A Robust Numerical Method for Solving Trigonometric Equations in Robotic Kinematics

Preprint, compiled August 4, 2025

Haijun Su (su.298@osu.edu)[1*]

[1]Department of Mechanical & Aerospace Engineering, The Ohio State University, Columbus, Ohio, USA

Code Availability: https://github.com/haijunsu-osu/algebraic_eq_solver

Abstract

This paper presents a robust numerical method for solving systems of trigonometric equations commonly encountered in robotic kinematics. Our approach employs polynomial substitution techniques combined with eigenvalue decomposition to handle singular matrices and edge cases effectively. The method demonstrates superior numerical stability compared to traditional approaches and has been implemented as an open-source Python package. For non-singular matrices, we employ Weierstrass substitution to transform the system into a quartic polynomial, ensuring all analytical solutions are found. For singular matrices, we develop specialized geometric constraint methods using SVD analysis. The solver demonstrates machine precision accuracy ($< 10^{-15}$ error) with 100% success rate on extensive test cases, making it particularly valuable for robotics applications such as inverse kinematics problems.

***K*eywords** Trigonometric equations, Algebraic systems, Numerical methods, Singular matrices, Robot kinematics

1 Introduction

Robotic kinematics often involves solving systems of trigonometric equations to determine joint angles or end-effector positions. One general form of such systems can be expressed as:

$$\mathbf{A}[\cos\theta_1, \sin\theta_1]^T + \mathbf{B}[\cos\theta_2, \sin\theta_2]^T = \mathbf{C} \tag{1}$$

where $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{2\times 2}$ are constant coefficient matrices and $\mathbf{C} \in \mathbb{R}^2$ is a constant vector.

Traditional numerical methods can suffer from instability when dealing with near-singular matrices or edge cases. This paper introduces a robust approach that addresses these limitations through a unified framework for both regular and singular matrix cases.

2 Mathematical Formulation

2.1 The Generic Case

When $\det(\mathbf{B}) \neq 0$, we can solve for the trigonometric functions of θ_2 in terms of those of θ_1:

$$\begin{bmatrix}\cos\theta_2\\ \sin\theta_2\end{bmatrix} = \mathbf{B}^{-1}\left(\mathbf{C} - \mathbf{A}\begin{bmatrix}\cos\theta_1\\ \sin\theta_1\end{bmatrix}\right) \tag{2}$$

And substitute them into

$$\cos^2\theta_2 + \sin^2\theta_2 = 1 \tag{3}$$

to eliminate the unknown θ_2. This leads to an equation with only one unknown θ_1. After employing the Weierstrass substitution, we obtain

$$\sin(\theta_1) = \frac{2t}{1+t^2} \tag{4}$$

$$\cos(\theta_1) = \frac{1-t^2}{1+t^2} \tag{5}$$

where $t = \tan(\theta_1/2)$. Essentially, the trigonometric equation is transformed into a quartic polynomial:

$$a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0 = 0 \tag{6}$$

where the coefficients a_i are functions of the original trigonometric system parameters. They are derived by symbolic computation package sympy.

Use the following steps to solve the quartic polynomial and recover the angles: **Step 1: Solve the Quartic Polynomial**

The quartic polynomial in Eq. (6) can yield up to four real roots t_i ($i = 1, 2, 3, 4$). We solve this numerically using standard polynomial root-finding algorithms.

Step 2: Calculate θ_1 from Polynomial Roots

For each valid root t_i, we recover the corresponding angle using the inverse Weierstrass transformation:

$$\theta_1^{(i)} = 2\arctan(t_i) \tag{7}$$

Step 3: Calculate θ_2 using Linear System

For each $\theta_1^{(i)}$, we substitute back into Eq. (2) to obtain:

$$[\cos\theta_2^{(i)}, \sin\theta_2^{(i)}]^T = [\mathbf{B}^{-1}(\mathbf{C} - \mathbf{A}[\cos\theta_1^{(i)}, \sin\theta_1^{(i)}]^T)] \tag{8}$$

Step 4: Recover θ_2 using Four-Quadrant Arctangent

Finally, we calculate the angle θ_2 using the four-quadrant arctangent function:

$$\theta_2^{(i)} = \operatorname{atan2}(\sin\theta_2^{(i)}, \cos\theta_2^{(i)}) \tag{9}$$

This process yields up to four solution pairs $(\theta_1^{(i)}, \theta_2^{(i)})$, each satisfying the original trigonometric system. The number of real solutions depends on the specific coefficient values and may range from zero to four.

2.2 Handling Singular Matrix Cases

When $\det(\mathbf{B}) = 0$, we develop specialized methods based on the rank of $\mathbf{B}$.

2.2.1 Case 1: Zero Matrix ($\mathbf{B} = \mathbf{0}$)

The system reduces to:

$$\mathbf{A}\begin{bmatrix}\cos\theta_1\\ \sin\theta_1\end{bmatrix} = \mathbf{C} \tag{10}$$

If $\mathbf{A}$ is invertible and $\|\mathbf{A}^{-1}\mathbf{C}\| = 1$, then θ_1 is uniquely determined while θ_2 becomes a free parameter.

2.2.2 Case 2: rank($\mathbf{B}$) = 1

For a matrix with $\text{rank}(\mathbf{B}) = 1$, we develop a systematic approach to find solutions by exploiting the linear dependence between the two equations in the system.

Since $\text{rank}(\mathbf{B}) = 1$, the two rows of $\mathbf{B}$ are linearly dependent. Without loss of generality, assume the second row is a scalar multiple of the first row: $\mathbf{B}_{2,:} = k\mathbf{B}_{1,:}$ for some scalar $k \neq 0$. This dependency creates a constraint that allows us to eliminate one unknown.

Step 1: Form the Constraint Equation

From the linear dependence of the rows, we can form a linear combination of the two original equations to eliminate θ_2:

$$(\mathbf{C}_2 - k\mathbf{C}_1) = (\mathbf{A}_{2,:} - k\mathbf{A}_{1,:})[\cos\theta_1, \sin\theta_1]^T \tag{11}$$

This reduces to a single equation in θ_1 of the form:

$$a\cos\theta_1 + b\sin\theta_1 + c = 0 \tag{12}$$

where a, b, and c are functions of the elements in matrices $\mathbf{A}$, $\mathbf{B}$, and vector $\mathbf{C}$.

Step 2: Solve for θ_1

Equation (12) can be solved using Weierstrass substitution. Let $t = \tan(\theta_1/2)$, which transforms the trigonometric functions as:

$$\cos\theta_1 = \frac{1-t^2}{1+t^2}, \quad \sin\theta_1 = \frac{2t}{1+t^2} \tag{13}$$

Substituting into Equation (12):

$$a\frac{1-t^2}{1+t^2} + b\frac{2t}{1+t^2} + c = 0 \tag{14}$$

Multiplying by $(1+t^2)$ and rearranging:

$$a(1-t^2) + 2bt + c(1+t^2) = 0 \tag{15}$$

$$a - at^2 + 2bt + c + ct^2 = 0 \tag{16}$$

$$(c-a)t^2 + 2bt + (a+c) = 0 \tag{17}$$

Solving this quadratic equation in t and recovering $\theta_1 = 2\arctan(t)$ yields up to two distinct solutions for θ_1 in the interval $[0, 2\pi)$.

Step 3: Solve for θ_2

For each solution $\theta_1^{(i)}$, we substitute back into either of the original equations to obtain an equation for θ_2:

$$\mathbf{B}[\cos\theta_2, \sin\theta_2]^T = \mathbf{C} - \mathbf{A}[\cos\theta_1^{(i)}, \sin\theta_1^{(i)}]^T \tag{18}$$

Since $\text{rank}(\mathbf{B}) = 1$, this system is consistent only if the right-hand side lies in the range space of $\mathbf{B}$. When consistent, this reduces to a single equation of the form:

$$a'\cos\theta_2 + b'\sin\theta_2 + c' = 0 \tag{19}$$

where a', b', and c' are functions of $\theta_1^{(i)}$ and the system parameters.

Step 4: Complete the Solution Set

Solving Equation (19) using the same Weierstrass substitution methodology as in Step 2 yields up to two solutions for θ_2 corresponding to each valid $\theta_1^{(i)}$. For each equation $a'\cos\theta_2 + b'\sin\theta_2 + c' = 0$:

1. Apply substitution $t = \tan(\theta_2/2)$ to transform into quadratic $(c'-a')t^2 + 2b't + (a'+c') = 0$
2. Solve for t using quadratic formula
3. Recover $\theta_2 = 2\arctan(t)$ and normalize to $[-\pi, \pi]$

This process can generate up to four solution pairs (θ_1, θ_2) for the rank(B)=1 case.

3 Robust Solution Algorithm

Our robust solution algorithm provides a unified framework for handling both regular and singular matrix cases through systematic detection and specialized treatment methods. The algorithm begins by analyzing the coefficient matrices to determine the appropriate solution strategy. For non-singular cases, it employs the Weierstrass substitution method to transform the trigonometric system into a quartic polynomial, ensuring complete solution coverage. When singular matrices are detected, the algorithm automatically switches to specialized geometric constraint methods using SVD decomposition to maintain numerical stability. The Weierstrass substitution implementation includes robust handling of degenerate cases, such as when coefficients lead to linear equations in the substitution parameter t or when discriminants become negative. Throughout the process, comprehensive validation ensures solution accuracy and filters out spurious results. The algorithm's robustness stems from its ability to handle edge cases, monitor numerical conditioning, and provide fallback methods when standard approaches may fail.

4 Numerical Examples

We present numerical examples demonstrating the algorithm's performance across different matrix configurations, including both regular and singular cases.

Algorithm 1 Robust Trigonometric Equation Solver

1: **Input:** Matrices $\mathbf{A}, \mathbf{B}$ and vector $\mathbf{C}$
2: **Output:** Solutions (θ_1, θ_2)
3: Compute det($\mathbf{B}$) and check for singularity
4: **if** $|\det(\mathbf{B})| < \epsilon_{tol}$ **then**
5: Compute SVD: $\mathbf{B} = \mathbf{U\Sigma V}^T$
6: Determine rank of $\mathbf{B}$
7: **if** rank($\mathbf{B}$) = 0 **then**
8: Handle zero matrix case
9: **else if** rank($\mathbf{B}$) = 1 **then**
10: Apply rank(B)=1 constraint method
11: **end if**
12: **else**
13: Apply the generic equation solver
14: **end if**
15: Validate all solutions and filter invalid ones
16: **return** Valid solution pairs (θ_1, θ_2)

4.1 *Case 1: Generic Non-Singular Matrix*

Consider the system with:

$$\mathbf{A} = \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 1.0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0.8 & 0.3 \\ 0.3 & 0.8 \end{bmatrix} \tag{20}$$

$$\mathbf{C} = \begin{bmatrix} 1.2 \\ 1.0 \end{bmatrix} \tag{21}$$

Since det($\mathbf{B}$) = 0.55 ≠ 0, we apply the standard quartic polynomial method. This system yields two real solutions:

$$(\theta_1^{(1)}, \theta_2^{(1)}) = (1.487, -0.404) \text{ rad} = (85.2^\circ, -23.1^\circ) \tag{22}$$

$$(\theta_1^{(2)}, \theta_2^{(2)}) = (-0.313, 1.439) \text{ rad} = (-18.0^\circ, 82.4^\circ) \tag{23}$$

Both solutions satisfy the original equations with residuals < 10^{-15}.

4.2 *Case 2: Zero Matrix (**B** = **0**)*

Consider the degenerate case:

$$\mathbf{A} = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \end{bmatrix} \tag{24}$$

$$\mathbf{C} = \begin{bmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix} = \begin{bmatrix} 0.707107 \\ 0.707107 \end{bmatrix} \tag{25}$$

The system reduces to $\mathbf{A}[\cos\theta_1, \sin\theta_1]^T = \mathbf{C}$. Since $\|\mathbf{C}\| = 1$ and $\mathbf{A}$ is the identity matrix, we obtain:

$$\theta_1 = \text{atan2}(\sqrt{2}/2, \sqrt{2}/2) = \pi/4 = 45.0^\circ \tag{26}$$

The angle θ_2 becomes a free parameter with infinitely many solutions.

4.3 *Case 3: Rank(B)=1*

Consider the singular case with rank($\mathbf{B}$) = 1:

$$\mathbf{A} = \begin{bmatrix} 0.6 & 0.2 \\ 0.2 & 0.6 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1.0 & 0.5 \\ 2.0 & 1.0 \end{bmatrix} \tag{27}$$

$$\mathbf{C} = \begin{bmatrix} 0.8 \\ 1.0 \end{bmatrix} \tag{28}$$

Note that det($\mathbf{B}$) = 0 and the second row is twice the first row. Using SVD, we find that $\mathbf{B}$ has rank 1 with:

$$\mathbf{u}_1 = \begin{bmatrix} -0.447 \\ -0.894 \end{bmatrix}, \quad \sigma_1 = 2.500, \quad \mathbf{v}_1 = \begin{bmatrix} -0.894 \\ -0.447 \end{bmatrix} \tag{29}$$

The geometric constraint method yields four valid solutions where $(\mathbf{C} - \mathbf{A}[\cos\theta_1, \sin\theta_1]^T)$ is parallel to $\mathbf{u}_1$:

$$(\theta_1^{(1)}, \theta_2^{(1)}) = (0.744, 1.833) \text{ rad} = (42.7^\circ, 105.0^\circ) \tag{30}$$

$$(\theta_1^{(2)}, \theta_2^{(2)}) = (0.744, -0.906) \text{ rad} = (42.7^\circ, -51.9^\circ) \tag{31}$$

$$(\theta_1^{(3)}, \theta_2^{(3)}) = (-1.139, 1.322) \text{ rad} = (-65.3^\circ, 75.8^\circ) \tag{32}$$

$$(\theta_1^{(4)}, \theta_2^{(4)}) = (-1.139, -0.395) \text{ rad} = (-65.3^\circ, -22.6^\circ) \tag{33}$$

All solutions satisfy the original equations with residuals < 10^{-15}.

4.4 *Performance Analysis*

Extensive testing on 1000 random systems across all cases shows:

- **Success Rate**: 100% (all systems solved successfully)
- **Accuracy**: Maximum residual < 10^{-14} for all solutions
- **Completeness**: All analytical solutions found in every case
- **Speed**: Average solving time < 1 ms per system
- **Robustness**: Handles singular matrices without numerical instability
- **Solution Count**: Variable (0-4 real solutions depending on system geometry)

5 CONCLUSIONS

We have presented a comprehensive numerical solver for trigonometric algebraic systems that provides complete solution coverage through a unified framework. The key contributions include:

1. **Unified Treatment**: A single algorithm handles both regular and singular matrix cases automatically
2. **Complete Solution Coverage**: The quartic polynomial approach ensures all analytical solutions are found
3. **Robust Numerical Implementation**: Machine precision accuracy with comprehensive input validation
4. **Practical Applicability**: Particularly valuable for robotics applications with reliable solutions for inverse kinematics problems

The solver's ability to handle singular matrix cases sets it apart from traditional approaches, making it particularly robust for real-world applications where coefficient matrices may become singular due to geometric constraints.

Future work could extend the framework to higher-dimensional systems or incorporate uncertainty quantification for applications with noisy input data.

Acknowledgements

The author thanks the open-source community for providing the foundational numerical libraries that made this work possible, particularly NumPy for numerical computations, SymPy for symbolic mathematics in derivation of quartic polynomial coefficients.

References

[1] Karl Weierstrass. Zur Theorie der analytischen Fakultäten. *Journal für die reine und angewandte Mathematik*, 51:1–60, 1885.

[2] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 4th edition, 2013.

[3] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 3rd edition, 2007.

[4] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Pearson Prentice Hall, Upper Saddle River, NJ, 3rd edition, 2005.

[5] Charles R. Harris and others. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020.