Annals of Optimization With Applications

www.anowa.reapress.com

Ann. Optim. Appl. Vol. 2, No. 1 (2026) 75–90.

Paper Type: Original Article

The Detect-Condition Algorithm for Robotic Inverse Kinematics and Common Root Analysis

Mahdi Dehghani Darmian*

Department of Basic Siences, Technical and Vocational University (TVU), Tehran, Iran; m.dehghanidarmian@gmail.com.

Citation:

Received: 10 October 2025 *Revised: 18 December 2025* *Accepted: 28 February 2026*	Dehghani Darmian, M. (2026). The detect-condition algorithm for robotic inverse kinematics and common root analysis. *Annals of optimization withapplications*, *2*(1), 75-90.

Abstract

CThis paper presents the Detect-Condition algorithm, a novel method based on Grobner systems for identifying conditions on the parameters of polynomial systems to determine their common roots. The power of this algrithm is demonstrated through a central application: automating the inverse kinematics (IK) for a planar 3R robotic manipulator. We translate the IK problem, finding the joint angles for a desired end-effector pose, into a system of parametric polynomials. The algorithm then partitions the parameter space (link lengths and target pose) to derive the exact algebraic conditions for solutions to exist. Specifically, it symbolically computes the robot's workspace boundary and provides closed-form expressions for the joint angles. Implemented in Mapleand validated with a numerical example, this approach offers a complete, symbolic, and automated solution. The results provide significant advantages over iterative numerical methods, including guaranteed precision and deep insight into kinematic constraints, establishing the algorithm as a powerful tool for robotic design and analysis.

Keywords: Detect-Condition algorithm, Inverse Kinematics, Parametric polynomials, Grobner systems, 3R Robot, Workspace analysis.

1 | Introduction

Identifying common roots of polynomial equations is a fundamental challenge with far-reaching implications across mathematics and engineering. A particularly compelling application arises in robotics, specifically in solving the inverse kinematics (IK) problem. The IK problem for a robotic manipulator—determining the joint angles that place theend-effector at a desired pose—can be formulated as a system of polynomial equations.

Corresponding Author: m.dehghanidarmian@ipm.ir

https://doi.org/10.48314/anowa.v2i1.68

A solution to this system corresponds to a *common root* that satisfies all equations simultaneously, representing a viable robot configuration. Thus, the search for common roots is not merely an abstract algebraic exercise but is directly tied to enabling precise robotic control and automation. Establishing the conditions for the existence of these roots reveals the fundamental geometric constraints of the mechanism. While the robotic example provides a powerful motivation, the general mathematical problem of identifying conditions and relationships between coefficients to determine common roots of polynomial equations remains challenging, especially as the number and degree of equations grow. Manual derivation becomes infeasible for complex systems.
For further illustration, let's assume two different quadratic polynomial equations, namely

$$p_1x^2 + q_1x + r_1 = 0 \text{ and } p_2x^2 + q_2x + r_2 = 0.$$

We aim to manually identify the conditions under which these two equations share a single root or two common roots. We assume a common root, denoted as α, meaning that substituting α into both equations yields zero. Thus, by replacing x with α, we obtain:

$$p_1\alpha^2 + q_1\alpha + r_1 = 0 \text{ and } p_2\alpha^2 + q_2\alpha + r_2 = 0.$$

Using the Cramer's rule, we can solve the above two equations.

$$\frac{\alpha^2}{\begin{vmatrix} q_1 & r_1 \\ q_2 & r_2 \end{vmatrix}} = \frac{-\alpha}{\begin{vmatrix} p_1 & r_1 \\ p_2 & r_2 \end{vmatrix}} = \frac{1}{\begin{vmatrix} p_1 & q_1 \\ p_2 & q_2 \end{vmatrix}}$$

$$\Rightarrow \frac{\alpha^2}{q_1r_2 - q_2r_1} = \frac{-\alpha}{p_1r_2 - p_2r_1} = \frac{1}{p_1q_2 - p_2q_1}.$$

Equating the first and second equivalents to the third one, we obtain,

$$\alpha^2 = \frac{q_1r_2 - q_2r_1}{p_1q_2 - p_2q_1}, \alpha = \frac{-(p_1r_2 - p_2r_1)}{p_1q_2 - p_2q_1}$$

$$\Rightarrow (p_1r_2 - p_2r_1)^2 = (q_1r_2 - q_2r_1)(p_1q_2 - p_2q_1).$$

The equation obtained indicates the condition for a common root between two quadratic equations. We will now establish the criteria for having two common roots, α and β, and derive the corresponding formula.

For the first equation, Vieta's formulas [26] yield the sum and product of its roots as $\alpha + \beta = \frac{-q_1}{p_1}$ and $\alpha \cdot \beta = \frac{r_1}{p_1}$. For the second equation, the expressions are $\alpha + \beta = \frac{-q_2}{p_2}$ and $\alpha \cdot \beta = \frac{r_2}{p_2}$. Since the roots are equal, the sums and products for both equations must also be equal. By setting the sums and products equal, we obtain:

$$\frac{-q_1}{p_1} = \frac{-q_2}{p_2} \text{ and } \frac{r_1}{p_1} = \frac{r_2}{p_2}.$$

Thus, the condition for having two common roots is:

$$\frac{p_1}{p_2} = \frac{q_1}{q_2} = \frac{r_1}{r_2}.$$

Deriving conditions for more equations or higher degrees quickly becomes intractable. This complexity underscores the need for an automated algorithmic approach. In light of these challenges, this article presents the DETECT-CONDITION algorithm, which utilizes Gröbner systems—a powerful computer algebra tool—to automatically partition the parameter space and derive the exact algebraic conditions for the existence of common roots. The algorithm is general but finds a potent application in solving the IK problem for a planar 3R robot, symbolically deriving its workspace boundary and providing closed-form solutions for joint angles.

2|Gröbner Systems

The primary challenge associated with the manual method arises particularly when faced with a situation where two equations possess different degrees. Moreover, as we consider an increasing number of polynomial equations coupled with a higher degree for each of these equations, the task of verifying the condition for common roots through manual calculations becomes increasingly intricate and time-consuming. The complexity escalates with each additional equation and each degree rise, making it a daunting task for anyone trying to perform these verifications by hand. In light of these challenges, this article is dedicated to finding solutions to both issues in an automated fashion by utilizing a powerful computer algebra tool known as Gröbner systems. To provide a comprehensive understanding of Gröbner systems, we will begin with a review of the essential concept of Gröbner bases, as discussed in [1, 4]. This preliminary discussion will set the stage for a deeper exploration of how Gröbner systems can effectively address the challenges presented by the need for common root verification in polynomial equations.

Let $R = \mathbb{K}[\mathbf{x}]$ be the polynomial ring over the field $\mathbb{K}$ in variables $\mathbf{x} = x_1, \ldots, x_n$. We denote a monomial $x_1^{\alpha_1} \cdots x_n^{\alpha_n} \in R$ as $\mathbf{x}^\alpha$, where $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ comprises non-negative integers. A monomial order $\prec$ on R is a total order on all monomials with two properties:

1) For all $\alpha, \beta, \gamma \in \mathbb{N}^n$, if $\mathbf{x}^\alpha \prec \mathbf{x}^\beta$, then $\mathbf{x}^{\alpha+\gamma} \prec \mathbf{x}^{\beta+\gamma}$.

2) The constant monomial is the smallest; specifically, $1 \prec \mathbf{x}^\alpha$ for all $\alpha \in \mathbb{N}^n$.

A classic example of a monomial ordering is the lexicographical order, denoted $\prec_{lex}$, where $\mathbf{x}^\beta \prec_{lex} \mathbf{x}^\alpha$ if the left-most non-zero entry of $\alpha - \beta$ is positive.

For $f \in R$ and a monomial order $\prec$, the leading monomial of f is the greatest monomial (with respect to $\prec$) in f, denoted $\mathrm{LM}(f)$. The leading monomial ideal of $I = \langle f_1, \ldots, f_m \rangle \subset R$ is defined as $\mathrm{LM}(I) = \langle \mathrm{LM}(f) | f \in I \rangle$. A finite subset $\{g_1, \ldots, g_k\} \subset I$ is a Gröbner basis for I with respect to $\prec$ if $\mathrm{LM}(I) = \langle \mathrm{LM}(g_1), \ldots, \mathrm{LM}(g_k) \rangle$.

Gröbner bases, introduced by Bruno Buchberger in 1965 along with Buchberger's algorithm [3], are named after his advisor Wolfgang Gröbner. Buchberger also proposed criteria to eliminate unnecessary S-polynomials to enhance the algorithm's efficiency [2]. However, due to its high complexity, Buchberger's algorithm may not always be practical. In 1983, Lazard described an algorithm for computing Gröbner bases using linear algebra techniques [20]. Faugère later introduced F_4 and F_5 algorithms in 1999 and 2002, respectively [9, 10]. Gao et al. then presented the incremental G2V algorithm [11], an efficient variant of F_5, and later introduced the GVW algorithm [12], a new framework for computing Gr¨ bases of ideals.

Gröbner bases have widespread applications across mathematics, science, and engineering. Their extension is vital for addressing practical problems involving systems of polynomial equations with parameters, playing significant roles in areas such as parametric linear algebra [5, 7, 16], automated geometry theorem proving [22], robotics [21, 23], algebraic geometry [13,22,23,27], and electrical networks [23,24], and more. These applications often require repeated analysis with varying parameter values. Gröbner systems extend the concept of Gröbner bases from polynomial ideals with numerical coefficients to those with parametric coefficients. A Gröbner system consists of a finite set of triples, each containing parametric constraints and a parametric polynomial set. For any specialization of parameters, a corresponding branch exists that satisfies the assignment, and the specialized polynomial set forms a Gröbner basis for the parametric ideal after parameter substitution. We will briefly overview parametric Gröbner systems and the existing literature on this subject.

In this article, we consider $S = \mathbb{K}[\mathbf{a}, \mathbf{x}]$, where $\mathbb{K}$ is an arbitrary field, $\mathbf{a} = a_1, \ldots, a_m$ is a sequence of parameters, and $\mathbf{x} = x_1, \ldots, x_n$ is a sequence of variables. Let $\prec_{\mathbf{x}}$ and $\prec_{\mathbf{a}}$ be monomial orders on variables and parameters, respectively. The combination of these orders defines a new ordering on S, denoted $\prec_{\mathbf{x},\mathbf{a}}$: for all $\alpha, \beta \in \mathbb{N}^n$ and $\gamma, \delta \in \mathbb{N}^m$, we have $\mathbf{x}^\alpha \mathbf{a}^\gamma \prec_{\mathbf{x},\mathbf{a}} \mathbf{x}^\beta \mathbf{a}^\delta$ if $\mathbf{x}^\alpha \prec_{\mathbf{x}} \mathbf{x}^\beta$, or $\mathbf{x}^\alpha = \mathbf{x}^\beta$ and $\mathbf{a}^\gamma \prec_{\mathbf{a}} \mathbf{a}^\delta$.

Definition 2.1. Let $F \subset S = \mathbb{K}[\mathbf{a}, \mathbf{x}]$ be a finite set, and $G = \{(N_i, W_i, G_i)\}_{i=1}^{\ell}$ be a finite triple set where $N_i, W_i \subset \mathbb{K}[\mathbf{a}]$ and $G_i \subset S$. The set G is a Gröbner system for $\langle F \rangle$

with respect to $\prec_{\mathbf{x},\mathbf{a}}$ if, for any i and homomorphism $\sigma : \mathbb{K}[\mathbf{a}] \to \mathbb{K}' \supseteq \mathbb{K}$, the following hold:

I. $\sigma(G_i) \subset \mathbb{K}'[\mathbf{x}]$ is a Gröbner basis for $\sigma(\langle F\rangle) \subset \mathbb{K}'[\mathbf{x}]$ with respect to $\prec_{\mathbf{x}}$.
II. $\sigma(p) = 0$ for all $p \in N_i \subset \mathbb{K}[\mathbf{a}]$ and $\sigma(q) \neq 0$ for all $q \in W_i \subset \mathbb{K}[\mathbf{a}]$.

Obner system concept was first introduced by Weispfenning in 1992, who proved that any parametric polynomial ideal has a finite Gröbner system and provided the initial computation algorithm [27]. In 1997, M. Kalkbrenner proposed key criteria regarding the specialization of parametric polynomial ideals and the stability of Gröbner bases under specialization [**18**]. Later, in 2002, Montes developed a more efficient algorithm called DisPGB for computing Gröbner systems [23]. This algorithm builds upon Buchberger's algorithm. In 2013, Hashemi et al. integrated two Buchberger criteria into DisPGB to enhance its efficiency [**15**]. Additionally, in 2006, Manubens and Montes improved the DisPGB algorithm through the concept of discriminant ideals [**21**]. Following Kalkbrenner's stability criterion, Suzuki and Sato created an effective Suzuki-Sato algorithm, for computing Gröbner systems [**25**]. In 2010, Kapur-Sun-Wang developed an efficient algorithm, PGBMain or the KSW algorithm, based on the Weispfenning and Suzuki-Sato algorithms [**19**]. This algorithm utilizes a minimal Dickson basis to streamline computations and includes radical membership checks for non-null parametric conditions, thus reducing unnecessary branches in the output.
Recently, Dehghani et al. further advanced the computation of Gröbner systems by exploring parametric polynomial ideals and creating optimized algorithms in this field [**7**, **8**, **13**, **14**, **17**]. In 2024, Dehghani introduced the efficient GES-GVW-CGS algorithm for computing these systems [**6**].

Example 2.2. Let $F = \{(a-b)xy + (b-1)y, x^2 + (2-b)y\} \subset \mathbb{K}[a, b, x, y]$, where x, y are variables and a, b are parameters. A Gröbner system of F with respect to the product of monomial orderings $y \prec_{drl} x$ and $c \prec_{drl} b \prec_{drl} a$ via our `Maple` implementation of the GES-GVW-CGS algorithm is as follows:

$$\begin{cases} ([b-1, a-1], [\,], & \{x^2 + y\}) \\ ([b-2, a-2], [\,], & \{y, x^2\}) \\ ([b-a], [b-2, b-1], & \{by - y, x^2\}) \\ ([b-2], [a-2], & \{axy - 2xy + y, x^2, y\}) \\ ([\,], [a-b, b-2], & \{axy - bxy + by - y, -by + x^2 + 2y, \\ & \quad (2b^2 - a^2b + 2ab^2 - b^3 + 2a^2 - 4ab)y^2 + (b^2 - 2b + 1)y\}). \end{cases}$$

This obner system consists of five triples. By setting $a = 3$ and $b = 2$, the fourth branch corresponds to these values, yielding $\{xy + y, x^2, y\}$ as a Gröbner basis for the ideal $\langle F\rangle|_{a=3,b=2}$.

3|Conditions System and Detect-Condition Algorithm

This section identifies the conditions for common roots of multiple polynomial equations and introduces a *conditions system* for these equations, along with the DetectCondition algorithm for its computation. We employ Gröbner systems to divide the parameter space into a finite number of cells, each corresponding to a set of polynomials that define the relationships between the coefficients. By choosing specific parameter values, we pinpoint the relevant cell and its associated polynomials. Identifying the conditions for common roots of polynomial equations is essential in algebraic geometry, facilitating computations and enhancing insight into their structures.

Definition 3.1. Let $I = \langle f_1, \ldots, f \rangle \subset S = \mathbb{K}[\mathbf{a}][x]$ be an ideal including univariate polynomial equations. We call $V = \{(N_i, W_i, V_i)\}_{i=1}^{\ell}$ a *conditions system* of I if for each i and any specialization $\sigma : \mathbb{K}[\mathbf{a}] \to \mathbb{K}' \supseteq \mathbb{K}$ satisfying the parametric constraint (N_i, W_i), the set $V_i \subset \mathbb{K}'$ is the common roots of $\sigma(f_1), \ldots, \sigma(f_t)$.

This indicates that all triples encompass the pairs (N_i, W_i), which contain parametric conditions based on the number of common roots of $f_1 = 0, \ldots, f_t = 0$. Based on this definition, we present the simple Detect-Condition algorithm for computing a conditions system. Detect-Condition algorithm receives as input a finite list of positive integers $[n_1, \ldots, n_t]$, the degree of univariate polynomial equations $f_1 = \sum_{i=0}^{n_1} a_{1_i} x^i = 0, \ldots, f_t = \sum_{i=0}^{n_t} a_{t_i} x^i = 0$, and returns a conditions system of $\langle f_1, \ldots, f_t \rangle$.

Algorithm 1 Detect-Condition

Require: $A = [n_1, \ldots, n_t]$ is a finite list of degrees of univariate polynomial equations
Ensure: A conditions system of $\langle f_1, \ldots, f_t \rangle$
  $F := \{\ \}$
  for j from 1 to t **do**
    $f_j := \sum_{i=0}^{n_j} a_{j_i} x^i$
    add f_j into F
  end for
  $V := \{\ \}$
  $I :=$ an ideal generated by the elements of F
  $G := \{(N_i, W_i, G_i)\}_{i=1}^{\ell}$ a Gröbner system of I with respect to $\prec_{x,\mathbf{a}}$
  for $(N_i, W_i, G_i) \in G$ **do**
    $V_i :=$ the solution of G_i
    $V := V \bigcup \{(N_i, W_i, V_i)\}$
  end for
  Return (V)

Theorem 3.2. *The* Detect-Condition *algorithm terminates in a finite number of steps and correctly computes a conditions system.*

Proof. The termination of the Detect-Condition algorithm is ensured by the termination of Gröbner system computations. Its correctness is based on established methods for solving polynomial systems. Specifically, consider the triple (N_i, W_i, G_i) representing the Gröbner system F at the i-th step of the second **for**-loop. Since all parameters in G_i are non-zero, we can apply standard solution methods for G_i. Thus, any V_i is a solution of F that satisfies the parametric conditions (N_i, W_i).

The following examples are computed using our `Maple` implementation of the Detect-Condition algorithm: first, a simple case of finding common roots for two different quadratic polynomials, and second, a more complex scenario involving conditions for three equations consisting of linear, quadratic, and cubic polynomials.

Example 3.3. Let us consider $A = [2, 2]$ as the input of the Detect-Condition algorithm. this means that the following two different quadratic polynomial equations are created.

$f_1 = p_1 x^2 + q_1 x + r_1$ and $f_2 = p_2 x^2 + q_2 x + r_2$.

Thus, an ideal I is formed $I = \langle p_1x^2+q_1x+r_1, p_2x^2+q_2x+r_2\rangle$ and then a Gröbner system of I is calculated w.r.t. $r_2 \prec_{lex} r_1 \prec_{lex} q_2 \prec_{lex} q_1 \prec_{lex} p_2 \prec_{lex} p_1 \prec_{lex} x$ when $p_1, p_2 \neq 0$

$$G = \begin{cases} ([(p_1r_2 - p_2r_1)^2 - (q_1r_2 - q_2r_1)(p_1q_2 - p_2q_1)], [p_1, p_2, p_1q_2 - p_2q_1], & [xp_1q_2 - xp_2q_1 + p_1r_2 - p_2r_1]) \\ ([p_2q_1r_2 - p_2q_2r_1, p_1r_2 - p_2r_1, p_1q_2 - p_2q_1], [p_1, p_2], & [x^2p_1 + xq_1 + r_1]) \\ ([\,], [p_1, p_2, p_1q_2 - p_2q_1, (p_1r_2 - p_2r_1)^2 - (q_1r_2 - q_2r_1)(p_1q_2 - p_2q_1)], & [1]) \\ ([p_1q_2 - p_2q_1], [p_1, p_2, p_1r_2 - p_2r_1], & [1]) \end{cases}$$

Now, we have a conditions system of $I = \langle p_1x^2 + q_1x + r_1, p_2x^2 + q_2x + r_2\rangle$

$$V = \begin{cases} ([(p_1r_2 - p_2r_1)^2 - (q_1r_2 - q_2r_1)(p_1q_2 - p_2q_1)], [p_1, p_2, p_1q_2 - p_2q_1], & \{x = \frac{p_2r_1 - p_1r_2}{p_1q_2 - p_2q_1}\}) \\ ([p_2q_1r_2 - p_2q_2r_1, p_1r_2 - p_2r_1, p_1q_2 - p_2q_1], [p_1, p_2], & \{x_1 = \frac{-q_1 + \sqrt{q_1^2 - 4p_1r_1}}{2p_1}, \\ & x_2 = \frac{-q_1 - \sqrt{q_1^2 - 4p_1r_1}}{2p_1}\}) \\ ([\,], [p_1, p_2, p_1q_2 - p_2q_1, (p_1r_2 - p_2r_1)^2 - (q_1r_2 - q_2r_1)(p_1q_2 - p_2q_1)], & \{\text{no common root}\}) \\ ([p_1q_2 - p_2q_1], [p_1, p_2, p_1r_2 - p_2r_1], & \{\text{no common root}\}) \end{cases}$$

For instance, if $p_1 = -6, p_2 = 3, q_1 = \frac{27}{5}, q_2 = \frac{1}{2}, r_1 = -\frac{6}{5}, r_2 = -1$ then the first branch corresponds to these values of parameters (these values satisfy (N_1, W_1)). Therefore, substituting these parameters' values into $V_1 = \{\frac{p_2r_1 - p_1r_2}{p_1q_2 - p_2q_1}\}$ we have $V_1 = \{\frac{1}{2}\}$. So, according to parametric conditions (N_1, W_1) the common root is $\frac{1}{2}$. Now, let $p_1 = \sqrt{2}, p_2 = -1, q_1 = -\sqrt{3}, q_2 = \frac{\sqrt{6}}{2}, r_1 = -1, r_2 = \frac{\sqrt{2}}{2}$ which hold the conditions pair (N_2, W_2). By setting these values in V_2 we have the following common roots for f_1 and f_2 under this specialization:

$$\left\{x_1 = \frac{1}{4}\sqrt{2}(\sqrt{3} + \sqrt{3 + 4\sqrt{2}}\,)\right\}, \left\{x_2 = \frac{1}{4}\sqrt{2}(\sqrt{3} - \sqrt{3 + 4\sqrt{2}}\,)\right\}$$

This means that for any $1 \leq i \leq 4$ and each specialization σ holding (N_i, W_i) we obtain conditions for common roots of quadratic polynomial equations, automatically. According to the above system, no common root exists in the third and fourth condition sets.

Next is a more complex example of identifying common roots among three polynomial equations: linear, quadratic, and cubic.

Example 3.4. Let us consider $A = [1, 2, 3]$ as the input of the Detect-Condition algorithm. This means that the following three polynomial equations are formed.

$f_1 = a_1x^3 + b_1x^2 + c_1x + d_1$ and $f_2 = a_2x^2 + b_2x + c_2$ and $f_3 = a_3x + b_3$.

Using our `Maple` implementation of the GES-GVW-CGS algorithm, we obtained the Gröbner system of $\langle f_1, f_2, f_3\rangle$ w.r.t. $d_1 \prec_{lex} c_3 \prec_{lex} c_2 \prec_{lex} b_3 \prec_{lex} b_2 \prec_{lex} b_1 \prec_{lex} a_3 \prec_{lex} a_2 \prec_{lex} a_1 \prec_{lex} x$ when $a_1, a_2, a_3 \neq 0$. This Gröbner system consists of the following four branches, excluding those containing the Gröbner basis $\{1\}$.

$N_1 = [a_2b_3^2 + a_3^2c_2 - a_3b_2b_3, a_1b_3c_2^2 + a_2a_3c_2d_1 + a_2b_2b_3d_1 - a_2b_3c_1c_2 - a_3b_1c_2^2 - a_3b_2^2d_1 + a_3b_2c_1c_2, a_1b_3^2c_2 + a_2a_3b_3d_1 - a_3^2b_2d_1 + a_3^2c_1c_2 - a_3b_1b_3c_2, a_1b_3^3 - a_3^3d_1 + a_3^2b_3c_1 - a_3b_1b_3^2, a_1a_3c_2^2 - a_1b_2b_3c_2 - a_2^2b_3d_1 + a_2a_3b_2d_1 - a_2a_3c_1c_2 + a_2b_1b_3c_2, a_1a_3b_3c_2 - a_1b_2b_3^2 + a_2a_3^2d_1 - a_2a_3b_3c_1 - a_3^2b_1c_2 + a_3b_1b_2b_3, a_1a_2b_3c_2 + a_1a_3b_2c_2 - a_1b_2^2b_3 + a_2^2a_3d_1 - a_2^2b_3c_1 - a_2a_3b_1c_2 + a_2b_1b_2b_3, a_1^2c_2^3 + 3a_1a_2b_2c_2d_1 - 2a_1a_2c_1c_2^2 - a_1b_1b_2c_2^2 - a_1b_2^2d_1 + a_1b_2^2c_1c_2 + a_2^2d_1^2 - 2a_2^2b_1c_2d_1 - a_2^2b_2c_1d_1 + a_2^2c_1^2c_2 + a_2b_1^2c_2^2 + a_2b_1b_2^2d_1 - a_2b_1b_2c_1c_2],$

$W_1 = [a_1, a_2, a_3, a_1a_2c_2 - a_1b_2^2 - a_2^2c_1 + a_2b_1b_2],$

$G_1 = [xa_1a_2c_2 - xa_1b_2^2 - xa_2^2c_1 + xa_2b_1b_2 - a_1b_2c_2 - a_2^2d_1 + a_2b_1c_2]$

$N_2 = [a_3^2c_2^2d_1 - a_3b_2b_3c_2d_1 + b_1b_3^2c_2^2 + b_2^2b_3^2d_1 - b_2b_3^2c_1c_2, a_2c_2d_1 - b_1c_2^2 - b_2^2d_1 + b_2c_1c_2, a_2b_3^2 + a_3^2c_2 - a_3b_2b_3, a_1c_2^2 + a_2b_2d_1 - a_2c_1c_2, a_1b_3^2c_2 + a_2a_3b_3d_1 - a_3^2b_2d_1 + a_3^2c_1c_2 - a_3b_1b_3c_2, a_1b_3^3 - a_3^3d_1 + a_3^2b_3c_1 - a_3b_1b_3^2, a_1b_2c_2 + a_2^2d_1 - a_2b_1c_2, a_1b_2^2b_3^2 + a_2^2a_3b_3d_1 - a_2a_3^2b_2d_1 - a_2a_3b_1b_3c_2 + a_2a_3b_2b_3c_1 + a_3^2b_1b_2c_2 - a_3b_1b_2^2b_3, a_1b_3^2 + a_3^2d_1 - a_2^2b_1c_2 + a_2^2b_2c_1 - a_2b_1b_3^2, a_1a_3b_3c_2 - a_1b_2b_3^2 + a_2a_3^2d_1 - a_2a_3b_3c_1 - a_3^2b_1c_2 + a_3b_1b_2b_3, a_1a_2c_2 - a_1b_2^2 - a_2^2c_1 + a_2b_1b_2],$

$W_2 = [a_1, a_2, a_3, b_2, b_3],$

$G_2 = [xa_1b_2b_3 + a_1b_3c_2 + a_2a_3d_1 - a_2b_3c_1 - a_3b_1c_2 + b_1b_2b_3]$

$N_3 = [b_1b_2^2b_3c_2^2d_1 - b_1b_2b_3c_1c_2^3 + b_2^4b_3d_1^2 - 2b_2^3b_3c_1c_2d_1 + b_2^2b_3c_1^2c_2^2, a_3b_2^2c_2d_1 - a_3b_2c_1c_2^2 - b_2^3b_3d_1 + b_2^2b_3c_1c_2, a_3b_2^3d_1^2 - a_3b_2c_1^2c_2^2 + b_1b_2^2b_3c_2d_1 - b_1b_2b_3c_1c_2^2 - 2b_2^3b_3c_1d_1 + 2b_2^2b_3c_1^2c_2, a_3^2c_2^2d_1 - a_3b_2b_3c_2d_1 + b_1b_3^2c_2^2 + b_2^2b_3^2d_1 - b_2b_3^2c_1c_2, a_3^2b_2d_1 - a_3b_2b_3c_1 + b_1b_2b_3^2, a_3^2b_2c_1c_2^2 + a_3b_2^2b_3d_1 - 2a_3b_2^2b_3c_1c_2 + b_1b_2^2b_3^2c_2, a_2c_2d_1 - b_1c_2^2 - b_2^2d_1 + b_2c_1c_2, a_2b_2^2 + a_3^2c_2 - a_3b_2b_3, a_2b_2b_3d_1 - a_3b_2^2d_1 + a_3b_2c_1c_2 - b_1b_2b_3c_2, a_2b_2b_3c_1c_2 - a_3b_2^2d_1 + a_3b_2^2c_1c_2 - b_1b_2^2b_3c_2, a_2a_3b_2d_1 - a_2b_2b_3c_1 - a_3b_1b_2c_2 + b_1b_2^2b_3, a_2a_3b_2c_1c_2 - a_2b_2^2b_3c_1 - a_3b_1b_2^2c_2 + b_1b_2^3b_3, a_2^2b_3d_1 - a_2b_1b_3c_2, a_2^2b_2b_3c_1 - a_2b_1b_2^2b_3, a_2^2a_3d_1 - a_2a_3b_1c_2, a_1c_2^2 + a_2b_2d_1 - a_2c_1c_2, a_1b_3c_2 + a_2a_3d_1 - a_2b_3c_1 - a_3b_1c_2 + b_1b_2b_3, a_1b_3^2 - a_3^2d_1 + a_3^2b_3c_1 - a_3b_1b_3^2, a_1b_2c_2 + a_2^2d_1 - a_2b_1c_2, a_1b_2b_3, a_1b_2^3 + a_3^2d_1 - a_2^2b_1c_2 + a_2^2b_2c_1 - a_2b_1b_2^2, a_1a_3b_2d_1, a_1a_2c_2 - a_1b_2^2 - a_2^2c_1 + a_2b_1b_2]$,

$W_3 = [a_1, a_3, b_3, c_2]$,

$G_3 = [xa_1b_3^2 + a_3^2d_1 - a_3b_3c_1 + b_1b_3^2]$

$N_4 = [b_1b_3c_2^2 + b_2^2b_3d_1 - b_2b_3c_1c_2, a_3c_2 - b_2b_3, a_3b_2d_1 + b_1b_3c_2 - b_2b_3c_1, a_3^2d_1 - a_3b_3c_1 + b_1b_3^2, a_2c_2d_1 - b_1c_2^2 - b_2^2d_1 + b_2c_1c_2, a_2b_3, a_2a_3d_1, a_1c_2^2 + a_2b_2d_1 - a_2c_1c_2, a_1b_3c_2, a_1b_3^2, a_1b_2c_2 + a_2^2d_1 - a_2b_1c_2, a_1b_2b_3, a_1b_3^2 + a_3^2d_1 - a_2^2b_1c_2 + a_2^2b_2c_1 - a_2b_1b_2^2, a_1a_3b_3d_1, a_1a_3b_2^2 + a_2^2a_3c_1 - a_2a_3b_1b_2, a_1a_2c_2 - a_1b_2^2 - a_2^2c_1 + a_2b_1b_2]$,

$W_4 = [a_1, a_2, a_3]$,

$G_4 = [xa_3 + b_3]$

Next we have the condition system of $\langle f_1, f_2, f_3\rangle$ utilizing our `Maple` implementation of the DETECT-CONDITION algorithm:

$N_1 = [a_2b_3^2 + a_3^2c_2 - a_3b_2b_3, a_1b_3c_2^2 + a_2a_3c_2d_1 + a_2b_2b_3d_1 - a_2b_3c_1c_2 - a_3b_1c_2^2 - a_3b_2^2d_1 + a_3b_2c_1c_2, a_1b_3^2c_2 + a_2a_3b_3d_1 - a_3^2b_2d_1 + a_3^2c_1c_2 - a_3b_1b_3c_2, a_1b_3^3 - a_3^3d_1 + a_3^2b_3c_1 - a_3b_1b_3^2, a_1a_3c_2^2 - a_1b_2b_3c_2 - a_2^2b_3d_1 + a_2a_3b_2d_1 - a_2a_3c_1c_2 + a_2b_1b_3c_2 - a_2^2a_3d_1 - a_2a_3^2b_3c_2 - a_1b_2b_3^2 + a_2a_3^2d_1 - a_2a_3b_3c_1 - a_3^2b_1c_2 + a_3b_1b_2b_3, a_1a_2b_3c_2 + a_1a_3b_2c_2 - a_1b_2^2b_3 + a_2^2a_3d_1 - a_2^2b_3c_1 - a_2a_3b_1c_2 + a_2b_1b_2b_3, a_1^2c_2^3 + 3a_1a_2b_2c_2d_1 - 2a_1a_2c_1c_2^2 - a_1b_1b_2c_2^2 - a_1b_2^2d_1 + a_1b_2^2c_1c_2 + a_2^3d_1^2 - 2a_2^2b_1c_2d_1 - a_2^2b_2c_1d_1 + a_2^2c_1^2c_2 + a_2b_1^2c_2^2 + a_2b_1b_2^2d_1 - a_2b_1b_2c_1c_2]$,

$W_1 = [a_1, a_2, a_3, a_1a_2c_2 - a_1b_2^2 - a_2^2c_1 + a_2b_1b_2]$,

$V_1 = \{x = \frac{(a_1b_2c_2 + a_2^2d_1 - a_2b_1c_2)}{(a_1a_2c_2 - a_1b_2^2 - a_2^2c_1 + a_2b_1b_2)}\}$

$N_2 = [a_3^2c_2^2d_1 - a_3b_2b_3c_2d_1 + b_1b_3^2c_2^2 + b_2^2b_3^2d_1 - b_2b_3^2c_1c_2, a_2c_2d_1 - b_1c_2^2 - b_2^2d_1 + b_2c_1c_2, a_2b_3^2 + a_3^2c_2 - a_3b_2b_3, a_1c_2^2 + a_2b_2d_1 - a_2c_1c_2, a_1b_3^2c_2 + a_2a_3b_3d_1 - a_3^2b_2d_1 + a_3^2c_1c_2 - a_3b_1b_3c_2, a_1b_3^3 - a_3^3d_1 + a_3^2b_3c_1 - a_3b_1b_3^2, a_1b_2c_2 + a_2^2d_1 - a_2b_1c_2, a_1b_2^2b_3^2 + a_2^2a_3b_3d_1 - a_2a_3^2b_2d_1 - a_2a_3b_1b_3c_2 + a_2a_3b_2b_3c_1 + a_3^2b_1b_2c_2 - a_3b_1b_2^2b_3, a_1b_2^3 + a_3^2d_1 - a_2^2b_1c_2 + a_2^2b_2c_1 - a_2b_1b_2^2, a_1a_3b_3c_2 - a_1b_2b_3^2 + a_2a_3^2d_1 - a_2a_3b_3c_1 - a_3^2b_1c_2 + a_3b_1b_2b_3, a_1a_2c_2 - a_1b_2^2 - a_2^2c_1 + a_2b_1b_2]$,

$W_2 = [a_1, a_2, a_3, b_2, b_3]$,

$V_2 = \{x = \frac{-(a_1b_3c_2 + a_2a_3d_1 - a_2b_3c_1 - a_3b_1c_2 + b_1b_2b_3)}{(a_1b_2b_3)}\}$

$N_3 = [b_1b_2^2b_3c_2^2d_1 - b_1b_2b_3c_1c_2^3 + b_2^4b_3d_1^2 - 2b_2^3b_3c_1c_2d_1 + b_2^2b_3c_1^2c_2^2, a_3b_2^2c_2d_1 - a_3b_2c_1c_2^2 - b_2^3b_3d_1 + b_2^2b_3c_1c_2, a_3b_2^3d_1^2 - a_3b_2c_1^2c_2^2 + b_1b_2^2b_3c_2d_1 - b_1b_2b_3c_1c_2^2 - 2b_2^3b_3c_1d_1 + 2b_2^2b_3c_1^2c_2, a_3^2c_2^2d_1 - a_3b_2b_3c_2d_1 + b_1b_3^2c_2^2 + b_2^2b_3^2d_1 - b_2b_3^2c_1c_2, a_3^2b_2d_1 - a_3b_2b_3c_1 + b_1b_2b_3^2, a_3^2b_2c_1c_2^2 + a_3b_2^2b_3d_1 - 2a_3b_2^2b_3c_1c_2 + b_1b_2^2b_3^2c_2, a_2c_2d_1 - b_1c_2^2 - b_2^2d_1 + b_2c_1c_2, a_2b_2^2 + a_3^2c_2 - a_3b_2b_3, a_2b_2b_3d_1 - a_3b_2^2d_1 + a_3b_2c_1c_2 - b_1b_2b_3c_2, a_2b_2b_3c_1c_2 - a_3b_2^2d_1 + a_3b_2^2c_1c_2 - b_1b_2^2b_3c_2, a_2a_3b_2d_1 - a_2b_2b_3c_1 - a_3b_1b_2c_2 + b_1b_2^2b_3, a_2a_3b_2c_1c_2 - a_2b_2^2b_3c_1 - a_3b_1b_2^2c_2 + b_1b_2^3b_3, a_2^2b_3d_1 - a_2b_1b_3c_2, a_2^2b_2b_3c_1 - a_2b_1b_2^2b_3, a_2^2a_3d_1 - a_2a_3b_1c_2, a_1c_2^2 + a_2b_2d_1 - a_2c_1c_2, a_1b_3c_2 + a_2a_3d_1 - a_2b_3c_1 - a_3b_1c_2 + b_1b_2b_3, a_1b_3^2 - a_3^2d_1 + a_3^2b_3c_1 - a_3b_1b_3^2, a_1b_2c_2 + a_2^2d_1 - a_2b_1c_2, a_1b_2b_3, a_1b_2^3 + a_3^2d_1 - a_2^2b_1c_2 + a_2^2b_2c_1 - a_2b_1b_2^2, a_1a_3b_2d_1, a_1a_2c_2 - a_1b_2^2 - a_2^2c_1 + a_2b_1b_2]$,

$W_3 = [a_1, a_3, b_3, c_2]$,

$V_3 = \{x = \frac{-(a_3^2d_1 - a_3b_3c_1 + b_1b_3^2)}{(a_1b_3^2)}\}$

$N_4 = [b_1b_3c_2^2 + b_2^2b_3d_1 - b_2b_3c_1c_2, a_3c_2 - b_2b_3, a_3b_2d_1 + b_1b_3c_2 - b_2b_3c_1, a_3^2d_1 - a_3b_3c_1 + b_1b_3^2, a_2c_2d_1 - b_1c_2^2 - b_2^2d_1 + b_2c_1c_2, a_2b_3, a_2a_3d_1, a_1c_2^2 + a_2b_2d_1 - a_2c_1c_2, a_1b_3c_2, a_1b_3^2, a_1b_2c_2 + a_2^2d_1 - a_2b_1c_2, a_1b_2b_3, a_1b_3^2 + a_3^2d_1 - a_2^2b_1c_2 + a_2^2b_2c_1 - a_2b_1b_2^2, a_1a_3b_3d_1, a_1a_3b_2^2 + a_2^2a_3c_1 - a_2a_3b_1b_2, a_1a_2c_2 - a_1b_2^2 - a_2^2c_1 + a_2b_1b_2]$,

$W_4 = [a_1, a_2, a_3]$,

$V_4 = \{x = \frac{-b_3}{a_3}\}$

Letting $a_1 = -2, a_2 = -1, a_3 = 2, b_1 = 1, b_2 = 2, b_3 = 3, c_1 = \frac{33}{2}, c_2 = \frac{21}{4}, d_1 = \frac{63}{4}$ satisfies the conditions of the pair (N_2, W_2). Substituting these values into

$$V_2 = \{x = -\frac{(a_1b_3c_2 + a_2a_3d_1 - a_2b_3c_1 - a_3b_1c_2 + b_1b_2b_3)}{(a_1b_2b_3)}\}$$

reveals the common root $V_2 = \{x = -\frac{3}{2}\}$ for f_1, f_2, and f_3 under this specialization.

Application in Robotic Inverse Kinematics

In robotics, a manipulator is composed of a series of rigid links connected by joints. The 3R manipulator, a classic and fundamental non-trivial structure, consists of three links connected by three revolute (rotational) joints. Determining the joint angles $\theta_1, \theta_2, \theta_3$ that place the end-effector at a specific coordinate (x, y, ϕ) in the 2D plane is known as the inverse kinematics problem. This problem is central to robot control and is inherently algebraic, making it an ideal application for the DETECT-CONDITION algorithm.

4.1.1 Denavit-Hartenberg (D-H) Convention. To systematically describe the robot's geometry, we use the Denavit-Hartenberg (D-H) convention. This method assigns a coordinate frame to each joint and defines four parameters to describe the transformation from one frame to the next:

- a_i: Link length – the distance along x_i from z_{i-1} to z_i
- α_i: Link twist – the angle around x_i from z_{i-1} to z_i
- d_i: Link offset – the distance along z_{i-1} from x_{i-1} to x_i
- θ_i: Joint angle – the angle around z_{i-1} from x_{i-1} to x_i

For our planar 3R robot, operating in the X-Y plane, the D-H parameters are simplified. All joints are revolute, so θ_i are the variables. The links lie in the plane, so all twist angles α_i are zero. We assume no link offsets ($d_i = 0$). The D-H parameter table is as follows:

- **Link 1**: $a_1 = L_1$, $\alpha_1 = 0$, $d_1 = 0$, $\theta_1 = \theta_1$
- **Link 2**: $a_2 = L_2$, $\alpha_2 = 0$, $d_2 = 0$, $\theta_2 = \theta_2$
- **Link 3**: $a_3 = L_3$, $\alpha_3 = 0$, $d_3 = 0$, $\theta_3 = \theta_3$

4.1.2 Homogeneous Transformation Matrices. The transformation from frame $\{i-1\}$ to frame $\{i\}$ is given by the homogeneous transformation matrix ${}^{i-1}_{i}T$:

$$ {}^{i-1}_{i}T = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i \sin\alpha_{i-1} \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i \cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1} $$

Given our parameters ($\alpha_i = 0, d_i = 0$), this simplifies to a series of planar rotations and translations:

$$ {}^{0}_{1}T = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & L_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & L_1\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2} $$

$$ {}^{1}_{2}T = \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & L_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & L_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3} $$

$$ {}^{2}_{3}T = \begin{pmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & L_3\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & L_3\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4} $$

The overall transformation from the base frame $\{0\}$ to the end-effector frame $\{3\}$ is found by multiplying these matrices:

$$ {}^0_3T = {}^0_1T \cdot {}^1_2T \cdot {}^2_3T \tag{5} $$

The position (x, y) and orientation ϕ of the end-effector are extracted from this matrix ${}^0_3T$:

$$ {}^0_3T = \begin{pmatrix} \cos\phi & -\sin\phi & 0 & x \\ \sin\phi & \cos\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6} $$

By performing the matrix multiplication ${}^0_3T = {}^0_1T \cdot {}^1_2T \cdot {}^2_3T$ and equating the result to the desired end-effector pose matrix, we derive our system of equations.
The $(1, 4)$ and $(2, 4)$ elements give the x and y coordinates:

$$ x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) + L_3 \cos(\theta_1 + \theta_2 + \theta_3) \tag{7} $$

$$ y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) + L_3 \sin(\theta_1 + \theta_2 + \theta_3) \tag{8} $$

The orientation ϕ is the sum of the joint angles for a planar robot:

$$ \phi = \theta_1 + \theta_2 + \theta_3 \tag{9} $$

This is a system of three equations (for x, y, and ϕ) in three unknowns (θ_1, θ_2, θ_3).
To convert this into an algebraic system suitable for the Detect-Condition algorithm, we use the standard half-angle substitution:

$$ t_i = \tan(\theta_i/2), \quad \sin\theta_i = \frac{2t_i}{1+t_i^2}, \quad \cos\theta_i = \frac{1-t_i^2}{1+t_i^2} \tag{10} $$

This substitution transforms the transcendental equations into a system of rational polynomials in t_1, t_2, t_3. Clearing the denominators yields a system of polynomial equations.

We now define the specific input for our algorithm. Let us fix the link lengths as symbolic parameters: $a = L_1, b = L_2, c = L_3$. The desired end-effector pose is also defined by parameters: $x = P_x$, $y = P_y$, $\phi = \Phi$. The variables are the half-angle tangents: $x_1 = t_1, x_2 = t_2, x_3 = t_3$.

After applying the half-angle substitution and clearing denominators, we obtain three polynomial equations f_1, f_2, f_3 and the input to the Detect-Condition algorithm is the list of degrees. The parameters of the system are $\mathbf{a} = (a, b, c, P_x, P_y, \Phi)$. and the variable is x_1. Using our `Maple` implementation of the Detect-Condition algorithm we compute the condition system for the ideal $I = \langle f_1, f_2, f_3 \rangle$ with respect to a lexicographic order that eliminates the variable, e.g., $\Phi \prec_{\text{lex}} P_y \prec_{\text{lex}} P_x \prec_{\text{lex}} c \prec_{\text{lex}} b \prec_{\text{lex}} a \prec_{\text{lex}} x_1$. It is worth noting that in the following system; $\Delta = 4a^2[(P_y - c\sin\Phi)^2 + (P_x - b\cos\Phi - c\sin\Phi)^2 - (b^2 + c^2 - P_y^2)]$.

$$
V = \left\{ \begin{array}{l}
\left(\begin{array}{l}
N_1 : \{(a^2+b^2+c^2-P_x^2-P_y^2)^2 - 4a^2(b^2+c^2-P_y^2) \\
\quad -4a^2(P_x - b\cos\Phi - c\sin\Phi)^2\} \\
W_1 : \{a, b, c, \quad P_x^2+P_y^2-a^2-b^2-c^2+2a(b\cos\Phi + c\sin\Phi), \\
\quad P_x^2+P_y^2-a^2-b^2-c^2-2a(b\cos\Phi + c\sin\Phi)\} \\
V_1 : \left\{ t_1 = \dfrac{2a(P_y - c\sin\Phi) \pm \sqrt{\Delta}}{P_x^2+P_y^2-a^2-b^2-c^2+2a(P_x - b\cos\Phi - c\sin\Phi)} \right\}
\end{array} \right), \\
\left(\begin{array}{l}
N_2 : \{P_x^2+P_y^2 = (a+b+c)^2\} \\
W_2 : \{a, b, c\} \\
V_2 : \left\{ t_1 = \dfrac{P_y}{P_x+a+b+c} \right\}
\end{array} \right), \\
\left(\begin{array}{l}
N_3 : \{P_x^2+P_y^2 - (a-b-c)^2\} \\
W_3 : \{a, b, c\} \\
V_3 : \left\{ t_1 = \dfrac{P_y}{P_x+a-b-c} \right\}
\end{array} \right), \\
\left(\begin{array}{l}
N_4 : \{P_x - b\cos\Phi - c\sin\Phi, P_y - b\sin\Phi + c\cos\Phi, a\} \\
W_4 : \{b, c\} \\
V_4 : \{t_1 \in \mathbb{R}\} \quad \text{(Infinite solutions)}
\end{array} \right), \\
\left(\begin{array}{l}
N_5 : \{\} \\
W_5 : \{(a^2+b^2+c^2-P_x^2-P_y^2)^2 - 4a^2(b^2+c^2-P_y^2) \\
\quad -4a^2(P_x - b\cos\Phi - c\sin\Phi)^2\} \\
V_5 : \{\} \quad \text{(No solution)}
\end{array} \right)
\end{array} \right.
$$

The algorithm returns a **conditions system** $V = \{(N_i, W_i, V_i)\}_{i=1}^{5}$. This system partitions the 6-dimensional parameter space $(a, b, c, P_x, P_y, \Phi)$ into regions. For each region, defined by the constraints (N_i, W_i), it provides the solution V_i for the common root x_1 (and from it, θ_1). A critically important branch is the one that defines the condition for a reachable pose inside the workspace. This branch's constraints (N_1, W_1) include the non-degeneracy conditions $a \neq 0, b \neq 0, c \neq 0$ and, most importantly, the **kinematic constraint equation**:

$$(P_x^2+P_y^2-a^2-b^2-c^2)^2 + 4a^2(b^2+c^2-P_y^2) - 4a^2(P_x - b\cos\Phi - c\sin\Phi)^2 = 0 \tag{11}$$

This equation **must hold** for the specified pose (P_x, P_y, Φ) to be reachable. It is the precise algebraic condition defining the boundary of the robot's workspace.
For parameters satisfying this and other non-degeneracy conditions, the solution set V_i provides the expression for the joint angle:

$$V_1 = \left\{ x_1 = t_1 = \frac{2a(P_y - c\sin\Phi) \pm \sqrt{\Delta}}{P_x^2+P_y^2-a^2-b^2-c^2+2a(P_x - b\cos\Phi - c\sin\Phi)} \right\} \tag{12}$$

This symbolic expression for $\tan(\theta_1/2)$ is the **common root** of the three polynomial equations f_1, f_2, f_3 under the specified parametric conditions.

Visualization of the Five Branches in 3R Robot Inverse Kinematics

The DETECT-CONDITION algorithm partitions the parameter space of the planar 3R robot into five distinct branches, each corresponding to different kinematic configurations. These branches represent the complete solution space for the inverse kinematics problem, providing both existential conditions and explicit solutions.

Branch 1: General Workspace Configuration. The primary branch represents the general case where the target end-effector pose lies within the robot's workspace but is not at any singular configuration. The kinematic constraint equation:

$$(P_x^2 + P_y^2 - L_1^2 - L_2^2 - L_3^2)^2 = 4L_1^2(L_2^2 + L_3^2 - P_y^2) + 4L_1^2(P_x - L_2\cos\phi - L_3\sin\phi)^2$$

must be satisfied for a solution to exist. This equation defines the precise algebraic boundary of the robot's workspace. For parameters satisfying this condition, the algorithm provides the closed-form solution:

$$\theta_1 = 2\arctan\left(\frac{2L_1(P_y - L_3\sin\phi) \pm \sqrt{\Delta}}{P_x^2 + P_y^2 - L_1^2 - L_2^2 - L_3^2 + 2L_1(P_x - L_2\cos\phi - L_3\sin\phi)}\right)$$

where Δ represents the discriminant ensuring real solutions.

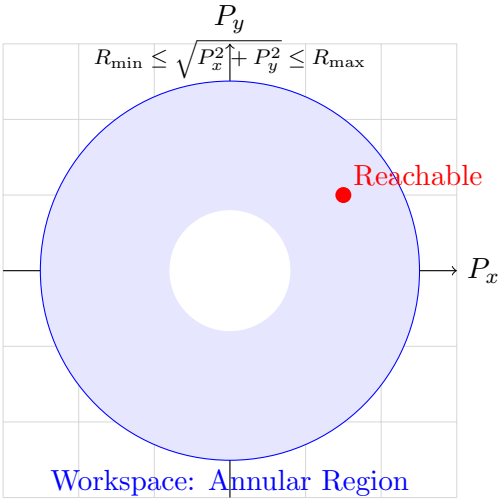

Fig. 1. Workspace of planar 3R robot. The workspace is an annular region bounded by inner radius $R_{\min} = |L_1 - L_2 - L_3|$ and outer radius $R_{\max} = L_1 + L_2 + L_3$.

Branch 2: Fully Extended Configuration. This branch corresponds to the case where the end-effector is located at the maximum reachable distance from the base:

$$P_x^2 + P_y^2 = (L_1 + L_2 + L_3)^2$$

In this configuration, all links are collinear, and the robot arm is fully stretched. The solution becomes unique, with the joint angle given by:

$$\theta_1 = \arctan\left(\frac{P_y}{P_x + L_1 + L_2 + L_3}\right)$$

This represents a singular configuration where the Jacobian matrix loses rank, and the robot has reduced mobility in certain directions.

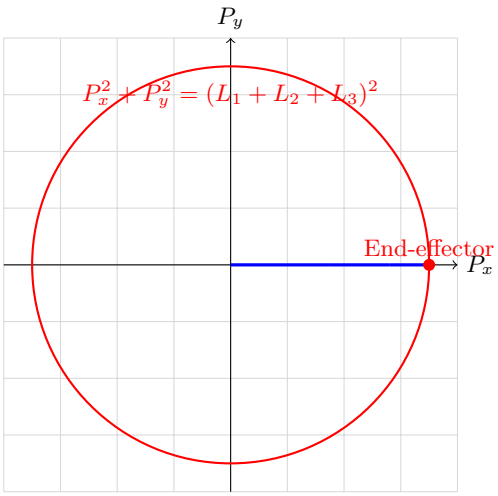

Fig. 2. Fully extended arm configuration. The robot links are collinear, reaching the maximum possible distance from the base.

Branch 3: Fully Retracted Configuration. The third branch occurs when the end-effector is at the minimum reachable distance:

$$P_x^2 + P_y^2 = (L_1 - L_2 - L_3)^2$$

In this case, the arm is completely folded back on itself. The solution is again unique:

$$\theta_1 = \arctan\left(\frac{P_y}{P_x + L_1 - L_2 - L_3}\right)$$

This configuration represents another singular point in the workspace where the robot's mobility is constrained.

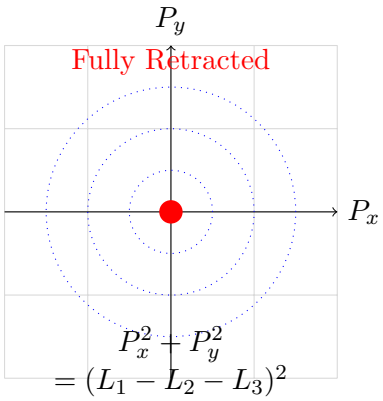

Fig. 3. Fully retracted arm configuration. The robot links are completely folded back, minimizing the reachable distance.

Branch 4: Degenerate Case with Infinite Solutions. This branch represents a degenerate case where the kinematic equations become dependent, leading to infinitely many solutions. This occurs when:

$$P_x = L_2 \cos\phi + L_3 \sin\phi, \quad P_y = L_2 \sin\phi - L_3 \cos\phi, \quad L_1 = 0.$$

In this configuration, the base joint coincides with the end-effector position, and any combination of joint angles that maintains the end-effector orientation is valid. This represents a workspace singularity where the inverse kinematics problem is underconstrained.

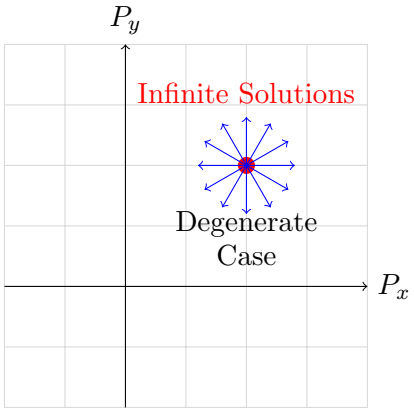

Fig. 4. Degenerate configuration with infinite solutions. The base joint coincides with the end-effector position, allowing multiple joint angle combinations.

Branch 5: Unreachable configurations. The final branch encompasses all parameter values for which no real solution exists. This occurs when the target pose lies outside the robot's workspace, violating the fundamental kinematic constraints. The algorithm correctly identifies these cases by the failure of all previous conditions, returning an empty solution set.

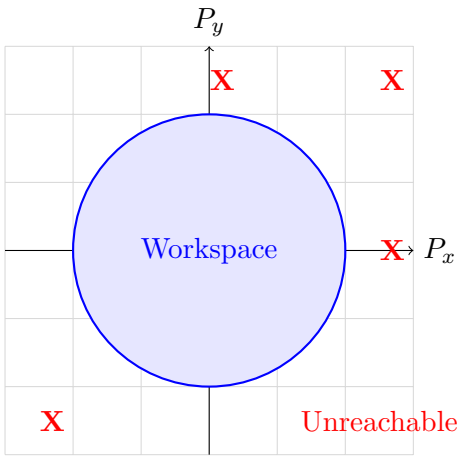

Fig. 5. Unreachable end-effector positions. Red X marks indicate target poses outside the robot's workspace boundary.

DETECT-CONDITION **Algorithm**
Five Solution Branches Summary

1. **Workspace Boundary**: General case with closed-form solutions
2. **Fully Extended**: Maximum reach singularity
3. **Retracted**: Minimum reach singularity
4. **Infinite Solutions**: Degenerate underconstrained case
5. **No Solution**: Unreachable target poses

Planar 3R Robot: $L_1 = 5$, $L_2 = 3$, $L_3 = 2$
Complete parameter space partitioning

Fig. 6. Comprehensive overview of the five solution branches identified by the DETECT-CONDITION algorithm.

Algorithmic Significance and Practical Implications. The partitioning of the solution space into these five branches provides several key advantages:

I. **Completeness**: The algorithm guarantees that all possible cases are considered, leaving no region of the parameter space unclassified.
II. **Symbolic insight**: Each branch provides explicit algebraic conditions that reveal the fundamental geometric constraints of the robot mechanism.
III. **Numerical robustness**: By identifying singular configurations (Branches 2, 3, and 4) explicitly, the algorithm avoids numerical instabilities that plague iterative methods near singularities.
IV. **Design guidance:** The workspace boundary equation (Branch 1) provides direct insight into how link length selection affects reachable workspace.
V. **Control strategy**: The branch classification enables intelligent control strategies that can avoid singular configurations or handle degenerate cases appropriately.

Numerical verification. Let the link lengths be fixed: $a = L_1 = 5, b = L_2 = 3, c = L_3 = 2$. Let the desired end-effector pose be: $P_x = 6, P_y = 4, \Phi = \pi/4$ radians.

First, we verify the kinematic constraint *Eq. (11)*. Substituting the values into the LHS of the constraint equation yields a value very close to zero (within numerical precision), confirming the pose is reachable. These parameters satisfy Branch 1 conditions:

$$\begin{aligned}\Delta &= 4 \cdot 25 \cdot \left[(4 - 2 \cdot \sqrt{2}/2)^2 + (6 - 3 \cdot \sqrt{2}/2 - 2 \cdot \sqrt{2}/2)^2 - (9 + 4 - 16)\right] \\ &= 100 \cdot \left[(4 - \sqrt{2})^2 + (6 - 5\sqrt{2}/2)^2 + 3\right] \\ x_1 = t_1 &= \frac{2 \cdot 5 \cdot (4 - \sqrt{2}) + \sqrt{\Delta}}{36 + 16 - 25 - 9 - 4 + 2 \cdot 5 \cdot (6 - 5\sqrt{2}/2)} \approx 0.455\end{aligned}$$

Thus, $\theta_1 = 2 \arctan(0.455) \approx 49.0°$. Back-substitution yields $\theta_2 \approx 23.4°$, $\theta_3 \approx -27.4°$. Forward verification confirms the end-effector reaches $(6, 4)$ with orientation $45°$.

This result can be verified by forward kinematics: using $\theta_1 \approx 49.0°$ and solving for θ_2, θ_3 from the remaining equations places the end-effector at $(x, y) \approx (6, 4)$ with an orientation $\phi \approx 45°$.

4|Conclusion

This application demonstrates the profound utility of the DETECT-CONDITION algorithm. It automatically derives the exact algebraic condition (the workspace boundary equation) that must be satisfied for a solution to exist, a non-trivial task that would be manually tedious. Furthermore, it provides a closed-form symbolic solution for the joint angle. The advantages are clear:

I. **Completeness:** the algorithm provides all possible solutions and the precise conditions under which they are valid.

II. Insight: it reveals the fundamental kinematic constraints of the robot design symbolically, which is invaluable for robot design and workspace analysis.

III. Precision: solutions are exact and symbolic, avoiding numerical instability issues common in iterative IK solvers for near-singular configurations.

IV. Automation: the entire process of deriving the complex IK equations and solving them is automated, saving significant time and preventing human error.

This makes the DETECT-CONDITION algorithm, powered by Gröbner systems, a powerful tool for roboticists and engineers dealing with the symbolic analysis of complex mechanisms.

Acknowledgment

This research was partially funded by the Institute for Research in Fundamental Sciences (IPM). I also appreciate the reviewers' valuable feedback, comments, and suggestions, which enhanced the article's quality.

References

[1] Becker, T., & Weispfenning, V. (1993). Gröbner bases. In *gröbner bases: A computational approach to commutative algebra* (pp. 187-242). New York, NY: Springer New York. https://doi.org/10.1007/978-1-4612-0913-3_6

[2] Buchberger, B. (1979). A criterion for detecting unnecessary reductions in the construction of Gröbner-bases. In *international symposium on symbolic and algebraic manipulation* (pp. 3-21). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-09519-5_52

[3] Buchberger, B. (2006). Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of symbolic computation*, *41*(3-4), 475-511. https://doi.org/10.1016/j.jsc.2005.09.007

[4] Cox, D., Little, J., O'shea, D., & Sweedler, M. (1997). *Ideals, varieties, and algorithms. An introduction to computational algebraic geometry and commutative algebra.* Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-319-16721-3

[5] Darmian, M. D. (2024). Efficient algorithm for computing inverse of parametric matrices. *Scientific annals of computer science*, *34*(1), 1-22. https://publications.info.uaic.ro/files/sacs/XXXIV1/XXXIV1_0.pdf

[6] Dehghani Darmian, M. (2024). Improvement of an incremental signature-based comprehensive Gröbner system algorithm. *Mathematics in computer science*, *18*(2), 12. https://doi.org/10.1007/s11786-024-00587-w

[7] Darmian, M. D., & Hashemi, A. (2017). Parametric FGLM algorithm. *Journal of symbolic computation*, *82*, 38-56. https://doi.org/10.1016/j.jsc.2016.12.006

[8] Dehghani Darmain, M., & Hashemi, A. (2024). A Parametric $ F_4 $ Algorithm. *Iranian journal of mathematical sciences and informatics*, *19*(1), 117-133.

[9] Faugere, J. C. (1999). A new efficient algorithm for computing Gröbner bases (F4). *Journal of pure and applied algebra*, *139*(1-3), 61-88. https://doi.org/10.1016/S0022-4049(99)00005-5

[10] Faugere, J. C. (2002). A new efficient algorithm for computing Gröbner bases without reduction to zero (F 5). In *proceedings of the 2002 international symposium on Symbolic and algebraic computation* (pp. 75-83). https://doi.org/10.1145/780506.780516

[11] Gao, S., Guan, Y., & Volny IV, F. (2010). A new incremental algorithm for computing Gröbner bases. In *proceedings of the 2010 international symposium on symbolic and algebraic computation* (pp. 13-19). https://doi.org/10.1145/1837934.1837944

[12] Gao, S., Volny IV, F., & Wang, M. (2016). A new framework for computing Gröbner bases. *Mathematics of computation*, *85*(297), 449-465. https://pubs.ams.org/journals/mcom/0000-000-00/S0025-5718-2015-02969-X/S0025-5718-2015-02969-X.pdf

[13] Amir Hashemi, Mahdi Dehghani Darmian, & Marzieh Barkhordar. (2017). *Gröbner systems conversion*. *Mathematics in computer science*, *11*(1), 61–77. https://doi.org/10.1007/s11786-017-0295-3

[14] Hashemi, A., Darmian, M. D., & Barkhordar, M. (2018, July). Universal Gröbner basis for parametric polynomial ideals. In *international congress on mathematical software* (pp. 191-199). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-96418-8_23

[15] Hashemi, A., Darmian, M. D., & M-Alizadeh, B. (2012). Applying buchberger's criteria on montes's dispgb algorithm. *Bulletin of the Iranian mathematical society*, *38*(3), 715. https://openurl.ebsco.com/contentitem/gcd:88915848?sid=ebsco:plink:crawler-gcd&id=ebsco:gcd:88915848&crl=c&jrnl=10186301

[16] Hashemi, A., M.-Alizadeh, B., & Darmian, M. D. (2013). Minimal polynomial systems for parametric matrices. *Linear and multilinear algebra*, *61*(2), 265-272. https://doi.org/10.1080/03081087.2012.670235

[17] Hashemi, A., & Dehghani, D. M. (2017). Computing Comprehensive Gr¨ obner Systems: A Comparison of Two Methods. *Computer science journal of moldova*, *75*(3), 278-302. https://ibn.idsi.md/sites/default/files/imag_file/pp278-302_Computing%20Comprehensive%20Gr%20%CC%88obner%20Systems%20A%20Comparison%20of%20Two%20Methods.pdf

[18] Kalkbrener, M. (1999). On the complexity of Gröbner bases conversion. *Journal of Symbolic Computation*, *28*(1–2), 265–273. https://doi.org/10.1006/jsco.1998.0276

[19] Kapur, D., Sun, Y., & Wang, D. (2010, July). A new algorithm for computing comprehensive Gröbner systems. In *proceedings of the 2010 international symposium on symbolic and algebraic computation* (pp. 29-36). https://doi.org/10.1145/1837934.1837946

[20] Lazard, D. (1983, March). Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In *european conference on computer algebra* (pp. 146-156). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-12868-9_99

[21] Manubens, M., & Montes, A. (2006). Improving the DISPGB algorithm using the discriminant ideal. *Journal of symbolic computation*, *41*(11), 1245-1263. https://doi.org/10.1016/j.jsc.2005.09.013

[22] Manubens, M., & Montes, A. (2009). Minimal canonical comprehensive Gröbner systems. *Journal of symbolic computation*, *44*(5), 463-478. https://doi.org/10.1016/j.jsc.2007.07.022

[23] Montes, A., & Castro, J. (1995). Solving the load flow problem using gröbner basis. *ACM SIGSAM Bulletin*, *29*(1), 1-13. https://doi.org/10.1145/216685.216686

[24] Suzuki, A., & Sato, Y. (2006). A simple algorithm to compute comprehensive Gröbner bases using Gröbner bases. In *proceedings of the 2006 international symposium on symbolic and algebraic computation* (pp. 326-331). https://doi.org/10.1145/1145768.1145821

[25] Viète, F. (1970). *Variorum de rebus mathematicis responsorum liber VIII*. https://books.google.com/books

[26] Weispfenning, V. (1992). Comprehensive gröbner bases. *Journal of symbolic computation*, *14*(1), 1-29. https://doi.org/10.1016/0747-7171(92)90023-W