OPEN ACCESS Freely available online

Review Article

Vol. 1, Issue 1, January-December 2026

Journal of Mathematics, Physics and Mechanics

Degeneracy in Linear Programming and the Simplex Methods - A Theoretical and Computational Analysis with Illustrative Examples

Aminu Salisu Taambu[1*], Pramod Mehta[2], Karuna Laddha[3], Habibu Muhammad Haris[4], Abubakar Sulaiman Muhammad[5], Adam Musa Garba[6], Najib bello halilu[7], Kabiru Isah[8], Isah Tasiu Basiru[9]

[1,2,3,4,5,6,7,8,9]*Department of Mathematics, Mewar University, Gangrar, India*

***Correspondence to:** Aminu Salisu Taambu, Department of Mathematics, Mewar University, Gangrar, India, E-mail: aminutaambu@gmail.com

Received: August 18, 2026; **Manuscript No:** JMPM-26-2703; **Editor Assigned:** August 20, 2026; **PreQc No:** JMPM-26-2703 (PQ); **Reviewed:** August 26, 2026; **Revised:** August 31, 2026; **Manuscript No:** JMPM-26-2703 (R); **Published:** October 07, 2026

Citation: Taambu AS, Mehta P, Laddha K, Haris HM, Muhammad SA, Garba AM, et al (2026). Degeneracy in Linear Programming and the Simplex Methods - A Theoretical and Computational Analysis with Illustrative Examples. J. Math. Phys. Mech. Vol.1 Iss.1, October (2026), pp:208-246.

Copyright: © 2026 Aminu Salisu Taambu, Pramod Mehta, Karuna Laddha, Habibu Muhammad Haris, Abubakar Sulaiman Muhammad, Adam Musa Garba, Najib bello halilu, Kabiru Isah,Isah Tasiu Basiru. 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

Degeneracy is a fundamental feature of linear programming that can affect the behaviour of the Simplex Method without changing the feasible solution or objective value. Although degeneracy does not necessarily cause difficulties, a sequence of degenerate pivots may result in stalling and, under an unfortunate pivot-selection rule, cycling. This paper provides a theoretical and illustrative analysis of degeneracy, stalling, and cycling in the Simplex Method. The distinction between a degenerate basic feasible solution, a degenerate pivot, and a tied minimum-ratio test is first clarified through basic simplex formulations and an illustrative numerical example. A classical cycling example is then used to demonstrate how repeated zero-step pivots can return the algorithm to a previously visited basis. Several anti-cycling strategies are subsequently examined, including Bland's rule, lexicographic pivoting, perturbation methods, and numerical safeguards. Particular attention is given to Bland's rule and its finite-termination property. The paper also illustrates the relevance of degeneracy and anti-cycling procedures to structured linear programming applications. The analysis shows that degeneracy itself does not imply cycling; rather, cycling depends on the interaction between degeneracy and the pivot-selection rule. The study provides a concise theoretical framework for understanding these issues and highlights the importance of deterministic tie-breaking procedures in reliable implementations of the Simplex Method.

Keywords: Linear Programming; Simplex Method; Degeneracy; Degenerate Basic Feasible Solution; Stalling; Cycling; Bland's Rule; Anti-Cycling Rules

INTRODUCTION

Linear programming (LP) is one of the basic tools of mathematical optimization. It is used to model problems in areas such as production planning, resource allocation, transportation, scheduling, and network design. When the objective function and the constraints are linear, the problem can be written as a linear program and studied using well-established optimization methods [1-4].

Among the methods developed for solving linear programs, the Simplex Method has a central place in the theory and practice of linear programming. The method, introduced by Dantzig, moves from one basic feasible solution to another through a sequence of pivot operations. Its geometric interpretation is based on the fact that, when an optimal solution exists, an optimum can be found at an extreme point of the feasible region [1-3].

The usual geometric interpretation of the Simplex Method is straightforward when the current basic feasible solution is nondegenerate. The situation is different when degeneracy is present. A basic feasible solution is called degenerate when at least one of its basic variables is equal to zero [1, 2]. From the geometric point of view, a degenerate vertex may have more active constraints than are needed to define the vertex. Consequently, the same geometric point may be represented by more than one basis [2,5].

Degeneracy can affect the behaviour of simplex iterations. At a degenerate basic feasible solution, a pivot may change the basis without changing the represented point or the value of the objective function. Such an iteration is often described as a degenerate pivot. A sequence of iterations with no improvement in the objective value can lead to stalling. These effects make the distinction between a geometric point and its algebraic representation by a basis particularly important [6,7].

Degeneracy should not be confused with cycling. Degeneracy is a property of a basic feasible solution, while cycling refers to the repetition of a previously visited basis. Stalling describes a sequence of iterations in which the objective value does not improve. Thus, a linear program may be degenerate without cycling, and a degenerate pivot does not by itself imply that the Simplex Method will cycle [8,9,10].

Cycling is nevertheless an important theoretical issue in simplex optimization. In a cycling sequence, the algorithm returns to a basis that has already occurred and can therefore repeat the same sequence of pivots indefinitely. Beale gave a classical example of cycling in the Simplex Method [11]. Later work showed that cycling examples can also be constructed systematically for different pivot-selection rules [5,12].

The possibility of cycling led to the development of several anti-cycling rules. Bland's rule uses an index-based choice for entering and leaving variables and guarantees finite termination of the Simplex Method [8]. Lexicographic and perturbation techniques provide other ways of resolving ties caused by degeneracy [13-15]. Practical procedures have also been developed. Gill, Murray, Saunders, and Wright proposed an anti-cycling procedure for linearly constrained optimization, including the Simplex Method, based on controlled infeasibility and an adjusted working feasibility tolerance [16]. Gal and Geue also studied pivoting strategies for degeneracy-related problems, including cycling and stalling [7].

Other pivot frameworks have been developed as alternatives to the standard primal Simplex Method. Fukuda and Terlaky studied criss-cross methods as a different class of pivot algorithms for linear programming [17]. Practical developments of the dual Simplex method have also been studied for large-scale linear programming problems [18].

The study of degeneracy also extends beyond the question of cycling. Degeneracy can be related to strict feasibility and numerical stability in linear programming. In particular, recent work has shown that when strict feasibility fails, every basic feasible solution of a standard-form linear program can be degenerate. The same work also discusses implicit redundancies and preprocessing procedures for such problems [10]. Related research has examined stability under nondegeneracy assumptions in primal-dual approaches to linear programming [19].

Worst-case behaviour of the Simplex Method is another related topic. The classical Klee–Minty construction demonstrates that the Simplex Method can require an exponential number of iterations for a particular family of linear programs and pivot rules [20]. This result is different from cycling: exponential running time and repeated bases are distinct phenomena.

Although the literature contains extensive work on degeneracy, stalling, cycling, pivot selection, and anti-cycling procedures, these topics are often discussed separately. A clear distinction between them is useful, especially when simplex iterations are examined through worked examples. The purpose of this paper is therefore to provide a unified treatment of degeneracy in the Simplex Method and to explain its relationship with degenerate pivots, stalling, and cycling.

The paper focuses on the algebraic and geometric meaning of degeneracy, the way degeneracy appears during simplex iterations, the conditions under which cycling can occur, and the main approaches used to prevent cycling. A carefully verified numerical example is included to illustrate how the basis may change without a corresponding change in the geometric solution or the objective value.

The contribution of the paper is mainly analytical and expository. It brings together classical results on degeneracy and cycling and explains their relationship through simplex calculations. Particular attention is given to the distinction between a degenerate basic feasible solution, a degenerate pivot, stalling, and cycling. The paper does not claim a new anti-cycling algorithm or a new complexity result.

The remainder of the paper is organized as follows. Section 2 reviews the relevant literature on linear programming, degeneracy, cycling, and anti-cycling procedures. Section 3 introduces the basic concepts of linear programming needed for the discussion. Section 4 reviews the Simplex Method and its pivot mechanism. Section 5 discusses degeneracy from algebraic and geometric viewpoints. Section 6 examines degenerate pivots and stalling. Section 7 discusses cycling. Section 8 reviews the main approaches used to handle degeneracy and cycling. Section 9 presents a verified numerical example. Sections 10 and 11 discuss the implications of the results and conclude the paper.

LITERATURE REVIEW

The literature on linear programming and the Simplex Method is extensive. For the present study, the relevant work can be grouped into four closely related areas: the development of the Simplex Method, degeneracy and degenerate pivots, cycling and anti-cycling procedures, and more recent work on the numerical and structural effects of degeneracy.

Foundations of the Simplex Method

The Simplex Method was developed by Dantzig as a systematic procedure for solving linear programming problems through a sequence of basic feasible solutions [1]. The method is based on the algebraic structure of bases and has a natural geometric interpretation in terms of extreme points of the feasible region. Standard treatments of linear programming provide the mathematical foundations for basic solutions, basic feasible solutions, bases, reduced costs, pivot operations, and optimality [2-4].

The basic simplex procedure is effective when each pivot produces a clear change in the current solution and an improvement in the objective value. This simple interpretation becomes less direct when the current basic feasible solution is degenerate. Degeneracy therefore represents an important part of the theoretical study of the Simplex Method.

Degeneracy in Linear Programming

A basic feasible solution is degenerate when at least one of its basic variables is zero. This definition is purely algebraic, but degeneracy also has a geometric interpretation. At a degenerate vertex, more constraints may be active than are required to identify the vertex. Consequently, more than one basis may represent the same geometric point [1,2,12].

The study of degeneracy goes back to the early development of linear programming. Charnes investigated optimality and degeneracy and considered ways of resolving difficulties associated with degenerate solutions [13]. Dantzig, Orden, and Wolfe developed a generalized Simplex Method, while Wolfe later proposed a technique for resolving degeneracy in linear programming [14, 15]. These early contributions established important ideas that later became associated with perturbation and lexicographic approaches.

Degeneracy is important because a pivot can change the basis without changing the represented geometric point. When the leaving basic variable has value zero, the minimumratio test produces a zero ratio. The resulting pivot changes the basis but may leave both the solution point and the objective value unchanged. Such a pivot is commonly referred to as a degenerate pivot [6, 7].

The distinction between degeneracy and cycling is important. Degeneracy concerns the representation of a basic feasible solution, whereas cycling concerns the repeated occurrence of a basis. A degenerate problem can therefore be solved without cycling. Similarly, a degenerate pivot does not by itself imply that the Simplex Method will enter a cycle.

Cycling and Pivot Selection

The possibility of cycling is one of the main theoretical difficulties associated with degenerate simplex iterations. Beale provided a classical example showing that the Simplex Method can cycle under a particular pivot rule [11]. In a cycling sequence, the algorithm returns to a previously visited basis and can repeat the same sequence of pivots. Since the repeated pivots do not improve the objective value, the usual progress argument for the Simplex Method is no longer sufficient to establish termination.

The existence of cycling examples led to considerable interest in pivot selection rules. Bland proposed a finite pivoting rule based on variable indices and proved that the rule prevents cycling [8]. The importance of pivot selection is also demonstrated by the systematic construction of cycling examples studied by Zrnig [5,12]. These studies show that cycling is a property of the interaction between degeneracy and the selected pivot rule rather than a consequence of degeneracy alone.

Anti-Cycling Procedures

Several approaches have been developed to prevent cycling and to deal with ties caused by degeneracy. Bland's rule is one of the clearest theoretical approaches because it imposes a fixed ordering on eligible entering and leaving variables and guarantees finite termination [8].

Lexicographic techniques provide another approach. When the minimum-ratio test produces a tie, an ordered comparison can be used to select a unique leaving variable. Perturbation techniques address the same problem by introducing sufficiently small changes that separate otherwise tied quantities. The development of these ideas can be traced to early work on degeneracy and generalized simplex procedures [13-15].

The theoretical rules are complemented by practical procedures. Gill, Murray, Saunders, and Wright proposed an anti-cycling procedure for linearly constrained optimization, including simplex-type methods. Their approach uses controlled infeasibility together with an adjusted working feasibility tolerance [16]. Gal and Geue studied pivoting techniques for several degeneracy-related problems, including cycling and stalling [7]. Dosios and Paparrizos examined degeneracy in a primal-dual simplex framework and showed the importance of suitable tie-breaking procedures for avoiding cycling [9].

These approaches have different purposes. Bland's rule gives a direct finite-termination guarantee. Lexicographic methods provide a systematic way of resolving ties. Perturbation separates tied quantities through a modified problem representation, while practical anticycling procedures seek to control the numerical behaviour of simplex iterations [8, 16].

Alternative Pivot Methods

The study of degeneracy has also led to alternative pivot frameworks. Fukuda and Terlaky developed the criss-cross method as a different type of pivot algorithm for linear programming [17]. Unlike the standard primal Simplex Method, criss-cross methods do not require the same separation between maintaining primal feasibility and improving the objective at every stage. This makes them relevant to the broader study of pivot algorithms and finite pivoting procedures.

The dual Simplex method is another important framework in practical linear programming. Koberstein and Suhl studied practical dual-phase procedures for large-scale linear programming problems and discussed computational aspects of the dual Simplex method

[18]. This work is relevant to the present discussion because degeneracy and pivot selection are also important when simplex procedures are applied to large-scale problems.

Degeneracy, Stability, and Modern Analysis

The consequences of degeneracy extend beyond cycling. Numerical stability and structural properties of the feasible region can also be affected by degeneracy and nondegeneracy assumptions. Gonzalez-Lima, Wei, and Wolkowicz studied a stable primal-dual approach to linear programming under nondegeneracy assumptions [19].

More recently, Im and Wolkowicz revisited the relationship between degeneracy, strict feasibility, and stability in linear programming [10]. Their analysis shows that failure of strict feasibility can be associated with degenerate basic feasible solutions and also discusses the role of implicit redundancy and preprocessing. These results provide a modern perspective on degeneracy that goes beyond the classical question of cycling.

Worst-Case Behaviour of the Simplex Method

Cycling should also be distinguished from the general worst-case complexity of the Simplex Method. The classical Klee–Minty construction demonstrated that, for a particular family of linear programs and pivot rule, the Simplex Method can require an exponential number of iterations [20]. This result concerns the number of iterations required by the algorithm and is therefore different from cycling, in which a basis is revisited.

Position of the Present Study

The existing literature provides strong theoretical results on degeneracy, cycling, pivot selection, and anti-cycling procedures. It also contains important work on the numerical and structural effects of degeneracy. However, these topics are often presented separately.

The present paper brings these ideas together in a single treatment. The main focus is the connection between a degenerate basic feasible solution, degenerate pivots, stalling, and cycling. The paper also compares the main anti-cycling approaches from an algorithmic and conceptual viewpoint and uses a verified numerical example to show how a basis can change without a corresponding change in the geometric solution or the objective value.

The aim is not to introduce a new anti-cycling algorithm or to claim a new complexity result. Instead, the paper provides a clear mathematical account of how degeneracy enters the Simplex Method, why it can lead to zero-improvement iterations, how cycling can arise, and how established pivot-selection procedures prevent repeated bases.

Basics of Linear Programming

Linear programming provides the mathematical framework needed for the study of the Simplex Method and degeneracy. In this section, the basic concepts used throughout the paper are introduced.

Linear Programming Problem

A linear programming problem can be written in the standard form

$$\min_{x} \quad c^T x \tag{1}$$

subject to

$$Ax = b, \qquad x \geq 0, \tag{2}$$

where

$$A \in \mathbb{R}^{m \times n}, \qquad x \in \mathbb{R}^n, \qquad b \in \mathbb{R}^m, \qquad c \in \mathbb{R}^n.$$

Here, x is the vector of decision variables, c is the objective coefficient vector, A is the constraint matrix, and b is the right-hand-side vector [1, 4, 3, 2].

A linear programming problem may initially be given in a different form, such as a maximization problem or a system containing inequality constraints. These forms can be converted into standard form by changing the sign of the objective when required and intro-ducing slack, surplus, or artificial variables as appropriate [4, 3].

Feasible Solution

A vector x is called a feasible solution if it satisfies all the constraints of the linear program-ming problem. For the standard form above, this means

$$Ax = b, \qquad x \geq 0. \tag{3}$$

The set of all feasible solutions is called the feasible region. Since both the equality constraints and the non-negativity restrictions are linear, the feasible region is a polyhedron [2].

If a feasible solution also gives the smallest value of the objective function in a minimiza-tion problem, it is called an optimal solution. The corresponding objective value is called the optimal objective value.

Basis and Basic Solution

Assume that the constraint matrix A has rank m. A basis is a set of m linearly independent columns of A. Let B denote the resulting m × m basis matrix. The variables associated with the columns of B are called basic variables, while the remaining variables are called non-basic variables [3, 2].

For a given basis B, the non-basic variables are set equal to zero. The basic variables are then obtained from

$$Bx_B = b, \tag{4}$$

and hence

$$x_B = B^{-1}b. \tag{5}$$

The resulting vector is called a basic solution. If

$$\text{B}-1\text{b} \geq 0, \tag{6}$$

then the basic solution is feasible and is therefore called a basic feasible solution (BFS) [1, 3].

Basic Feasible Solutions and Extreme Points

There is an important relationship between basic feasible solutions and the geometry of the feasible region. Under the usual rank assumptions, a basic feasible solution corresponds to an extreme point of the feasible polyhedron. Conversely, an extreme point can be represented by a basic feasible solution [2, 3].

This relationship provides the geometric basis of the Simplex Method. Rather than searching through every feasible point, the method moves between basic feasible solutions associated with extreme points of the feasible region.

For a linear program with a finite optimum, an optimal solution can be found at an extreme point whenever the feasible region is nonempty and the objective function is bounded below in the minimization case [1, 2].

Degenerate Basic Feasible Solution

A basic feasible solution is called degenerate if at least one of its basic variables is equal to zero. Thus, for a basis B, the BFS is degenerate if

$$(B^{-1}b)_i = 0 \tag{7}$$

for at least one basic variable x_{B_i} [1, 2].

For example, suppose that a basis contains three basic variables and the corresponding basic solution is

$$x_B = \begin{pmatrix} 2 \\ 0 \end{pmatrix}. \tag{8}$$

Although the solution is feasible, one of the basic variables is zero. Therefore, the basic feasible solution is degenerate.

Degeneracy has an important geometric interpretation. At a degenerate vertex, more constraints may be active than are required to identify the vertex. Consequently, more than one basis can represent the same geometric point [2, 12].

This distinction between a point and its basis representation is important for the later discussion. The Simplex Method operates through changes of basis, whereas the geometricinterpretation concerns movement between vertices of the feasible region.

Nondegenerate Basic Feasible Solution

A basic feasible solution is nondegenerate when all of its basic variables are strictly positive:

$$x_{B_i} > 0, \qquad i = 1, \ldots, m. \tag{9}$$

In a nondegenerate iteration, the minimum-ratio test normally gives a positive step length. The resulting pivot therefore moves the current solution to a different basic fea-sible solution. Degeneracy changes this behaviour because the minimum ratio can become zero.

Objective Function and Reduced Costs

For a basis B, let c_B denote the objective coefficients associated with the basic variables. The basic solution has objective value

$$z = c_B^T B^{-1} b. \tag{10}$$

For a non-basic variable x_j with column a_j and objective coefficient c_j, the reduced cost is

$$\bar{c}_j = c_j - c_B^T B^{-1} a_j. \tag{11}$$

For the minimization convention used in this paper, a negative reduced cost indicates a direction in which the objective function can be improved, provided that the corresponding variable can enter the basis while maintaining feasibility [3, 2].

The reduced costs therefore provide the algebraic information used to select an entering variable during a simplex iteration.

Pivot Operation

Suppose that x_j is selected as the entering variable. Let

$$d = B^{-1} a_j. \tag{12}$$

To maintain feasibility, only components satisfying $d_i > 0$ are considered in the minimum-ratio test. The allowable step is determined by

$$\theta^* = \min_{i:\, d_i > 0} \frac{(x_B)_i}{d_i}. \tag{13}$$

The corresponding basic variable leaves the basis. The basis is then updated by replacing the leaving column with the entering column [1, 3].

If

$$\theta^* > 0, \tag{14}$$

the new basic solution normally represents movement to a different point of the feasible region. If

$$\theta^* = 0, \tag{15}$$

the pivot is degenerate. In that case, the basis can change while the geometric solution and the objective value remain unchanged. This case is central to the discussion of degeneracy, stalling, and cycling in the following sections.

The Simplex Method

The Simplex Method solves a linear programming problem by moving from one basic feasible solution to another. Each movement is obtained by changing the current basis through a pivot operation. The method uses the reduced costs to identify a possible improving direction and the minimum-ratio test to maintain feasibility [1, 3, 2].

Initial Basic Feasible Solution

Consider the standard-form linear programming problem

$$\min \; c^T x \tag{16}$$

subject to

$$Ax = b, \qquad x \geq 0. \tag{17}$$

Let B be the current basis and let N contain the non-basic columns of A. The variables can then be partitioned as

$$x = \begin{pmatrix} x_B \\ \mathrm{x_N} \end{pmatrix}, \qquad A = [B \; N]. \tag{18}$$

The basic variables are obtained from

$$Bx_B = b, \tag{19}$$

so that

$$x_B = B^{-1}b. \tag{20}$$

The non-basic variables are initially set to zero. If

$$B^{-1}b \geq 0, \tag{21}$$

the resulting solution is a basic feasible solution [3, 2].

When slack variables are available, an initial basis can often be obtained directly from the identity columns associated with the slack variables. For problems in which such a basis is not immediately available, an auxiliary procedure, such as Phase I of the two-phase Simplex Method, can be used to find a feasible starting basis [4, 3].

Reduced Costs

Let c_B denote the objective coefficients corresponding to the basic variables. The objective value associated with the current basic solution is

$$z = c_B^T x_B = c_B^T B^{-1} b. \tag{22}$$

For a non-basic variable x_j, let a_j denote its column in A and let c_j be its objective coefficient. The reduced cost of x_j is defined by

$$\bar{c}_j = c_j - c_B^T B^{-1} a_j. \tag{23}$$

For the minimization convention used in this paper, if a non-basic variable has

$$\bar{c}_j < 0, \tag{24}$$

then increasing that variable from zero provides an improving direction, provided that fea-sibility can be maintained. Thus, a negative reduced cost can be used to select an entering variable [?, 2].

If all reduced costs satisfy

$$\bar{c}_j \geq 0 \qquad \text{for all non-basic variables,} \tag{25}$$

the current basic feasible solution is optimal for the minimization problem, provided that the problem has a finite optimum [3, 2].

Choosing the Entering Variable

Suppose that a non-basic variable x_j has been selected to enter the basis. Let

$$d = B^{-1}a_j. \tag{26}$$

If increasing x_j causes a decrease in one or more basic variables, the current solution can remain feasible only up to a certain step length. Therefore, the components of d that satisfy

$$d_i > 0 \tag{27}$$

are considered in the minimum-ratio test.

The entering variable is selected according to the chosen pivot rule. If more than one variable has an admissible improving reduced cost, different pivot rules may select different entering variables. This choice becomes particularly important when degeneracy is present because the resulting sequence of bases can depend on the pivot rule [8, 17].

Minimum-Ratio Test

Let the current basic solution be $x_B = B^{-1}b$. After increasing the entering variable x_j by an amount θ, the basic variables become

$$x_B(\theta) = x_B - \theta B^{-1}a_j. \tag{28}$$

For feasibility, we require

$$x_B - \theta B^{-1}a_j \geq 0. \tag{29}$$

Therefore, for every component satisfying

$$d_i = (B^{-1}a_j)_i > 0, \tag{30}$$

we must have

$$\theta \leq \frac{(x_B)_i}{d_i}. \tag{31}$$

The maximum feasible step is consequently

$$\theta^* = \min_{i:\, d_i>0} \frac{(x_B)_i}{d_i}. \tag{32}$$

The basic variable corresponding to the minimum ratio leaves the basis [1, 3, 2].

Degenerate Pivot

The minimum-ratio test is particularly important when the current basic feasible solution is degenerate. Suppose that

$$(x_B)_i = 0 \tag{33}$$

for one of the basic variables and that

$$d_i > 0. \tag{34}$$

The corresponding ratio is then

$$\frac{(x_B)_i}{d_i} = 0. \tag{35}$$

Hence,

$$\theta^* = 0. \tag{36}$$

The entering variable therefore increases by zero, and the current geometric point does not change. Nevertheless, the basis changes because one basic variable leaves and the entering variable becomes basic.

Such an iteration is called a degenerate pivot. It is one of the key ways in which degeneracy affects the Simplex Method [6, 7].

This observation gives an important distinction between a simplex iteration and geometric movement. A pivot does not necessarily imply movement from one vertex to another. Under degeneracy, the basis may change while the represented vertex remains the same.

Objective-Function Change

For a selected entering variable x_j, the change in the objective function can be expressed as

$$\Delta z = \bar{c}_j \theta. \tag{37}$$

When

$$\theta > 0 \tag{38}$$

and

$$\bar{c}_j < 0, \tag{39}$$

the objective value decreases in the minimization problem.

For a degenerate pivot,

$$\theta = 0. \tag{40}$$

Therefore,

$$\Delta z = 0. \tag{41}$$

Thus, a degenerate pivot can change the basis without changing the objective value. A sequence of such zero-improvement iterations is relevant to the phenomenon of stalling and, under an unsuitable pivot rule, can contribute to cycling [6, 11].

Tie in the Minimum-Ratio Test

Degeneracy can also produce a tie in the minimum-ratio test. Suppose that two or more basic variables give the same minimum ratio:

$$\frac{(x_B)_i}{d_i} = \frac{(x_B)_k}{d_k} = \theta^*. \tag{42}$$

Any of the tied basic variables may be selected as the leaving variable unless a specific tie-breaking rule is imposed. The choice can affect the subsequent sequence of bases.

This is one reason why pivot-selection rules are important in the study of degeneracy and cycling. Bland's rule, for example, resolves such choices using the indices of the eligible variables and guarantees that the Simplex Method does not cycle [8].

Termination of the Simplex Method

The Simplex Method can terminate in several ways. If the current basic feasible solution has no improving reduced cost, it satisfies the simplex optimality condition. If the minimum-ratio test has no eligible row for an improving entering variable, the objective can be unbounded in the corresponding direction.

Degeneracy introduces another possibility in the behaviour of the iterations. Several consecutive pivots may occur without improving the objective value. If a previous basis is eventually reached, the algorithm has entered a cycle. Thus, the absence of objective improvement should not by itself be interpreted as cycling; cycling requires the repetition of a basis [11, 8].

The distinction developed here will be used in the following sections to study stalling and cycling in more detail.

Degeneracy in Linear Programming

Degeneracy is one of the features of linear programming that can make the behaviour of the Simplex Method less direct than the usual geometric picture suggests. The key issue is that a single feasible point can be associated with more than one basis. As a result, the Simplex Method may change its basis without moving to a new point of the feasible region [1, 2, 12].

Algebraic View of Degeneracy

Let B be a basis and let the associated basic solution be

$$x_B = B^{-1}b. \tag{43}$$

The basic solution is degenerate if

$$(x_B)_i = 0 \tag{44}$$

for at least one basic variable x_{B_i}.

For a nondegenerate basic feasible solution,

$$(x_B)_i > 0, \qquad i = 1, \ldots, m. \tag{45}$$

Thus, the difference between a degenerate and a nondegenerate basic feasible solution is determined by the values of the basic variables. The non-basic variables are zero in both cases. The important difference is that, in the degenerate case, at least one of the basic variables is also zero [3, 2].

Consider a basis with three basic variables and suppose that

$$x_B = \begin{pmatrix} 3 \\ 0 \\ 5 \end{pmatrix}. \tag{46}$$

This solution is feasible because all components are non-negative. However, the second basic variable has value zero. Therefore, the solution is degenerate.

The zero value does not mean that the corresponding variable is non-basic. This point is important. A variable can be basic and still have value zero. The distinction is determined by the basis, not only by the numerical value of the variable.

Geometric Interpretation

The algebraic definition of degeneracy has a natural geometric interpretation. A feasible point is an extreme point when it cannot be written as a proper convex combination of two distinct feasible points. For a linear program satisfying the usual rank assumptions, basic feasible solutions correspond to extreme points of the feasible region [3, 2].

At a nondegenerate vertex, the active constraints have the expected number of indepen-dent equations needed to determine the vertex. At a degenerate vertex, additional constraints can be active at the same point. Therefore, the same geometric vertex can correspond to different bases [2, 12].

This explains why a change of basis does not always mean that the Simplex Method has moved to a different geometric point. Several different bases may describe the same vertex.

The distinction can be expressed as

$$\text{basis} \neq \text{geometric point}. \tag{47}$$

A single geometric point may have several basis representations when degeneracy is present.

Degeneracy and the Minimum-Ratio Test

The effect of degeneracy becomes particularly clear during a simplex pivot. Suppose that x_j is selected as the entering variable and define

$$d = B^{-1}a_j. \tag{48}$$

The basic variables after increasing x_j by θ are

$$x_B(\theta) = x_B - \theta d. \tag{49}$$

For every component satisfying $d_i > 0$, feasibility requires

$$\theta \leq \frac{(x_B)_i}{d_i}. \tag{50}$$

Hence the minimum-ratio test gives

$$\theta^* = \min_{i:\, d_i>0} \frac{(x_B)_i}{d_i}. \tag{51}$$

Now suppose that the current solution is degenerate and that

$$(x_B)_i = 0 \tag{52}$$

for some i with

$$d_i > 0. \tag{53}$$

Then

$$\frac{(x_B)_i}{d_i} = 0, \tag{54}$$

and therefore

$$\theta^* = 0. \tag{55}$$

The entering variable consequently remains equal to zero after the pivot. At the same time, the basis changes. This is the algebraic mechanism behind a degenerate pivot [6, 7].

Degenerate Pivot and Objective Value

Let $\bar{c}_j$ denote the reduced cost of the entering variable. The change in the objective value is

$$\Delta z = \bar{c}_j \theta^*. \tag{56}$$

For a degenerate pivot,

$$\theta^* = 0. \tag{57}$$

Consequently,

$$\Delta z = 0. \tag{58}$$

Thus, even when the selected entering variable has an improving reduced cost, the ob-jective value does not change when the pivot is degenerate.

This observation is important for understanding simplex stalling. The algorithm may perform a valid pivot while showing no improvement in the objective value. The basis has changed, but the represented point has not moved.

Multiple Bases at a Degenerate Vertex

Suppose that two different bases, B_1 and B_2, produce the same feasible point:

$$B_1^{-1}b = x \qquad \text{and} \qquad B_2^{-1}b = x. \tag{59}$$

If the point is degenerate, such multiple basis representations can arise because several constraints are active at the same vertex [2, 12].

This feature is central to the cycling problem. Since the Simplex Method operates on bases, rather than only on geometric points, it is possible for the algorithm to move between different bases representing the same point. If the pivot rule eventually returns to a previously visited basis, a cycle can occur.

Therefore, the sequence

$$B_1 \to B_2 \to B_3 \to \cdots \tag{60}$$

may represent little or no geometric movement when all the bases correspond to the same degenerate vertex.

Degeneracy Does Not Imply Cycling

It is important to distinguish the existence of degeneracy from the occurrence of cycling.

A degenerate basic feasible solution only tells us that at least one basic variable is zero. It does not tell us how the subsequent pivot choices will behave. The Simplex Method may leave the degenerate vertex and continue to an improving solution. It may also perform several degenerate pivots before making progress.

Cycling requires a stronger condition: a basis that has already appeared is visited again. Beale's classical example demonstrates that such behaviour can occur under a suitable com-bination of degeneracy and pivot selection [11]. Bland's rule later provided a pivot-selection rule that guarantees that such repetition does not occur [8].

Hence,

$$\text{degeneracy} \not\Rightarrow \text{cycling}. \tag{61}$$

This distinction is important throughout the remainder of the paper.

Degeneracy and Stalling

A sequence of simplex iterations may contain several pivots for which

$$\Delta z = 0. \tag{62}$$

When the objective value remains unchanged over a sequence of iterations, the behaviour is commonly described as stalling. Degeneracy is one source of such zero-improvement pivots [6, 7].

Stalling does not necessarily mean that the algorithm is cycling. The algorithm can perform several zero-improvement pivots and then make an improving pivot. Cycling occurs only when a previously visited basis is reached again and the subsequent pivot sequence repeats.

The distinction can therefore be summarized as

$$\begin{aligned} &\text{Degeneracy:} && \text{a basic variable is zero,} \\ &\text{Degenerate pivot:} && \theta^* = 0, \end{aligned} \tag{63}$$

(63)

Stalling: $\Delta z = 0$ over one or more iterations,
Cycling: a previous basis is revisited.

These four concepts are closely related, but they describe different properties of the simplex process.

Structural Sources of Degeneracy

Degeneracy can arise from the structure of the constraints. Redundant constraints, multiple active constraints at a feasible point, and particular relationships among the right-hand-side values can all contribute to degenerate basic feasible solutions.

Transportation models provide a useful example. In a balanced m × n transportation problem, the total supply equals the total demand. Consequently, one of the standard supply-demand equations is linearly dependent on the others, leaving m + n − 1 independent constraints. This structural dependence determines the number of basic positions in a non-redundant basis. However, balance itself does not imply that a basic allocation must be zero. A zero basic allocation depends on the particular data and basis.

Degeneracy can also be connected with the failure of strict feasibility. Recent work has shown that, for certain standard-form linear programs, the absence of a strictly feasible point can force every basic feasible solution to be degenerate. This provides a broader view of degeneracy that includes structural and numerical issues beyond simplex cycling [10].

Implications for the Simplex Method

The main computational implication of degeneracy is that the number of simplex pivots need not correspond directly to the amount of geometric progress. A pivot may change the basis without changing the solution point or the objective value.

This observation explains why iteration counts alone do not always describe the behaviour of the Simplex Method. When degeneracy is present, it is also useful to examine the basis sequence, the minimum ratios, and the objective change at each iteration.

The next section uses these observations to examine degenerate pivots and stalling in greater detail.

Degenerate Pivots and Stalling

Degeneracy becomes especially important during a simplex iteration. When the current basic feasible solution contains a zero basic variable, the minimum-ratio test can produce a zero step. The resulting pivot changes the basis but does not move the current solution to a different point. This behaviour is the main connection between degeneracy and stalling [6, 7].

Degenerate Pivot

Consider a current basis B and let the corresponding basic feasible solution be

$$x_B = B^{-1}b. \tag{64}$$

Suppose that x_j is selected as the entering variable. Define

$$d = B^{-1}a_j, \tag{65}$$

where a_j is the column of A associated with x_j.

After increasing x_j by an amount θ, the basic variables become

$$x_B(\theta) = x_B - \theta d. \tag{66}$$

Feasibility requires

$$x_B - \theta d \geq 0. \tag{67}$$

Therefore, the maximum feasible step is

$$\theta^* = \min_{i:\, d_i > 0} \frac{(x_B)_i}{d_i}. \tag{68}$$

Suppose that one of the basic variables satisfies

$$(x_B)_i = 0 \tag{69}$$

and

$$d_i > 0. \tag{70}$$

The corresponding ratio is then

$$\frac{(x_B)_i}{d_i} = 0. \tag{71}$$

Consequently,

$$\theta^* = 0. \tag{72}$$

The entering variable therefore remains equal to zero after the pivot. Nevertheless, the basis changes because the corresponding zero basic variable leaves the basis and the entering variable becomes basic.

This is a degenerate pivot. The important point is that a pivot has taken place even though the solution has not moved to a new point [6, 7].

Effect on the Objective Function

Let $\bar{c}_j$ be the reduced cost of the entering variable. For the minimization convention used in this paper, the change in the objective function is

$$\Delta z = \bar{c}_j \theta. \tag{73}$$

For a degenerate pivot,

$$\theta = 0. \tag{74}$$

Hence,

$$\Delta z = 0. \tag{75}$$

Thus, an improving reduced cost does not necessarily produce an immediate improvement in the objective value. If the minimum-ratio test gives a zero step, the basis changes while the objective value remains unchanged.

This is one of the reasons why the number of simplex iterations should be interpreted with care when degeneracy is present.

Stalling

Stalling refers to simplex iterations in which the objective value does not improve. Degen-erate pivots are a direct source of such zero-improvement iterations [6, 7].Suppose that the objective values generated during several consecutive iterations are

$$z^{(t)} = z^{(t+1)} = z^{(t+2)} = \cdots . \tag{76}$$

The algorithm is making basis changes without reducing the objective value. This be-haviour can occur because the corresponding pivots are degenerate.

Stalling should not be interpreted as a failure of the Simplex Method. A sequence of zero-improvement pivots may eventually be followed by a nondegenerate pivot with

$$\theta > 0, \tag{77}$$

after which the objective value improves.

Therefore,

$$\text{stalling} \not\Rightarrow \text{cycling}. \tag{78}$$

A stalled sequence becomes a cycling sequence only when the algorithm returns to a previously visited basis and the same pivot sequence can repeat.

Degenerate Pivot Versus Stalling

The terms degenerate pivot and stalling describe related but different events.

A degenerate pivot is a property of an individual iteration:

$$\theta = 0. \tag{79}$$

Stalling describes the behaviour of the objective value over one or more iterations:

$$\Delta z = 0. \tag{80}$$

For a degenerate pivot,

$$\theta = 0 \quad \Longrightarrow \quad \Delta z = 0. \tag{81}$$

However, a zero objective change should not automatically be used as the definition of a degenerate pivot. For example, an iteration may have a positive step but zero objective change when the selected reduced cost is zero. Such a case is different from a zero-step degenerate pivot.

This distinction is important when analysing simplex iterations numerically.

Stalling and Basis Changes

Although the objective value may remain constant during a degenerate iteration, the basis can change:

$$B^{(t)} \neq B^{(t+1)}. \tag{82}$$

At the same time, the represented feasible point may remain unchanged:

$$x^{(t)} = x^{(t+1)}. \tag{83}$$

Thus, it is possible to have

$$B^{(t)} \neq B^{(t+1)}, \qquad x^{(t)} = x^{(t+1)}, \qquad z^{(t)} = z^{(t+1)}. \tag{84}$$

This relation gives a compact algebraic description of the behaviour that makes degen-eracy difficult to recognize from objective values alone.

A record of the basis at every iteration is therefore useful when studying possible cy-cling. Two consecutive iterations with the same objective value are not sufficient evidence of cycling. The basis sequence must also be examined.

Zero Ratio and Ties

A degenerate pivot can occur when the minimum ratio is zero. Degeneracy can also create ties in the minimum-ratio test. Suppose that two basic variables give the same ratio:

$$\frac{(x_B)_i}{d_i} = \frac{(x_B)_k}{d_k} = \theta^*. \tag{85}$$

The pivot rule must determine which of the tied variables leaves the basis. Different choices can produce different subsequent basis sequences.

Tie-breaking is therefore important in the presence of degeneracy. The choice of leav-ing variable can affect whether the algorithm makes progress towards a different vertex or continues through additional degenerate bases. Classical anti-cycling rules were developed partly to control this type of behaviour [8, 13].

Illustrative Basis Sequence

To make the distinction clear, consider a sequence of bases

$$B^{(0)} \to B^{(1)} \to B^{(2)} \to B^{(3)}. \tag{86}$$

Suppose that

$$x^{(0)} = x^{(1)} = x^{(2)}, \tag{87}$$

and

$$z^{(0)} = z^{(1)} = z^{(2)}. \tag{88}$$

The basis has changed, but the solution point and objective value have not. These iterations are consistent with degenerate simplex behaviour.

If the algorithm subsequently reaches

$$B^{(0)} \tag{89}$$

again, then a previously visited basis has been repeated. At that point, the behaviour is no longer merely stalling; it is evidence of cycling if the pivot sequence repeats.

The distinction can therefore be written as

$$\begin{aligned} &B^{(t)} \neq B^{(t+1)}, \\ &x^{(t)} = x^{(t+1)}, \quad \Longrightarrow \quad \text{zero-improvement basis change}, \\ &z^{(t)} = z^{(t+1)} \end{aligned} \tag{90}$$

whereas

$$B^{(t+r)} = B^{(t)} \tag{91}$$

for some $r > 0$ indicates that a previous basis has been revisited.

The latter condition is the essential feature of cycling.

Practical Importance

The presence of degenerate pivots means that an iteration count alone does not always show how much progress has been made by the Simplex Method. A sequence can contain several basis changes while the objective value and the geometric solution remain unchanged.

For this reason, a careful analysis of degeneracy should record at least the basis, entering variable, leaving variable, minimum ratio, step length, and objective value at each iteration.

Such information makes it possible to distinguish a normal improving pivot from a de-generate pivot and to distinguish stalling from actual cycling. This distinction will be used in the numerical example presented later in the paper.

The Cycling Problem

Cycling is one of the most important theoretical difficulties associated with degeneracy in the Simplex Method. It occurs when a sequence of simplex pivots returns to a basis that has already been visited. Once the same basis is reached under the same pivot-selection rule, the subsequent sequence of pivots can repeat. The objective value then remains unchanged throughout the cycle [11, 8].

Definition of Cycling

Let

$$B^{(0)}, B^{(1)}, B^{(2)}, \ldots$$

denote the sequence of bases generated by the Simplex Method.
Cycling occurs if there exist integers t ≥ 0 and r > 0 such that

$$B^{(t+r)} = B^{(t)} \tag{92}$$

and the same pivot rule generates the same subsequent sequence of bases.

Thus, the essential feature of cycling is the repetition of a basis. A constant objective value alone is not sufficient to establish cycling.

For example, consider the basis sequence

$$B^{(0)} \to B^{(1)} \to B^{(2)} \to B^{(3)} \to B^{(1)} \to \cdots . \tag{93}$$

If the same pivot rule is applied again at $B^{(1)}$, the algorithm can repeat

$$B^{(1)} \to B^{(2)} \to B^{(3)} \to B^{(1)} \to \cdots . \tag{94}$$

The algorithm therefore continues indefinitely without producing a new basis.

Relationship Between Degeneracy and Cycling

Cycling is closely related to degeneracy. At a nondegenerate basic feasible solution, an improving pivot normally produces a positive step and changes the represented solution point. Under degeneracy, however, a pivot can have

$$\theta = 0. \tag{95}$$

The basis may then change without changing the current feasible point or objective value. A sequence of such basis changes can occur at the same degenerate vertex.

Therefore, degeneracy creates the possibility of basis changes without geometric progress. Under an unsuitable pivot-selection rule, such changes can eventually lead back to a previ-ously visited basis [11, 6].

It is important, however, that degeneracy alone does not imply cycling. A degenerate problem can contain several zero-step pivots and still terminate after a finite number of iterations.

Hence,

$$\text{degeneracy} \not\Rightarrow \text{cycling}. \tag{96}$$

Cycling requires the additional condition that a previous basis is revisited.

Classical Cycling Example

Beale presented a classical example showing that the Simplex Method can cycle under a particular pivot-selection procedure [11]. The example is important because it demonstrates that the usual intuition that the Simplex Method continually makes progress toward an optimum is not sufficient to guarantee termination when degeneracy is present.

The example contains a sequence of degenerate pivots for which the objective value does not improve. After a finite number of pivots, the algorithm returns to a basis that has already occurred. The same sequence of pivots can then be repeated.

The example does not show that the Simplex Method generally cycles. Rather, it demon-strates that cycling is theoretically possible when the pivot rule does not provide an appro-priate mechanism for resolving degeneracy.

Cycling and Objective Values

Suppose that the objective values along a sequence of iterations satisfy

$$z^{(0)} = z^{(1)} = z^{(2)} = \cdots . \tag{97}$$

This sequence shows that there is no objective improvement, but it does not by itself establish cycling.

To determine whether cycling has occurred, the basis sequence must also be examined. For example,

$$z^{(1)} = z^{(2)} \tag{98}$$

does not imply that

$$B^{(1)} = B^{(2)}. \tag{99}$$

A degenerate pivot can satisfy

$$B^{(1)} \neq B^{(2)}, \qquad x^{(1)} = x^{(2)}, \qquad z^{(1)} = z^{(2)}. \tag{100}$$

Cycling, in contrast, requires a repeated basis, such as

$$B^{(t+r)} = B^{(t)}. \tag{101}$$

This distinction is important when a simplex implementation is used to detect cycling.

Basis Repetition as a Cycling Test

A direct way to identify cycling is to keep a record of the bases visited during the simplex iterations. If a newly generated basis is already contained in the record, then a previous basis has been revisited.

Let

$$\mathcal{B}_t = \left\{B^{(0)}, B^{(1)}, \ldots, B^{(t)}\right\} \tag{102}$$

denote the set of bases visited up to iteration t.

If

$$B^{(t+1)} \in \mathcal{B}_t, \tag{103}$$

then the new basis has already appeared.

For an exact theoretical analysis, basis identity can be checked directly from the indices of the basic variables. In numerical implementations, care must be taken to distinguish the identity of a basis from approximate equality of floating-point solution vectors.

Cycling and Stalling

Stalling and cycling are related but different.

Stalling refers to a lack of objective improvement over a sequence of iterations. Cycling refers to the repetition of a previous basis.

Consequently,

$$\text{cycling} \Rightarrow \text{repeated zero-improvement behaviour}, \tag{104}$$

under the standard degenerate cycling setting, but

$$\text{stalling} \not\Rightarrow \text{cycling}. \tag{105}$$

A simplex algorithm may stall for several iterations and then make an improving pivot. It becomes a cycling process only if a previously visited basis is reached and the pivot sequence repeats.

This distinction prevents a common misunderstanding in the analysis of simplex com-putations. A constant objective value should be treated as a signal to inspect the basis sequence, not as proof that cycling has occurred.

Consequences of Cycling

If cycling is not prevented, the Simplex Method may continue indefinitely without reaching a new basis. Since the same sequence of pivots can be repeated, the algorithm cannot reach a new optimal basis through that sequence.

This creates a theoretical termination problem. The existence of a finite optimal solution does not by itself prevent an unsuitable simplex pivot rule from entering a cycle.

The issue is therefore not the existence of an optimum, but the behaviour of the selected pivot rule in the presence of degeneracy [11, 8].

Pivot Rules and Cycling

The choice of pivot rule plays an important role in determining whether cycling can occur. If several variables are eligible to enter or leave the basis, different tie-breaking decisions can produce different sequences of bases.

A pivot rule that does not impose a suitable tie-breaking mechanism may allow a se-quence of degenerate pivots to return to a previously visited basis. Anti-cycling rules impose additional structure on these choices.

Blandôs rule is a classical example. By selecting the eligible variable with the smallest index for entering and, in the case of a tie, the eligible variable with the smallest index for leaving, Bland proved that cycling cannot occur [8].

Lexicographic pivoting and perturbation provide other approaches to the same general problem. These methods resolve ties in a systematic way so that the algorithm does not repeatedly follow the same degenerate sequence [13, 14, 15].

Summary of the Cycling Mechanism

The mechanism can be summarized as follows:

$$\text{Degeneracy} \rightarrow \text{zero-step pivots} \rightarrow \text{zero objective improvement} \rightarrow \text{possible repeated bases} \rightarrow \text{cycling} \tag{106}$$

The arrows in this expression should not be interpreted as logical implications in every case. Degeneracy can occur without a zero-step pivot, zero-improvement iterations can occur without cycling, and a degenerate problem can terminate normally. The expression describes the pathway through which cycling can arise under an unsuitable pivot rule.

The next section examines the main procedures used to prevent this behaviour, with particular attention to Bland's rule, lexicographic pivoting, perturbation, and practical anti-cycling procedures.

Methods to Handle Degeneracy and Cycling

The possibility of cycling has led to the development of several pivot selection and tie-breaking procedures. Their main purpose is to control the choice of entering and leaving variables when degeneracy creates more than one admissible choice. The main approaches considered here are Bland's rule, lexicographic pivoting, perturbation, and practical anti-cycling procedures [[8, 13, 14, 16]].

Bland's Rule

Bland's rule is one of the classical anti-cycling rules for the Simplex Method. It resolves ties by using the indices of the variables rather than making an arbitrary choice.

For a minimization problem, suppose that several non-basic variables have negative re-duced costs and are therefore eligible to enter the basis. Bland's rule selects the eligible variable with the smallest index.

If several basic variables attain the minimum ratio and are therefore eligible to leave the basis, the variable with the smallest index is chosen.

Thus, if

$$J = \{j : \bar{c}_j < 0\} \tag{107}$$

is the set of eligible entering variables, Bland's rule selects

$$j^* = \min J. \tag{108}$$

Similarly, if

$$I = \left\{ i : \frac{(x_B)_i}{d_i} = \theta^* \right\} \tag{109}$$

is the set of basic variables tied in the minimum-ratio test, the leaving variable is selected as

$$i^* = \min I. \tag{110}$$

Bland proved that this rule prevents cycling and guarantees finite termination of the Simplex Method under the usual assumptions [8].

The main advantage of Bland's rule is its simplicity. It does not require the objective function or the constraint matrix to be modified. It only changes how eligible variables are selected when a choice exists.

Lexicographic Pivoting

Lexicographic pivoting provides another systematic way of resolving ties created by degen-eracy. Instead of selecting a tied variable arbitrarily, the candidate rows are compared using an ordered sequence of coefficients.

The idea can be viewed as introducing an ordering between otherwise indistinguishable ratios. Suppose that two rows give the same minimum ratio:

$$\frac{(x_B)_i}{d_i} = \frac{(x_B)_k}{d_k}. \tag{111}$$

The primary ratio cannot distinguish the two candidates. Additional coefficients are then examined in a prescribed order until a unique choice is obtained.

The lexicographic approach is closely related to perturbation. A symbolic perturbation can be introduced so that tied quantities are separated without changing the essential struc-ture of the original problem [13, 14].

Lexicographic rules are useful because the tie-breaking procedure is deterministic. The same problem, starting from the same basis, produces the same sequence of choices under the same rule.

Perturbation Methods

Perturbation provides a different way to deal with degeneracy. The basic idea is to replace the original data by slightly perturbed values so that zero or tied quantities become distinct.

For example, a right-hand-side vector

$$b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \tag{112}$$

can conceptually be replaced by

$$b(\varepsilon) = \begin{pmatrix} b_1 + \varepsilon_1 \\ b_2 + \varepsilon_2 \\ \vdots \\ b_m + \varepsilon_m \end{pmatrix}, \tag{113}$$

where the perturbations are chosen sufficiently small and in a prescribed order.

The purpose is not to change the practical problem significantly, but to remove exact ties in the pivot calculations. Symbolic perturbation can achieve the same effect without requiring a numerical value to be assigned to an extremely small parameter [13, 14].

Perturbation techniques are therefore closely connected with lexicographic methods. Both impose an ordering on choices that would otherwise be tied.

Generalized Simplex Approaches

Early work on degeneracy also considered modifications of the simplex procedure itself. Charnes studied degeneracy and proposed methods for resolving the difficulties associated with degenerate solutions [13]. Dantzig, Orden, and Wolfe developed a generalized Simplex Method in which perturbation ideas could be incorporated into the solution process [14].

Wolfe later proposed a method specifically concerned with degeneracy in linear program-ming [15]. These contributions are important because they show that the treatment of degeneracy has been part of simplex research since the early development of linear program-ming.

Practical Anti-Cycling Procedures

Theoretical anti-cycling rules provide termination guarantees, but practical implementations may also need to deal with numerical effects. Floating-point arithmetic can turn exact zeros and exact ties into small numerical quantities. Therefore, a practical implementation must distinguish between mathematical degeneracy and numerical tolerance.

Gill, Murray, Saunders, and Wright developed an anti-cycling procedure for linearly constrained optimization that uses controlled infeasibility and an adjusted working feasibility tolerance [16].

The important point for the present study is that numerical anti-cycling procedures should not be confused with the mathematical definition of cycling. Mathematical cycling is based on the repetition of a basis. Numerical safeguards may instead be used to prevent unstable or repetitive behaviour caused by finite-precision computations.

Comparison of the Main Approaches

The main approaches considered in this paper can be summarized as follows.

These approaches solve the same broad problem in different ways. Bland's rule changes the selection rule directly. Lexicographic pivoting imposes an ordering on tied candidates. Perturbation changes the representation of the problem so that ties are separated, while practical procedures focus on numerical behaviour [8, 13, 14, 16].

Why Tie-Breaking Matters

Suppose that the minimum-ratio test gives

$$\theta^* = \frac{(x_B)_i}{d_i} = \frac{(x_B)_k}{d_k}. \qquad (114)$$

Table 1: Main approaches for handling degeneracy and cycling

Method	Main idea	Role
Bland's rule	Choose the smallest-index eligible variable	Provides a simple finite-termination anti-cycling rule
Lexicographic pivoting	Resolve ties using an ordered comparison of candidate rows	Provides deterministic tie-breaking under degeneracy
Perturbation	Introduce sufficiently small ordered changes to separate ties	Removes exact degeneracy or ties in a controlled man-ner
Practical anti-cycling proce-dures	Use numerical safe-guards and controlled feasibility	Addresses computational effects that can arise in numerical implementations

Both variables are valid candidates for leaving the basis. If no rule is specified, the choice may depend on the implementation.

In a nondegenerate situation, such a tie may simply lead to different valid simplex paths. In a degenerate situation, however, repeated ties can contribute to a sequence of zero-improvement pivots. A suitable tie-breaking rule provides a consistent way of choosing between the alternatives.

This is why anti-cycling procedures are closely connected with degeneracy. They do not remove the mathematical possibility of degenerate solutions. Instead, they control how the algorithm behaves when degeneracy creates multiple pivot choices.

Bland's Rule and the Present Study

Bland's rule is particularly useful in the present study because it gives a clear theoretical reference point. The rule demonstrates that cycling is not an unavoidable consequence of degeneracy. A suitable pivot-selection strategy can guarantee that the algorithm does not revisit a previous basis [8].

The numerical example developed later in this paper will therefore be analysed by record-ing the entering variable, leaving variable, minimum ratio, step length, objective value, and basis at each iteration. This makes it possible to distinguish ordinary simplex progress from degenerate pivots, stalling, and basis repetition.

Limitations of Anti-Cycling Rules

No single anti-cycling approach is best for every purpose. A rule designed to provide a simple theoretical termination guarantee may require more iterations than a rule designed primarily for practical efficiency. Similarly, perturbation can simplify tie-breaking but introduces a modified representation of the original data.

For the purposes of this paper, the main goal is not to propose a new anti-cycling pro-cedure. Instead, the established methods are used to explain why cycling occurs and how systematic pivot selection can prevent it.

The next section presents a numerical example in which the simplex iterations are ex-amined step by step. Particular attention is given to the basis, the minimum-ratio test, the step length, and the objective value.

Illustrative Numerical Example: Degeneracy at an Optimal BFS

This section illustrates how degeneracy can arise at an optimal basic feasible solution. The example also shows why a tied minimum-ratio test requires a clear tie-breaking rule.
Consider the linear program

$$\min z = -x_1 - 2x_2 \tag{115}$$

subject to

$$x_1 + x_2 + s_1 = 4, \tag{116}$$
$$x_2 + s_2 = 2, \tag{117}$$
$$x_1 + x_2 + s_3 = 4, \tag{118}$$

where

$$x_1, x_2, s_1, s_2, s_3 \geq 0.$$

The first and third constraints have the same coefficients for x_1 and x_2 and the same right-hand side. Hence,

$$s_1 = s_3. \tag{119}$$

At the optimal point considered below, both constraints are active. This allows different bases to represent the same geometric point and leads to a degenerate optimal basic feasible solution.

Initial Basic Feasible Solution

The initial basis is

$$B_0 = \{s_1, s_2, s_3\}.$$

Setting the nonbasic variables $x_1 = x_2 = 0$ gives

$$s_1 = 4, \qquad s_2 = 2, \qquad s_3 = 4.$$

Thus, the initial basic feasible solution is

$$(x_1, x_2, s_1, s_2, s_3) = (0, 0, 4, 2, 4), \tag{120}$$

with

$$z = 0.$$

The initial simplex tableau is

Table 2: Initial simplex tableau.

Basis	x_1	x_2	s_1	s_2	s_3	RHS
s_1	1	1	1	0	0	4
s_2	0	1	0	1	0	2
s_3	1	1	0	0	1	4
z	-1	-2	0	0	0	0

For the minimization problem, the most negative reduced cost is -2, corresponding to x_2. Hence, x_2 is selected as the entering variable.

The minimum-ratio test gives

$$\frac{4}{1} = 4, \qquad \frac{2}{1} = 2, \qquad \frac{4}{1} = 4.$$

Therefore,

$$\theta = 2, \tag{121}$$

and s_2 leaves the basis.

Since $\theta > 0$, the first pivot is nondegenerate.

First Pivot

After pivoting on the $x2$ entry in the $s2$ row, the basis becomes

$$B_1 = \{s_1, x_2, s_3\}.$$

The resulting equations are

$$s_1 = 2 - x_1 + s_2, \tag{122}$$
$$x_2 = 2 - s_2, \tag{123}$$
$$s_3 = 2 - x_1 + s_2, \tag{124}$$

and the objective function becomes

$$z = -4 - x_1 + 2s_2. \tag{125}$$

Setting the nonbasic variables $x_1 = s_2 = 0$ gives

$$x_1 = 0, \qquad x_2 = 2, \qquad s_1 = 2, \qquad s_2 = 0, \qquad s_3 = 2.$$

Hence,

$$z = -4.$$

The corresponding tableau is

The remaining negative reduced cost is -1, corresponding to x_1. Therefore, x_1 is selected as the entering variable.

Table 3: Simplex tableau after the first pivot.

Basis	x_1	x_2	s_1	s_2	s_3	RHS
s_1	1	0	1	-1	0	2
x_2	0	1	0	1	0	2
s_3	1	0	0	-1	1	2
z	-1	0	0	2	0	-4

Tied Minimum-Ratio Test

For the x1 column, the positive coefficients occur in the rows corresponding to s1 and s3. The corresponding ratios are

$$\frac{2}{1} = 2, \qquad \frac{2}{1} = 2.$$

Thus,

$$\theta = 2 \tag{126}$$

is attained by both rows.

The minimum ratio is positive. Therefore, this tie does not constitute a degenerate pivot. It simply means that there are two eligible choices for the leaving variable.

Suppose that s1 is selected to leave. This is the choice made by Bland's rule because s1 has the smaller index than s3.

Second Pivot

Pivoting on the s1 row gives

$$x_1 = 2 - s_1 + s_2, \tag{127}$$
$$x_2 = 2 - s_2, \tag{128}$$
$$s_3 = s_1. \tag{129}$$

The objective function becomes

$$z = -6 + s_1 + s_2. \tag{130}$$

The new basis is

$$B_2 = \{x_1, x_2, s_3\}.$$

Setting the nonbasic variables

$$s_1 = s_2 = 0$$

gives

$$x_1 = 2, \qquad x_2 = 2, \qquad s_3 = 0.$$

Therefore,

$$(x_1, x_2, s_1, s_2, s_3) = (2, 2, 0, 0, 0), \tag{131}$$

and

$$z = -6. \tag{132}$$

Table 4: Simplex tableau after the second pivot.

Basis	x_1	x_2	s_1	s_2	s_3	RHS
x_1	1	0	1	-1	0	2
x_2	0	1	0	1	0	2
s_3	0	0	-1	0	1	0
z	0	0	1	1	0	-6

The resulting tableau is
The reduced costs of the nonbasic variables s_1 and s_2 are

$$\bar{c}_{s_1} = 1, \qquad \bar{c}_{s_2} = 1.$$

Since both reduced costs are non-negative, no further improvement is possible under the minimization convention. Hence, the current basic feasible solution is optimal.

The optimal solution is therefore

$$\boxed{x_1^* = 2, \qquad x_2^* = 2, \qquad z^* = -6.} \tag{133}$$

Since s_3 is a basic variable and

$$s3 = 0,$$

the optimal basic feasible solution is degenerate.

Alternative Leaving Variable

The minimum-ratio test in the previous iteration produced a tie between s1 and s3. It is therefore useful to examine the alternative choice.

If s_3 is selected to leave instead, the resulting basis is

$$B_2' = \{s_1, x_2, x_1\}.$$

From

$$s_3 = 2 - x_1 + s_2,$$

we obtain

$$x_1 = 2 + s_2 - s_3.$$

Since

$$x_2 = 2 - s_2$$

and

$$s_1 = s_3,$$

setting the nonbasic variables

$$s_2 = s_3 = 0$$

gives

$$x_1 = 2, \qquad x_2 = 2, \qquad s_1 = 0.$$

Thus, the same geometric point is obtained:

$$(x_1, x_2) = (2, 2),$$

with the same objective value

$$z = -6.$$

In this alternative basis, s1 is basic and has value zero. Therefore, this representation is also degenerate.

The two different bases

$$\{x_1, x_2, s_3\} \qquad \text{and} \qquad \{s_1, x_2, x_1\}$$

represent the same geometric point. This illustrates why a basis and a geometric vertex should not be regarded as identical objects in a degenerate linear program.

Verification of the Optimal Solution

The optimal point can also be checked directly from the original constraints. At

$$(x_1, x_2) = (2, 2),$$

the three equations give

$$2 + 2 + s_1 = 4,$$

$2 + s_2 = 2,$

and

$$2 + 2 + s_3 = 4.$$

Hence,

$$s_1 = s_2 = s_3 = 0,$$

so the point is feasible.

The objective value is

$$z = -x_1 - 2x_2 = -2 - 2(2) = -6.$$

The simplex tableau gives non-negative reduced costs for all nonbasic variables at this point. Therefore, the simplex optimality condition is satisfied, confirming that

$$\boxed{z^* = -6}.$$

Interpretation

The example demonstrates that degeneracy may occur at an optimal basic feasible solution. At the optimal point,

$$(x_1, x_2) = (2, 2),$$

all three slack variables are zero in the first basis representation. Consequently, the basic variable s_3 has value zero, making the optimal BFS degenerate.

The example also demonstrates the difference between a tied minimum-ratio test and a degenerate pivot. The tie occurs at the positive value

$$\theta = 2,$$

so the pivot itself is not degenerate. Degeneracy appears in the resulting BFS because one of the basic variables has value zero.

The alternative leaving-variable choice gives a different basis representing the same geo-metric point. This illustrates the role of tie-breaking rules in degenerate linear programs.

Importantly, this example does not produce cycling. No basis is revisited. Instead, it illustrates how degeneracy can create multiple basis representations of the same optimal vertex. This provides a useful foundation for the cycling example considered in the following section.

A Classical Cycling Example

The previous sections showed that degeneracy can lead to zero-step pivots and that a se-quence of such pivots may prevent the Simplex Method from making progress. This section presents the classical cycling example introduced by Beale [11]. It gives a direct illustration of how a degenerate simplex process can return to a previously visited basis.

Consider the linear programming problem

$$\min z = -\frac{3}{4}x_4 + 20x_5 - \frac{1}{2}x_6 + 6x_7 \tag{134}$$

subject to

$$x_1 + \frac{1}{4}x_4 - 8x_5 - x_6 + 9x_7 = 0, \tag{135}$$

$$x_2 + \frac{1}{2}x_4 - 12x_5 - \frac{1}{2}x_6 + 3x_7 = 0, \tag{136}$$

$$x_3 + x_6 = 1, \tag{137}$$

where

$$x_j \geq 0, \qquad j = 1, \ldots, 7. \tag{138}$$

The initial basis is

$$B_0 = \{x_1, x_2, x_3\}. \tag{139}$$

Setting the nonbasic variables

$$x_4 = x_5 = x_6 = x_7 = 0$$

gives

$$x_1 = 0, \qquad x_2 = 0, \qquad x_3 = 1. \tag{140}$$

Thus, the initial basic feasible solution is

$$x^{(0)} = (0, 0, 1, 0, 0, 0, 0), \tag{141}$$

with

$$z^{(0)} = 0. \tag{142}$$

The initial BFS is degenerate because x_1 and x_2 are basic variables with value zero.

Pivot Selection Rule

To demonstrate cycling, we use the following simplex pivot rule.

First, among the nonbasic variables with negative reduced cost, the variable with the most negative reduced cost is selected to enter the basis.

Second, the minimum-ratio test is applied to determine the leaving variable. If more than one basic variable attains the minimum ratio, the variable with the smallest index is selected.

The purpose of specifying the rule is important. Cycling is a property of the combination of the degenerate problem and the pivot-selection rule. A different tie-breaking rule can produce a different sequence of bases.

First Pivot

Initially, the reduced costs of the nonbasic variables are

$$\bar{c}_4 = -\frac{3}{4}, \qquad \bar{c}_5 = 20, \qquad \bar{c}_6 = -\frac{1}{2}, \qquad \bar{c}_7 = 6.$$

The most negative reduced cost is

$$-\frac{3}{4},$$

so x_4 enters the basis.

For the x_4 column, the two eligible rows give the ratios

$$\frac{0}{1/4} = 0, \qquad \frac{0}{1/2} = 0.$$

The minimum ratio is therefore

$$\theta_1 = 0.$$

There is a tie between x_1 and x_2. According to the specified tie-breaking rule, x_1 leaves the basis.

Hence,

$$B_1 = \{x_4, x_2, x_3\}. \tag{143}$$

Since

$$\theta_1 = 0,$$

the first pivot is degenerate.

The basic feasible solution remains

$$x^{(1)} = (0, 0, 1, 0, 0, 0, 0), \tag{144}$$

and

$$z^{(1)} = 0. \tag{145}$$

Second Pivot

After the first pivot, the most negative reduced cost corresponds to $x5$. Therefore,

$$x_5$$

enters the basis.

The minimum-ratio test selects x_2 as the leaving variable. The step length is again

$$\theta_2 = 0.$$

Consequently,

$$B_2 = \{x_4, x_5, x_3\}. \tag{146}$$

The basic feasible solution and objective value remain unchanged:

$$x^{(2)} = (0, 0, 1, 0, 0, 0, 0), \qquad z^{(2)} = 0. \tag{147}$$

Thus, the second pivot is also degenerate.

Third Pivot

At the third iteration, $x6$ is selected as the entering variable according to the most-negative-reduced-cost rule.

The minimum-ratio test selects x_4 as the leaving variable. Again,

$$\theta_3 = 0.$$

The new basis is

$$B_3 = \{x_6, x_5, x_3\}. \tag{148}$$

The represented basic feasible solution is still

$$x^{(3)} = (0, 0, 1, 0, 0, 0, 0), \tag{149}$$

with

$$z^{(3)} = 0. \tag{150}$$

The third pivot is therefore degenerate.

Fourth Pivot

At the fourth iteration, $x7$ enters the basis. The minimum-ratio test selects $x5$ as the leaving variable, with

$$\theta_4 = 0.$$

The resulting basis is

$$B_4 = \{x_6, x_7, x_3\}. \tag{151}$$

Again,

$$x^{(4)} = (0, 0, 1, 0, 0, 0, 0), \qquad z^{(4)} = 0. \tag{152}$$

Thus, the fourth pivot is degenerate.

Fifth Pivot

At the fifth iteration, $x1$ enters the basis. The minimum-ratio test selects $x6$ to leave the basis, again with

$$\theta_5 = 0.$$

The new basis is

$$B_5 = \{x_1, x_7, x_3\}. \tag{153}$$

The basic feasible solution remains

$$x^{(5)} = (0, 0, 1, 0, 0, 0, 0), \tag{154}$$

and the objective value remains

$$z^{(5)} = 0. \tag{155}$$

Sixth Pivot and Return to the Initial Basis

At the sixth iteration, $x2$ enters the basis. The minimum-ratio test selects $x7$ as the leaving variable, with

$$\theta_6 = 0.$$

The resulting basis is

$$B_6 = \{x_1, x_2, x_3\}. \tag{156}$$

Therefore,

$$B_6 = B_0. \tag{157}$$

The algorithm has returned exactly to the initial basis.
The complete sequence is

$$\boxed{\begin{aligned} B_0 &= \{x_1, x_2, x_3\}, \\ B_1 &= \{x_4, x_2, x_3\}, \\ B_2 &= \{x_4, x_5, x_3\}, \\ B_3 &= \{x_6, x_5, x_3\}, \\ B_4 &= \{x_6, x_7, x_3\}, \\ B_5 &= \{x_1, x_7, x_3\}, \\ B_6 &= \{x_1, x_2, x_3\} = B_0. \end{aligned}} \tag{158}$$

Hence, after six degenerate pivots, the simplex method has returned to the basis from which it started.

Summary of the Pivot Sequence

The complete pivot sequence is summarized in Table 5.

Table 5: Verified pivot sequence for the classical cycling example.

Iteration	Entering	Leaving	Step θ	New basis	Objective
0	–	–	–	$\{x_1, x_2, x_3\}$	0
1	x_4	x_1	0	$\{x_4, x_2, x_3\}$	0
2	x_5	x_2	0	$\{x_4, x_5, x_3\}$	0
3	x_6	x_4	0	$\{x_6, x_5, x_3\}$	0
4	x_7	x_5	0	$\{x_6, x_7, x_3\}$	0
5	x_1	x_6	0	$\{x_1, x_7, x_3\}$	0
6	x_2	x_7	0	$\{x_1, x_2, x_3\}$	0

Every pivot in the cycle has

$$\theta = 0.$$

Therefore, none of the six pivots changes the represented feasible point. The objective value also remains equal to zero throughout the sequence.

Why This Is Genuine Cycling

The example is not merely a case of stalling. Stalling would describe a sequence of iterations in which the objective value does not improve. Here, a stronger condition is satisfied: the basis itself is repeated.

Specifically,

$$B_6 = B_0. \tag{159}$$

Once the algorithm returns to B_0, the same pivot-selection rule produces the same en-tering and leaving variables:

$$x_4 \to x_1, \quad x_5 \to x_2, \quad x_6 \to x_4, \quad x_7 \to x_5, \quad x_1 \to x_6, \quad x_2 \to x_7.$$

Consequently, the same six-pivot sequence can be repeated indefinitely:

$$B_0 \to B_1 \to B_2 \to B_3 \to B_4 \to B_5 \to B_0 \to \cdots. \tag{160}$$

This establishes cycling.

Role of Degeneracy

The example also makes clear why degeneracy is central to the cycling problem. At the initial BFS,

$$x1 = x2 = 0$$

even though both variables are basic.

The first pivot therefore has a zero step. The same pattern continues through the subse-quent pivots:

$\theta 1 = \theta 2 = \cdots = \theta 6 = 0.$

The basis changes at every iteration, but the represented feasible point does not move. Thus, the algorithm can make a sequence of basis changes without making geometric progress.

The mechanism can be summarized as

$$\boxed{\text{degenerate BFS} \longrightarrow \text{zero-step pivots} \longrightarrow \text{basis changes without movement} \longrightarrow \text{repeated basis} \longrightarrow \text{cycling.}} \tag{161}$$

This example therefore provides a direct numerical demonstration of the theoretical dis-cussion in the preceding sections.

Importance of the Pivot Rule

The cycle is not an unavoidable property of the feasible region. It results from applying a particular pivot-selection and tie-breaking procedure to a degenerate problem.

This point is important because an anti-cycling rule can change the sequence of bases and prevent the repeated cycle. In particular, Bland's rule provides a systematic variable-selection rule that guarantees finite termination of the Simplex Method [8].

The next section applies Bland's rule to the degeneracy problem and explains how an appropriate tie-breaking strategy prevents the type of repeated basis sequence observed in this example.

Prevention of Cycling Using Bland's Rule

The cycling example in the previous section shows that degeneracy can cause the Simplex Method to change bases without changing the feasible point or the objective value. If the pivot rule repeatedly makes the same choices, the algorithm can return to a previously visited basis.

A classical way to prevent this behaviour is Bland's rule. The rule uses the indices of the variables to resolve ties in a fixed manner. Bland proved that the Simplex Method cannot cycle when this rule is used [8].

Statement of Bland's Rule

Bland's rule consists of two parts.

First, among all eligible nonbasic variables that can enter the basis, the variable with the smallest index is selected.

Second, if more than one basic variable attains the minimum ratio and is therefore eligible to leave the basis, the basic variable with the smallest index is selected.

Let

$$J = \{j : \bar{c}_j < 0\} \tag{162}$$

denote the set of eligible entering variables for the minimization convention used in this paper. Bland's entering-variable rule selects

$$j^* = \min J. \tag{163}$$

Suppose that the minimum-ratio test produces the set

$$I = \left\{ i : \frac{(x_B)_i}{d_i} = \theta^* \right\}. \tag{164}$$

Bland's leaving-variable rule selects

$$i^* = \min I. \tag{165}$$

The rule therefore provides a deterministic way to resolve ties.

Application to the Cycling Problem

Consider again the classical cycling example discussed in Section 10. The cycle was generated by a sequence of degenerate pivots:

$$B_0 \to B_1 \to B_2 \to B_3 \to B_4 \to B_5 \to B_0. \tag{166}$$

Every pivot in this sequence had zero step length:

$$\theta_i = 0, \qquad i = 1, \ldots, 6. \tag{167}$$

Thus, the feasible point and objective value remained unchanged while the basis changed. The important feature of the cycle is the presence of tied choices in the pivot procedure. Bland's rule removes the arbitrary part of these choices by selecting the eligible variable with the smallest index.

Consequently, the pivot sequence generated by Bland's rule is not the cycling sequence described in Section 10. The rule imposes a different ordering on the possible basis changes.

Why Bland's Rule Prevents Cycling

The main theoretical result associated with Bland's rule is that repeated bases cannot occur under the rule. Bland proved that if the Simplex Method uses his smallest-index rule for both entering and leaving variables, the algorithm terminates after a finite number of pivots [8].

Suppose, for contradiction, that Bland's rule produced a cycle. Then there would exist two iterations p < q such that

$$B^{(p)} = B^{(q)}. \tag{168}$$

Since the same basis is reached again, the simplex process would have returned to a previously visited state. The subsequent sequence of choices would therefore repeat.

Bland's result shows that such a repeated basis cannot occur under the smallest-index pivot rule. Hence,

$$\boxed{\text{Bland's rule} \quad \Longrightarrow \quad \text{no cycling.}} \tag{169}$$

The result is particularly important for degenerate linear programs because it provides a finite-termination guarantee even when several eligible variables are tied.

Comparison with an Arbitrary Tie-Breaking Rule

The difference between an arbitrary rule and Bland's rule can be seen in a simple tied minimum-ratio test. Suppose that

$$\frac{(x_B)_i}{d_i} = \frac{(x_B)_k}{d_k} = \theta^*, \qquad i < k. \tag{170}$$

An arbitrary implementation may select either i or k. Bland's rule selects

$$i^* = \min\{i, k\} = i. \tag{171}$$

The same principle applies to entering variables. If

$$\bar{c}_i < 0$$

and

$$\bar{c}_k < 0, \qquad i < k,$$

then Bland's rule selects i.

Thus, the rule replaces an arbitrary choice with a fixed ordering based on variable indices.

Bland's Rule and Degenerate Pivots

Bland's rule does not eliminate degeneracy. A degenerate BFS can still contain zero basic variables, and a pivot selected by Bland's rule can still have

$$\theta = 0. \tag{172}$$

The purpose of the rule is different. It controls which eligible variable is selected when degeneracy produces multiple choices.

Therefore,

$$\text{Bland's rule} \not\Rightarrow \text{absence of degeneracy.} \tag{173}$$

Instead,

$$\text{Bland's rule} \Rightarrow \text{cycling is prevented.} \tag{174}$$

This distinction is important. Degeneracy is a property of a basic feasible solution, whereas cycling is a property of the sequence of bases generated by the pivot rule.

Bland's Rule Versus the Classical Cycling Sequence

The classical cycling example can be used to illustrate the role of the rule without claiming that every possible pivot sequence must cycle.

Under the cycling rule used in Section 10, the sequence was

$$\begin{aligned} B_0 &= \{x_1, x_2, x_3\}, \\ B_1 &= \{x_4, x_2, x_3\}, \\ B_2 &= \{x_4, x_5, x_3\}, \\ B_3 &= \{x_6, x_5, x_3\}, \\ B_4 &= \{x_6, x_7, x_3\}, \\ B_5 &= \{x_1, x_7, x_3\}, \\ B_6 &= \{x_1, x_2, x_3\}. \end{aligned} \tag{175}$$

The final basis satisfies

$$B_6 = B_0, \tag{176}$$

which establishes the cycle.

Bland's rule does not permit this repeated-basis behaviour. Whenever the pivot proce-dure encounters competing eligible variables, the smallest index is selected. The resulting sequence therefore follows a different path through the collection of feasible bases.

The important result is not that Bland's rule necessarily requires fewer iterations. Its significance here is that the rule gives a theoretical guarantee against cycling.

Advantages and Limitations

Bland's rule has a simple definition and does not require perturbing the constraint coeffi-cients or the right-hand-side vector. Its main theoretical advantage is the finite-termination guarantee proved by Bland [8].

Its simplicity also makes it useful when explaining degeneracy and cycling. The rule can be stated entirely in terms of variable indices, so the tie-breaking decision can be reproduced from the simplex tableau.

The rule is primarily a theoretical anti-cycling device in the present study. Practical sim-plex implementations may use other strategies or additional numerical safeguards. There-fore, the theoretical guarantee of Bland's rule should not be interpreted as a claim that every practical simplex implementation uses this rule.

Summary

The numerical example in Section 10 demonstrated that a sequence of degenerate pivots can return the Simplex Method to a previously visited basis. Bland's rule addresses this problem by imposing a fixed smallest-index choice for both entering and leaving variables.

The main distinction can be written as

Degeneracy may produce several eligible pivot choices
↓
Bland's rule selects the smallest index
↓
Repeated basis sequences are prevented
↓
Finite termination (177)

Thus, Bland's rule provides a clear theoretical response to the cycling problem. It does not remove degeneracy from the linear program; rather, it controls the pivot decisions that can otherwise allow degeneracy to produce a cycle.

The following section compares the main anti-cycling approaches discussed in this paper and highlights their respective roles in theoretical and computational treatments of degen-eracy.

Comparison of Anti-Cycling Methods

The examples discussed in the previous sections show that degeneracy can affect the be-haviour of the Simplex Method in different ways. A degenerate pivot may leave the objective value unchanged, and a sequence of such pivots can lead to cycling. For this reason, several strategies have been developed to control the choice of entering and leaving variables.

The main approaches considered in this study are Bland's rule, lexicographic pivoting, perturbation methods, and numerical safeguards. Although these methods have the same broad purpose, they deal with degeneracy in different ways.

Bland's Rule

Bland's rule resolves ties by assigning priority according to the indices of the variables. The eligible nonbasic variable with the smallest index is selected to enter, while the eligible basic variable with the smallest index is selected to leave.

The main advantage of Bland's rule is its theoretical simplicity. It does not require changes to the original linear program, and it provides a finite-termination guarantee for the Simplex Method [8].

Its basic structure can be written as

$$j^* = \min\{j : \bar{c}_j < 0\}, \tag{178}$$

for the entering variable, together with

$$i^* = \min I \tag{179}$$

for the leaving variable, where I is the set of variables attaining the minimum ratio.

Bland's rule therefore provides a direct and reproducible way to resolve ties.

Lexicographic Pivoting

Lexicographic pivoting uses an ordered comparison of the tableau rows when the minimum-ratio test produces a tie. Instead of relying only on the variable index, the rows are compared lexicographically.

Suppose that two or more rows produce the same minimum ratio. The lexicographic rule compares the corresponding normalized rows and selects the row that is smallest under the lexicographic ordering.

The main idea is to replace an ambiguous comparison by a strict ordering:

$$R_i <_{\text{lex}} R_j \tag{180}$$

whenever the first component at which the two normalized rows differ is smaller for R_i.

This approach provides a systematic way of resolving ties and is closely related to the useof infinitesima l perturbations. Under exact arithmetic, lexicographic pivoting can be usedas an anti-cycling procedure.

Perturbation Methods

Another approach is to perturb the right-
hand sides or other quantities in the linear programby sufficiently small values. The purpose is to remove e xact ties and thereby make the pivotsequence unique.

For example, the right-hand side vector can be conceptually replaced by

$$b(\varepsilon) = b + \varepsilon b_1 + \varepsilon^2 b_2 + \cdots, \tag{181}$$

where

$0 < \varepsilon \ll 1$.

The perturbation is treated symbolically or through an equivalent lexicographic proce-
dure. After the simplex process has been completed, the perturbation is interpreted astending to zer o.

The main benefit of this approach is that it removes the exact degeneracy that causestied ratio tests . However, an explicit numerical perturbation can introduce additional nu-
merical issues if the perturbation is chosen poorly. For this reason, symbolic or lexicographicimplementatio ns are often preferred when a rigorous anti-cycling guarantee is required.

Numerical Safeguards

Practical simplex implementations also use numerical safeguards to deal with the effects offinite-
precision arithmetic. Very small numbers may be treated as zero, and tolerances maybe used when compa ring ratios or reduced costs.

For example, a feasibility tolerance $\epsilon_f > 0$ may be used so that a computed value satis-fying

$$|x_i| \le \epsilon_f \tag{182}$$

is treated as numerically zero.

Similarly, a reduced cost may be regarded as zero when

$$|\bar{c}_j| \le \epsilon_c. \tag{183}$$

These safeguards are useful in computational work, but they should not be confused witha mathematic al anti-cycling proof. Numerical tolerances can reduce instability caused byfloating-
point arithmetic, whereas rules such as Bland's rule provide a theoretical guaranteeconcerning cycling.

Comparison of the Main Approaches

The main characteristics of the approaches discussed above are summarized in Table 6.

The comparison shows that there is no need to remove degeneracy from the linear programming proble m itself in order to prevent cycling. Instead, the pivotselection procedurecan be designed to control the choices produced by degeneracy.

Table 6: Comparison of methods for handling degeneracy and cycling.

Method	Main idea	Main advantage	Main limitation
Bland's rule	Choose the smallest-indexeligible variable	Simple rule with a finite-terminationguarantee	May require moreiterations in someproblems
Lexicographicpivoting	Use lexicographic ordering to resolve ties	Provides a systematic anti-cycling procedure	Requires additional row comparisons
Perturbation	Introduceafinitesmallchangesremove ties	Provides a clearway to interpretdegeneracy	Explicit numerical perturbations may affect numerical stability
Numerical safeguards	Use tolerances and checks numerical	Useful in practical floating-pointimplementations	Does not by itself provide a mathematical anti-cycling guarantee

Theoretical and Computational Perspectives

There is an important distinction between theoretical anticycling rules and practical numerical safeguards.

From a theoretical perspective, Bland's rule and lexicographic pivoting are particularlyuseful because their behaviour can be stated precisely. They provide deterministic rules forresolving ties and can be analysed without relying on floating-point tolerances.

From a computational perspective, numerical safeguards are also important. Real implementations use finite-precision arithmetic, so exact equality tests may not always be reliable.Small numerical errors can make a problem appear degenerate even when the exact mathematical problem is not, or can hide a degeneracy that exists in exact arithmetic.

For this reason, a robust implementation should distinguish between mathematical degeneracy and numerical near-degeneracy.

Relationship Between Degeneracy, Stalling, and Cycling

The discussion in this paper can be summarized through the following relationship:

$$\text{Degeneracy} \longrightarrow \text{possible zero-step pivot} \longrightarrow \text{stalling} \longrightarrow \text{possible cycling.} \qquad (184)$$

The arrows in this expression indicate possibility rather than necessity. A degenerateBFS does not necessarily lead to stalling, and stalling does not necessarily lead to cycling.

The examples in Sections 9 and 10 demonstrate these distinctions. Section 9 showed adegenerate optimal BFS without cycling, whereas Section 10 showed a sequence of degeneratepivots that returned to a previously visited basis.

The anti-cycling methods discussed in this section act mainly at the pivot selectionstage. Their purpose is to prevent the sequence of basis changes from becoming trapped in a repeated cycle.

Choice of Method for the Present Study

For the theoretical discussion in this paper, Bland's rule is particularly appropriate because of its simple formulation and well-established finite-termination result [8]. It also provides a clear way to explain how a deterministic tie-breaking rule changes the behaviour of the Simplex Method.

Lexicographic pivoting provides a useful complementary perspective because it shows that cycling can also be avoided by imposing a strict ordering on tied candidates.

Numerical safeguards are considered mainly from the computational perspective. They are useful when implementing the Simplex Method in finite precision, but they are not treated as substitutes for a formal anti-cycling rule.

Thus, the main theoretical focus of the present study is Bland's rule, while lexicographic pivoting, perturbation, and numerical safeguards are discussed as related approaches to handling degeneracy and cycling.

CONCLUSION AND FUTURE RESEARCH DIRECTIONS

This paper examined degeneracy in linear programming and its connection with the Simplex Method. The discussion focused on the effect of zero-valued basic variables, tied minimum-ratio tests, stalling, and cycling.

Degeneracy occurs when at least one basic variable has value zero. Although the corre-sponding basic feasible solution remains feasible, the simplex method may perform a pivot with zero step length. In such a case, the basis changes while the represented solution and objective value remain unchanged. This behaviour explains why degeneracy can cause a lack of apparent progress during simplex iterations.

A numerical example was used to illustrate degeneracy at an optimal basic feasible so-lution. The example also showed that a tied minimum-ratio test does not necessarily mean that the pivot is degenerate. When the common minimum ratio is positive, the tie only indicates that more than one leaving variable is eligible. Degeneracy occurs when a basic variable has value zero in the resulting BFS.

The paper then considered the classical cycling example of Beale. In that example, a sequence of zero-step pivots changes the basis without changing the feasible point or the objective value. After a finite number of pivots, the method returns to a previously visited basis. The repeated basis establishes genuine cycling.

Several approaches for dealing with degeneracy and cycling were also discussed. Bland's rule provides a simple deterministic tie-breaking procedure and guarantees finite termination of the Simplex Method [8]. Lexicographic pivoting and perturbation provide alternative ways of resolving ties, while numerical safeguards are useful for handling the effects of finite-precision arithmetic in practical implementations.

The main lesson is that degeneracy itself does not imply cycling. A degenerate basic feasible solution may be optimal and may never lead to a cycle. Cycling requires a particular sequence of pivot decisions that returns the algorithm to a previously visited basis. Therefore, the distinction between degeneracy, stalling, and cycling is important both in theoretical analysis and in computational implementations of the Simplex Method.

Future Research Directions

Several directions can be considered for further study. One possibility is to develop computa-tional experiments comparing different anti-cycling rules on degenerate linear programming problems. Such experiments could examine the number of pivots, computational time, and the frequency of degenerate pivots under different problem structures.

Another useful direction is the study of numerical degeneracy in finite-precision imple-mentations. Exact degeneracy and near-degeneracy may behave differently when floating-point arithmetic is used. A detailed study of this issue could help clarify the interaction between theoretical anti-cycling rules and practical numerical tolerances.

Further work could also consider degeneracy in other simplex variants, including the dual simplex method and revised simplex implementations. Comparing their behaviour on structured degenerate problems may provide additional insight into the relationship between pivot rules and algorithmic performance.

Finally, hybrid anti-cycling strategies could be investigated. Such strategies could com-bine a theoretically guaranteed rule, such as Bland's rule or lexicographic pivoting, with numerical safeguards designed for finite-precision computation. This may provide a useful balance between theoretical reliability and practical computational performance.

REFERENCES

1. Dantzig GB. Linear programming and extensions. Princeton (NJ): Princeton University Press; 1963.
2. Schrijver A. Theory of linear and integer programming. Chichester: John Wiley & Sons; 1998.
3. Bertsimas D, Tsitsiklis JN. Introduction to linear optimization. Belmont (MA): Athena Scientific; 1997.
4. Chvátal V. Linear programming. New York: W. H. Freeman; 1983.
5. Zörnig P. Systematic construction of examples for cycling in the simplex method. Comput Oper Res. 2006;33(8):2247-2262.
6. Dantzig GB. Making progress during a stall in the simplex algorithm. Linear Algebra Appl. 1989;114-115:251-259.
7. Gal T, Geue F. A new pivoting rule for solving various degeneracy problems. Oper Res Lett. 1992;11(1):23-32.
8. Bland RG. New finite pivoting rules for the simplex method. Math Oper Res. 1977;2(2):103-107.
9. Dosios K, Paparrizos K. Resolution of the problem of degeneracy in a primal and dual simplex algorithm. Oper Res Lett. 1997;20(1):45-50.
10. Im H, Wolkowicz H. Revisiting degeneracy, strict feasibility, stability, in linear programming. Eur J Oper Res. 2023;310(2):495-510.
11. Beale EML. Cycling in the dual simplex algorithm. Naval Res Logist Q. 1955;2(4):269-275.
12. Zörnig P. Degeneracy graphs and simplex cycling. Lecture Notes in Economics and Mathematical Systems. Vol. 357. Berlin: Springer; 1991.
13. Charnes A. Optimality and degeneracy in linear programming. Econometrica. 1952;20(2):160-170.
14. 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.
15. Wolfe P. A technique for resolving degeneracy in linear programming. J Soc Ind Appl Math. 1963;11(2):205-211.
16. Gill PE, Murray W, Saunders MA, Wright MH. A practical anti-cycling procedure for linearly constrained optimization. Math Program. 1989;45(1-3):437-474.
17. Fukuda K, Terlaky T. Criss-cross methods: a fresh view on pivot algorithms. Math Program. 1997;79(1-3):369-395.
18. Koberstein A, Suhl UH. Progress in the dual simplex method for large scale LP problems: practical dual phase 1 algorithms. Comput Optim Appl. 2007;37(1):49-65.
19. Gonzalez-Lima MD, Wei H, Wolkowicz H. A stable primal-dual approach for linear programming under nondegeneracy assumptions. Comput Optim Appl. 2009;44(2):213-247.
20. Klee V, Minty GJ. How good is the simplex algorithm?. In: Shisha O, editor. Inequalities III. New York: Academic Press; 1972. p. 159-175.