Understanding Straight Line forms: Analytical Geometry and beyond (the case of Linear Programming)

Jose Portillo

Abstract: This easy walk the path of Descartes himself armed with algebra and geometry: extract pure spatial meaning from symbolic expressions. No measuring tape, no compass, no guesswork. Just: equations, logic and system solving. This easy was written with a little help from AI GPT. It's a use case of human subject using the AI as copilot for math syntax checking, LaTeX formatting, and checking overall math ideas.

1 Straight Line Equations: Forms and Applications

A straight line in the Cartesian plane can be represented in several forms, each with its own advantages and applications. The most common forms include:

1.1 Slope-Intercept Form

The slope-intercept form is given by

$$y = mx + b, \tag{1}$$

where m is the slope of the line and b is the y-intercept. This form is especially convenient for graphing since it directly shows how the line behaves with respect to the y-axis.

1.2 General Form

The general form of a line is written as

$$Ax + By + C = 0, \tag{2}$$

where A, B, and C are real constants, and A and B are not both zero. This implicit form is very powerful because it can represent all lines, including vertical lines which cannot be expressed in slope-intercept form due to undefined slope.

From the general form, the slope m can be recovered as

$$m = -\frac{A}{B}, \quad \text{provided } B \neq 0. \tag{3}$$

1.3 Point-Slope Form

Given two points (x_1, y_1) and (x_2, y_2), the slope is

$$m = \frac{y_2 - y_1}{x_2 - x_1}. \tag{4}$$

The point-slope form of the line passing through (x_1, y_1) is

$$y - y_1 = m(x - x_1). \tag{5}$$

Note: the coordinates x and y are not a second fixed point, but rather variables representing ANY point on the line. The pair (x_1, y_1) is the GIVEN, fixed point used to construct the line, and m is the known slope. Once these are known, the equation defines all possible (x, y) points that lie along the same straight path.

While the slope-intercept form is intuitive for graphing, the general form is indispensable for:

- Representing vertical lines ($x = k$) where slope is undefined.
- Algebraic manipulation and solving systems of linear equations.
- Applications in higher dimensions (planes and hyperplanes).
- Implicit representation of lines, useful in computational geometry.

1.4 Example: Constructing and Solving a Triangle System

Consider three points in the plane:

$$A(1,2), \quad B(4,3), \quad C(2,5).$$

Before using three points to define a triangle, we must first confirm that they are not collinear — that is, they do not all lie on the same straight line. If they are collinear, they cannot enclose any area, and thus, no triangle can be formed.

Given three points $A(x_1, y_1)$, $B(x_2, y_2)$, and $C(x_3, y_3)$, we can check for collinearity using the area formula for a triangle derived from the determinant:

$$\text{Area} = \frac{1}{2} \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \tag{6}$$

This evaluates to:

$$\text{Area} = \frac{1}{2} \left| x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) \right| \tag{7}$$

If this area is zero, then the points are collinear — geometrically aligned — and no triangle exists.
This check is essential before proceeding with triangle-based constructions or region definitions in linear programming, as degeneracies (like zero-area shapes) can lead to incorrect or undefined results.
Note: See Appendix A for more on this determinant.
Back to our three point example

$$\text{Area} = \frac{1}{2}\left\|\begin{matrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{matrix}\right\| = \frac{1}{2}\left|1(3-5)-4(2-5)+2(2-3)\right| = \frac{1}{2}|-2+12-2| = \frac{1}{2}\cdot 8 = 4. \tag{8}$$

Since the area is non-zero, the points are not collinear and do form a triangle.
We seek the equations of the lines forming the triangle ABC and solve their systems to recover the vertices algebraically.
Note: Collinear points have zero area and thus cannot define a triangle. Always verify this before proceeding with symbolic derivations.

Line AB: Calculate the slope:

$$m_{AB} = \frac{3-2}{4-1} = \frac{1}{3}.$$

Point-slope form using point $A(1,2)$:

$$y-2 = \frac{1}{3}(x-1).$$

Solve for y:

$$y = \frac{1}{3}x + \frac{5}{3}.$$

Multiply both sides by 3 so we get the line's *General form*:

$$3y = x+5 \implies x-3y+5=0 \quad \text{(General form)}.$$

Line BC: Calculate the slope:

$$m_{BC} = \frac{5-3}{2-4} = -1.$$

Point-slope form using point $B(4,3)$

$$y-3 = -1(x-4).$$

Solve for y:

$$y = -x+7.$$

Rewrite so we get the line's *General form*:

$$x+y-7=0 \quad \text{(General form)}.$$

Line CA: Calculate the slope:

$$m_{CA} = \frac{2-5}{1-2} = 3.$$

Point-slope form using point $C(2,5)$

$$y-5 = 3(x-2).$$

Solve for y:

$$y = 3x-1.$$

Rewrite so we get the line's *General form*:

$$-3x+y+1=0.$$

Solving Intersections: Intersection of AB and BC:

$$\begin{cases} x-3y+5=0, \\ x+y-7=0. \end{cases}$$

Subtract second from first:

$$(x-3y+5)-(x+y-7) = -4y+12 = 0 \implies y=3.$$

Substitute back:

$$x+3-7=0 \implies x=4.$$

Vertex $B=(4,3)$.
Intersection of BC and CA:

$$\begin{cases} x+y-7=0, \\ -3x+y+1=0. \end{cases}$$

Subtract second from first:

$$(x+y-7)-(-3x+y+1) = 4x-8=0 \implies x=2.$$

Substitute back:

$$2+y-7=0 \implies y=5.$$

Vertex $C=(2,5)$.
Intersection of CA and AB:

$$\begin{cases} -3x+y+1=0, \\ x-3y+5=0. \end{cases}$$

Multiply the second equation by 3:

$$3x-9y+15=0.$$

Add to the first:

$$(-3x+y+1)+(3x-9y+15) = -8y+16=0 \implies y=2.$$

Substitute back:

$$x-3(2)+5=0 \implies x=1.$$

Vertex $A=(1,2)$.
Lets see a PYTHON implementation:

```
import matplotlib.pyplot as plt
import numpy as np

# Define points
A = np.array([1, 1])
B = np.array([4, 1])
C = np.array([2, 3])
points = np.array([A, B, C, A])  # Close triangle

# Function to compute triangle area using determinant
def triangle_area(p1, p2, p3):
    return 0.5 * abs(
        p1[0]*(p2[1] - p3[1]) +
        p2[0]*(p3[1] - p1[1]) +
        p3[0]*(p1[1] - p2[1])
    )

area = triangle_area(A, B, C)

if area == 0:
    print("The points are collinear | no triangle can be formed.")
else:
    print(f"Triangle area: {area:.2f} | points are NOT collinear. Proceeding...\n")

    # Function to compute line equation
    def line_equation(p1, p2):
        if p2[0] != p1[0]:
            m = (p2[1] - p1[1]) / (p2[0] - p1[0])
            b = p1[1] - m * p1[0]
            return m, b
        else:
            return np.inf, p1[0]  # Vertical line

    # Compute and print equations
    lines = {'AB': (A, B), 'BC': (B, C), 'CA': (C, A)}
    print("Line equations:")
    for name, (p1, p2) in lines.items():
        m, b = line_equation(p1, p2)
        if m == np.inf:
            print(f"{name}: vertical line at x = {b}")
        else:
            print(f"{name}: y = {m:.2f}x + {b:.2f}")

    # Plot triangle
    plt.figure(figsize=(6, 6))
    plt.plot(points[:, 0], points[:, 1], 'bo-', label='Triangle ABC')
    plt.fill(points[:, 0], points[:, 1], color='skyblue', alpha=0.3)

    # Label points
    for P, name in zip([A, B, C], ['A', 'B', 'C']):
        plt.text(P[0] + 0.1, P[1] + 0.1, name, fontsize=12, fontweight='bold')

    # Axis decoration
    plt.xlabel("x")
    plt.ylabel("y")
    plt.grid(True)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.xlim(0, 5)
    plt.ylim(0, 4)
    plt.title("Triangle from Three Points")
    plt.legend()
    plt.show()
```

This procedure can be applied to *any* three distinct, non-collinear points in the Cartesian plane. For any such triple, one can:

- Derive the line equations connecting each pair of points,
- Express those lines in general form,
- Solve the pairwise systems of equations,
- Recover the original triangle's vertices purely through algebraic means.

This approach demonstrates the power of analytic geometry: using symbolic equations and systems of linear equations to understand and reconstruct geometric figures without requiring any prior graphical or coordinate plotting.

2 Beyond the Triangle: Polygons and Linear Programming

Having established the power of line equations and algebraic systems in reconstructing a triangle, we now ask deeper questions: Can this procedure be extended to more complex geometric structures? And is this type of system solving related to techniques used in optimization problems such as Linear Programming (LP)? The answer to both is yes with important nuances.

2.1 Connection to Linear Programming (LP)

Let's now revisit our old friend, the straight line $y = mx+b$, and explore how this humble object turns into the central character in the theory of Linear Programming (LP). The idea is this: LP problems are built entirely out of straight lines — some act as boundaries, and one plays the role of the quantity we want to optimize.

In LP, we deal with:

- A collection of **linear constraints** — these are inequalities like $a_1x + a_2y \leq b$. Each one defines not just a line, but a *region*: a half-plane on one side of that line.
- A **feasible region** — this is the intersection of all those constraint regions. In two dimensions, it forms a polygon; in higher dimensions, a more general shape called a **polytope**.
- A **linear objective function** — something like $z = cx + dy$ — which we aim to either maximize or minimize. This too defines a straight line, but it doesn't restrict the region. Instead, we imagine sliding this line (or its level curves) across the feasible region to find where it reaches its highest or lowest value.

Geometrically, the LP setup looks like this:

- The constraints draw lines on the plane, then pick one half-plane (via inequality signs).
- The feasible region is the intersection of all the half-planes determined by the constraint inequalities.
- The objective function defines a direction — and we slide it along that direction until it touches the farthest edge of the region.

Note. On the Direction of the Objective Function. *The objective function $z = cx + dy$ doesn't merely define a line — it defines an entire family of parallel lines, one for each fixed value of z. These lines are called* level curves *of the function.*

The vector (c, d) formed by the coefficients tells us the direction in which z increases. This is known as the gradient vector, and it points perpendicular (normal) to the level curves. When solving an LP problem, we "slide" the objective line in this direction — that is, we move it along the vector (c, d) — in search of the highest or lowest value of z that still touches the feasible region. The optimal solution lies where this last sliding line makes contact: typically on a vertex of the feasible polygon.

This is the geometric essence of the Simplex method: it searches over the vertices *of the feasible polytope, not the whole region or every constraint line.*

So the entire LP framework is just an elegant game of straight lines. The constraints form a polygonal "arena" — or more generally, a **polytope**, which is a bounded region formed by the intersection of half-spaces in n-dimensional space. You can think of it as the high-dimensional cousin of a polygon or polyhedron.

Now, here's the reason behind the name *Simplex Method.* A **simplex** is the simplest possible polytope in any given dimension:

- In 1D: a line segment (2 vertices),
- In 2D: a triangle (3 vertices),
- In 3D: a tetrahedron (4 vertices),
- In nD: the convex hull of $n+1$ affinely independent points.

The Simplex algorithm moves from vertex to vertex of the feasible polytope, following the edges — but it never wastes time wandering inside the region or sliding along the faces. Why? Because in linear programming, the optimal value of the objective function is always achieved at one of the **vertices** (or extreme points) of the feasible region — assuming the problem is bounded and the constraints define a convex region. So, rather than search the whole region, the method hops efficiently from corner to corner, improving the value step by step until it hits the best one.

So when you hear "Simplex Method," think of a polite but no-nonsense traveler who's only interested in standing at the corners of the polytope, peering out to see where the view (value) is best — never stopping in the middle for a picnic.

In this sense, our previous exploration of finding a triangle from three intersecting lines was a kind of warm-up for LP: in LP, we also look for intersections of lines (constraints), but we care not just about the corners — we care about which corner makes us the most profit (or the least cost), according to the direction of our objective line.

2.2 Generalizing to Polygons with N Sides

Let us now generalize our triangle reconstruction method to arbitrary polygons with N sides. Suppose we have a simple, closed polygon defined not by its vertices but by the equations of its N sides. Our goal is to recover the polygon's corner points (vertices) using only algebra.

Step-by-step Procedure:

1. **Line Equations:** Begin with N linear equations, each representing one side of the polygon. These should be in general form:
$$A_i x + B_i y + C_i = 0, \quad \text{for } i = 1, 2, \ldots, N.$$
2. **Solve Intersections:** For each pair of adjacent lines L_i and L_{i+1}, solve the system:
$$\begin{cases} A_i x + B_i y + C_i = 0, \\ A_{i+1} x + B_{i+1} y + C_{i+1} = 0. \end{cases}$$
This yields the vertex V_{i+1}, the point of intersection between sides i and $i+1$.
3. **Closing the Polygon:** To complete the loop, solve the final system between the last side and the first:
$$\begin{cases} A_N x + B_N y + C_N = 0, \\ A_1 x + B_1 y + C_1 = 0, \end{cases}$$
giving the closing vertex V_1.

Conditions and Assumptions:

For this process to work reliably:

- The polygon must be **simple** (its sides should not intersect except at the vertices).
- The sides must be ordered such that each one connects logically to the next (i.e., the lines form a loop).
- No three consecutive sides should be concurrent (i.e., intersect at a single point), and adjacent lines should not be parallel (to avoid degenerate cases).

This approach allows the polygon to be reconstructed entirely through symbolic means — no coordinates required up front. Each vertex emerges naturally from the algebraic solution of systems of equations.

Apendix A

Determinants and Their Relation to the Geometry of a Triangle

A **matrix** is a rectangular array of numbers arranged in rows and columns. For example, a 2×2 matrix is written as:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Where a, b, c, and d are the elements of the matrix. The **determinant** of a square matrix is a special value that can be computed from its elements and provides important information about the matrix.

For a 2×2 matrix, the determinant is calculated as:

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$$

For a 3×3 matrix, the calculation of the determinant is more complex but follows a similar principle. The determinant of a 3×3 matrix is:

$$\det \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

The **determinant** of a matrix can tell us many things, including whether the matrix has an inverse and how the transformation associated with that matrix affects areas or volumes in geometric spaces.

Determinants and the Area of a Triangle

Given three non-collinear points in a two-dimensional plane, we can calculate the area of the triangle they form using the coordinates of these points. If the three points are represented by (x_1, y_1), (x_2, y_2), and (x_3, y_3), the area of the triangle can be computed using the following formula:

$$\text{Area} = \frac{1}{2} \left| x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) \right|$$

This formula can also be interpreted as half of the absolute value of the determinant of a 3×3 matrix whose rows are formed by the coordinates of the points, as shown below:

$$\text{Area} = \frac{1}{2} \left| \det \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \right|$$

The determinant of the matrix captures the "signed" area of the parallelogram formed by vectors corresponding to the points. The absolute value ensures that we get a positive area, regardless of the order of the points.

Why Does This Work?

The determinant essentially captures the *geometry* of the points in space. Specifically, it provides information about how the three points are positioned relative to one another in two-dimensional space. In the context of a triangle, the determinant measures the "scaled area" of the parallelogram formed by the vectors corresponding to the points.
If the determinant is zero, this means that the three points lie on a straight line, i.e., they are **collinear**. When the points are collinear, they do not form a triangle with any area, because the area of the parallelogram (or triangle) formed by them is zero. This is because there is no "width" to the shape — it is effectively a straight line.

The determinant captures how the points "spread out" in space. In other words, it quantifies the area of the parallelogram (or triangle) that these points form by using their positions relative to each other. When we calculate the determinant, we are measuring how much space is enclosed by the vectors formed by the points in question.
The determinant is a measure of how much one vector "scales" or "rotates" the other. In a geometric sense:

- If the determinant is **positive**, the orientation of the triangle formed by the points is counterclockwise.
- If the determinant is **negative**, the orientation is clockwise.
- If the determinant is **zero**, the points are collinear, and no triangle is formed.

Thus, the determinant provides a powerful tool not only to compute areas but also to understand the relative positions and orientations of geometric objects.

Note: The presence of the 1s in the third column might appear arbitrary at first, but they are both intentional and essential. Here's why. The determinant is only defined for square matrices — those with the same number of rows and columns. In the case of the triangle, we have three points, so three rows. By adding a third column (filled with 1s), we form a 3×3 square matrix, making the determinant operation valid. By writing each point as $(x, y, 1)$, we are effectively lifting our 2D points into a form known as *homogeneous coordinates*, common in projective geometry. This extension allows affine computations, like area, to be handled uniformly through determinants. More abstractly, the determinant can be interpreted as a signed volume (or area in 2D). It tells us not only how much space is enclosed, but also the orientation (clockwise or counterclockwise) of the points.

Note: The determinant is a remarkable scalar value associated with a square matrix that serves multiple roles, notably in algebra and geometry. Understanding why it can both determine the uniqueness of solutions in a linear system and measure volumes in geometry unveils the deep connection between these perspectives. Given a system of linear equations represented as

$$A\mathbf{x} = \mathbf{b},$$

where A is an $n \times n$ matrix, the determinant plays a crucial role in determining the system's behavior:

$$\det(A) \neq 0 \implies \text{the system has a unique solution.}$$

When $\det(A) = 0$, the matrix is singular, implying either infinitely many solutions or none at all. Intuitively,

this is because the rows (or columns) of A are linearly dependent, collapsing the solution space.
Geometrically, the determinant measures how the linear transformation associated with A scales volumes in n-dimensional space. For example, if you consider n vectors in $\mathbb{R}^n$ arranged as columns of A, the absolute value $|\det(A)|$ gives the volume of the parallelepiped spanned by these vectors.
Thus,

$$\text{Volume} = |\det(A)|.$$

If $\det(A) = 0$, the parallelepiped collapses into a lower-dimensional shape with zero volume, reflecting the linear dependence of the vectors.
The determinant acts as a bridge between algebra and geometry by encoding linear independence and volume scaling in one number. When $\det(A) \neq 0$, the vectors forming the matrix A are linearly independent—this guarantees a unique solution algebraically, and a nonzero volume geometrically.
Conversely, when $\det(A) = 0$, the vectors lie in a lower-dimensional subspace, causing the volume to vanish and the system to lose uniqueness.
In essence, the determinant reveals whether a linear transformation is invertible (algebra) and how it distorts space (geometry) simultaneously.
Ther are two famous "rumors" about the determinant of a square matrix A: 1. It tells us whether a system of linear equations $A\mathbf{x} = \mathbf{b}$ has a unique solution or not. 2. It measures the area (in 2D), volume (in 3D), or hyper-volume (in higher dimensions) of the shape formed by the columns of A.
But why would one number possibly do both jobs? Let's unpack this intriguing mystery step-by-step.
Consider a system of linear equations with n variables and n equations, which can be written in matrix form as:

$$A\mathbf{x} = \mathbf{b},$$

where A is an $n \times n$ matrix, $\mathbf{x}$ is the unknown vector, and $\mathbf{b}$ is the output vector.
The determinant $\det(A)$ tells us if A is invertible. If $\det(A) \neq 0$, the matrix has an inverse A^{-1}, and we can find a unique solution:

$$\mathbf{x} = A^{-1}\mathbf{b}.$$

Why? Because $\det(A) \neq 0$ means the rows (and columns) of A are linearly independent—they don't "overlap" or "collapse" onto each other. In other words, the system is well-behaved, and there's exactly one set of values for $\mathbf{x}$ that satisfies all equations.
If $\det(A) = 0$, the system either has infinitely many solutions or none at all, because the equations are "redundant" or "contradictory" — geometrically, the matrix squashes space so much that it loses dimensions.
Now, switch gears to geometry. Imagine the columns of A as vectors in n-dimensional space. These vectors can be seen as edges of a parallelepiped (a generalized parallelogram).
The absolute value of the determinant, $|\det(A)|$, tells you how much this parallelepiped scales the unit cube. Specifically:
- In 2D, it measures the area of the parallelogram formed by the column vectors. - In 3D, it measures the volume of the parallelepiped. - In higher dimensions, it measures the hyper-volume.
If $\det(A) = 0$, it means the vectors lie in a lower-dimensional space (e.g., a plane or line instead of filling 3D space), so the volume collapses to zero.
At first glance, these two applications seem different—one algebraic, one geometric. But they are deeply connected through the concept of linear independence and dimension.
- When $\det(A) \neq 0$, the columns of A form a "full-dimensional" shape with nonzero volume. Algebraically, this means no column is redundant, so the system has a unique solution. - When $\det(A) = 0$, the columns are linearly dependent, lying on a lower-dimensional subspace. Geometrically, the volume is zero. Algebraically, the system loses uniqueness.
This unity is why the determinant is such a powerful tool: it encodes both whether the linear transformation represented by A squashes space and whether the system of equations is solvable uniquely.
The determinant acts as a measure of how a linear transformation stretches or squashes space (geometry) and whether the transformation is invertible (algebra). Understanding this dual role deepens our appreciation of the beautiful harmony between algebra and geometry in linear algebra.

Connection Between Determinant and Area for Polygons

The determinant can be interpreted geometrically as representing the area of the parallelogram (or triangle, in the case of three points) formed by vectors.
For three points $A(x_1, y_1)$, $B(x_2, y_2)$, and $C(x_3, y_3)$, the area of the triangle formed by these points can be calculated using the determinant of the matrix formed by these coordinates. This area is given by:

$$\text{Area} = \frac{1}{2}\,|\text{determinant}|$$

So, when the determinant is zero, the area is zero, which means the points lie on the same straight line (because there's no triangle formed). In other words, if the points are collinear, the "triangle" collapses into a straight line, and the area is zero.
When the determinant is non-zero, there's a non-zero area, and the points are not collinear — they form a valid triangle.

In a sense, the determinant reflects how "spread out" the points are in the plane. When they're collinear, they all lie along the same line, so there's no area, and hence the determinant is zero.
For a polygon with more than three vertices, the determinant can still be used to calculate its area. The area of a polygon with n vertices (where $n \geq 3$) can be calculated using the shoelace formula (or Gauss's area formula). The determinant plays a central role in the shoelace formula, which is derived from the geometry of the polygon and the determinants of matrices.

Shoelace Formula for Area of a Polygon

Let's say we have a polygon with vertices $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, ordered in a counter-clockwise (or clockwise) direction. To compute the area of the polygon, the formula is:

$$A = \frac{1}{2} \left| \sum_{i=1}^{n-1} (x_i y_{i+1} - y_i x_{i+1}) + (x_n y_1 - y_n x_1) \right|$$

This formula is derived from the fact that the determinant reflects how the vertices are spread out in space. Specifically, it calculates the signed area of the polygon, where each signed area corresponds to a triangle formed by two consecutive vertices and the origin.
When using determinants, one can compute the signed area for each triangle formed by the vertices and sum them up. This approach, extended from the triangle case, can be applied to polygons with more vertices.
The determinant of a matrix formed by the coordinates of the vertices of the polygon can be interpreted as the signed area of the shape. If the points are collinear (or degenerate in some way), the area of the polygon will be zero. On the other hand, if the points form a closed, non-degenerate shape, the area will be non-zero.

Glosary

Polygons: Any closed shape with straight sides. (Triangles, quadrilaterals, pentagons, etc.)
Quadrilaterals: A specific type of polygon with four sides. (Squares, rectangles, trapezoids, parallelograms, etc.)
Parallelograms: A special subset of quadrilaterals where opposite sides are parallel and equal in length. (Rectangles, squares, rhombuses, etc.)