

Cycling in the Simplex and Dual Simplex Methods: A Comparative Study

Abdulazeez Hussaini Aikawa^{1*}, Aminu Salisu Taambu², Habibu Muhammad Haris², Mahmud Dalhat Mahmud², Tasiu Isah Basiru², Maryam Isah², Muhammad Sabiu², Firdausi Umar Muhammad², Ahmad Mustapha Abubakar³

¹Department of Mathematics, Northwest University, Kano, Nigeria

²Department of Mathematics, Mewar University, Chittorgarh, Rajasthan, India

³Yusuf Maitama Sule Collage of education and advanced studies, Ghari, Kano state, Nigeria

*Correspondence to: Abdulazeez Hussaini Aikawa, Department of Mathematics, Northwest University, Kano, Nigeria, E-mail: Abdulazeezaikawa@gmail.com

Received: August 27, 2026; Manuscript No: JMPM-26-6003; Editor Assigned: September 01, 2026; PreQc No: JMPM-26-6003 (PQ); Reviewed: October 23, 2026; Revised: October 26, 2026; Accepted: October 29, 2026; Published: November 23, 2026

Citation: Aikawa AH, Taambu AS, Haris HM, Mahmud MD, Basiru TI, Isah M, et al (2026). Cycling in the Simplex and Dual Simplex Methods: A Comparative Study. J. Math. Phys. Mech. Vol.1 Iss.1, November (2026), pp:404-416.

Copyright: © 2026 Abdulazeez Hussaini Aikawa, Aminu Salisu Taambu, Habibu Muhammad Haris, Mahmud Dalhat Mahmud, Tasiu Isah Basiru, Maryam Isah, Muhammad Sabiu, Firdausi Umar Muhammad, Ahmad Mustapha Abubakar. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

ABSTRACT

Cycling in the simplex and dual simplex methods is a long-recognized theoretical obstacle in linear programming (LP): in the presence of degeneracy, an unsuitable pivot-selection or tie-breaking rule can allow the algorithm to revisit a previously encountered basis without any improvement in the objective. Although the first examples were given by Hoffman and Beale for the primal simplex method, the dual simplex method, which is widely used for reoptimization in branch-and-bound solvers, is susceptible to an analogous mechanism through dual degeneracy. This paper presents a structured comparative analysis of cycling in both methods, examining the conditions that enable it, the structural similarities and practical differences, and the anti-cycling strategies available in each framework. The classical Beale example is worked in full, with every pivot listed and the return to the initial basis verified in exact arithmetic, and its dual counterpart is used to exhibit cycling of the dual simplex method. Bland's rule, lexicographic ordering, and perturbation are reviewed for both methods, and geometric and physical interpretations are presented as illustrative analogies. The study concludes that, although persistent cycling is uncommon in many practical LP computations, it remains a theoretically important phenomenon, and the analysis of dual degeneracy and cycling is practically relevant to LP reoptimization and branch-and-bound computations.

Keywords: Linear Programming; Simplex Method; Dual Simplex Method; Cycling; Degeneracy; Bland's Rule; Lexicographic Pivot; Perturbation; Convergence; Pivot Rules

INTRODUCTION

Linear programming occupies a central place in operations research and applied mathematics. The ability to optimize a linear objective over a polyhedral feasible region underpins decision-making in production scheduling, supply chain design, financial portfolio optimization, and network flow management, among many other domains. The simplex method, developed by Dantzig in 1947 (see Dantzig, 1951), has remained the algorithm of choice for LP over more than seven decades [1, 2]. Its practical efficiency on sparse real-world instances has kept it competitive with polynomial-time interior-point alternatives, and it continues to serve as the workhorse of modern commercial LP solvers. At the heart of the simplex method is a sequence of pivot operations that moves between adjacent vertices of the feasible polytope, each step ideally reducing the objective. When the problem is non-degenerate (meaning every basic feasible solution has exactly m positive basic variables), the method terminates in a finite number of steps because each distinct vertex can be visited at most once. Degeneracy complicates this picture considerably. A degenerate basic feasible solution has at least one basic variable equal to zero, and in such cases a pivot may change the basis representation without moving to a new vertex. When an unfortunate sequence of such pivots closes a loop, returning to a basis that was already visited, the algorithm enters a non-terminating cycle with no convergence. The dual simplex method, proposed by Lemke, operates from the perspective of dual feasibility, maintaining it throughout while progressively restoring primal feasibility [3]. It has become indispensable in sensitivity analysis and, more importantly, in the branch-and-bound frameworks used for integer programming, where branching decisions routinely produce dual-feasible, primal-infeasible subproblems

requiring warm-start re-optimization. Despite this central role, the dual simplex shares the same structural vulnerability to cycling as its primal counterpart, with dual degeneracy playing the role that primal degeneracy plays in the classic setting. Classical analyses of cycling tend to focus on the primal simplex method, with the dual case often treated by symmetry. The goal of this paper is to present a clear, side-by-side account of how cycling arises, what it implies, and how it can be prevented in each framework.

Contributions of This Paper

The main contributions of this work are as follows:

- A unified theoretical comparison of the cycling mechanism in both the simplex and dual simplex methods, demonstrating that the root cause is structurally analogous but operationally distinct.
- A structural classification of degeneracy-induced cycling across primal and dual frameworks, with worked tableau examples for each case.
- A fully worked and reproducible cycling example (Section 8.3): the classical Beale (1955) LP, with every tableau of the six-pivot cycle listed and verified in exact arithmetic, together with its dual counterpart for the dual simplex method (Section 5.3).
- A proposition (Section 6.1) stating the conditions that enable cycling in primal and dual simplex-type algorithms, which makes the structural parallel between the two frameworks precise.
- A synthesis of classical and modern anti-cycling strategies (Bland's rule, lexicographic ordering, and perturbation) evaluated for applicability in both primal and dual settings, alongside geometric and physical interpretations connecting the theory to practical LP contexts.

LITERATURE REVIEW

The table below presents selected representative contributions to the study of cycling, degeneracy, and pivot-rule design in linear programming. The selection spans foundational constructions, anti-cycling rules, and more recent solver-oriented and machine-learning work. The sources include journal articles, books, a technical report, a thesis, a conference paper, and a preprint; each entry describes the topic of the cited work and is not a claim about results established in this paper.

No.	Author(s), Year	Source	Key Contribution	Theme
1	Hoffman [4]	NBS Report 2974	Gave the first published example of cycling in the simplex method.	Foundational
2	Beale [5]	Naval Res. Logistics Q.	Gave a small LP on which the simplex method cycles under a naïve pivot rule; the standard textbook example of cycling.	Foundational
3	Charnes [6]	Econometrica	Introduced the perturbation approach for resolving degeneracy in linear programming.	Anti-cycling
4	Dantzig, Orden & Wolfe [1, 2]	Pacific J. Math.	Presented the generalized simplex method with a lexicographic rule for choosing the leaving variable, which prevents cycling.	Anti-cycling
5	Bland [7]	Math. Oper. Res.	Introduced smallest-index pivoting rules that guarantee finite termination of the simplex method.	Anti-cycling
6	Murty [8]	<i>Linear Programming</i> , Wiley	Provided textbook treatment of the simplex and dual simplex methods, including degeneracy.	Textbook

7	Ryan & Osborne [9]	Mathematical Programming	Proposed a technique for solving highly degenerate linear programs.	Degeneracy
8	Terlaky & Zhang [10]	Annals of Operations Research	Surveyed pivot rules for linear programming and their theoretical properties.	Survey
9	Fukuda & Terlaky [11]	Mathematical Programming	Surveyed criss-cross methods, which are finite pivoting algorithms that do not maintain primal or dual feasibility.	Alternative methods
10	Spielman & Teng [12, 13]	Journal of the ACM	Presented smoothed analysis showing that the simplex method with the shadow-vertex rule has polynomial smoothed complexity; a running-time result rather than a result about cycling.	Theory
11	Goldfarb & Reid [14]	Mathematical Programming	Introduced a practicable steepest-edge pricing rule for the simplex method.	Pricing
12	Todd [15]	Mathematical Programming	Surveyed linear programming, covering the simplex method among other topics.	Survey
13	Koberstein [16]	Doctoral Thesis, Paderborn	Developed techniques for fast and stable implementation of the dual simplex method, including treatment of degeneracy and cycling.	Implementation
14	El Hallaoui [17]	INFORMS Journal on Computing	Proposed an improved primal simplex algorithm designed for degenerate linear programs.	Degeneracy
15	Zanette, Fischetti & Balas [18]	Mathematical Programming	Examined the role of lexicography and degeneracy in pure cutting-plane algorithms.	Cutting planes
16	Achterberg [19]	Mathematical Programming Computation	Described the SCIP framework for constraint integer programming, in which LP relaxations are solved within branch-and-bound.	Branch-and-bound
17	Huangfu & Hall [20]	Mathematical Programming Computation	Described the parallelization of the dual revised simplex method used in the HiGHS solver.	Modern solver

18	Li [21]	Science China Mathematics	Proposed a reinforcement-learning/ Monte Carlo tree-search approach to finding optimal pivot paths of the simplex method.	Machine learning
19	Gill, Murray, Saunders & Wright [22]	Mathematical Programming	Described the EXPAND procedure for preventing cycling in active-set methods for linearly constrained optimization, including the simplex method.	Anti-cycling
20	Hall & McKinnon [23]	Mathematical Programming	Introduced a class of simple LP examples on which the simplex method cycles and showed that the EXPAND procedure is not guaranteed to prevent cycling.	Foundational
21	Zörnig [24]	Computational Optimization and Applications	Presented systematic procedures for constructing LP examples that cycle under several pivot-selection strategies.	Cycling examples
22	Yang [25]	arXiv:2101.01805	Collected small LP problems that cycle under Dantzig's rule or the steepest-edge rule, providing test cases for new simplex algorithms.	Cycling examples
23	Bach, Black, Huiberts & Kafer [26]	STOC 2026	Proposed a by-the-book analysis of the simplex method, giving a polynomial running-time result under assumptions modeled on implementations; a running-time result rather than a result about cycling.	Theory

Table 1: Selected Representative Contributions on Cycling, Degeneracy, And Pivot Rules in Linear Programming

Preliminaries

Standard Form of LP

A linear program in standard form is written as

$$\text{minimize } c^T x \text{ subject to } Ax = b, x \geq 0, \quad (1)$$

where $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, and A has full row rank. For a basis B consisting of m linearly independent columns of A , the corresponding basic solution is $x_B = B^{-1}b$, with the nonbasic variables set to zero. When $x_B \geq 0$ the solution is a basic feasible solution (BFS). The BFS is degenerate if at least one basic variable is equal to zero.

The Simplex Method

Starting from an initial BFS, the simplex method selects an entering variable with a strictly negative reduced cost and a leaving variable via the minimum-ratio test. A pivot updates the basis and, in the non-degenerate case, strictly decreases the objective. When degenerate pivots are possible (the minimum-ratio test yields a zero-valued ratio), a basis change occurs with no objective improvement, which opens the possibility of basis repetition and cycling.

The Dual Simplex Method

The dual simplex method begins from a dual-feasible, primal-infeasible basis. A basic variable with a negative value is selected to leave the basis (under the largest-infeasibility rule, the most negative one), and the entering variable is selected by the dual ratio test so as to maintain dual feasibility. Dual degeneracy occurs when one or more nonbasic variables have zero reduced cost, which can produce ties in the dual ratio test. Under arbitrary tie-breaking in this situation, the dual simplex method may execute a pivot that changes the basis without changing the dual objective value, which creates the conditions for a cycle.

Notation and Sign Conventions

Throughout the paper, $z = c^T x$ denotes the objective value of the minimization problem under consideration, and the reduced cost of a nonbasic variable x_j is $r_j = c_j - c^T B^{-1} A_j$. Every tableau lists the entries a_{ij} and the basic-variable values u_i of the current basis, and its last row shows the reduced costs r_j and, in the RHS column, the value $-z$. The Beale LP (4) has objective z . The LP (3) obtained from its dual has objective $z_D = s_3$, and its tableaux use the same convention with z_D in place of z . The relation between the two optimal values, $z^* = -z^*$, is derived in Section 5.3.

Cycling in the Simplex Method

The Degeneracy–Cycling Connection

A degenerate BFS admits multiple basis representations at the same geometric vertex. The simplex method, operating in the space of bases rather than the space of points, can pivot among these representations without ever leaving the vertex. Each such pivot is degenerate: the objective does not change, no new vertex is reached, and no real progress is made toward optimality. An adversarial sequence of degenerate pivots can return the algorithm to a basis it has already visited, at which point the same sequence of pivots will repeat indefinitely. Cycling is therefore a consequence of an insufficiently designed pivot rule on a problem that happens to have degenerate vertices; it is not a symptom of a poorly posed problem.

Beale's and Hoffman's Examples

The earliest example of cycling in the simplex method is due to Hoffman. Beale later gave a smaller and more widely cited example in which, under the naïve pivot rule specified in Section 8.2, the simplex method returns to its initial basis after six pivots, forming a non-terminating cycle. In both examples the cycle consists entirely of degenerate pivots: the initial basic feasible solution is degenerate, the objective value does not change during the cycle, and the cycle can be avoided by a different pivot choice. These examples established that cycling is not merely an abstract concern but a concrete and reproducible failure mode. The Beale example is worked in full in Section 8.

Primal Tableau Illustration

Table 2 shows the initial tableau of the Beale example (the LP is stated in Section 8.1, Eq. (4)). The basic variables x_1 and x_2 both have right-hand side zero, so the initial basic feasible solution is degenerate. Under the naïve rule specified in Section 8.2 (entering variable: most negative reduced cost; leaving variable: minimum ratio, ties broken by the smallest index), six consecutive pivots return to exactly this tableau.

Basis	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS
x_1	1	0	0	$1/4$	-8	-1	9	0
x_2	0	1	0	$1/2$	-12	$-1/2$	3	0
x_3	0	0	1	0	0	1	0	1
r_j	0	0	0	$-3/4$	20	$-1/2$	6	0

Table 2: The initial tableau of the Beale

Cycling in the Dual Simplex Method

Dual Degeneracy as the Enabling Condition

In the dual simplex method, the entering variable is chosen by the dual ratio test: among columns with a negative pivot element in the leaving variable's row, the one achieving the smallest ratio of reduced cost to absolute pivot entry is selected. When two or more nonbasic variables attain the minimum ratio, in particular when several of them have zero reduced cost, the dual ratio test yields a tie that requires an explicit tie-breaking rule. Under arbitrary tie-breaking, the resulting pivot may change the basis without changing the

dual objective value and without making progress toward restoring primal feasibility. A sequence of such degenerate dual pivots can return to a previously visited basis, completing a cycle in which the algorithm never terminates.

Dual Cycling via the Primal–Dual Correspondence

The dual simplex method applied to a linear program is, pivot for pivot, the primal simplex method applied to its dual: a basis of one problem corresponds to the complementary basis of the other, a variable that leaves the basis in one problem is the variable that enters in the other, and the objective values coincide. Consequently, every cycle of the primal simplex method has a counterpart cycle of the dual simplex method on the dual LP, with the same length and the same constant objective value, and primal degeneracy in the first problem appears as dual degeneracy in the second. For background on the simplex and dual simplex methods and on degeneracy see Murty and Koberstein. The example below is not a new construction: it is the dual of the classical Beale (1955) LP, and its pivot sequence has been verified in exact arithmetic.

Dual Counterpart of the Beale Example

The dual of the Beale LP (Section 8.1, Eq. (4)) is

$$\text{maximize } b^T y \quad \text{subject to } A^T y + s = c, \quad s \geq 0, \quad (2)$$

with $b = (0, 0, 1)^T$ and y free. The constraints for x_1, x_2, x_3 give $y_i = -s_i$ ($i = 1, 2, 3$), so the free dual variables can be eliminated. Moreover $b^T y = y_3 = -s_3$, so maximizing $b^T y$ is equivalent to minimizing

s_3 . This yields the following LP in the nonnegative variables $s_1, \dots, s_7$:

$$\begin{aligned} \text{minimize } z_D = s_3 \\ \text{subject to } & \frac{1}{4}s_1 - \frac{1}{2}s_2 + s_4 = -\frac{3}{4}, \\ & 8s_1 + 12s_2 + s_5 = 20, \\ & s_1 + \frac{1}{2}s_2 - s_3 + s_6 = -\frac{1}{2}, \\ & -9s_1 - 3s_2 + s_7 = 6, \\ & s_1, s_2, \dots, s_7 \geq 0. \end{aligned} \quad (3)$$

Sign Convention

The Beale LP (4) minimizes z , with optimal value $z^* = -5/4$. By strong duality $\max b^T y = z^* = -5/4$, and since $b^T y = -s_3$, the LP (3) has optimal value $z^*_D = 5/4 = -z^*$. Along the cycle both objectives are constant, $z = 0$ and $z_D = 0$. The initial basis $\{s_4, s_5, s_6, s_7\}$ is dual feasible (the reduced costs of the nonbasic variables s_1, s_2, s_3 are 0, 0, 1) and primal infeasible ($s_4 = D - 3/4$ and $s_6 = -1/2$ are negative). The zero reduced costs of s_1 and s_2 make the basis dual degenerate. Table 3 gives the initial tableau. The naïve dual rule used here is: the leaving variable is the basic variable with

Basis	s_1	s_2	s_3	s_4	s_5	s_6	s_7	RHS
s_4	$-1/4$	$-1/2$	0	1	0	0	0	$-3/4$
s_5	8	12	0	0	1	0	0	20
s_6	1	$1/2$	-1	0	0	1	0	$-1/2$
s_7	-9	-3	0	0	0	0	1	6
r_j	0	0	1	0	0	0	0	0

Table 3: Initial tableau of the dual counterpart (3) of the Beale LP, with basis $\{s_4, s_5, s_6, s_7\}$. The last row gives the reduced costs r_j and, in the RHS column, $-z_D$. The basis is dual feasible and primal infeasible ($s_4, s_6 < 0$), and the zero reduced costs of s_1 and s_2 make it dual degenerate the most negative value (ties: smallest index); the entering variable is determined by the dual ratio test, that is, the minimum of $r_j / |a_{rj}|$ over columns with $a_{rj} < 0$ (ties: smallest index). Under this rule the dual simplex method performs the six pivots listed in Table 4. Every pivot is degenerate (the objective $z_D = s_3$ stays at 0), and the basis after pivot 6 is the initial basis $\{s_4, s_5, s_6, s_7\}$; the tableau is then identical to Table 3 up to the order of its rows, so the sequence repeats indefinitely. Comparing with Table 6, each pivot of the dual simplex method on this LP is the corresponding pivot of the primal simplex method on the Beale LP, with the complementary basis; the last column of Table 4 lists the primal pivot concerned

Iter.	Leaving(u_i)	Entering	Tied columns	Basis after pivot	z_D	Degenerate	Primal pivot (Table 6)
-------	------------------	----------	--------------	-------------------	-------	------------	------------------------

0	n/a	n/a	n/a	{s4,s5,s6,s7}	0	n/a	n/a
1	s4 (−3/4)	s1	s1,s2	{s1,s5,s6,s7}	0	Yes	x4 in,x1 out
2	s5(−4)	s2	none	{s1,s2,s6,s7}	0	Yes	x5 in,x2 out
3	s6(−2)	s4	s4,s5	{s1,s2,s4,s7}	0	Yes	x6 in,x4 out
4	s7(−3)	s5	none	{s1,s2,s4,s5}	0	Yes	x7 in,x5 out
5	s1(−1)	s6	s6,s7	{s2,s4,s5,s6}	0	Yes	x1 in,x6 out
6	s2(−2)	s7	none	{s4,s5,s6,s7}	0	Yes	x2 in,x7 out

Table 4: Six-pivot cycle of the dual simplex method on the dual counterpart (3) of the Beale LP under the naïve rule. The value u_i of the leaving variable is shown in parentheses, and “tied columns” are those that attain the minimum dual ratio. Basis sets are listed in increasing index order

Operational Consequence

The dual simplex method plays an important role in LP reoptimization and branch-and-bound computations, making the analysis of dual degeneracy and cycling practically relevant. The consequences of cycling differ in the two methods. When the primal simplex method cycles, every basis visited is primal feasible, so the current point is a feasible, though not necessarily optimal, solution. When the dual simplex method cycles, every basis visited is dual feasible but primal infeasible, so no progress is made toward primal feasibility, although the dual objective value, which bounds the optimal value, is preserved. In a solver that embeds the dual simplex method as a subroutine, a non-terminating dual cycle would therefore stall the enclosing procedure, which is why anti-cycling safeguards are needed there as well.

Comparative Analysis of Cycling Behavior

Table 5 draws together the key dimensions of comparison between cycling in the two methods. The structural parallel is clear: both methods cycle through the same degeneracy-driven mechanism, both are governed by the pivot rule, and both require the same class of remedies. The differences lie primarily in which type of feasibility is at stake and in the operational consequences for the solver.

Feature	Simplex Method	Dual Simplex Method
Feasibility maintained	Primal feasibility throughout	Dual feasibility throughout
Root cause	Primal degeneracy: some basic variable equals zero	Dual degeneracy: zero reduced cost for some nonbasic variable, causing ties in the dual ratio test
Degenerate pivot effect	Objective unchanged; vertex does not change	Dual objective unchanged; primal infeasibility is not repaired
Consequence of cycling	Stalls at a feasible but non-optimal vertex	Stalls without reaching primal feasibility; dual objective value does not change
Classical example	Beale (1955): six-pivot cycle (Section 8)	Dual of the Beale LP: six-pivot cycle (Section 5.3)
Anti-cycling tools	Bland’s rule; lexicographic rule; perturbation	Dual Bland’s rule; dual lexicographic rule; dual perturbation
Detection in practice	Objective stagnation; basis repetition check	Dual objective stagnation; basis repetition check
Practical frequency	Uncommon in practice; solvers include perturbation and related safeguards	Uncommon in practice; analogous safeguards address dual degeneracy
Theoretical study depth	Studied since Hoffman (1953) and Beale (1955)	Anti-cycling theory transfers from the primal case by LP duality

Table 5: Structured Comparison of Cycling in the Simplex and Dual Simplex Methods Across Nine Dimensions

Formal Characterization

Proposition 1 (Conditions Enabling Cycling in Simplex Frameworks). Cycling in simplex-type pivot algorithms is enabled by degeneracy together with a pivot-selection or tie-breaking rule that permits the algorithm to revisit a previously encountered basis. Primal cycling is associated with degenerate basic feasible solutions, while analogous cycling in the dual simplex framework is associated with dual degeneracy and ties in the relevant ratio test. Degeneracy alone does not imply cycling; whether a cycle occurs depends on the pivot rule and the sequence of basis changes.

This proposition summarizes what Table 5 shows informally: the distinction between primal and dual cycling lies not in the combinatorial mechanism (basis repetition under degeneracy) but in the type of degeneracy that enables it and the type of feasibility that is maintained during the cycle. Because the dual simplex method applied to an LP is the primal simplex method applied to its dual, pivot rules that prevent basis repetition in one framework have direct analogues in the other. What the table cannot fully capture is the deeper shared insight: cycling is not a property of the LP problem itself, but of the combination of the problem's degeneracy structure with a pivot rule that fails to guard against basis repetition. An LP on which one rule cycles may terminate correctly under another, as Section 8.2 shows for Bland's rule.

Anti-Cycling Strategies

Bland's Rule

Bland's rule is a simple provably cycle-free pivot rule [7]. It states: always select the entering variable with the smallest index among all candidates with a strictly negative reduced cost, and always select the leaving variable with the smallest index among those tied in the minimum ratio test. The proof of finiteness proceeds by contradiction: assuming that a cycle exists under this rule leads to a contradiction with the index ordering. The dual analogue applies the same index ordering to the leaving variable selection and to the tie-breaking of the dual ratio test, and the finiteness proof carries over by LP duality. Bland's rule is often not the most efficient choice on typical LP problems: its insistence on index ordering can lead it to select pivots that are far from the most improving, which may result in more iterations than the Dantzig rule. For this reason, it is better suited as a fallback when stagnation is detected than as the primary rule throughout.

Lexicographic Rule

The lexicographic pivot rule was developed by Dantzig, Orden and Wolfe, building on the perturbation idea of Charnes [2, 6]. It resolves ties in the minimum ratio test by comparing rows of the simplex tableau lexicographically: each tied row is divided by its entry in the pivot column, and the lexicographically smallest result determines the leaving variable. Under this rule the rows of the tableau remain lexicographically positive and the objective row changes in a strictly lexicographically monotone way, so no basis can be repeated and termination is guaranteed. The rule is equivalent to the perturbation of Section 7.3 with a formal infinitesimal. A dual lexicographic rule applies the analogous ordering in the dual ratio test and gives the same guarantee for the dual simplex method.

Perturbation Method

Rather than modifying the pivot rule, perturbation addresses cycling by modifying the problem. A small positive quantity ϵ_i is added to each right-hand side value b_i , producing a perturbed LP that is non-degenerate by construction when the ϵ_i are chosen in general position, for example, symbolically as ϵ_i for a formal infinitesimal ϵ . Since a non-degenerate LP cannot cycle, the perturbed problem terminates finitely, and the solution converges to an optimal solution of the original as $\epsilon \rightarrow 0$. The dual perturbation method applies a corresponding perturbation to the cost vector, eliminating dual degeneracy in a directly analogous way.

Implementation in Modern Solvers

Modern LP solvers incorporate numerical and algorithmic safeguards to reduce the practical impact of degeneracy and cycling. These safeguards are important even though persistent cycling is uncommon in many practical LP computations. Commonly used devices include perturbation of the right-hand side or cost vector and pricing rules such as steepest-edge pricing Goldfarb and Reid, which tend to avoid long degenerate sequences although they provide no formal anti-cycling guarantee [14]. Implementation techniques for the dual simplex method, including the handling of degeneracy and cycling, are discussed by Koberstein, and the dual revised simplex method used in the HiGHS solver is described by Huangfu and Hall [16, 20].

Illustrative Examples

Problem Statement: Beale's Example

To ground the theoretical discussion in something concrete, consider the following LP, adapted from Beale (1955) in the form commonly given in LP textbooks:

$$\begin{aligned} \text{minimize } z = & -\frac{3}{4}x_4 + 20x_5 - \frac{1}{2}x_6 + 6x_7 \\ \text{subject to } & x_1 + \frac{1}{4}x_4 - 8x_5 - x_6 + 9x_7 = 0, \\ & x_2 + \frac{1}{2}x_4 - 12x_5 - \frac{1}{2}x_6 + 3x_7 = 0, \\ & x_3 + x_6 = 1, \\ & x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0. \end{aligned} \tag{4}$$

The initial basis is $B = \{x_1, x_2, x_3\}$, with basic feasible solution $x_1 = x_2 = 0$, $x_3 = 1$ and $x_4 = x_5 = x_6 = x_7 = 0$. Because the basic variables x_1 and x_2 are both zero, the solution is degenerate. The objective value at this vertex is $z = 0$, and the reduced costs of x_4, x_5, x_6, x_7 are $-3/4, 20, -1/2, 6$. The simplex method may nevertheless cycle under the naïve pivot rule stated below, as the following pivots demonstrate.

Cycling Under Naïve Rule vs. Bland's Rule

The naïve rule used here is: the entering variable is the nonbasic variable with the most negative reduced cost (ties: smallest index), and the leaving variable is determined by the minimum-ratio test (ties: smallest index of the leaving variable). Table 6 lists every pivot. Under this rule the Beale example returns to its initial basis after six pivots, forming a non-terminating cycle. Under Bland's rule the first three pivots coincide with the naïve rule. At pivot 4, however, Bland's rule brings in x_1 (the smallest index with a negative reduced cost) instead of x_7 , and x_5 leaves. Pivot 5 (x_2 enters, x_3 leaves) is the first pivot that decreases the objective, to $z = -1/2$. Pivot 6 (x_4 enters, x_2 leaves) gives the basis $\{x_6, x_1, x_4\}$ with $x_1 = 3/4$, $x_4 = 1$, $x_6 = 1$ and all other variables zero; all reduced costs are then nonnegative, so this solution is optimal with $z = -5/4$. Bland's rule therefore terminates after six pivots, whereas the naïve rule never terminates. The same holds for the dual LP (3). With the dual analogue of Bland's rule (Section 7.1) the dual simplex method performs the pivots (entering, leaving) (s_1, s_4) , (s_2, s_5) , (s_4, s_6) , (s_5, s_1) , (s_3, s_2) , (s_2, s_4) and terminates at the basis $\{s_5, s_3, s_2, s_7\}$ with $s_5 = 2$, $s_3 = 5/4$, $s_2 = 3/2$, $s_7 = 21/2$, so $z^* = 5/4 = -z^*$, in agreement with Section 5.3.

Iter.	Entering (r_j)	Leaving	Tied rows	Basis after pivot	z	Degenerate
0	n/a	n/a	n/a	$\{x_1, x_2, x_3\}$	0	n/a
1	$x_4 (-3/4)$	x_1	x_1, x_2	$\{x_2, x_3, x_4\}$	0	Yes
2	$x_5 (-4)$	x_2	none	$\{x_3, x_4, x_5\}$	0	Yes
3	$x_6 (-2)$	x_4	x_4, x_5	$\{x_3, x_5, x_6\}$	0	Yes
4	$x_7 (-3)$	x_5	none	$\{x_3, x_6, x_7\}$	0	Yes
5	$x_1 (-1)$	x_6	x_6, x_7	$\{x_1, x_3, x_7\}$	0	Yes
6	$x_2 (-2)$	x_7	none	$\{x_1, x_2, x_3\}$	0	Yes

Table 6: The six pivots of the Beale example under the naïve rule. The entering reduced cost r_j is shown in parentheses, and “tied rows” are the basic variables that attain the minimum ratio. The basis after pivot 6 is the initial basis $\{x_1, x_2, x_3\}$ (and the tableau is identical to Table 2), so the sequence repeats indefinitely. Every pivot is degenerate and $z = 0$ throughout.

Verified Classical Cycling Example

To provide a fully reproducible demonstration, this subsection uses a verified classical cycling example from the literature. The complete LP formulation, initial basis, pivot-selection rule, successive tableaux, and final return to the initial basis are given explicitly. The LP is the Beale (1955) example of Section 8.1 with the naïve rule of Section 8.2; all tableaux were computed in exact rational arithmetic. Table 2 is the initial tableau (basis $\{x_1, x_2, x_3\}$); Tables 7 to 11 give the tableaux after pivots 1 to 5, and pivot 6 returns to Table 2. The last row of each tableau shows the reduced costs r_j and, in the RHS column, $-z$. Pivot 6 (x_2 enters, x_7 leaves) produces a tableau identical to Table 2: the basis $\{x_1, x_2, x_3\}$ is revisited with the same tableau, so the same six pivots repeat indefinitely and the method never terminates. Every pivot in the sequence is degenerate: the right-hand sides remain 0, 0 and 1 and the objective value stays at 0. This example also illustrates why the ratio test alone cannot prevent cycling: when several rows attain the same (here zero) minimum ratio, the tie-breaking rule completely determines which basis is selected, and hence whether the sequence closes on itself [27]. Under Bland's rule (Section 8.2) the same LP is solved to optimality in six pivots. The exact-arithmetic computation behind Tables 2 to 11 is reproduced by the supplementary script `verify_cycling.py`.

Basis	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS
x_4	4	0	0	1	-32	-4	36	0
x_2	-2	1	0	0	4	$3/2$	-15	0
x_3	0	0	1	0	0	1	0	1
r_j	3	0	0	0	-4	$-7/2$	33	0

Table 7: Tableau after Pivot 1 (x_4 enters, x_1 leaves); basis $\{x_2, x_3, x_4\}$, $z = 0$

Basis	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS
x_4	-12	8	0	1	0	8	-84	0
x_5	-1/2	1/4	0	0	1	$3/8$	$-15/4$	0

x_3	0	0	1	0	0	1	0	1
η_j	1	1	0	0	0	-2	18	0

Table 8: Tableau after Pivot 2 (x_5 enters, x_2 leaves); basis $\{x_3, x_4, x_5\}$, $z = 0$

Basis	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS
x_6	$-3/2$	1	0	$1/8$	0	1	$-21/2$	0
x_5	$1/16$	$-1/8$	0	$-3/64$	1	0	$3/16$	0
x_3	$3/2$	-1	1	$-1/8$	0	0	$21/2$	1
η_j	-2	3	0	$1/4$	0	0	-3	0

Table 9: Tableau after Pivot 3 (x_6 enters, x_4 leaves); basis $\{x_3, x_5, x_6\}$, $z = 0$

Graphical Representation and Physical Interpretation

Geometric Visualization of Cycling

Figure 1 provides a schematic comparison of cycling behavior and correct algorithmic termination on a small feasible polytope. Vertex V_2 is degenerate: it lies at the intersection of three binding constraints in a two-dimensional space, so it corresponds to three distinct basis representations at the same geometric point. The left panel illustrates a naïve pivot rule moving among these basis representations without geometric progress. The right panel illustrates an anti-cycling rule (Bland's rule) imposing a strict index ordering that leads through V_1, V_2, V_3, V_4 and finally to the optimal vertex V_5 in finitely many steps. Figure 1 is a conceptual illustration only: it is not computed from the LP of Section 8, and the positions of its vertices carry no numerical meaning. The numerical results derived from the actual LP examples are those of Tables 2 to 11. Figure 2 displays them graphically: it plots the six basis transitions of Table 6 for the Beale LP (4), together with the complementary dual bases visited by the dual simplex method in Table 4. Unlike Figure 1, Figure 2 is computed from the exact pivot sequences.

Basis	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS
x_6	2	-6	0	$-5/2$	56	1	0	0
x_7	$1/3$	$-2/3$	0	$-1/4$	$16/3$	0	1	0
x_3	-2	6	1	$5/2$	-56	0	0	1
η_j	-1	1	0	$-1/2$	16	0	0	0

Table 10: Tableau after Pivot 4 (x_7 enters, x_5 leaves); basis $\{x_3, x_6, x_7\}$, $z = 0$

Basis	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS
x_1	1	-3	0	$-5/4$	28	$1/2$	0	0
x_7	0	$1/3$	0	$1/6$	-4	$-1/6$	1	0
x_3	0	0	1	0	0	1	0	1
η_j	0	-2	0	$-7/4$	44	$1/2$	0	0

Table 11: Tableau after Pivot 5 (x_1 enters, x_6 leaves); basis $\{x_1, x_3, x_7\}$, $z = 0$

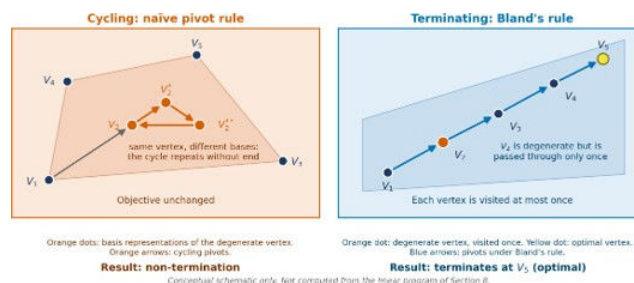


Figure 1: Conceptual Schematic (not computed from the LP of Section 8) of Cycling among Alternative Basis Representations of a Degenerate Vertex and Finite Progression under an Anti-cycling Pivot Rule

Physical Interpretation: Transportation and Production

The interpretations in Sections 9.2 and 9.3 are illustrative analogies intended to build intuition; they are not mathematical results of this paper. In a transportation LP, decision variables represent flow volumes along routes between supply and demand nodes. A basic variable at zero means that a route selected in the basis carries no flow. Degeneracy in a transportation problem occurs when a subset

of the supplies exactly matches a subset of the demands. Viewed as an analogy, the simplex method entering a non-terminating cycle corresponds to the optimizer repeatedly reassigning zero-flow routes without improving the total shipping cost: the algorithm reorganizes an empty part of the network without making any productive change to it. In production planning, a degenerate BFS arises when more resource constraints are binding than are needed to determine the vertex. As an analogy, a simplex algorithm that cycles in this context resembles a planner who reconsiders the same marginal resource trade-offs in sequence (substituting machine A for machine B, then B for C, then back to A) without ever committing to a production plan. Anti-cycling rules impose an external ordering that forces a definitive choice and ensures the algorithm moves to a genuinely different operating point.

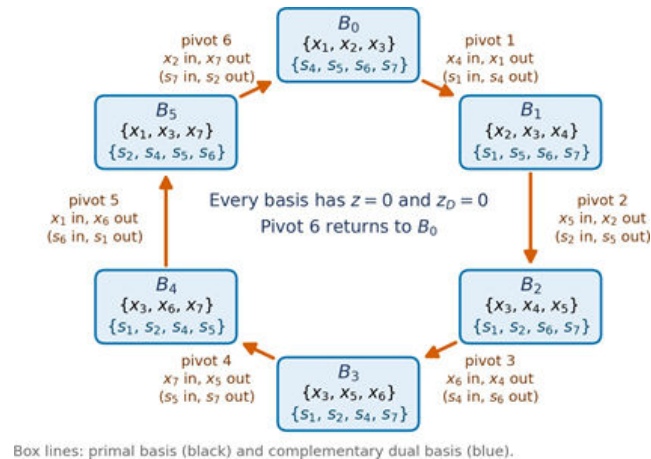


Figure 2: Basis-transition Cycle Computed in Exact Arithmetic. Each box shows a basis visited under Rule P on the Beale LP (4) (Table 6, black) and the complementary basis of the dual LP (3) (Table 4, blue). Arrows are the six pivots, and pivot 6 returns to B0

Physical Interpretation: Dual Simplex and Shadow Prices

The dual variables of an LP can be interpreted as shadow prices: the marginal value of relaxing a constraint by one unit. Dual degeneracy, meaning that a nonbasic variable has a reduced cost of exactly zero so that bringing it into the basis leaves the dual objective unchanged, corresponds to the existence of several price vectors with the same dual objective value. As an illustrative interpretation, a cycling dual simplex method can be viewed as moving among such equivalent price vectors without ever reaching one that yields primal feasibility [27-30]. This viewpoint may be suggestive for column generation and Dantzig-Wolfe decomposition, where the dual values of the restricted master problem drive the pricing of new columns. It is offered as an analogy only: establishing that a particular master problem cycles in this way would require a specific reference or computational evidence, which is outside the scope of this paper.

DISCUSSION

The analysis presented in this paper points to several conclusions worth drawing together explicitly. The first and most fundamental is that cycling is, in a precise sense, a property of the pivot rule rather than of the problem. The same LP that causes cycling under one rule will terminate correctly under another. This means that any theoretical claim about the correctness of a simplex-based algorithm must be accompanied by a specification of the pivot rule, and any implementation that omits an anti-cycling safeguard is, strictly speaking, algorithmically incomplete on the full class of LP instances. The second observation concerns the relative emphasis in the literature. Primal cycling has been studied since Hoffman's 1953 example, and anti-cycling rules such as Bland's rule, lexicographic ordering, and perturbation are well established for the primal simplex method. Because the dual simplex method applied to an LP is the primal simplex method applied to its dual, these tools transfer to the dual simplex method. The dual simplex method plays an important role in LP reoptimization and branch-and-bound computations, making the analysis of dual degeneracy and cycling practically relevant. A third observation concerns smoothed analysis. Spielman and Teng showed that the simplex method with the shadow-vertex pivot rule has polynomial smoothed complexity under random perturbation of the input data. This helps to explain the practical efficiency of the simplex method, but it is a running-time result and not a statement about cycling. Practical robustness against cycling in solvers is instead obtained through safeguards of the kind discussed in Section 7.4. Structured degeneracy, such as that found in LP relaxations of combinatorial problems, may still produce long degenerate sequences, so anti-cycling safeguards remain an important part of any correct implementation.

SUMMARY AND CONCLUSION

Summary

This paper has provided a structured comparative analysis of cycling in the simplex and dual simplex methods. The treatment began with a precise definition of degeneracy and described the cycling mechanism in both methods through the classical Beale (1955) example, worked in full with every tableau of the six-pivot cycle (Section 8.3), and through its dual counterpart (Section 5.3). The

comparative analysis in Section 6, summarized in Proposition 1, showed that the enabling conditions are structurally analogous (degeneracy together with a pivot rule that permits basis repetition), while the operational consequences differ. Anti-cycling strategies including Bland's rule, lexicographic ordering, and perturbation were reviewed for both frameworks. The geometric and physical interpretations in Section 9 were presented as illustrative analogies connecting the abstract mechanism to logistics, production planning, and shadow-price analysis.

Key Conclusions

Four conclusions stand out. First, cycling is a property of the pivot rule under degeneracy, not of the LP itself; the choice of pivot rule is, in degenerate problems, a question of algorithmic correctness rather than mere computational preference. Second, the dual simplex method is not immune to cycling: since it is the primal simplex method applied to the dual, every primal cycling example has a dual counterpart. The dual simplex method plays an important role in LP reoptimization and branch-and-bound computations, making the analysis of dual degeneracy and cycling practically relevant. Third, Bland's rule, lexicographic ordering, and perturbation each provide cycle-free guarantees applicable to both frameworks, with different practical trade-offs in efficiency. Fourth, modern LP solvers incorporate numerical and algorithmic safeguards against degeneracy, but these engineering measures do not substitute for formal anti-cycling guarantees in theoretical analysis and algorithm design. This highlights that cycling is not merely a pathological curiosity, but a structural phenomenon that must be addressed explicitly in both theoretical analysis and algorithm design.

Directions for Further Work

Several directions follow from this analysis. A computational study of how often degeneracy-related safeguards are triggered in large-scale dual simplex reoptimization, for example in branch-and-bound LP relaxations, would help quantify their practical role. On the theoretical side, a characterization of the LP structural properties (sparsity patterns, constraint matrix geometry) that make long degenerate sequences most likely could support structure-aware pivot rules. Finally, machine-learning approaches to pivot selection (for example, Li et al., 2024, who use reinforcement learning to search for optimal pivot paths) raise the question of whether learned pivot rules can be combined with formal cycle-freedom guarantees, a question that remains open.

REFERENCES

1. Dantzig GB. Maximization of a linear function of variables subject to linear inequalities. In: Koopmans TC, editor. *Activity Analysis of Production and Allocation*. New York: Wiley; 1951. p. 339-347.
2. Dantzig GB, Orden A, Wolfe P. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pac J Math*. 1955;5(2):183-195.
3. Lemke CE. The dual method of solving the linear programming problem. *Nav Res Logist Q*. 1954;1(1):36-47.
4. Hoffman AJ. Cycling in the simplex algorithm. National Bureau of Standards Report 2974. 1953.
5. Beale EML. Cycling in the dual simplex algorithm. *Nav Res Logist Q*. 1955;2(4):269-275.
6. Charnes A. Optimality and degeneracy in linear programming. *Econometrica*. 1952;20(2):160-170.
7. Bland RG. New finite pivoting rules for the simplex method. *Math Oper Res*. 1977;2(2):103-107.
8. Murty KG. *Linear Programming*. New York: Wiley; 1983.
9. Ryan DM, Osborne MR. On the solution of highly degenerate linear programmes. *Math Program*. 1988;41:385-392.
10. Terlaky T, Zhang S. Pivot rules for linear programming: A survey on recent theoretical developments. *Ann Oper Res*. 1993;46:203-233.
11. Fukuda K, Terlaky T. Criss-cross methods: A fresh view on pivot algorithms. *Math Program*. 1997;79(1-3):369-395.
12. Spielman DA, Teng SH. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. In: *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC 2001)*. New York: ACM; 2001. p. 296-305.
13. Spielman DA, Teng SH. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J ACM*. 2004;51(3):385-463.
14. Goldfarb D, Reid JK. A practicable steepest-edge simplex algorithm. *Math Program*. 1977;12:361-371.
15. Todd MJ. The many facets of linear programming. *Math Program*. 2002;91(3):417-436.
16. Koberstein A. The dual simplex method, techniques for a fast and stable implementation. Doctoral dissertation. Universität Paderborn; 2005.
17. El Hallaoui I, Metrane A, Desaulniers G, Soumis F. An improved primal simplex algorithm for degenerate linear programs. *INFORMS J Comput*. 2011;23(4):569-577.
18. Zanette A, Fischetti M, Balas E. Lexicography and degeneracy: Can a pure cutting plane algorithm work?. *Math Program*. 2011;130(1):153-176.
19. Achterberg T. SCIP: Solving constraint integer programs. *Math Program Comput*. 2009;1(1):1-41.
20. Huangfu Q, Hall JAJ. Parallelizing the dual revised simplex method. *Math Program Comput*. 2018;10(1):119-142.
21. Li A, Guo T, Han C, Li B, Li H. Optimal pivot path of the simplex method for linear programming based on reinforcement learning. *Sci China Math*. 2024;67:1263-1286.
22. Gill PE, Murray W, Saunders MA, Wright MH. A practical anti-cycling procedure for linearly constrained optimization. *Math Program*. 1989;45:437-474.
23. Hall JAJ, McKinnon KIM. The simplest examples where the simplex method cycles and conditions where EXPAND fails to prevent cycling. *Math Program*. 2004;100(1):133-150.
24. Zörnig P. Systematic construction of examples for cycling in the simplex method. *Comput Oper Res*. 2006;33(8):2247-2262.

25. Yang Y. A collection of cycling problems in linear programming. arXiv:2101.01805. 2021.
26. Bach E, Black AE, Huiberts S, Kafer S. Beyond smoothed analysis: Analyzing the simplex method by the book. In: Proceedings of the 58th Annual ACM Symposium on Theory of Computing (STOC 2026). New York: ACM; 2026. p. 222-233.
27. Andersen ED, Andersen KD. Presolving in linear programming. Math Program. 1995;71(2):221-245.
28. Berthold T, Gleixner AM. Undercover: A primal MINLP heuristic exploring a largest sub-MIP. Math Program. 2014;144(1-2):315-346.
29. Bixby RE. A brief history of linear and mixed-integer programming computation. Doc Math. 2012;Extra Vol Optimization Stories:107-121.
30. Borgwardt KH. The Simplex Method: A Probabilistic Analysis. Berlin: Springer; 1987.