k + 1$
16: end while
```

Note that the Wolfe algorithm terminates after a finite number of steps. The same is true for the LP-Newton algorithm [17].

## 3. Naive separation algorithm (NSA)

At Step 4, Fujishige's LP-Newton algorithm relies on the Wolfe minimum norm point algorithm. In this study, we introduce an algorithm generating a supporting hyperplane to separate the zonotope from the origin as an alternative to the Wolfe algorithm.

Let $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N$ be given points in $\mathbb{R}^n$ and $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\}$, and assume that the origin $\boldsymbol{0}$ does not belong to the convex hull $C(P)$. The following *separation theorem* is well-known in convex analysis.

**Theorem 3.1** (Separation theorem)**.** *If $C(P)$ does not contain $\boldsymbol{0}$, then there exist a unit vector $\boldsymbol{x}^*$ and a constant $\gamma > 0$ such that*

$$\boldsymbol{x}^{*\top} \boldsymbol{p} \geq \gamma \qquad (\forall \boldsymbol{p} \in P),$$

*i.e.,* $\gamma = \min_{\boldsymbol{p} \in P} \boldsymbol{x}^{*\top} \boldsymbol{p} > 0$.

Suppose that $\boldsymbol{x}_k$ is obtained in the algorithm. We have the following stopping criterion for Algorithm 2.

**Lemma 3.1** (Stopping criterion implies separation)**.** *Let $\delta_k := \min_{\boldsymbol{p} \in P} \boldsymbol{h}_k^\top \boldsymbol{p}$. If $\delta_k > 0$, then the hyperplane*

$$H(\boldsymbol{h}_k) := \{\boldsymbol{x} \mid \boldsymbol{h}_k^\top \boldsymbol{x} = \delta_k\}$$

*separates $C(P)$ from the origin.*

The proposed separation algorithm is summarized below.

**Algorithm 2** Naive Separation Algorithm (NSA)

```
Require: Vertex set $P$ (with $\mathbf{0} \notin C(P)$), tolerance $\varepsilon > 0$, parameter $\eta \in (0, 1)$
Ensure: Normal vector $\boldsymbol{h}^*$ and intercept $\gamma^*$
 1: $k := 0$, set $\boldsymbol{x}_0 := \boldsymbol{p}_j$ for some $\boldsymbol{p}_j \in P$ and $\boldsymbol{h}_0 := \boldsymbol{x}_0/\|\boldsymbol{x}_0\|$
 2: while true do
 3:     $\boldsymbol{p}_k \in \arg\min_{\boldsymbol{p}\in P} \boldsymbol{h}_k^\top \boldsymbol{p}$
 4:     if $\boldsymbol{h}_k^\top \boldsymbol{p}_k > \varepsilon$ then
 5:         return $\boldsymbol{h}^* := \boldsymbol{h}_k,\ \gamma^* := \boldsymbol{h}_k^\top \boldsymbol{p}_k$
 6:     else
 7:         $\theta_k := \dfrac{\boldsymbol{p}_k^\top(\boldsymbol{p}_k - \boldsymbol{x}_k)}{\|\boldsymbol{p}_k - \boldsymbol{x}_k\|^2}$
 8:         $\alpha_k := \min\{\max\{\theta_k, 0\},\ 1 - \eta\}$
 9:         $\boldsymbol{x}_{k+1} := \alpha_k \boldsymbol{x}_k + (1 - \alpha_k)\boldsymbol{p}_k$
10:         $\boldsymbol{h}_{k+1} := \boldsymbol{x}_{k+1}/\|\boldsymbol{x}_{k+1}\|$
11:         $k := k + 1$
12:     end if
13: end while
```

At Step 8, the step size $\alpha_k$ is chosen as $\min\{\max\{\theta_k, 0\}, 1 - \eta\}$ so that the update in Step 9 keeps $\boldsymbol{x}_{k+1}$ inside the convex hull $C(P)$. Since $\boldsymbol{x}_k$ and $\boldsymbol{p}_k$ lie in $C(P)$, their convex combination $\boldsymbol{x}_{k+1}$ also lies in $C(P)$. Under the following additional assumption: the point $\boldsymbol{p}_k$ selected in Algorithm 2 satisfies $\boldsymbol{h}^{*\top}\boldsymbol{p}_k = \gamma$, where $\gamma = \min_{\boldsymbol{p}\in C(P)} \boldsymbol{h}^{*\top}\boldsymbol{p}$, the convergence analysis with respect to the fixed normal vector $\boldsymbol{h}^*$ becomes tractable, and we obtain the following result.

**Theorem 3.2** (Asymptotic convergence)**.** *Let $\boldsymbol{h}^* \in \mathbb{R}^n$ be a fixed normal vector and define $\gamma :=$* $\min_{\boldsymbol{p}\in C(P)} \boldsymbol{h}^{*\top}\boldsymbol{p}$, $d_k := \boldsymbol{h}^{*\top}\boldsymbol{x}_k - \gamma$ $(\geq 0)$*. Assume that the point $\boldsymbol{p}_k$ selected in Algorithm 2 always satisfies $\boldsymbol{h}^{*\top}\boldsymbol{p}_k = \gamma$. Then the update rule of Algorithm 2 yields*

$$d_{k+1} = \alpha_k d_k \leq (1 - \eta)d_k.$$

*Consequently, $d_k \to 0$ as $k \to \infty$.*

*Proof.* Applying $\boldsymbol{h}^{*\top}$ to the update $\boldsymbol{x}_{k+1} = \alpha_k \boldsymbol{x}_k + (1 - \alpha_k)\boldsymbol{p}_k$ gives

$$\boldsymbol{h}^{*\top}\boldsymbol{x}_{k+1} = \alpha_k \boldsymbol{h}^{*\top}\boldsymbol{x}_k + (1 - \alpha_k)\boldsymbol{h}^{*\top}\boldsymbol{p}_k.$$

Since $\boldsymbol{h}^{*\top}\boldsymbol{p}_k = \gamma$ by assumption,

$$d_{k+1} = \boldsymbol{h}^{*\top}\boldsymbol{x}_{k+1} - \gamma = \alpha_k(\boldsymbol{h}^{*\top}\boldsymbol{x}_k - \gamma) = \alpha_k d_k.$$

Because $\alpha_k \leq 1 - \eta$, it follows that $d_{k+1} \leq (1 - \eta)d_k$, and hence $d_k \to 0$. □

Building upon the asymptotic convergence established in Theorem 3.2, we next derive a theoretical upper bound on the number of iterations required for the NSA algorithm to achieve a specified accuracy.

**Theorem 3.3** (Upper bound of NSA)**.** *Suppose that* $d_k := \boldsymbol{h}^{*\top}\boldsymbol{x}_k - \gamma$ $(\geq 0)$ *is defined with the same fixed normal vector* $\boldsymbol{h}^*$ *and constant* $\gamma$ *as in Theorem 3.2. If*

$$k = \left\lceil \frac{\log(d_0/\varepsilon)}{\log(1/(1-\eta))} \right\rceil,$$

*then* $d_k \leq \varepsilon$ *holds.*

*Proof.* From Theorem 3.2, the update in Algorithm 2 satisfies

$$d_{k+1} = \alpha_k d_k \leq (1-\eta)d_k, \qquad 0 \leq \alpha_k \leq 1-\eta.$$

By induction,

$$d_k \leq (1-\eta)^k d_0.$$

To ensure $d_k \leq \varepsilon$, it suffices that

$$(1-\eta)^k d_0 \leq \varepsilon,$$

which is equivalent to

$$k \geq \frac{\log(\varepsilon/d_0)}{\log(1-\eta)} = \frac{\log(d_0/\varepsilon)}{\log(1/(1-\eta))}.$$

Taking the smallest such integer yields the desired bound. □

## 4. Numerical experiments

In this section, we present numerical experiments to evaluate the effectiveness of NSA. We consider two classes of linear programs: (i) a standard-form LP with box constraints (Type 1) and (ii) an $\ell_1$-minimization problem (Type 2). For each class, we benchmark NSA against representative classical and modern algorithms.

### *4.1. Type 1: Standard-form LP with box constraints*

Following the problem setting used for evaluating the LP-Newton method in [17], we consider the following linear programming problem with box constraints:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{maximize}} \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} \quad & A\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}. \end{aligned} \tag{4.1}$$

The data for this problem are generated as follows:

- Each entry of the coefficient matrix $A \in \mathbb{R}^{m\times n}$ is independently drawn from the uniform distribution $\mathcal{U}[0, 1]$.
- Each entry of the objective coefficient vector $\boldsymbol{c} \in \mathbb{R}^n$ is independently sampled from the uniform distribution $\mathcal{U}[-0.5, 0.5]$.
- The true solution $\boldsymbol{x}_{\text{true}} \in [0, 10]^n$ is generated from the uniform distribution $\mathcal{U}([0, 10]^n)$.
- The right-hand side vector $\boldsymbol{b}$ is computed as $\boldsymbol{b} = A\boldsymbol{x}_{\text{true}}$.
- The box constraints are set as $\boldsymbol{l} = \boldsymbol{0}$ and $\boldsymbol{u} = 10 \times \boldsymbol{1}$.

### 4.2. *Type 2: $\ell_1$-minimization problem*

Next, we consider the $\ell_1$-minimization problem, which plays a central role in sparse signal recovery and compressed sensing, as discussed in [14, 35]:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \|\boldsymbol{x}\|_1 \\ \text{subject to} \quad & A\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}. \end{aligned} \tag{4.2}$$

The data for this problem are generated as follows:

- Each entry of the coefficient matrix $A \in \mathbb{R}^{m \times n}$ is independently drawn from the standard normal distribution $\mathcal{N}(0, 1)$.
- The true solution $\boldsymbol{x}_{\text{true}}$ is a sparse vector in $[0, 10]^n$, where the number of nonzero entries is at most 10% of $n$.
- The right-hand side vector $\boldsymbol{b}$ is computed as $\boldsymbol{b} = A\boldsymbol{x}_{\text{true}}$.
- The box constraints are set in the same way as in Type 1: $\boldsymbol{l} = \boldsymbol{0}$ and $\boldsymbol{u} = 10 \times \boldsymbol{1}$.

### 4.3. *Compared methods and experimental settings*

We compare five methods: the simplex method, the primal-dual interior point method, the LP-Newton method based on the Wolfe algorithm, the LP-Newton method with Perceptron, and the LP-Newton method with NSA. For the simplex and interior point methods, both Type 1 and Type 2 problems are first converted into standard equality form before being solved.

Computational settings. The method-specific settings are as follows:

- **Simplex method:** We adopt a two-phase procedure to obtain an initial feasible solution. To avoid cycling and guarantee finite termination, we use Bland's rule for selecting entering and leaving basic variables.
- **Primal-dual interior point method:** The primal and dual variables are updated using Mehrotra's predictor-corrector steps. Step-size control and automatic adjustment of the barrier parameter are employed to ensure stable convergence.
- **LP-Newton method:** Based on [17, 22], parameters related to the stopping criteria are adjusted appropriately.
- **LP-Newton method with Perceptron:** For the separation process, the Perceptron algorithm, which is widely used in the field of machine learning, was employed. The maximum number of iterations was set to $10^6$, and the stopping tolerance for the separation theorem was set to $\varepsilon = 10^{-10}$.
- **LP-Newton method with NSA:** The step-size parameter is set as $\eta = 10^{-3}$, the maximum number of iterations is $10^6$, and the stopping tolerance is $\varepsilon = 10^{-10}$.

All experiments were conducted on a MacBook Air (2022, Apple M2, 8-core CPU, 8GB RAM) running macOS 14.6.1 with MATLAB R2024a. Runtime was measured using `CPU time`. Each experiment was performed 10 times, and the average runtime was reported. In the numerical evaluation, we computed the objective gap with respect to the optimal values of problems (4.1) and (4.2), corresponding to the Type 1 and Type 2 formulations, respectively. We compare all five methods for $(m, n) \in \{10, 30, 50, 100, 110\} \times \{200, 350, 500\}$, and the results are summarized in Tables 1 and 2.

**Table 1.** Average `CPU time` (in seconds, 10 trials) for various methods and $(m, n)$ dimensions (Type 1 problems).

| Dimension $(m, n)$ | Simplex method | Primal-dual interior point method | LP-Newton (Wolfe) | LP-Newton (Perceptron) | LP-Newton (NSA) |
|---|---|---|---|---|---|
| (10, 200) | 2.7 | 0.04 | 0.05 | 3.1 | 1.5 |
| (10, 350) | 9.0 | 0.05 | 0.07 | 4.4 | 2.8 |
| (10, 500) | 20.2 | 0.16 | 0.07 | 4.9 | 3.4 |
| (30, 200) | 11.2 | 0.11 | 0.4 | 9.5 | 3.9 |
| (30, 350) | 56.2 | 0.36 | 0.5 | 12.9 | 5.2 |
| (30, 500) | 124.1 | 0.29 | 0.5 | 19.7 | 6.7 |
| (50, 200) | 23.2 | 0.35 | 2.0 | 17.8 | 6.2 |
| (50, 350) | 109.7 | 0.65 | 25.5 | 341.8 | 140.9 |
| (50, 500) | 275.1 | 0.51 | 21.4 | 477.6 | 148.0 |
| (100, 200) | 164.0 | 0.65 | 110.7 | 1193.5 | 181.0 |
| (100, 350) | 1361.6 | 3.73 | 167.5 | 1226.8 | 187.6 |
| (100, 500) | 4227.8 | 2.05 | 231.6 | 1259.0 | 211.6 |
| (110, 200) | 167.9 | 0.85 | 292.1 | 1146.5 | 222.6 |
| (110, 350) | 1708.5 | 1.74 | 445.0 | 1071.8 | 188.5 |
| (110, 500) | 4563.0 | 4.96 | 558.6 | 1299.3 | 214.7 |

**Table 2.** Average `CPU time` (in seconds, 10 trials) for various methods and $(m, n)$ dimensions (Type 2 problems).

| Dimension $(m, n)$ | Simplex method | Primal-dual interior point method | LP-Newton (Wolfe) | LP-Newton (Perceptron) | LP-Newton (NSA) |
|---|---|---|---|---|---|
| (10, 200) | 6.09 | 0.07 | 0.08 | 1.8 | 2.7 |
| (10, 350) | 21.9 | 0.12 | 0.3 | 105.9 | 130.3 |
| (10, 500) | 52.6 | 0.19 | 0.5 | 107.9 | 97.6 |
| (30, 200) | 28.3 | 0.36 | 0.6 | 5.1 | 6.6 |
| (30, 350) | 120.2 | 0.79 | 4.0 | 187.3 | 210.1 |
| (30, 500) | 323.2 | 0.66 | 4.6 | 219.4 | 190.6 |
| (50, 200) | 60.7 | 0.57 | 13.7 | 147.4 | 125.3 |

| Dimension $(m, n)$ | Simplex method | Primal-dual interior point method | LP-Newton (Wolfe) | LP-Newton (Perceptron) | LP-Newton (NSA) |
|---|---|---|---|---|---|
| (50, 350) | 296.0 | 0.82 | 20.6 | 296.0 | 146.9 |
| (50, 500) | 816.8 | 0.95 | 30.4 | 303.9 | 208.7 |
| (100, 200) | 514.0 | 3.23 | 82.4 | 247.5 | 306.1 |
| (100, 350) | 3551.1 | 2.84 | 192.3 | 400.0 | 287.5 |
| (100, 500) | 9560.9 | 6.75 | 241.6 | 425.8 | 369.6 |
| (110, 200) | 552.8 | 3.34 | 181.7 | 224.7 | 299.5 |
| (110, 350) | 3985.5 | 3.09 | 259.1 | 367.9 | 350.0 |
| (110, 500) | 10540.7 | 6.90 | 622.5 | 444.5 | 389.1 |

Preliminary results suggest that NSA is competitive and, in some instances, achieves superior performance on large-scale problems (e.g., $(m, n) = (100, 500)$). As the problem size increases, the runtime becomes comparable to that of the LP-Newton algorithm based on the Wolfe algorithm, despite the latter requiring exact minimum norm point computations. Furthermore, when compared under identical settings, the primal-dual interior point method exhibited the fastest and most stable convergence, especially for medium to large-scale problems.

To evaluate accuracy, Table 3 reports the objective gaps of NSA against the exact optimal values.

**Table 3.** Average objective gap (10 trials) of NSA.

| | (10, 200) | (10, 350) | (10, 500) | (30, 200) | (30, 350) | (30, 500) | (50, 200) | (50, 350) |
|---|---|---|---|---|---|---|---|---|
| Type 1 | 0.010 | 0.006 | 0.009 | 0.060 | 0.050 | 0.120 | 0.150 | 0.120 |
| Type 2 | 0.019 | 0.016 | 0.010 | 0.064 | 0.050 | 0.072 | 0.112 | 0.075 |

| | (50, 500) | (100, 200) | (100, 350) | (100, 500) | (110, 200) | (110, 350) | (110, 500) |
|---|---|---|---|---|---|---|---|
| Type 1 | 0.130 | 0.640 | 0.460 | 0.380 | 0.720 | 0.520 | 0.420 |
| Type 2 | 0.072 | 0.326 | 0.172 | 0.140 | 0.428 | 0.219 | 0.152 |

Table 3 shows that NSA tends to suffer from larger objective gaps as the number of constraints $m$ increases, particularly in Type 1 problems. On the other hand, relatively high accuracy was maintained in high-dimensional settings with large $n$, which was consistently observed across experiments.

To quantitatively analyze these trends, we conducted a second-order polynomial regression of the objective gap with respect to the number of constraints $m$ and variables $n$. The resulting approximation is given by:

$$\mathrm{gap}(m, n) \approx 0.0054 \cdot m - 0.0007 \cdot n + 0.000036 \cdot m^2 - 0.000012 \cdot (m \cdot n) + 0.000001 \cdot n^2 + 0.0243.$$

This regression model clearly indicates that the number of constraints $m$ has a dominant positive impact on the gap, leading to accuracy degradation as $m$ increases. In contrast, the effect of $n$ is relatively minor, and in some cases, better accuracy was observed in higher-dimensional settings. These findings suggest that NSA remains applicable even in large-scale problems with high variable dimensions.

To evaluate the reliability of this regression model, we computed the coefficient of determination $R^2$ and the root mean squared error (RMSE). The results showed that $R^2 = 0.989$ and RMSE was approximately 0.0247, indicating that the model fits the observed data well.

To further examine the structural characteristics of the gap behavior, we applied the Clough-Tocher interpolation method to the observed discrete data. This method is a multivariate generalization of Lagrange interpolation that constructs a piecewise quadratic spline over a triangulated domain and ensures global $C^1$ continuity while passing exactly through all observed points. It is particularly effective in capturing local variations and nonlinearities that are difficult to express using global polynomial regression.

From the resulting smooth interpolation surface, we sampled 200 internal points and performed another second-order polynomial fit, which yielded the following approximation:

$$\mathrm{gap}(m,n) \approx -0.0437 + 5.7425\times10^{-3}\cdot m - 3.7219\times10^{-4}\cdot n + 3.8252\times10^{-5}\cdot m^2 - 1.3613\times10^{-5}\cdot(m\cdot n) + 1.3173\times10^{-6}\cdot n^2.$$

This interpolated expression also indicates that the increase in the objective gap is primarily driven by the number of constraints $m$, and it is consistent with the regression-based findings. The result provides additional structural insight into the numerical characteristics of NSA.

Finally, we compare the LP-Newton method based on the Perceptron, which is categorized as a first-order method. For Type 1 problems, the LP-Newton method with NSA consistently outperformed the Perceptron-based methods, achieving up to six times faster performance improvement in large-scale instances. In Type 2 problems, the Perceptron-based method exhibited slightly better performance on small-scale instances; however, the LP-Newton method with NSA was more stable and faster for medium- and large-scale problems. Moreover, for Type 1 problems, the optimality error of the LP-Newton method with Perceptron increased rapidly with problem size, growing from approximately 0.03 in small instances to significantly larger values in large-scale cases. In contrast, as shown in Table 3, the LP-Newton method with NSA maintained consistently small errors across all scales and preserved high accuracy even in large problems, demonstrating its superiority in the separating-hyperplane strategy.

## 5. Conclusions

This paper proposes a new LP-Newton algorithm that constructs separating hyperplanes asymptotically, thereby eliminating the need for the Wolfe algorithm in solving linear programming problems with box constraints.

In a series of pilot numerical experiments, NSA exhibited longer computation times than the LP-Newton method based on the Wolfe algorithm on small-scale instances. However, as the problem size increased, this difference diminished, and comparable performance was observed at $(m, n) = (100, 500)$. Experiments with larger instances could not be completed due to numerical precision limitations in the current computational environment. Compared with the LP-Newton method using the Perceptron, NSA exhibited more stable performance. On the other hand, similar to other methods, NSA also faces challenges in constructing high-quality separating hyperplanes near the optimal solution.

As reported in [17], computation time tends to increase significantly as $m$ grows, and our results are consistent with this observation. Similarly, the relatively minor effect of $n$ on computation time also agrees with previous findings. Given these trends, it is reasonable to expect that the observed behavior may persist even for larger-scale problems, suggesting the potential applicability of the proposed method to high-dimensional settings.

A potential direction for future work is to enhance the current approach, which updates hyperplanes based on two points, by exploring constructions that incorporate multiple points simultaneously. Related strategies can also be found in the Dual Approach [23]. Although the present experiments achieved accuracy up to approximately one significant digit, introducing hyperplane rotations may lead to more precise optimization. It was also observed that the hyperplane algorithm requires substantially more computation time when the distance between the convex hull and the origin is small. Therefore, an alternative approach worth exploring is the direct computation of the intersection between a convex hull and a line, which may be applicable to the initialization of algorithms such as NSA in this study.

## Use of Generative-AI tools declaration

Portions of this manuscript were polished for English clarity using AI-based language assistance (OpenAI's ChatGPT). The scientific content, mathematical formulations, and conclusions were developed entirely by the author.

## Acknowledgments

The author would like to express his sincere gratitude to Professor Jianming Shi of Tokyo University of Science for his invaluable guidance, constructive suggestions, and continuous encouragement throughout this research.

## Conflict of interest

The author declares no conflict of interest.

## References

1. J. Dongarra, F. Sullivan, Guest editors introduction to the top 10 algorithms, *Comput. Sci. Eng.*, **2** (2000), 22–23. https://doi.org/10.1109/MCISE.2000.814652
2. G. B. Dantzig, Maximization of a linear function of variables subject to linear inequalities, In: *Activity analysis of production and allocation*, New York: Wiley, 1951, 339–347.
3. V. Klee, G. J. Minty, How good is the simplex algorithm?, In: *Inequalities III*, New York: Academic Press, 1972, 159–175.
4. L. G. Khachiyan, A polynomial algorithm in linear programming, *Sov. Math. Dokl.*, **20** (1979), 191–194.
5. N. Karmarkar, A new polynomial-time algorithm for linear programming, *Combinatorica*, **4** (1984), 373–395. https://doi.org/10.1007/BF02579150
6. M. Kojima, S. Mizuno, A. Yoshise, A primal-dual interior point algorithm for linear programming, In: *Progress in mathematical programming: Interior point and related methods*, New York: Springer, 1989, 29–47.

7. N. Megiddo, Pathways to the optimal set in linear programming, In: *Progress in mathematical programming: interior-point and related methods*, New York: Springer, 1989, 131–158. https://doi.org/10.1007/978-1-4613-9617-8_8
8. S. Mizuno, M. Kojima, A. Yoshise, A polynomial-time algorithm for a class of linear complementarity problems, *Math. Program.*, **44** (1989), 1–26. https://doi.org/10.1007/BF01587074
9. F. L. Hitchcock, The distribution of a product from several sources to numerous localities, *J. Math. Phys.*, **20** (1941), 224–230. https://doi.org/10.1002/SAPM1941201224
10. G. B. Dantzig, D. R. Fulkerson, S. M. Johnson, Solution of a large-scale traveling-salesman problem, *Oper. Res.*, **2** (1954), 393–410. https://doi.org/10.1287/opre.2.4.393
11. L. R. Ford, D. R. Fulkerson, Maximal flow through a network, *Canad. J. Math.*, **8** (1956), 399–404. https://doi.org/10.4153/CJM-1956-045-5
12. A. Ben-Tal, A. Nemirovski, Robust solutions of uncertain linear programs, *Oper. Res. Lett.*, **25** (1999), 1–13. https://doi.org/10.1016/S0167-6377(99)00016-4
13. S. Agrawal, Z. Wang, Y. Ye, A dynamic near-optimal algorithm for online linear programming, *arXiv preprint*, 2009, arXiv:0911.2974. https://doi.org/10.48550/arXiv.0911.2974
14. S. S. Chen, D. L. Donoho, M. A. Saunders, Atomic decomposition by basis pursuit, *SIAM Rev.*, **43** (2001), 129–159. https://doi.org/10.1137/S003614450037906X
15. Y. Ye, The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate, *Math. Oper. Res.*, **36** (2011), 593–603. https://doi.org/10.1287/moor.1110.0516
16. S. Mizuno, A strongly polynomial simplex method for totally unimodular LP, *Optimization Online, Technical Report*, 2014. Available at: `https://optimization-online.org/wp-content/uploads/2014/07/4452.pdf`
17. S. Fujishige, T. Hayashi, K. Yamashita, U. Zimmermann, Zonotopes and the LP-Newton method, *Optim. Eng.*, **10** (2009), 193–205. https://doi.org/10.1007/s11081-008-9067-x
18. T. Kitahara, S. Mizuno, J. Shi, The LP-Newton method for standard form linear programming problems, *Oper. Res. Lett.*, **41** (2013), 426–429. https://doi.org/10.1016/j.orl.2013.05.004
19. M. Tanaka, T. Okuno, Extension of the LP-Newton method to conic programming problems via semi-infinite representation, *Numer. Algorithms*, **86** (2021), 1285–1302. https://doi.org/10.1007/s11075-020-00933-6
20. T. Kitahara, N. Sukegawa, A simple projection algorithm for linear programming problems, *Algorithmica*, **81** (2019), 167–178. https://doi.org/10.1007/s00453-018-0436-3
21. Y. Matsuno, J. Shi, Rethinking linear programming: A new finite-termination LP-Newton approach, In: *Advances in data science and optimization of complex systems*, **1311** (2025), 76–82. https://doi.org/10.1007/978-3-031-90606-0_7
22. P. Wolfe, Finding the nearest point in a polytope, *Math. Program.*, **11** (1976), 128–149. https://doi.org/10.1007/BF01580381
23. S. Fujishige, P. Zhan, A dual algorithm for finding the minimum-norm point in a polytope, *J. Oper. Res. Soc. Japan*, **33** (1990), 188–195. https://doi.org/10.15807/jorsj.33.188

24. S. Fujishige, H. Sato, P. Zhan, An algorithm for finding the minimum-norm point in the intersection of a convex polyhedron and a hyperplane, *Japan J. Ind. Appl. Math.*, **11** (1994), 245–264. https://doi.org/10.1007/BF03167224

25. S. Fujishige, X. Liu, X. Zhang, An algorithm for solving the minimum-norm point problem over the intersection of a polytope and an affine set, *J. Optim. Theory Appl.*, **105** (2000), 113–147. https://doi.org/10.1023/A:1004666028951

26. K. Sekitani, Y. Yamamoto, Recursive algorithm for finding the minimum norm point in a polytope and a pair of closest points in two polytopes, *Math. Program.*, **61** (1993), 233–249. https://doi.org/10.1007/BF01582149

27. S. Fujishige, T. Kitahara, L. A. Végh, An update-and-stabilize framework for the minimum-norm-point problem, *Math. Program.*, **210** (2025), 281–311. https://doi.org/10.1007/s10107-024-02077-0

28. F. Bach, Learning with submodular functions: A convex optimization perspective, *Found. Trends Mach. Learn.*, **6** (2013), 145–373. https://doi.org/10.1561/2200000039

29. S. Fujishige, Lexicographically optimal base of a polymatroid with respect to a weight vector, *Math. Oper. Res.*, **5** (1980), 186–196. https://doi.org/10.1287/moor.5.2.186

30. S. Fujishige, *Submodular functions and optimization*, Amsterdam: North-Holland, 1991.

31. I. Shanu, C. Arora, P. Singla, Min norm point algorithm for higher order MRF-MAP inference, In: *2016 IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, 5365–5374. https://doi.org/10.1109/CVPR.2016.579

32. E. Klafszky, J. Mayer, T. Terlaky, Linearly constrained estimation by mathematical programming, *Eur. J. Oper. Res.*, **42** (1989), 254–267. https://doi.org/10.1016/0377-2217(89)90437-2

33. H. Takehara, An interior point algorithm for large scale portfolio optimization, *Ann. Oper. Res.*, **45** (1993), 373–386. https://doi.org/10.1007/BF02282059

34. J. A. De Loera, J. Haddock, A. Rademacher, The minimum Euclidean-norm point in a convex polytope: Wolfe's combinatorial algorithm is exponential, *SIAM J. Comput.*, **49** (2020), 138–169. https://doi.org/10.1137/18M1221072

35. E. J. Candès, T. Tao, Near-optimal signal recovery from random projections: Universal encoding strategies?, *IEEE Trans. Inf. Theory*, **52** (2006), 5406–5425. https://doi.org/10.1109/TIT.2006.885507