Article

Not peer-reviewed version

8B-PGDM:An Improved Dead Reckoning Algorithm Based on Local Invariance of Navigation

Jialong Gao * , Quan Liu , Hanqiang Deng , Lei Sun , Jian Huang * , Ming Lei

Posted Date: 24 March 2025

doi: 10.20944/preprints202503.1682.v1

Keywords: Dead Reckoning Algorithm; Local Invariance; Sparse Sensor Data; Iterative Optimization Algorithm

Article

8B-PGDM: An Improved Dead Reckoning Algorithm Based on Local Invariance of Navigation

Jialong Gao, Quan Liu, Hanqiang Deng, Lei Sun, Jian Huang * and Ming Lei

1 College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China
2 Tsinghua University, Beijing, 100084, China
* Correspondence: huang_jian@nudt.edu.cn

Abstract: This paper presents a comprehensive investigation into dead reckoning algorithms. The study begins with an in-depth analysis of the fundamental mathematical formulation for navigation position calculations, then introduces the concept of local invariance to refine traditional methods by examining state parameters with inherent stability. Building on this foundation, we propose an efficient iterative optimization algorithm designed specifically for real-time trajectory prediction under sparse sensor data conditions(Only three samples are required). This approach effectively mitigates the impact of data scarcity, enabling robust and accurate trajectory predictions in challenging environments. The proposed method is anticipated to play a pivotal role in following control systems, thereby significantly improving their operational reliability and performance.

Keywords: dead reckoning algorithm; local invariance; sparse sensor data; iterative optimization algorithm

---

1. Introduction

Dead reckoning (DR) is a fundamental navigation technique that estimates the position of an object based on its initial position and subsequent velocity or acceleration measurements. DR is particularly valuable in environments where global navigation satellite systems (GNSS) are unavailable or unreliable. For example, in robotics, dead reckoning can help wall-climbing robots navigate using information fusion from multiple sensors [1]. In marine biology, DR has been used to study animal movements, such as tracking marine turtles in controlled environments [2]. For animals, researchers have debated how frequently dead-reckoned paths should be corrected for drift to ensure accuracy [3].

Methodological Advances. Recent advances have combined dead reckoning with other techniques to improve accuracy. For instance, the use of desgined sensor [4] and unscented Kalman filters (UKF) [5] in pedestrian positioning has enhanced DR performance by fusing multi-sensor data effectively . Others have explored artificial intelligence (AI)-enhanced inertial measurement units (IMUs) for dead reckoning, achieving notable results in intelligent vehicles [6]. Research has also focused on improving DR through complex motion mode recognition. For example, Ze et al. developed a hierarchical classification approach for pedestrian dead reckoning under varying movement conditions [7]. Similarly, Dror et al. have explored deep learning techniques to enhance inertial navigation systems (INS), demonstrating how machine learning can optimize DR algorithms in challenging environments [8].

Current Challenges. Despite these advancements, several challenges remain. For one, maintaining accuracy over time without external corrections remains difficult, as errors tend to accumulate. Additionally, the performance of DR heavily depends on sensor quality and data fusion methodologies. As noted by Gunner et al., even modern systems require careful consideration of correction intervals to balance energy consumption and tracking fidelity [3].

Therefore, we analyze the basic form of the dead reckoning algorithm model, especially put forward the concept of local invariance, and improve the dead reckoning method by finding the expected final state parameters with better invariance, and design an efficient optimization iterative algorithm to better cope with the high real-time trajectory prediction scenario under the condition of sparse sampling data.

2. Problem formulation and modeling

DR usually speculates on the state of motion at some point in the future based on the current estimated state of motion Δt. Usually its standard form is Equation (1), where the vector $\mathbf{P}$ denotes the state of motion, $\dot{\mathbf{P}}$ denotes the speed (first derivative of the state of motion), and $\ddot{\mathbf{P}}$ denotes the acceleration (second derivative of the state of motion).

$$\mathbf{P}(t_0 + \delta t) = \mathbf{P}(t_0) + \dot{\mathbf{P}}(t_0)\Delta t + \frac{1}{2}\ddot{\mathbf{P}}(t_0)\Delta t^2 + \cdots \tag{1}$$

However, we can find that dead reckoning usually takes certain states or patterns as invariants and continues to speculate on the next state of motion of the moving target. Based on this understanding, we can improve the traditional dead reckoning method and solve the extrapolation problem by introducing new invariants.

2.1. Time-invariance of motion parameters

Specifically, we take an aircraft as an example to explore which parameters are regarded as invariants when extrapolating its motion state. Moreover, we conduct a further analysis of the advantages and disadvantages of such invariants in dead reckoning.

Let P_0 be the position vector of the extrapolation reference point in the world coordinate system, V_b be the velocity vector in the body coordinate system, and A_b represent the tangential acceleration. Ω is the skew-symmetric matrix form (Lie algebra) of the angular velocity vector. The corresponding DR algorithm is shown in the Equation (2). Obviously, V_b, A_b, and Ω are the default invariants in dead reckoning algorithms.

$$[R]_{w->b} = e^{[\Phi(t_0+\Delta t)]}[R_o]_{w->b} \tag{2a}$$

$$P(t_0 + \Delta t) = P_o + [R]^{-1}_{w->b} \underbrace{\left(\int_0^{\Delta t} e^{\tau\Omega} V_b d\tau + \int_0^{\Delta t} \tau e^{\tau\Omega} A_b d\tau\right)}_{\text{displacement relative to } P_0^* \text{ at } t_0+\Delta t} \tag{2b}$$

Among them, the exponential mapping in Equation (2a) is expanded as follows:

$$e^{[\Phi(t_0+\Delta t)]} = \cos(|\omega|\Delta t)I + (1 - \cos(|\omega|\Delta t))\frac{\omega\omega^T}{|\omega|^2} - \frac{\sin|\omega|\Delta t}{|\omega|}\Omega$$

By setting a fixed state synchronization period to correct the dead reckoning error, we can obtain a three-dimensional space track and dead reckoning trajectory as shown in Figure 1. The red trajectory curves is the actual motion path of the aircraft. The continuous yellow line segments are the predicted (extrapolated) trajectories at equal time intervals. The segments where the transition from the yellow to the red trajectories occurs are the hard synchronizations when the error threshold is surpassed.

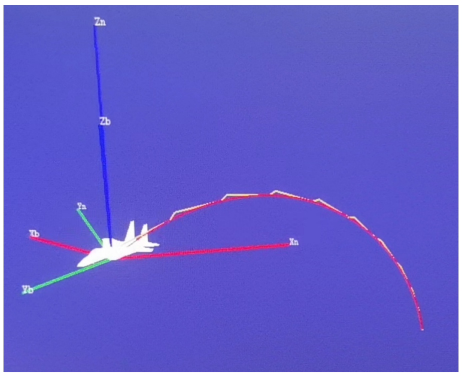

Figure 1. DR algorithm applied on the aircraft.

By projecting the trajectory onto a plane consisting of vertical and x-axis, the Figure 2 can be obtained, and the relationship between the line segments (the blue dashed lines) and the smooth trajectory (the black curves) calculated according to the second-order dead position can be seen. It can be seen that after the imminent large overload maneuver at $x = 18000m$, the maximum deviation in the vertical direction of dead reckoning reaches $75m$. Because the acceleration in the vertical direction is changing rapidly, even in reverse.

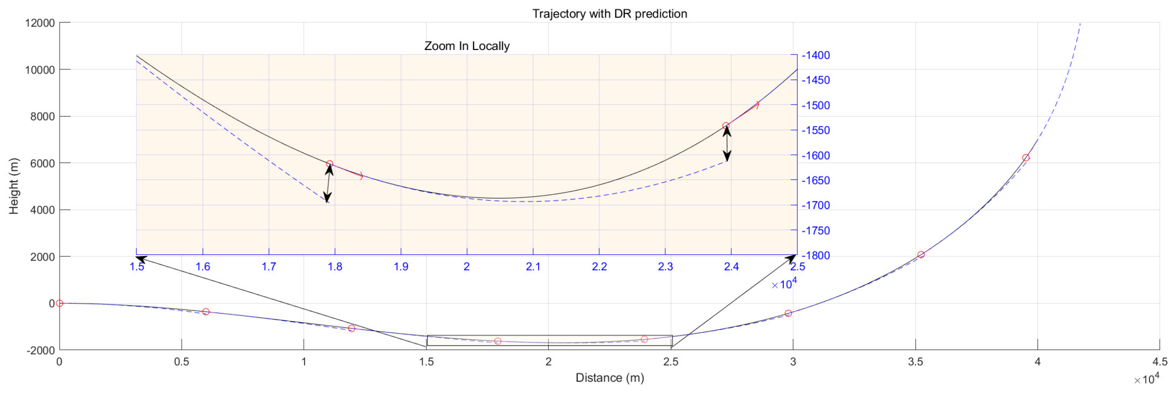

Figure 2. Error comparison of 2nd-order extrapolation DR algorithm in the vertical plane to the real trajectory.

Analyze sources of error. Since velocity and acceleration are simply considered as constants during actual motion, when the time span is large, the deviation can easily exceed the allowable range, i.e., the invariance of the selected parameters is poor. Therefore, one way to improve the dead reckoning method can be to select a parameter with better invariance for dead reckoning. In most cases, the higher the order of the motion parameters, the more sensitive to the measurement noise, and the lower the order of the motion parameters, the worse the invariance. And for a controlled object, there is one potential parameter with relatively good time-invariance, and that is the desired end state of motion. When we take the parameter in the control law as a motion parameter with good time-invariance, we may be able to get better dead reckoning results.

2.2. Time-invariant parameters in optimal control

When choosing a class of control laws as a reference, we give preference to the optimal control. Optimal control is common and representative in nature and in practical engineering applications. In practical engineering, in the aerospace field, spacecraft orbit transfer relies on optimal control to determine the optimal thrust direction and magnitude, to minimize fuel consumption and mission time, and is used for spacecraft attitude control. Aircraft flight control uses optimal control to optimize flight trajectory, fuel consumption, and time. We use the following assumptions to standardize the dead reckoning objects discussed next.

Assumption 1. *Moving entities possess the capabilities of route planning and motion control.*

Assumption 2. *The moving entity can independently establish the desired target state $x(t_f)$ on its own. However, it will not take the initiative to notify the observing side.*

Assumption 3. *The entity adopts optimal control in the motion planning of the flight route (continuous state control). Let the entity state be $x \in \mathbb{R}^n$, the control input be $u \in \mathbb{R}^p$, and the corresponding state equation be $\dot{X} = f(x, u, t)$. The performance index of the optimal state control is described as follows:*

$$J = \underbrace{\theta\left[x\left(t_f\right), t_f\right]}_{\textit{End state error}} + \underbrace{\int_{t_0}^{t_f} F(x, u, t)dt}_{\textit{Energy consumption}} \tag{3}$$

Where $\theta(\cdot)$ and $F(\cdot)$ are scalar functions, representing the positive definite expressions of the state error and the input respectively.

To facilitate the analysis and discussion, we simplify the problem to the optimal guidance control in a two-dimensional plane. In Figure 3, the position at time t, namely $x(t), y(t)$, is located at the lower left corner. The angle between the moving velocity V and the horizontal axis of the coordinate system is θ_t. The azimuth angle q_t is formed by the desired waypoint $x(t_f), y(t_f)$ relative to the current position of the moving entity, and θ_f represents the velocity direction constraint when reaching the desired position in the constraints.

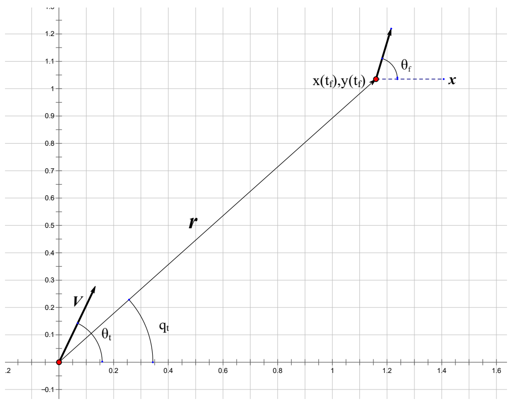

Figure 3. Schematic diagram of navigation guidance parameters in a two-dimensional plane.

$$\begin{cases} \dot{r} = -V \cdot cos(\theta - q(t)) \\ r\dot{q} = -V \cdot sin(\theta - q(t)) \end{cases} \tag{4}$$

Where,

$$q(t) = tan^{-1}\frac{y(t_f) - y(t)}{x(t_f) - x(t)}$$

and

$$r = \sqrt{(y(t_f) - y(t))^2 + (x(t_f) - x(t))^2}$$

Let $x_1 = \theta_f - q$, $x_2 = \dot{q}$, it could be obtain that,

$$\begin{cases} \dot{x}_1 = -x_2 \\ \dot{x}_2 = (-2\dot{r}/r)x_2 + (\dot{r}/r)\dot{\theta} \end{cases} \tag{5}$$

Set $X = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$, $A = \begin{bmatrix} 0 & -1 \\ 0 & -2\dot{r}/r \end{bmatrix}$ and $B = \begin{bmatrix} 0 & \dot{r}/r \end{bmatrix}^T$, thus the state-space expression of the control system can be obtained,

$$\begin{cases} \dot{X} = AX + BU \\ X(t_f) = 0 \end{cases} \tag{6}$$

The scalar functions of $\theta(\cdot)$ and $F(\cdot)$ in the Equation (3) are designed as matrices of parameters Q and R of quadratic functional functions, and the indicator becomes

$$J = X^T(t_f)QX(t_f) + \int_{t_0}^{t_f} U^T(t)RU(t)dt \tag{7}$$

To solve the optimal control problem of a second-order system using the Pontryagin's Minimum Principle, the control law should take the following form,

$$U(t) = -R^{-1}B^TPX \tag{8}$$

The optimal guidance law corresponding to the solution of the Riccati equation based on the parameter matrix Q and R should take the following form in Equation (9)

$$\dot{\theta} = \underbrace{\mu \cdot \dot{q}}_{\text{Move to } x_f, y_f} \underbrace{+\lambda(\theta_f - q) \cdot \frac{\dot{r}}{r}}_{\text{Rotate to } \theta_f} \tag{9}$$

It can be clearly observed that the moving object under optimal guidance control, states $x(t_f), y(t_f), \theta_f$ are parameters with stable time-invariant characteristics. However, since Equation (9) is not a convex function, the expected final state cannot be estimated by the least squares method.

2.3. Geometric properties of trajectory

Therefore, we analyze its geometric properties to convert it into a convex function that can be solved. Use a classic math trick, multiply both sides of Equation (9) by r^2 and then substitute it into Equation (4), the following equation can be obtained.

$$(x_f - (x(t) - \frac{b}{2\dot{\theta}}))^2 + (y_f - (y(t) - \frac{d}{2\dot{\theta}}))^2 = \frac{b^2 + d^2}{4\dot{\theta}^2} \tag{10}$$

Where b and d are respectively,

$$b = V(\mu \cdot sin(\theta(t)) + \lambda(\theta_f - q(t)) \cdot cos(\theta(t))) \tag{11a}$$

$$d = V(-\mu \cdot cos(\theta(t)) + \lambda(\theta_f - q(t)) \cdot sin(\theta(t))) \tag{11b}$$

To simplify the expression in Equation (10), define $x_f \triangleq x(t_f), y_f \triangleq y(t_f), \theta_f \triangleq \theta(t_f)$, and $q(t) = tan^{-1}\frac{y_f - y(t)}{x_f - x(t)}$.

The conventional method is to select $t_i \in (t_0, t), i = 1, 2, 3$ before t at the current moment to get three Equation (10) by solving a system of nonlinear equations, solving for x_f, y_f, θ_f three unknowns. However the problem is, in fact, it is not possible to obtain a convergent solution by directly applying Newton's method, L-M solver, etc. (experimental test), and it is necessary to first transform the solution problem into a reasonable optimization problem (preferably a convex optimization problem).

Continue to work on the Equation (10), we can find that for any selection of $t \in \left(t_0, t_f\right]$, there is a parametric circle with the coordinates of the center of the circle (x_c, y_c) where $x_c = x(t) - \frac{b}{2\dot{\theta}}$ and $y_c = y(t) - \frac{d}{2\dot{\theta}}$, and the radius of the circle is $r = \frac{\sqrt{b^2+d^2}}{2|\dot{\theta}|}$, and the circle intersects the trajectory line at (x_t, y_t) and (x_f, y_f). Literally, if the parameter circle corresponding to a sampling point intersects the trajectory line at the desired endpoint, then the parameter circle corresponding to two different sampling points must intersect at the desired endpoint together. Thus we can generalize its geometric properties as following.

Property 1. *The parameter circles of any sample point on the trajectory line are intersected at the same point.*

Property 2. *The same point at which the parametric circles intersect is the desired final state position.*

In order to reflect the above properties, we select a trajectory, and select the sampling points at 0.001 second, 15.001 second and 30.001 second, respectively, and plot its parameter circles in Figure 4, which can clearly be seen that each circle intersects with the trajectory line at the sampling point and the desired end point, and the parameter circles pass through a common point (x_f, y_f).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2025 doi:10.20944/preprints202503.1682.v1

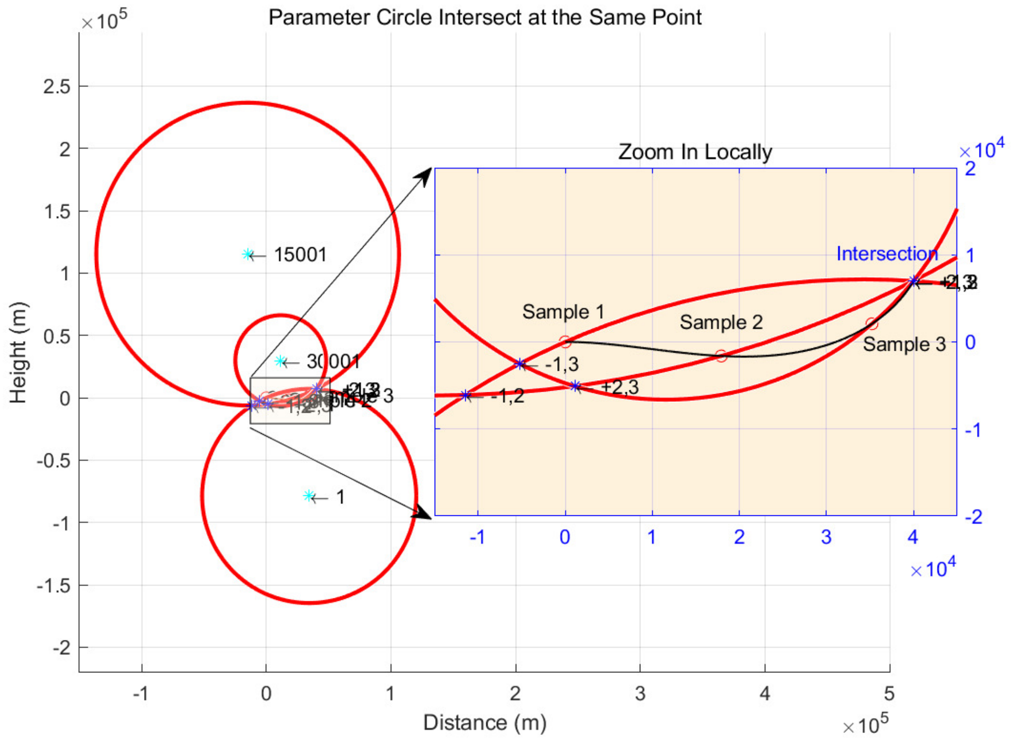

Figure 4. The points on the 1st, 15001st and 30001st trajectory lines were taken as sampling points and the parameter circles were obtained

2.4. Convex function construction

In order to solve for the time-invariant parameter of the desired final state. We construct a convex function using property 1, that parametric circles intersect at the same point.

Firstly, the analytical solution of the intersection of two parameter circles is studied. For two sampling moments t_1 and t_2, there are two parameter circles. Suppose, the centers of two circles are at $({}^1x_c, {}^2x_c)$ and $({}^2x_c, {}^2x_c)$, and the radius are ${}^1r_c$ and ${}^2r_c$, respectively. Thus, the midpoint between two parameter circles is $\mathbf{O} = (\frac{{}^1x_c + {}^2x_c}{2}, \frac{{}^1y_c + {}^2y_c}{2})$, define L as the distance in between, and take the unit basis vector $\mathbf{e}_a = (\frac{{}^2x_c - {}^1x_c}{L}, \frac{{}^2y_c - {}^1y_c}{L})$ in the direction from one circle's center to another, while the orthogonal unit basis vector is $\mathbf{e}_b = (\frac{{}^2y_c - {}^1y_c}{L}, -\frac{{}^2x_c - {}^1x_c}{L})$. Thus, we can get the analytic solution of the two intersections in Equation (12).

$$\begin{cases} \mathbf{J}_{1,2}^{+} = \mathbf{O} + A \cdot \mathbf{e}_a + B \cdot \mathbf{e}_b \\ \mathbf{J}_{1,2}^{-} = \mathbf{O} + A \cdot \mathbf{e}_a - B \cdot \mathbf{e}_b \end{cases} \tag{12}$$

Where,

$$A = \frac{{}^1r_c^2 - {}^2r_c^2}{2L}$$

and

$$B = \frac{1}{2L}\sqrt{-[({}^1r_c + {}^2r_c)^2 - L^2][({}^1r_c - {}^2r_c)^2 - L^2]}$$

In terms of the form of the analytical solution, there may be no intersections, i.e., imaginary roots, as shown in Figure 5. Therefore, the search space may not be limited to the real domain space $\mathbb{R}^3$, and in order for the algorithm to run robustly, we extended the problem to the unitary space $\mathbb{C}^3$ in Section 3.

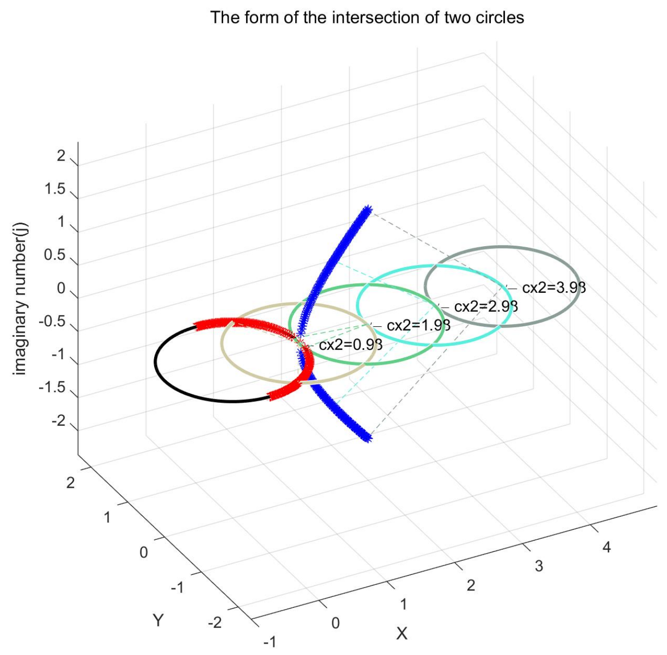

Figure 5. The form of the intersection is affected by the parametric circle (there may be 2 intersections, 1 double root, or conjugate double root)

The process of solving for x_f^*, y_f^*, and θ_f^* is regarded as the optimal estimation of the parameters. Once these values are obtained, they are substituted into Equation (10), which allows us to obtain the set of three parametric circles $\mathbb{P}_c = \{({}^i x_c, {}^i y_c, {}^i r_c) | i = 1,2,3\}$ for $t_i \in (t_0, t), i = 1,2,3$ with $t_i \in (t_0, t), i = 1,2,3$. There are up to 6 intersections between the 3 circles on the xOy plane, and according to the geometry of the problem, 3 of them intersect on the trajectory line at the desired end point of the trajectory. If we take any two parametric circles from them, say those corresponding to indices i and j , there will be two intersection points $J_{i,j}^+$ and $J_{i,j}^-$, which constitute the geometric properties of the system. It can be observed that there is a $\frac{1}{2}$ probability of selecting the potential common point (end point of the trajactory) from the intersection points provided by each pair of circles. Consequently, there are 8 possible combinations. It is preferable to use binary s_i (where the symbol $+$ represents the positive root and the symbol $-$ represents the negative root) to represent and define the intersection point combinations simultaneously.

$$\mathbb{C}_{s_2,s_1,s_0} = \{J_{1,2}^{s_2}, J_{2,3}^{s_1}, J_{3,1}^{s_0}\}, s_i \in \{+,-\} \tag{13}$$

However, there is only one combination belong to $\mathbb{C}_{s_2,s_1,s_0}$ that can be optimized to obtain the final optimization result. According to property 1, a certain intersection and common point between the parametric circle and the parametric circle, we get the 2-norm corresponding to the first part of the optimization objective function, and according to property 2, the common point is also the desired end point of the trajectory line, we can get the 2-norm corresponding to the second part of the objective function in Equation (14).

$$V_{s_2,s_1,s_0} = \underbrace{\sum_{J_{i,j}^s, J_{j,k}^s \in \mathbb{C}_{s_2,s_1,s_0}} \frac{1}{2} ||J_{i,j}^s - J_{i,j}^s||^2}_{V^o\text{:Norm with other intersecions}} + \underbrace{\sum_{J_{i,j}^s \in \mathbb{C}_{s_2,s_1,s_0}} \frac{1}{2} ||(x_f, y_f) - J_{i,j}^s||^2}_{V^d\text{:Norm with desired position}} \tag{14}$$

Thus, the parameter estimation problem becomes the one shown in Equation (15), which is solved so that the sum of two sqares of the 2-norms is taken as the smallest combination s_2, s_1, s_0 of intersection points $J_{1,2}^{s_2}, J_{2,3}^{s_1}, J_{3,1}^{s_0}$ and the corresponding target end state x_f, y_f, and θ_f. Obviously, the optimization goal V is positively definite, and if $\dot{V}$ is negatively definite, and the problem model is a convex optimization problem, these would ensures that any local minimum is also a global minimum.

Thus, the optimization problem for parameter estimation takes the form shown in Equation (15). That is, the sum of the distances between the intersections is minimal, and the sum of the distances between the intersections and the final state positions is also minimal. To make the sum of squares of the norm representing the distances equal to 0, two conditions need to be met simultaneously: 1. the selected intersection group $J_{1,2}^{s_2}, J_{2,3}^{s_1}, J_{3,1}^{s_0}$ is correct; 2. the solution is precisely the corresponding target ending states x_f, y_f, and θ_f.

$$\arg\min_{x_f, y_f, \theta_f} V_{s_2,s_1,s_0} = \arg\min_{x_f, y_f, \theta_f} (V^d + V^o) \tag{15}$$

3. Methods

As mentioned in the previous section, there are a total of 8 possible branches that need to be optimized and iterated to find the final optimal solution, so we propose the Pseudoinverse Gradient Descent Method with Eight Branch Directions (8B-PGDM). Since V is the sum of two sqares of the 2-norms, it is obviously a convex function, assuming that the intersections can be extended to unitary space, then it is not affected by the undefined imaginary intersection point, and the variable $x_f^*, , y_f^*, \theta_f^* \in \mathbb{R}$ can be considered unconstrained, then the optimization objective function V is a convex optimization problem. Since it is a convex optimization problem, common optimization solvers, such as Pseudoinverse Gradient Descent Method can be used to solve the problem.

3.1. Pseudoinverse Gradient Descent Method

Considering that the optimization target V is positively definite, then, as long as the appropriate $\delta x_f^*, \delta y_f^*, \delta\theta_f^*$ is taken to make $\Delta V(x_f^*, y_f^*, \theta_f^*)$ negatively determined, then it can converge asymptotically to 0 according to Lyapunov's theorem. The existence of the optimal solution is proven. Based on this existence result, we develop a method named Pseudoinverse Gradient Descent Method (PGDM), which leverages the properties of the problem. Specifically, the existence of the optimal solution guarantees that our algorithm has a theoretical target to converge to. Additionally, by exploiting the convergence properties of the solution, we design the update rules, step size adjustment strategies, and termination conditions of PGDM to ensure that it can efficiently find the optimal solution."

Since the dimension of $[\delta x_f^*, \delta y_f^*, \delta\theta_f^*]^T$ is 3, we need at least 3 linearly independent equations to find the inverse of the relevant matrix in order to solve for these unknowns. To this end, we consider expanding V_d and V_o in terms of the X and Y axes by expressing V_d and V_o as functions of the components on the X and Y axes through projection. Then, by taking derivatives of these functions with respect to $[\delta x_f^*, \delta y_f^*, \delta\theta_f^*]^T$, we obtain a system of equations. If the resulting coefficient matrix of this system of equations is not invertible, we can use the Moore-Penrose pseudoinverse to solve for the optimized gradient direction. This is because the pseudoinverse provides a least-squares solution, which is particularly useful when dealing with overdetermined or underdetermined systems of equations.

$$\begin{bmatrix} \delta x_f^* \\ \delta y_f^* \\ \delta\theta_f^* \end{bmatrix} = -\begin{bmatrix} \frac{\partial V_x^d}{\partial x_f^*} & \frac{\partial V_x^d}{\partial y_f^*} & \frac{\partial V_x^d}{\partial \theta_f^*} \\ \frac{\partial V_y^d}{\partial x_f^*} & \frac{\partial V_y^d}{\partial y_f^*} & \frac{\partial V_y^d}{\partial \theta_f^*} \\ \frac{\partial V_x^o}{\partial x_f^*} & \frac{\partial V_x^o}{\partial y_f^*} & \frac{\partial V_x^o}{\partial \theta_f^*} \\ \frac{\partial V_x^o}{\partial x_f^*} & \frac{\partial V_x^o}{\partial y_f^*} & \frac{\partial V_x^o}{\partial \theta_f^*} \end{bmatrix}^+ \cdot \begin{bmatrix} V_x^d \\ V_y^d \\ V_x^o \\ V_x^o \end{bmatrix} \tag{16}$$

According to the situation of the current branch, the pseudo-inverse of the Jacobian matrix at the current iterative solution of the Lyapunov function is used to perform a local linearization operation on the function. On this basis, the gradient iterative vector $[\delta x_f^*, \delta y_f^*, \delta\theta_f^*]$ is further obtained to make the error asymptotic convergence as shown in Equation (17). $[V_x^d, V_y^d, V_x^o, V_x^o]^T$ is the projection component vector of objective functions V^d and V^o at $[x_f^*, y_f^*, \theta_f^*]_k$. Thus, the solution for the next iteration is defined by Equation (17).

$$[x_f^*, y_f^*, \theta_f^*]_{k+1} = [x_f^*, y_f^*, \theta_f^*]_k + [\delta x_f^*, \delta y_f^*, \delta\theta_f^*] \tag{17}$$

where $[x_f^*, y_f^*, \theta_f^*]_k$ is the parameter vector of the k th iteration.

3.2. *Intermediate complexity of methods*

Since there may be no intersection of parameter circles during the iterative solution process, we consider discussing the complex roots. To handle this situation, we introduce the concept of the distance of complex roots in definition 2. In the intermediate steps of our algorithm, complex numbers are introduced to expand the solution space. However, the final solution is required to be in the real space. Therefore, we apply a specific mapping operation to transform the intermediate complex-valued iterative vectors back into the real space. This procedure reflects the Intermediate Complexity characteristics of the algorithm, which means that although the algorithm involves complex numbers in the intermediate stage, the final solution is restricted to the real domain.

Definition 1 (Distance in a Unitary Space)**.** *Let V_u be a unitary space over the complex field $\mathbb{C}$ with an inner product $\langle\cdot,\cdot\rangle : V_u \times V_u \to \mathbb{C}$ that satisfies the following properties:*

1. *For all $u, v, w \in V_u$ and $a, b \in \mathbb{C}$, $\langle au + bv, w\rangle = a\langle u, w\rangle + b\langle v, w\rangle$ (linearity).*
2. *$\langle u, v\rangle = \overline{\langle v, u\rangle}$ (conjugate symmetry), where $\overline{\langle v, u\rangle}$ denotes the complex conjugate of $\langle v, u\rangle$.*
3. *For all $u \in V_u$, $\langle u, u\rangle \geq 0$, and $\langle u, u\rangle = 0$ if and only if $u = 0$ (positive definiteness).*

Definition 2. *The distance $d(u, v)$ between two vectors $u, v \in V_u$ is defined as $d(u, v) = \|u - v\|$, where the norm $\|u\|$ of a vector u is given by $\|u\| = \sqrt{\langle u, u\rangle}$. In the case of an n-dimensional unitary space $V_u = \mathbb{C}^n$ with the standard inner product $\langle u, v\rangle = \sum_{i=1}^n u_i\overline{v_i}$, the distance formula can be explicitly written as $d(u, v) = \sqrt{\sum_{i=1}^n (u_i - v_i)\overline{(u_i - v_i)}}$.*

Next, assuming that the current branch is about the intersection of $J_{1,2}^+$, $J_{2,3}^+$, and $J_{3,1}^+$, we will take the elements in the Jaccobi matrix as an example to introduce how to extend the method to the complex root.

According to the chain rule, the partial derivatives item $\frac{\partial V_x^d}{\partial x_f^*}$ in the Jacobian matrix. In the following formula, we use two important symbols $\Re$ and $\Im$ related to complex numbers. The symbol $\Re(z)$ represents the real part and symbol $\Im(z)$ represents the imaginary part of a complex number $z = \Re(z) + i\Im(z)$. The distance $d(u, v)$ in definition 2 can be expressed as $d^2(u, v) = \Re^2(u - v) + \Im^2(u - v)$. Let $x_{i,j}^+$ represent the x-axis projection component of $J_{i,j}^+$.

$$\begin{aligned}
\frac{\partial V_x^d}{\partial x_f^*} &= \frac{1}{2}\sum_{i \neq j}^{i,j} \frac{\partial((x_f^* - \Re(x_{i,j}^+))^2 + \Im^2(x_{i,j}^+))}{\partial x_f^*} \\
&= \sum_{i \neq j}^{i,j} (x_f^* - \Re(x_{i,j}^+))(1 - \frac{\partial\Re(x_{i,j}^+)}{\partial x_f^*}) + \Im(x_{i,j}^+)\frac{\partial\Im(x_{i,j}^+)}{\partial x_f^*}
\end{aligned} \tag{18}$$

As for $\frac{\partial\Re(x_{i,j}^+)}{\partial x_f^*}$ and $\frac{\partial\Im(x_{i,j}^+)}{\partial x_f^*}$, they can be obtained by substituting Equation (12) and projecting $\mathbf{O}$, $\mathbf{e}_a$ and $\mathbf{e}_b$ on the x-axis.

$$\frac{\partial\Re(x_{i,j}^+)}{\partial x_f^*} = \frac{\partial(\frac{{}^i x_c + {}^j x_c}{2} + A \cdot \frac{{}^j x_c - {}^i x_c}{L} + \Re(B) \cdot \frac{{}^j y_c - {}^i y_c}{L})}{\partial x_f^*} \tag{19}$$

$$\frac{\partial\Im(x_{i,j}^+)}{\partial x_f^*} = \frac{\partial(\Im(B) \cdot \frac{{}^j y_c - {}^i y_c}{L})}{\partial x_f^*} \tag{20}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2025 doi:10.20944/preprints202503.1682.v1

Next, to get the full form of the partial derivative, it need to substitute $\frac{\partial^i x_c}{\partial x_f^*}, \frac{\partial^i y_c}{\partial x_f^*}, \frac{\partial^i r_c}{\partial x_f^*}$ from Equation (11a) and Equation (11b) into the above equation.

3.3. Convergence analysis

As explained at the beginning of this section, this problem is a convex optimization problem, and in order to ensure that the optimal solution can be approximated in the iterative solution process, it is necessary to use the derivative of the lyapunov function to prove the convergence of the error.

The negative definite property of the derivative of the lyapunov function can be obtained by its definition 3.

Definition 3 (Lyapunov Function Derivative in Discrete Time Systems). *Let $x_k \in \mathbb{R}^n$ be the state vector at the k-th iteration of a discrete-time dynamical system, and $g : \mathbb{R}^n \to \mathbb{R}^n$ be the state transition function such that $x_{k+1} = g(x_k)$. Given a Lyapunov function $V(x) : \mathbb{R}^n \to \mathbb{R}$, its derivative is approximated as follows. Let $\mathbf{J}_V(x)$ be the Jacobian matrix of $V(x)$, which is a $1 \times n$ row vector with elements $\mathbf{J}_V(x) = \left[\frac{\partial V}{\partial x_1}, \frac{\partial V}{\partial x_2}, \cdots, \frac{\partial V}{\partial x_n}\right]$. The derivative of the Lyapunov function, denoted by $\dot{V}(x_k)$, is defined as:*

$$\dot{V}(x_k) \approx V(x_{k+1}) - V(x_k)$$

By using the Taylor expansion of $V(g(x))$ at $x = x_k$, and let $\Delta x \triangleq g(x_k) - x_k$we have:

$$\begin{aligned} V(x_{k+1}) &= V(g(x_k)) \\ &\approx V(x_k) + \mathbf{J}_V(x_k)(\Delta x) + O(\|\Delta x\|^2) \end{aligned} \tag{21}$$

Thus, the first-order approximation of the Lyapunov function derivative is given by:

$$\dot{V}(x_k) \approx \mathbf{J}_V(x_k)(g(x_k) - x_k) \tag{22}$$

From the full differentiation of Equation (14), there is the form of the derivative of the lyapunov function $\dot{V}(\mathbf{x}_k)$. Substituting Equation (16) into Equation (22), still allowing $\mathbf{x}_k$ to represent the vector $[x_f^*, y_f^*, \theta_f^*]_k$, gives the derivative of the lyapunov function as shown in the following equation.

$$\begin{aligned} \dot{V}(\mathbf{x}_k) &\approx V_x^d \cdot \triangle V_x^d + V_y^d \cdot \triangle V_y^d + V_x^o \cdot \triangle V_x^o + V_x^o \cdot \triangle V_x^o \\ &\approx \left[\begin{array}{c} V_x^d, V_y^d, V_x^o, V_x^o \end{array}\right] \mathbf{J}_V(\mathbf{x}_k)(g(\mathbf{x}_k) - \mathbf{x}_k) \\ &\approx \begin{bmatrix} V_x^d \\ V_y^d \\ V_x^o \\ V_x^o \end{bmatrix}^T \mathbf{J}_V(x_k) \cdot -\mathbf{J}_V(x_k)^+ \begin{bmatrix} V_x^d \\ V_y^d \\ V_x^o \\ V_x^o \end{bmatrix} \\ &\approx -((V_x^d)^2 + (V_y^d)^2 + (V_x^o)^2 + (V_x^o)^2) \leq 0 \end{aligned} \tag{23}$$

At this point, the convergence has been proven.

3.4. 8B-PGDM

In the problem description and modeling section, we have analyzed that there are 8 branches in the combination of intersections, and only one of them can obtain the optimal estimate of the final state through iterative optimization due to the uniqueness of the solution required by geometric properties. We will elaborate on the optimization solution algorithm logic below in the case of 8 branches (Algorithm 1).

Initialization Step. The algorithm starts by setting up the necessary initial conditions. In general, first-order extrapolation can be used for the most recent sample points to obtain the initial state of the solution vector $[x_f^*, y_f^*, \theta_f^*]_0$ to be optimized. Considering the actual computing power and real-time requirements, the error tolerance ϵ and maximum iteration count N_{max} are also defined as stopping criteria.

Iterative Process. The while loop ensures that the algorithm continues until either the maximum number of iterations is reached or the error tolerance is met. The for loop allows the algorithm to explore different configurations of $s_2, s_1, s_0 \in \{+, -\}$, which might represent different combination of intersections in the optimization problem. The computation of $[\delta x_f^*, \delta y_f^*, \delta\theta_f^*]$ using Equation (16) is the key step for updating the parameters $[x_f^*, y_f^*, \theta_f^*]_k$ in the direction of optimization. The subsequent update step modifies the parameters based on these increments. The computation of V_{s_2,s_1,s_0} using Equation (14) is going to evaluating the objective function (a Lyapunov-like function) associated with the optimization problem. Selecting the minimum value of V_{s_2,s_1,s_0} on behalf of the error of the optimal solution among the 8 branches

Termination and Output. By judging whether the optimal solution in the 8 branches meets the convergence requirements ($V < \epsilon$), the iterative solution process is terminated. The stopping criteria $k > N_{max}$ ensure that the algorithm does not run indefinitely. Upon termination, the algorithm outputs the optimized values V_{s_2,s_1,s_0} of the parameters $[x_f^*, y_f^*, \theta_f^*]$, which represent the solution found by the optimization process.

Algorithm 1 Iterative Optimization Solver for 8B-PGDM

```
1:  Initialize $[x_f^*, y_f^*, \theta_f^*]_0$ (Initial guess)
2:  Set $\epsilon = 5$ (Error tolerance)
3:  Set $k = 0$(iteration counter)
4:  Set maximum number of iterations $N_{max}$
5:  while $k < N_{max}$ and $V > \epsilon$ do
6:      for all $s_2, s_1, s_0 \in \{+, -\}$ do
7:          Compute $[\delta x_f^*, \delta y_f^*, \delta\theta_f^*]$—Equation (16)
8:          Update $[x_f^*, y_f^*, \theta_f^*] \leftarrow [x_f^*, y_f^*, \theta_f^*] + [\delta x_f^*, \delta y_f^*, \delta\theta_f^*]$
9:          Compute $V_{s_2,s_1,s_0}$—Equation (14)
10:     end for
11:     $V = \min(V_{s_2,s_1,s_0})$
12:     $k = k + 1$
13: end while
14: Return $x_f, y_f, \theta_f$ and $s_2, s_1, s_0$
```

4. Results

To ensure the validity and reliability of these results, a well-designed experimental setup was employed. Here are the results obtained from our experiments and analyses.

4.1. Design of Experiments

Configuration of experimental conditions. In our experiment, it is assumed that there is a cruise missile, the initial position is at the origin $(0, 0)$, the attitude is parallel to the horizontal axis ($\theta = 0$), with a constant velocity $v = 400$, the optimal control law parameters $\mu = 4, \lambda = 2$, and the flight under optimal guidance is performed to the desired position $(4000, -40000)$ and the desired attitude of $\pi/3rad$. About numerical simulation, Continuous simulations were performed with a simulation step size of $1ms$ for a total of $300s$.

Control group settings for the experimental method. We employed four different methods as a control group in our experiment, they are 1st-order Dead Reckoning Algorithm(1st-order DR), and 2nd-order Dead Reckoning Algorithm(2nd-order DR). For the 1st-order DR and 2nd-order DR methods, two different experimental conditions were set up. In the first condition, the update period was fixed at 1 second, and in the second condition, the update was triggered based on an error threshold of $10m$, while our method (8B-PGDM) was tested based on three sampling points at $0.001s$, $15.001s$ and $30.001s$, without specifying an update period nor error threshold.

4.2. Method validation

To verify our method, the experimental design was structured in the following way. Firstly, we use continuous system simulation to obtain the motion trajectory under optimal guidance control (Blue curves in Figure 6). Then, based on three sampling points, applying the 8B-PGDM method, the 8 branches are optimized and iterated synchronously, and from the error convergence decline in Figure 7, it can be seen that the logarithmic curve of the 7 th (–+) iterative error converges to the lowest point, and the corresponding cases of the branch line are the negative intersection of the first parameter circle and the second parameter circle ($s_2 = -$), the negative intersection point of the second parameter circle and the third parameter circle ($s_1 = -$), the orthogonal point of the third parameter circle and the first parameter circle ($s_0 = +$), and the error function (Lyapunov) when the termination condition occurs converges to 1.2562. In contrast, the other branches do not have an optimal solution where the error converges to within the 0 neighborhood. And the estimated values of the parameter vector $[x_f^*, y_f^*, \theta_f^*]$ is $[3998, -39999, 1.0471]$.

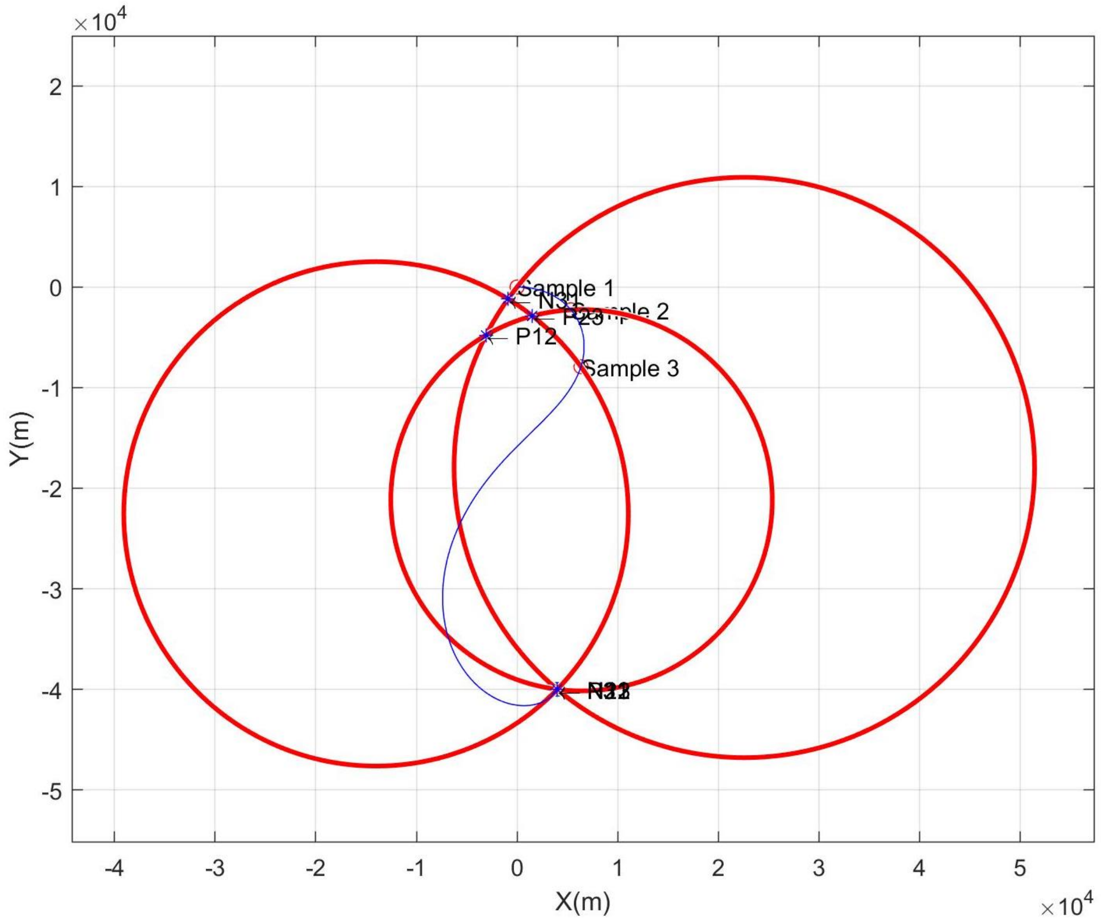

Figure 6. In the numerical simulation, the blue curve is the trajectory line, the small red ring on the trajectory line is the sampling point, and the three large circles are the parameter circles that are solved and intersect at the desired end point of the trajectory.

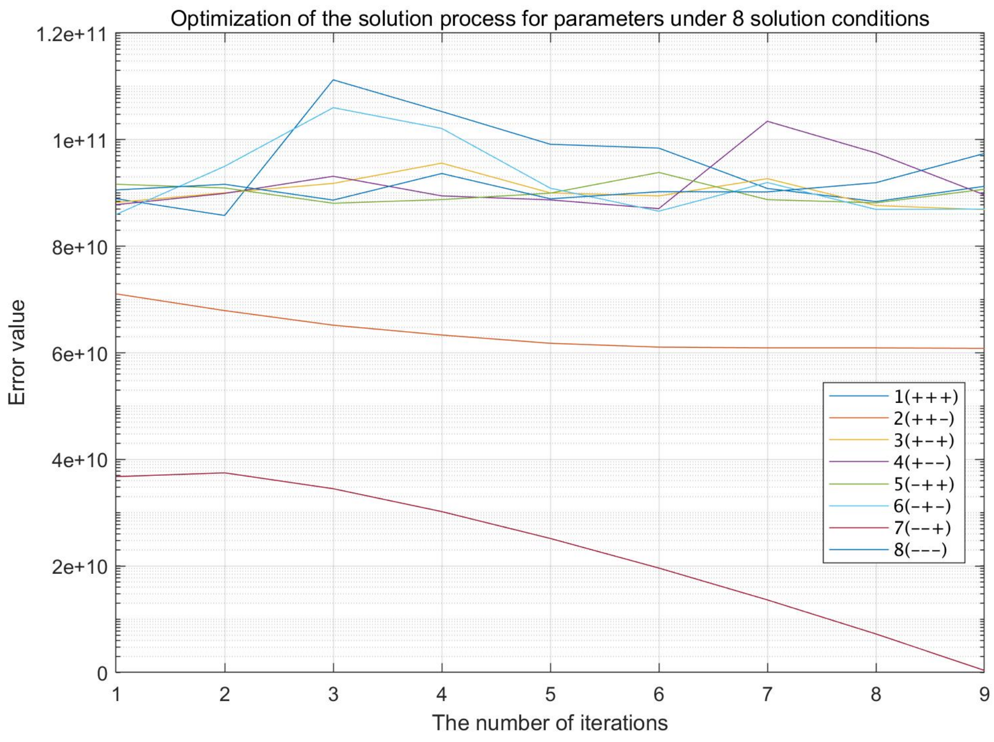

Figure 7. The optimization iterative process of the 8 branches of 8B-PGDM. Only the 7 th branch has significantly converged to $\log_{10}(1.2562)$ after 9 iterations.

At this point, we have verified the effectiveness of our method, which can determine the intersection point combination of parameter circles and the corresponding motion state estimation parameters in the process of synchronous optimization of 8 branches. In addition, the optimal solution was reached in only 9 iteration cycles, and the convergence speed was fast.

4.3. Superiority contrast

The following presents an analysis of the experimental results obtained from the different methods evaluated in our study. As shown in Table 1, we compared several methods, including our proposed method (8B-PGDM) and different orders of Dead Reckoning (DR) methods, under various conditions.

For our method, according to the parameter estimation ($x_f^* = 3998, y_f^* = -39999, \theta_f^* = 1.0471rad$) obtained by the 8B-PGDM, the flight trajectory of the missile can be obtained by introducing the guidance law of Formula in Equation (24) to a inner continuous system simulation process as Equation (4).

$$\begin{aligned} \dot{\theta} &= 4 \cdot \dot{q} + 2(1.0471 - q) \cdot \frac{\dot{r}}{r} \\ q(0) &= tan^{-1}\frac{-39999 - y(0)}{3998 - x(0)} \\ r(0) &= \sqrt{3998^2 + (-39999)^2} \end{aligned} \tag{24}$$

As for the control group, the current state is estimated by extrapolation in the same way as the step size $1ms$ of the continuous system simulation, and the motion state of the object is obtained through observation or communication only when the update condition is triggered (threshold $\leq 10m$ or $T = 1s$).

The Mean Squared Error (MSE) values of different methods provide crucial insights into their performance. As shown in Table 1, our proposed method, 8B-PGDM, achieves an MSE of 0.94459, which is remarkably lower than those of the other methods. In contrast, the 1st-order DR method shows varying MSE values depending on the update criteria. When the update criteria is a fixed period of $1s$, the MSE is 18.053. However, when the update is triggered by the condition , the MSE increases slightly to 19.913. The 2nd-order DR method exhibits a significantly higher MSE of 53176 when the update criteria is a fixed period of $1s$, suggesting poor performance under this condition. Interestingly, when the update is based on the condition, the 2nd-order DR method has the same MSE as the 1st-order DR method under the same condition, with a value of 19.913. This comparison indicates that the 8B-PGDM method outperforms both the 1st and 2nd-order DR methods in terms of minimizing the MSE, regardless of the update criteria. The large MSE of the 2nd-order DR method under the fixed $1s$ update criteria might imply that the 2nd-order approximation is not effective under this setting. Meanwhile, the similar MSE of both 1st and 2nd-order DR methods under the condition suggests that the order of approximation may not be the sole factor influencing the MSE, and other factors related to the update criteria could have a more significant impact.

The number of samples required by each method also reveals important information about their computational cost, observation and communication needs. As presented in Table 1, the 8B-PGDM method uses only 3 samples, which is considerably fewer than the sample times of the other methods. The 1st and 2nd-order DR methods both require 146 samples when the update criteria is a fixed period of $1s$, indicating a higher computational cost associated with this update strategy. When the update is based on the condition ($\epsilon > 10m$), both methods use 122 samples, showing that the error-based update criteria reduce the number of samples required compared to the fixed $1s$ update criteria. The substantial difference in sample times between the 8B-PGDM method and the DR methods suggests that the 8B-PGDM method is more efficient in terms of computational cost, especially considering its relatively low MSE. The fact that the DR methods require more samples, especially under the fixed $1s$ update criteria, implies that more frequent sampling might be necessary to maintain a certain level of performance, whereas the 8B-PGDM method can achieve good performance with far fewer samples.

In summary, compared with other traditional methods in Figure 8, our method has the advantage of having fewer sampling times, smooth curve estimation and small cumulative error.

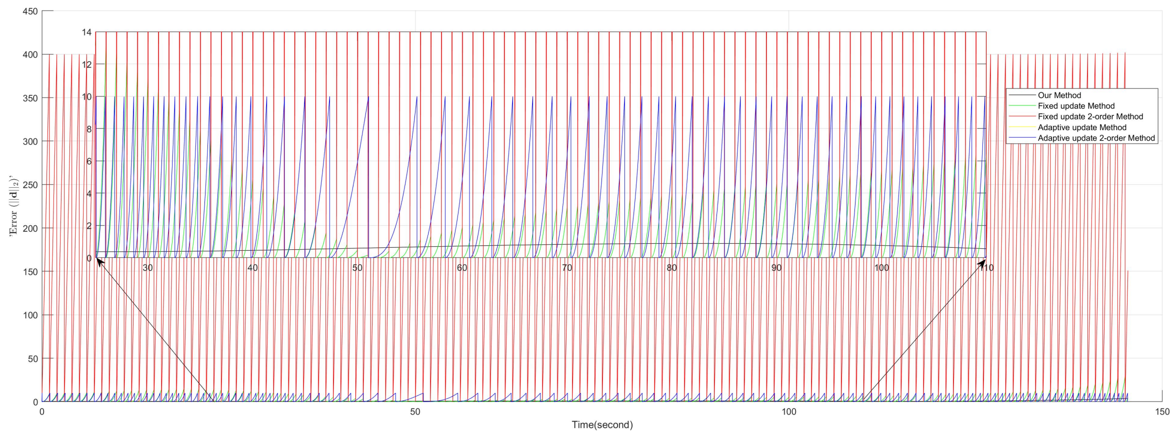

Figure 8. Comparison of our method with traditional methods, our 8B-PGDM has advantage in sampling times, smooth curve estimation, and small cumulative error.

Table 1. Experimental Results Comparison of Different Methods

Method	Update Criteria	Samples	MSE
8B-PGDM	-	3	0.94459
1st-order DR	1s	146	18.053
2nd-order DR	1s	146	53176
1st-order DR	$\epsilon > 10m$	122	19.913
2nd-order DR	$\epsilon > 10m$	122	19.913

5. Conclusion

In this paper, we conduct an in-depth analysis of the relationship between the local time invariance of motion parameters and the performance of the corresponding dead-reckoning algorithm. Given our awareness that the expected state exhibits superior time invariance compared to the temporary state, we devise a geometric approach. This approach transforms the parameter-estimation problem into a convex optimization problem. To solve the optimization problem and accurately estimate the expected state, we employ a pseudoinverse gradient descent method featuring 8 branch directions. We then validate our proposed method through an example, where four other classic dead reckoning algorithms are set as the control group.

In terms of effectiveness and significance, the following aspects can be highlighted. Firstly, the method is capable of achieving trajectory predictions with fewer sampling times, which is particularly advantageous in scenarios where the target is occluded or communication is intermittent. This feature enables the method to operate effectively under challenging conditions, ensuring reliable performance even when data acquisition is limited. Secondly, it can yield trajectory predictions with a smaller mean squared error. A lower mean squared error indicates a higher accuracy of the predictions, thereby improving the overall quality and reliability of the results obtained through the method. Thirdly, the predicted trajectories are smooth without jaggedness. Smooth trajectories are more desirable in many applications as they conform better to physical realities and are easier to interpret and utilize, enhancing the practical value of the predicted results. Finally, the iterative solution speed of the method is fast. Rapid iteration enables quick convergence and timely results, making the method suitable for time-sensitive applications where real-time or near-real-time processing is required, thereby enhancing its efficiency and responsiveness in dynamic situations.

The current method still exhibits several shortcomings in research. Firstly, it necessitates parallel iterative optimization across the eight branches. This not only augments the computational complexity but also imposes higher demands on hardware capabilities for parallel processing, thereby potentially restricting its practical applicability in resource-constrained environments. Such a requirement could pose challenges in systems with limited computational resources, impeding its seamless integration and deployment. Secondly, the method is strictly tailored to the corresponding guidance law, with the precondition that certain parameters, specifically mu and lambda, must be known. This inherent constraint significantly limits the method's universality. In real-world scenarios, accurately obtaining these parameters can be quite challenging, and moreover, the guidance law might vary depending on diverse mission requirements. This lack of flexibility may undermine the method's versatility and applicability in different operational contexts. Thirdly, a sensitivity analysis of this method regarding the noise present in the observations has not been performed. In practical applications, noise is an inescapable factor, and neglecting its influence can lead to inaccurate results and a degradation in the algorithm's reliability. Consequently, future research efforts should be concentrated on addressing these issues to enhance the method's overall effectiveness and practicality, thereby facilitating its broader adoption and more reliable performance under various conditions.

For the next steps, the following plans are envisioned. Firstly, a sensitivity analysis of the method will be conducted regarding the noise present in the observations. This analysis aims to investigate how the method's performance is affected by different levels and types of noise, providing valuable insights into its robustness and reliability under noisy conditions. Secondly, generalization studies will be carried out on other motion patterns and the time invariance of their associated parameters. By exploring a wider range of motion patterns, we hope to extend the applicability of the method and deepen our understanding of the time invariance properties in various dynamic systems, which could potentially lead to more comprehensive and versatile models. Finally, an attempt will be made to transform the method into a neural network unit. This unit will be designed for online identification of motion types and parameters, followed by online prediction. The neural network architecture will leverage its powerful learning and generalization capabilities to continuously adapt to different motion characteristics, enabling real-time motion recognition and prediction, thereby enhancing the method's utility in dynamic and complex environments.

Acknowledgments

I would like to express my sincere gratitude to all those who have supported me throughout the course of this research. First and foremost, I am deeply indebted to my advisors and colleagues at NUDT for their valuable guidance, insightful suggestions, and continuous encouragement. Their expertise and experience have been invaluable in shaping the direction and quality of this work. Special thanks also go to my friends and family for their understanding and patience during the long hours of research work. Their moral support and love have been a constant source of motivation, helping me to overcome numerous difficulties and persevere through the challenging times. Finally, I would like to acknowledge the assistance of all those who participated in the experiments, provided data, or offered their time and resources, without which this research could not have been completed.

Author Contributions: Conceptualization, J.G. and J.H.; methodology and writing—original draft, J.G.; writing—review and editing, Q.L., H.C., H.D., L.Z., L.S. and J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: While developing this architecture and the corresponding platform, the open-source community provides the model objects, and all enthusiasts contribute to robot systems that lay a good foundation for the design and implementation of our architecture. We would also thank Professor Huang for allowing us to conduct system performance testing in the multi-robot system simulation course, as well as to the 12 students who provided the hardware equipments and took the time to complete the test.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. L. Li, Z. Wei, T. Zhang, and D. Cai, "Three-dimensional dead reckoning of wall-climbing robot based on information fusion of compound extended kalman filter," *Journal of Field Robotics*, pp. 505–520, 2023.
2. P. Gogendeau, S. Bonhommeau, H. Fourati, D. D. Oliveira, V. Taillandier, A. Goharzadeh, and S. Bernard, "Dead-reckoning configurations analysis for marine turtle context in a controlled environment," *IEEE Sensors Journal*, pp. 12 298–12 306, 2022.
3. G. RM, H. MD, S. DM, H. P, S. ELC, F. AJ, G. B, Q. F, G.-L. A, Y. K, Y. T, E. H, F. S, G. D, V. P, B. A, van Schalkwyk OL, C. NC, T. V, B. L, R. J, B. SH, M. NJ, B. NC, T. MH, W. HJ, D. CM, van Rooyen MC, B. MF, T. CJ, and W. RP, "How often should dead-reckoned animal movement paths be corrected for drift?" *Animal biotelemetry*, p. 43, 2021.
4. J. Kuang, T. Liu, Y. Wang, X. Meng, and X. Niu, "Magnetic vector constraint pedestrian dead reckoning based on foot-mounted and waist-mounted imu," *IEEE Internet of Things Journal*, p. 1, 2025.
5. H. Li, H. Guo, Y. Qi, L. Deng, and M. Yu, "Research on multi-sensor pedestrian dead reckoning method with ukf algorithm(article)," *Measurement: Journal of the International Measurement Confederation*, p. 108524, 2021.
6. M. Brossard, A. Barrau, and S. Bonnabel, "Ai-imu dead-reckoning," *IEEE Transactions on Intelligent Vehicles*, pp. 585–595, 2020.

7. Z. Niu, L. Cong, H. Qin, and S. Cao, "Pedestrian dead reckoning based on complex motion mode recognition using hierarchical classification," *IEEE Sensors Journal*, pp. 4935–4947, 2024.
8. D. Hurwitz, N. Cohen, and I. Klein, "Deep-learning-assisted inertial dead reckoning and fusion," *IEEE Transactions on Instrumentation and Measurement*, pp. 1–9, 2025.