A Robust Numerical Method for Solving Trigonometric Equations in Robotic Kinematics

Preprint, compiled August 3, 2025

Haijun Su[1*]

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

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. These problems frequently lead to systems of the form:

$$a_1 \sin(\theta) + b_1 \cos(\theta) = c_1 \tag{1}$$

$$a_2 \sin(\theta) + b_2 \cos(\theta) = c_2 \tag{2}$$

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.

The general form of such systems can be expressed as:

$$\mathbf{A}[\cos\theta_1, \sin\theta_1]^T + \mathbf{B}[\cos\theta_2, \sin\theta_2]^T = \mathbf{C} \tag{3}$$

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

2 Mathematical Formulation

2.1 Polynomial Substitution Method

We employ the Weierstrass substitution $t = \tan(\theta/2)$, which transforms trigonometric equations into polynomial form:

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

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

This substitution converts our system into:

$$a_1 \frac{2t}{1+t^2} + b_1 \frac{1-t^2}{1+t^2} = c_1 \tag{6}$$

$$a_2 \frac{2t}{1+t^2} + b_2 \frac{1-t^2}{1+t^2} = c_2 \tag{7}$$

2.2 Quartic Polynomial Formation

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

$$\begin{bmatrix}\cos\theta_2\\ \sin\theta_2\end{bmatrix} = \mathbf{B}^{-1}\left(\mathbf{C} - \mathbf{A}\begin{bmatrix}\cos\theta_1\\ \sin\theta_1\end{bmatrix}\right) \tag{8}$$

After algebraic manipulation, we obtain a quartic polynomial:

$$At^4 + Bt^3 + Ct^2 + Dt + E = 0 \tag{9}$$

where the coefficients are functions of the original trigonometric system parameters.

3 Robust Solution Algorithm

3.1 Handling Singular Matrix Cases

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

3.1.1 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.

Algorithm 1 Robust Trigonometric Equation Solver

```
1: Input: Coefficients $a_1, b_1, c_1, a_2, b_2, c_2$
2: Output: Solutions $\theta$
3: Check for edge cases and handle special configurations
4: Form coefficient matrix $M$ and vector $\mathbf{b}$
5: Compute matrix condition number
6: if condition number $> 10^{12}$ then
7:     Use SVD decomposition for numerical stability
8: else
9:     Use standard linear algebra solution
10: end if
11: Form quartic polynomial coefficients
12: Solve quartic equation using companion matrix eigenvalues
13: Convert polynomial roots back to angular solutions
14: Filter and validate solutions
15: return Valid solutions
```

3.1.2 Case 2: Rank-1 Matrix

We employ SVD analysis: $\mathbf{B} = \mathbf{U\Sigma V}^T$ where $\mathbf{\Sigma} = \text{diag}(\sigma_1, 0)$.

4 Numerical Examples

4.1 Generic Non-Singular Case

Consider the system with:

$$\mathbf{A} = \begin{bmatrix} 2.0 & 1.0 \\ 1.0 & 3.0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1.5 & -0.5 \\ 0.5 & 2.0 \end{bmatrix} \tag{11}$$

$$\mathbf{C} = \begin{bmatrix} 2.939158 \\ 4.133792 \end{bmatrix} \tag{12}$$

This system yields two real solutions:

$$(\theta_1^{(1)}, \theta_2^{(1)}) = (0.524, 0.785) \text{ rad} = (30.0, 45.0) \tag{13}$$

$$(\theta_1^{(2)}, \theta_2^{(2)}) = (1.233, 0.246) \text{ rad} = (70.7, 14.1) \tag{14}$$

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

4.2 Performance Analysis

Extensive testing on 1000 random systems 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

5 Software Implementation

The algorithm has been implemented in Python with the following key features:

- Automatic detection and handling of singular matrices
- Edge case protection for division by zero
- Numerical stability through condition number monitoring
- Comprehensive solution validation
- Command-line interface for practical usage

The complete implementation is available as an open-source Python package:

```
pip install algebraic-eq-solver
trig-solver --help
```

The package includes core numerical solver with robustness features, command-line interface for direct usage, comprehensive test suite, and documentation with examples.

6 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 efficient array operations and SciPy for numerical algorithms.

Code Availability

The complete source code and documentation are available at: https://github.com/haijunsu-osu/algebraic_eq_solver

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.