Journal of AppliedMath 2023; 1(1): 42.
Communication

Novel learning for control of nonlinear spacecraft dynamics

Bo-Ruei Huang[1], **Timothy Sands**[2*]

[1] *Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, New York 14853, USA.*
[2] *Department of Mechanical Engineering (SCPD), Stanford University, Stanford 94305, USA. E-mail: dr.timsands@alumni.stanford.edu*

ARTICLE INFO

Received: 10 February 2023
Accepted: 4 March 2023
Available online: 12 April 2023

http://dx.doi.org/10.59400/jam.v1i1.42

ABSTRACT: With accurate dynamic system parameters (embodied in self-awareness statements), a controller can provide precise signals for tracking desired state trajectories. If dynamic system parameters are initially guessed inaccurately, a learning method may be used to find the accurate parameters. In the deterministic artificial intelligence method, self-awareness statements are formed as mathematical expressions of the governing physics. When the nonlinear, coupled expressions are precisely parameterized as the product of known matrix components and unknown vectrix (i.e., an intermediate between a dyadic and a matrix in regression form) tracking errors may be projected onto the known matrix to update the unknown vectrix in an optimal form (in a two-norm sense). In this work, a modified learning method is proposed and proved to have global convergence of both state error and parameter estimation error. The modified learning method is compared with those in the prequels using simulation experiments of three-dimensional rigid body dynamic rotation motion. The achieved state error convergence using the modified approach is two magnitudes better than using the methods in the prequels.

KEYWORDS: nonlinear systems; mechanics; spacecraft attitude control; deterministic artificial intelligence; regression; learning

1. Introduction

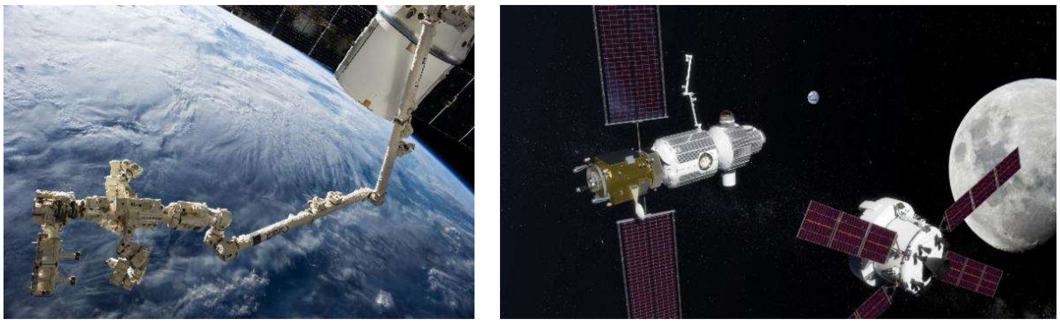


Figure 1. (**a**) The International Space Station's Canadarm2 and Dextre carry the RapidScat instrument assembly after removing it from the trunk of the SpaceX Dragon cargo ship (upper right), which is docked at the nadir port of the Harmony node. (**b**) NASA Gateway would support a growing space economy photos taken from [1] and [2] respectively in compliance with NASA's image use policy[3].

Consider intricate robotic operations in low-earth orbit near the space station as displayed in **Figure 1**, where considerable human intervention is available. Next, contemplate the requirements to autonomously do such operations in far distant cis-lunar orbits. The latter system must be able to learn in real-time dynamic changes that occur when the space robot grasps and grapples targeted spacecraft. Dynamics and control issues associated with rendezvous in Cis-lunar space near rectilinear halo orbits were investigated in [4], where a fully-safe, automatic rendezvous strategy was developed between a passive vehicle and an active one orbiting around the Earth-Moon L2 Lagrangian point. Bando *et al.*[4] proposed a chattering attenuation sliding mode control utilizing the eigen structure of the linearized flow around a libration point of the Earth-Moon circular restricted three-body problem, and this novel article serves as a reminder of the prevalence of linearization when dealing with multiple, coupled nonlinear equations. In 2021, Colombia presented a guidance, navigation and control framework for 6 degrees of freedom (6DOF) coupled Cislunar rendezvous and docking, and the article highlighted the importance of dealing with full, coupled translational-rotational dynamics of multi-body (i.e., highly flexible) dynamics seeking guaranteed coupled-state estimation[5]. Immediately that same year[6], new techniques for highly flexible multi-body space robotics were proposed as a competing narrative to the just-proposed “whiplash compensation” of flexible space robotics[7] establishing a thread of research offered by Cornell University. China now has two robotic arms attached to its space station[8], where a large robotic arm can “crawl” along the outside of the spacecraft[9].

An alternative thread of research is offered by Massachusetts Institute of Technology[10–16]. Noting that ubiquitous approaches rely on either simplifying assumptions in the dynamical model or on abundant computational resources, Lafarge *et al.*[10] proposed reinforcement learning for closed-loop control of onboard low-thrust guidance. Albee *et al.*[11] studied active interception of targets for autonomous repair and deorbiting must account for the tumbling motion of targets, which is oftentimes not known a priori. A model reference adaptive algorithmic approach was proposed to identify the state of the target’s tumble. In a more typical manner, Mehta *et al.*[12] proposed a quasi-physical dynamic reduced-order model that used a linear approximation of the underlying dynamics and effect of the drivers where data assimilation and model calibration utilized estimation of the model coefficients that represent the model parameters. One sequel article about autonomous docking with rotating targets via reinforcement learning was offered by Oestreich *et al.*[13] proposing learning policies. Following the initial target search[14], analytical closed expressions to compute the minimum distance between any two satellites (at the same altitude in circular orbits), Avendaño *et al.* proposed “flower constellations” to produce give an efficient method to compute the minimum angular distance between satellites. Reversing the method, Arnas *et al.*[15] proposed two-dimensional lattice flower constellations to design a low earth orbit slotting system to avoid collisions between compliant satellites (rather than intercept). Oestreich *et al.*[16] also highlighted dependence on on-orbit inspection (i.e., relative navigation and inertial properties estimation) to intercept tumbling debris objects or defunct satellites. In a late proposal following the M.I.T. approach, the master’s thesis by Roberts[17] continued to develop the stochastic artificial intelligence approach embodied in supervised learning. Ekal *et al.*[18] highlight key parametric uncertainties are mass and moment of inertia, and the Cornell line of research also adopts this premise.

Another line of work is presented by Stanford University[19–21]. Cassinis *et al.*[19] introduced an adaptive convolutional neural network–based unscented Kalman filter for the pose estimation of uncooperative spacecraft. Park *et al.*[20] followed the same approach using a shared multi-scale feature encoder and multiple prediction heads that perform different tasks on a shared feature output, while Park *et al.*[21] also followed a comparative line similar to the Cornell approach presented in this manuscript, where the (to be proposed) deterministic approach is supplemented by an adaptive neural network-based unscented Kalman filter.

Cornell's Zhang *et al*. proposed an adaptive control strategy based on the full, nonlinear equations accounting for modeling uncertainties using an adaptive neural network amidst external disturbances[22], where the Cornell approach stems from naval approaches proposed in 2020, called deterministic artificial intelligence[23], which stated that the system dynamics constitute a feedforward control when paired with analytic trajectories; and when the dynamics are expressed in a canonical regression form, optimal feedback (in the two-norm sense) can aid control of spacecraft attitude. The method stems from the incremental development of a common nonlinear adaptive scheme offered by Slotine[24] for spacecraft attitude control, where elements of classical feedback were eliminated in 2020 and foremost applied to unmanned underwater robotics[25]. The burgeoning lineage of research continued in 2022 when Sandberg *et al*.[26] compared several trajectory-generation schemes and a nominal learning method based on the regression model, where applied torque is estimated by an enhanced Luenberger observer. Very shortly afterwards, Raigoza[27] augmented Sandberg's trajectory generators with autonomous collision avoidance. In November 2022, Wilt examined efficacy in the face of simulated craft damage and environmental disturbances[28]. This sequel substantiates a short communication presenting significant findings that are part of the larger study of Slotine, Sands, Smeresky/Rizzo, Sandberg, Raigoza, and Wilt.

In prequel works[23–28], the error convergence property is obtained using the proper design of the trajectory generation process. However, if the external disturbance makes the current state deviate from the trajectory, even if the system parameter is already converged to an accurate value, the trajectory will need to be re-calculated to fit the current state, so that the deterministic artificial intelligence can continue to drive the system using an optimal feedforward control signal.

As a result, provided the initial error between the current state and the current desired trajectory as well as inaccurate initial parameter value, the goal of the modified learning approach proposed in this manuscript is to guarantee the convergence to zero of both parameter error and the state error. This work focuses on the rotation rate control problem of a spacecraft and provided 2 ways of modification to the learning phase of the deterministic artificial intelligence algorithm and compared them with the original deterministic artificial intelligence using simulation in MATLAB®. Moreover, the modified method can be proved to make the error converge to zero using a similar way as how Slotine and Li[24] proved the stability of the non-linear system controlled by some specific feed-forward/feed-back controllers. That is, the Lyapunov candidate function is provided, and the time derivative of the candidate function can be proved to be negative with the proposed modified learning method.

Main contribution of the study. This paper provides 2 novel unknown parameter learning methods, that is, the time derivative of the vector of unknown, which are able to not just bound the error in parameter estimations but also the difference between the current system state and the desired state with respect to the planned trajectory. For the second method proposed, we will further show the convergence of parameter estimation error, as well as how this leads to the convergence of the state tracking error. The paper also discussed how the provided methods may fail to converge under certain conditions.

2. Materials and methods

2.1 Spacecraft rotation rate control

The spacecraft rotation rate control problem focuses on applying torque so that the rotation rate of a spacecraft converges to the desired value. The dynamic can be described by the Euler equation (displayed in equation (1)). Euler's moment equations can be parameterized in canonical regression form. This full form of the coupled, nonlinear equations whose importance was highlighted by the research cited in the Introduction.

$$\tau = I\dot{\omega} + \omega \times I\omega = \underbrace{\begin{bmatrix} \dot{\omega}_x & \dot{\omega}_y - \omega_x\omega_z & \dot{\omega}_z + \omega_x\omega_y & -\omega_y\omega_z & \omega_y^2 - \omega_z^2 & \omega_y\omega_z \\ \omega_x\omega_z & \dot{\omega}_x + \omega_y\omega_z & \omega_z^2 - \omega_x^2 & \dot{\omega}_y & \dot{\omega}_z - \omega_x\omega_y & -\omega_x\omega_z \\ -\omega_x\omega_y & \omega_x^2 - \omega_y^2 & \dot{\omega}_x - \omega_y\omega_z & \omega_x\omega_y & \dot{\omega}_y + \omega_x\omega_z & \dot{\omega}_z \end{bmatrix}}_{\Phi} \underbrace{\begin{Bmatrix} I_{xx} \\ I_{xy} \\ I_{xz} \\ I_{yy} \\ I_{yz} \\ I_{zz} \end{Bmatrix}}_{\Theta} \tag{1}$$

The matrix Φ is the matrix of known, which is composed of the current state and the rate of the state (ω and $d\omega/dt$). The matrix Θ is the vector of the unknown, which is composed of system parameters, in this case, the moment of inertia. The way of formulation shows that it is possible to estimate the moment of inertia with the accurate measurement of the current state.

2.2 Original deterministic artificial intelligence control

The idea of deterministic artificial intelligence is that if the matrix of the unknown can be estimated and the desired trajectory of the state is given, the optimal control signal will be multiplying the desired matrix of known (Φ_d), which includes the information of the current desired state, with the best guess of the parameter ($\hat{\Theta}$). This turns the system dynamic to equation (2).

$$\tau = \Phi\Theta \ \rightarrow \tau_{applied} \equiv \Phi_d\hat{\Theta} \tag{2}$$

However, the $\hat{\Theta}$ can be inaccurate or changed in the middle of the operation. Therefore, a learning approach should be provided so that the vector of the unknown can converge to an accurate value. The original learning approach in the space rotation rate control problem is described in equation (3-a) and (3-b), which is provided by Smeresky *et al.*[12] and is equation (12) in his publication.

$$learned\ difference\ d \equiv \Theta - \hat{\Theta} = \Phi^H\left(\tau_{applied} - \Phi\hat{\Theta}\right) \tag{3-a}$$

$$\frac{d\hat{\Theta}}{dt} = a * d \tag{3-b}$$

Where $\tau_{applied}$ is the controller torque output, and the capital H means the pseudo inverse of a non-square matrix. In short, this provided a way to turn the difference between the applied torque and the expected torque into the parameter error d, which should be a minimal square error estimation using the information in the current time stamp. Concerning the stability of the parameter estimation, the learning of the parameter is applied incrementally, and this can be done using a first order low pass filter to smoothen the learned difference.

Table 1. Symbols used in section 2.2

Variable	Physical meaning	Variable	Physical meaning
Θ	Vector of unknown	I	Moment of inertia
$\hat{\Theta}$	Estimation on the unknown	ω	Angular speed vector
Φ_d	Matrix of known made by trajectory	d	Learned difference
Φ	Matrix of known	a	Filter time constant
τ	Applied torque		

Additionally, deterministic artificial intelligence requires a trajectory generation process to produce a trajectory that leads from the current state to the desired state. If the current state deviates undesirably from the trajectory, it is better to update the trajectory, or the error of the state may accumulate. Please be aware that the desired state of the trajectory generation is not the desired state of the controller, which follows the output of the trajectory generator by making the trajectory the desired state of the controller should follow. In this manuscript, all the "desired states" mentioned are the desired state for the controller, if not specifically noted.

The overall deterministic artificial intelligence can be expressed with the combination of control feedforward based on a desired trajectory as well as the current best estimation on the vector of unknown and a "learning" mechanism that updates the vector of unknown until it goes to the actual value. **Figure 2** presents the deterministic artificial intelligence as a block diagram and shows the relationship between each component. In sections 2.3 and 2.4, the discussion focuses on the learning part of deterministic artificial intelligence and the goal is to learn the vector of unknown and decrease the tracking error at the same time.

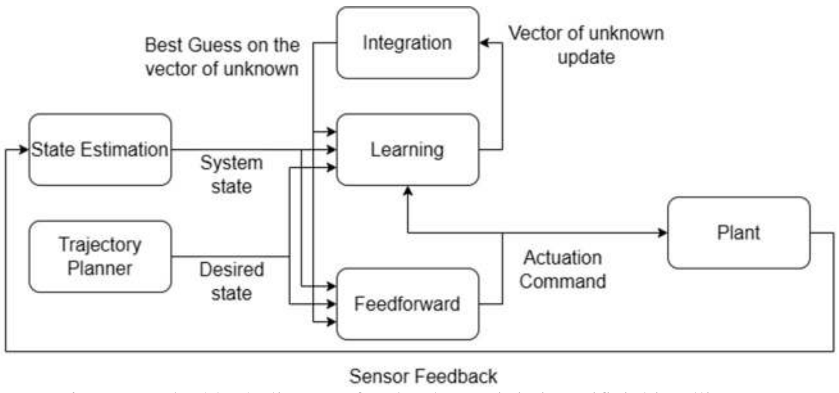


Figure 2. The block diagram for the deterministic artificial intelligence.

2.3 Modified learning method, a general version

The target of the modification is that if the learning approach can also guarantee to decrease the error in the current state when doing the parameter estimation, the chance of regenerating trajectory can be decreased because the error is kept from growing, which increases the robustness. In a general version of the modification, we consider all the systems that can be expressed in the regression form, as in equation (2), where the information of the current state is provided in the matrix of known. To study the error of the parameters and state, the error between the desired matrix of known and the current matrix of known is noted as ϕ, and the error of the unknown vector is noted as θ. Equation (2) can therefore be turned into equation (4). In this case, the goal becomes keeping both ϕ and θ bounded simultaneously using a modified learning method.

$$\Phi\theta + \phi\hat{\Theta} = 0 \; where \; \phi = \Phi_d - \Phi \; and \; \theta = \Theta - \hat{\Theta} \tag{4}$$

Considering the Lyapunov candidate function described in equation (5), the function value must decrease to 0 if both ϕ and θ go to 0. If there is a parameter update approach $\dot{\theta}$ that makes the candidate function globally stable, it is very likely that the error of the state ϕ goes to 0 together with θ. Equation (7) shows that if $\dot{\theta}$ is taken in the form of equation (6), and considering equation (4) and the time derivative of equation (4), the time derivative of the Lyapunov function will be negative semidefinite and leads to the global boundedness of the system as long as the matrix G is positive definitive.

$$V = \hat{\Theta}^T\phi^T\phi\hat{\Theta} + \theta^T\theta \tag{5}$$

$$\dot{\theta}^T = -\frac{d\widehat{\Theta}^T}{dt} = -\hat{\Theta}^T\phi^T((\Phi\dot{\Phi}^H)(\Phi^H + \Phi^T)^H + (\Phi^H + \Phi^T)^T G) \tag{6}$$

$$\frac{\dot{V}}{2} = \left[\hat{\Theta}^T \phi^T \Phi \dot{\Phi}^H + \dot{\theta}^T \Phi^H + \dot{\theta}^T \Phi^T\right] \phi \hat{\Theta} = -\hat{\Theta}^T \phi^T (\Phi^H + \Phi^T)^T G (\Phi^H + \Phi^T) \phi \hat{\Theta} \leq 0 \tag{7}$$

However, this candidate function only provided the boundedness of $\phi\hat{\Theta}$ and θ, and the derivation of equation (7) requires the matrix of known to be full rank. Furthermore, the convergence of $\phi\hat{\Theta}$, even if it happens, is not equivalent to the convergence of the state even if the matrix of known is full rank. For example, for the target application in this manuscript (equation (1)), the rank of the matrix of known is at most 3, while the parameter number in the vector of unknown is 6, this makes the learning method provided unable to guarantee convergence. It is possible that when the unknown parameter converges to an accurate value and the state error still exists at the same time, the state error will not be going to be zero. This can be seen in equation (2) that when $\hat{\Theta} = \Theta$, the term $\hat{\Theta}^T \phi^T = \hat{\Theta}^T (\Phi_d - \Phi)^T$ will always be 0. When ϕ has a smaller rank than the number of unknowns, it is possible that $\hat{\Theta}^T \phi^T = 0$ when ϕ is not zero.

Another concern of using this method is that the calculation of $\dot{\Phi}$ is prone to noises and will cause latency in the real-time calculation because it requires the knowledge of the double derivative of the rotation rate, which generally requires special treatments like the smoothing process.

All in all, this version of modification will not guarantee the convergence of the state tracking error, so a closer inspection of the system dynamic, rather than a generalized "matrix of known times vector of unknown" formulation, may be necessary, and will be shown in section 2.4.

Table 2. Symbols used in section 2.3

Variable	Physical meaning	Variable	Physical meaning
ϕ	Error in matrix of known	V	Lyapunov candidate function
θ	Error in vector of unknown	G	Arbitrary positive definite matrix

2.4 Modified learning method, a specific version

To avoid the problem mentioned in section 2.3, a specific version of the modified learning method is provided for the rotation rate controller. The non-regression form of the system dynamic is considered in equation (8), and the modified learning method is provided in equation (10) which utilizes both the state error as well as parameter error. Also, the character "*i*" means the error in the inertia matrix in a 3 × 3 form rather than in a 1 × 6 unknown vector. The torque input to the system is slightly modified from $\omega_d \times \hat{I}\omega_d$ to $\omega_d \times \hat{I}\omega$, which improves the global stability but won't affect the feed forward optimality in the deterministic artificial intelligence much when the state is very close to the desired value. (Or defined as applied torque (equation (8) in [12])).

$$I\dot{\omega} + \omega \times I\omega = \tau_{applied} \rightarrow \tau_{applied} \equiv \hat{I}\dot{\omega}_d + \omega_d \times \hat{I}\omega$$
$$Also, define\ i = I - \hat{I}\ and\ \omega' = \omega_d - \omega \tag{8}$$

$$I\dot{\omega}' = -(\omega' \times I\omega_d - \omega' \times I\omega') + (i\dot{\omega}_d + \omega_d \times i\omega_d - \omega_d \times i\omega') = \omega' \times C + K\theta \tag{9}$$

$$\dot{\theta} = -\frac{d\hat{\Theta}}{dt} = -Q\omega' - R\theta \tag{10}$$

The equation (8) is rearranged to equation (9), and the θ, again, means the inertia in an unknown vector form. To prove the global convergence of both state error ω' and parameter error θ, another Lyaponuv function (equation (11)) is provided, which has a physical meaning close to the square error of the whole system, where the state square error is weighted by the inertia. If the Q term in equation (10) is the transpose of the K term in equation (9), and the R term in equation (10) is positive definite, the Lyapunov function will be bounded globally, as shown in equation (12). About the parameter vector θ, it is chosen based on equation (13), which is the least square estimation same as the equation (3-a), and the value is used for the modified learning method in equation (10).

Finally, the parameter vector θ can be shown to converge in this case by applying Barbalat's lemma. Since the candidate function V is bounded, by equation (12) both tracking and estimation error is bounded, the desired trajectory has to be bounded, and K is a continuous function of $\omega', \omega_d,\ and\ \dot{\omega}_d$, it can be concluded that $\frac{\ddot{V}}{4}$ is bounded, which makes $\dot{V}$ converges. As a result, the estimation error is convergence.

$$V(\omega',\theta) = \omega'^T I\omega' + \theta^T\theta \tag{11}$$

$$\begin{aligned}\frac{\dot{V}}{2} &= \omega'^T I\dot{\omega}' + \theta^T\dot{\theta} = \omega'^T(\omega' \times C) + \theta^T(K^T - Q)\omega' - \theta^T R\theta \\ &= \theta^T(K^T - Q)\omega' - \theta^T R\theta = -\theta^T R\theta \le 0\end{aligned} \tag{12}$$

$$\theta = \Phi^H(\tau_{applied} - \Phi\hat{\Theta}) \tag{13}$$

$$\frac{\ddot{V}}{4} = \theta^T R K^T(\omega', \omega_d, \dot{\omega}_d)\omega' + \theta^T R^2\theta \tag{14}$$

The discussion of convergence of tracking error can be based on the time integral of equation (10), as shown in equation (15). Since θ is proven to be convergence, the right-hand side is now a constant and both terms at the right-hand side have finite value. As a result, it can be said that the term $K^T\omega'$ goes to zero as time goes infinity, and the tracking error ω' will be convergence as long as K^T always has a rank of 3.

$$-\theta(0) = \int_0^\infty \dot{\theta}\, dt = -\int_0^\infty K^T\omega'\, dt - \int_0^\infty R\theta\, dt \tag{15}$$

The conclusion on the Lyapunov candidate is still based on the fact that the matrix of known has to be full rank, due to equation (13). Provided a full rank matrix of known, the candidate function will be driven to zero from any positive value. When the candidate function is zero, the tracking error and unknown vector estimation error will have to be zero as well.

Table 3. Symbols used in section 2.4

Variable	Physical meaning	Variable	Physical meaning
i	Error of inertia matrix	C	A term for simplifying equation (9)
ω'	Error of angular velocity	K	A term for simplifying equation (9)
Q	Learning matrix for angular velocity error	R	Learning matrix for parameter estimation error

2.5 Simulation

The trajectory tracking of the rotation rate controller will be simulated. In the simulation, the trajectory is generated using arbitrary test torque, as shown in equation (16). The controller does not possess the test torque value, but instead receives a stream of desired rotation rate and the time derivative of the rotation rate. The idea is that if the deterministic artificial intelligence can track the test trajectory, it should also be able to track any trajectory generated by another trajectory planner.

$$I\dot{\omega}_d + \omega_d \times I\omega_d = \tau_{test} \tag{16}$$

Two types of performance matrices are considered: norm ratio of the state error, and the norm ratio of the parameter error, in equation (17). The result is plotted in section 3.

$$State\ error\ norm\ ratio = \left(\frac{||\omega'||_2^2}{||\omega_d||_2^2}\right)$$

$$Parameter\ error\ norm\ ratio = ln(\frac{||\theta'||_2^2}{||\Theta||_2^2}) \tag{17}$$

3. Results

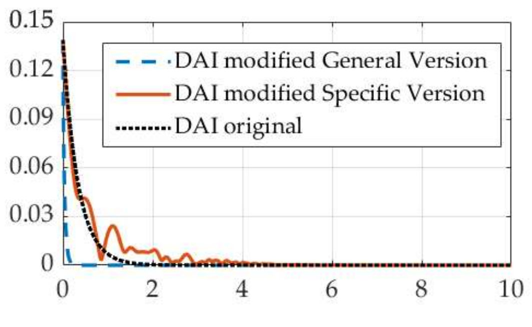
In this section, simulation results of the rotation rate problem (section 2.1) under different conditions are presented, and the performance of both types of modification (general version in section 2.3 and specific version in 2.4) is compared with the original deterministic artificial intelligence (section 2.2) learning approach.

3.1 Performance comparison without the product of inertia

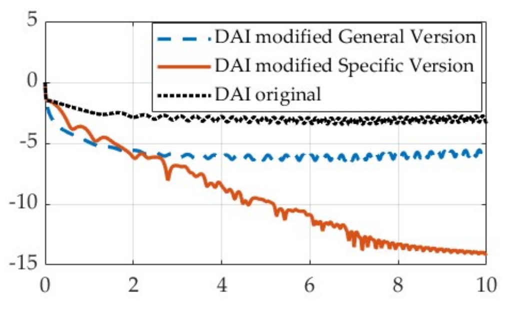
This case aims at testing the learning method when there is no product of inertia value in both the system's true parameter and the initial estimation of the unknown vector, which can be seen as an indication of control design vulnerability to coupling effects in governing equation. If the products of inertia have to be zero, then in equation (1), the size of the matrix of known will be reduced from 3 × 6 to 3 × 3, and the size of the vector of unknown will become 3 × 1. Intuitively speaking, we can have 3 equations and 3 unknowns in this case, making the unknown solvable using only the current information, as long as the matrix of known is full rank. The initial condition and the system parameters are listed in **Table 4**. The norm ratio of the state error and parameter error is shown in **Figure 3**. Also, the G in equation (7) and the R in equation (10) will be a scaler "r" multiplied by a 3 × 3 identity matrix, and this form of G and R will be used in all the cases presented in this manuscript.

Table 4. Initial condition for the simulation in section 3.1

Variable	Value	Variable	Value	Variable	Value
I_{xx}	1	I_{yy}	2	I_{zz}	3
I_{xy}	0.2	I_{xz}	0.3	I_{yz}	0.4
$\omega_{init,x}$	0.02	$\omega_{init,y}$	0.03	$\omega_{init,z}$	0.01
$\tau_{test,x}$	5	$\tau_{test,y}$	2	$\tau_{test,z}$	–2
r (specific)	3	r (general)	15	a	3



(a)



(b)

Figure 3. The convergence of the parameter error and state error. (**a**) Parameter error norm ratio on the ordinant versus time in seconds on the abscissa. (**b**) State error norm ratio on the ordinant versus time in seconds on the abscissa.

Table 5. Performance of inertia estimation and tracking errors

Figure of merit	Original method (prequels)	Proposed version general	Proposed version specific
Parameter error mean	0.0029	0.0058	0.0059
Parameter error deviation	0.0020	0.0038	0.0036
Mean tracking error	0.1709	0.0390	0.0076
Tracking error deviation	0.0862	0.0186	0.0083

3.2 Performance comparison with the product of inertia

This case is similar to section 3.1, but the product of inertia values in both the system's true parameter and the initial estimation of the unknown vector is not zero. The initial condition and the system parameters are listed in **Table 6**. The norm ratio of the state error and parameter error is shown in **Figure 4**.

Table 6. Initial condition for the simulation in section 3.2

Variable	Value	Variable	Value	Variable	Value
I_{xx}	1	I_{yy}	2	I_{zz}	1
I_{xy}	0.2	I_{xz}	0.3	I_{yz}	0.4
$\hat{I}_{xx,init}$	1.06	$\hat{I}_{yy,init}$	1.90	$\hat{I}_{zz,init}$	1.15
$\hat{I}_{xy,init}$	0.21	$\hat{I}_{xz,init}$	0.31	$\hat{I}_{yz,init}$	0.41
$\omega_{init,x}$	0.02	$\omega_{init,y}$	0.03	$\omega_{init,z}$	0.01
$\tau_{test,x}$	5	$\tau_{test,y}$	2	$\tau_{test,z}$	−2
r	3				

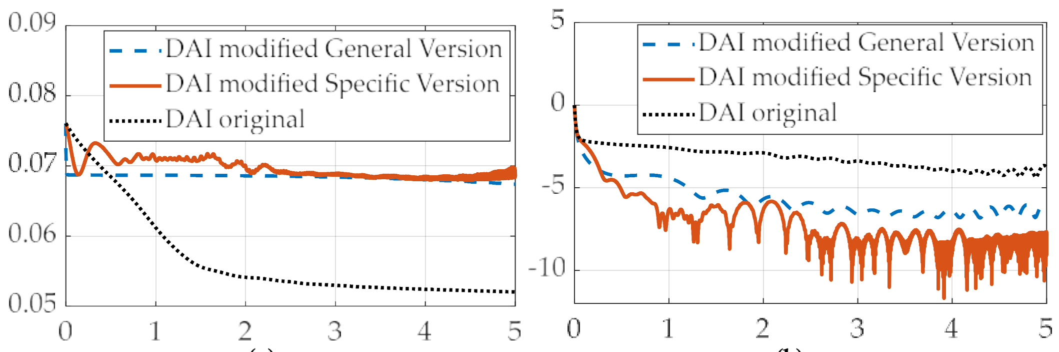


Figure 4. The convergence of the parameter error and state error. (**a**) Parameter error norm ratio on the ordinant versus time in seconds on the abscissa. (**b**) State error norm ratio on the ordinant versus time in seconds on the abscissa.

3.3 Performance comparison with different r value

This case shows for the modified learning method (Specific Version) how the *r* value, which can be seen as the "magnitude" of the *G* in equation (7) and the *R* in equation (10), affects the final result. The initial condition and parameters used in this case are identical to case 3.2 and can be checked in **Table 6**, except for the *r* value.

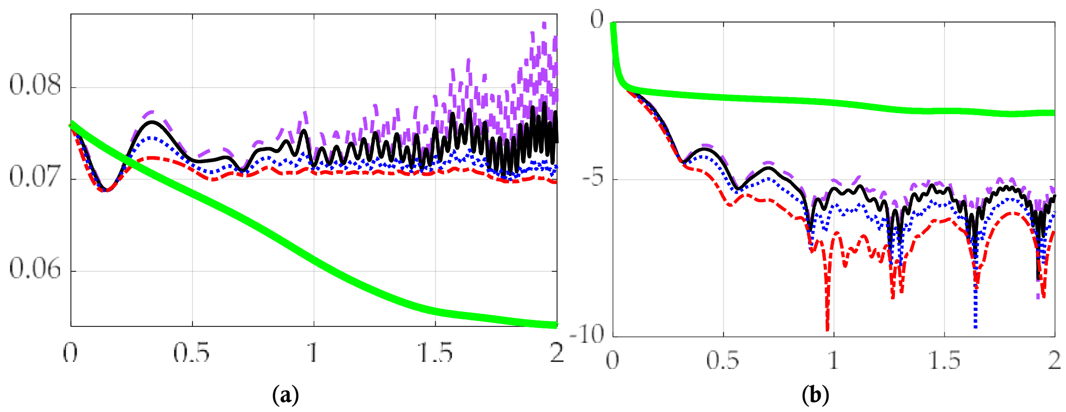


Figure 5. The convergence of the parameter error and state error. Original deterministic artificial intelligence is displayed by a thick, solid green line, dashed purple line displays $r = 0.5$, thin solid black line displays $r = 1$, dotted blue line displays $r = 2$, dot-dashed red line displays $r = 4$. (**a**) Parameter error norm ratio on the ordinant versus time in seconds on the abscissa. (**b**) State error norm ratio on the ordinant versus time in seconds on the abscissa.

Table 7. Convergence of inertia estimation and tracking errors

Figure of merit	Original method (prequels)	Modified with $r = 0.5$	Modified with $r = 1$	Modified with $r = 2$	Modified with $r = 4$
Parameter error mean	0.0247	0.0351	0.0348	0.0345	0.0341
Parameter error deviation	0.0124	0.0306	0.0272	0.0239	0.0217
Mean tracking error	–0.0401	–0.0033	–0.0033	–0.0033	–0.0034
Tracking error deviation	0.1761	0.0296	0.0246	0.0189	0.0143

4. Discussion

In sections 3.1 and 3.2, the modified learning method yields better state error convergence than the original method. For the specific version of the modified method, the final state error norm ratio is about 2 magnitudes smaller (rough order $\times\ e^{-8}$ compared with $\times\ e^{-3}$) than the original learning method, due to the data shown in both **Figures 3** and **4**.

In section 3.1, all the learning methods are able to make the parameter error converge to zero. This fits the expectation because in section 3.1 there is only 3 unknowns instead of 6. However, when the moment of inertia matric contains the nonzero product of inertia, as has been done in section 3.2, the left part of **Figure 4** shows that the modified methods are not better than the original method.

Table 8. Percent performance enhancement: Convergence of inertia estimation and tracking errors

Figure of merit	Original method (prequels)	Proposed version general	Proposed version specific
Parameter error mean	**0%**	42%	53%
Parameter error deviation	**0%**	16%	100%
Mean tracking error	0%	–77%	–99%
Tracking error deviation	0%	–91%	–96%

In section 3.3, **Figure 5** shows that when the magnitude of R in equation (10) goes bigger, the convergence rate also increases. Because equation (12) states that the convergence rate of the Lyapunov function (equation (11)) is only determined by the size of R and θ, the result in section 3.3 is reasonable.

Table 9. Percent performance enhancement: Convergence of inertia estimation and tracking errors

Figure of merit	Original method (prequels)	Modified with $r = 0.5$	Modified with $r = 1$	Modified with $r = 2$	Modified with $r = 4$
Parameter error mean	0.00%	–42.11%	–40.89%	–39.68%	–38.06%
Parameter error deviation	0.00%	–146.77%	–119.35%	–92.74%	–75.00%
Mean tracking error	0.00%	91.77%	91.77%	91.77%	91.52%
Tracking error deviation	0.00%	83.19%	86.03%	89.27%	91.88%

From the convergence condition of errors in **Figures 3** and **4**, it can be concluded that the convergence trajectories of the specific version of the modified learning method are "bumpier" and contains more jitters and oscillations. This phenomenon may result from the way of θ value determination provided in equation (13), which only consider the data in the current time stamp, and the indeterminate nature of equation (13), when the matrix of known is not full rank, makes the estimation of θ very unstable.

It can be concluded that the specific version of the modified learning method can achieve the convergence of both parameter error and state error in the simulation done in this manuscript, especially when the matrix of known is full rank, which can increase the robustness of the rotation rate controller.

4.1 Recommended future work

From the parameter error data of the specific version of the modified method in **Figures 4** and **5**, the increasing jitters can be observed. The reason for such instability after the convergence is unclear. It could result from the numerical instability of the chosen ODE solver and the options given to it, or the indeterminate way used for determining θ value in equation (13).

Moreover, the property of the "general version of modified learning method" hasn't been explored carefully because it is not suitable in this case by nature. Also, a better way of estimating θ may improve the result of the modified learning method as well. Finally, a better way of choosing the G in equation (7) and the R in equation (10) is also an interesting topic.

Conflict of interest

The authors declare no conflict of interest.

References

1. Johnson M. Space station robotic arms have a long reach [Internet]. Washington, DC: National Aeronautics and Space Administration; 2019 [updated 2022 Aug 13]. Available from: https://www.nasa.gov/mission_pages/station/research/news/b4h-3rd/hh-robotic-arms-reach.
2. Mahoney E. NASA seeks ideas for commercial uses of gateway [Internet]. Washington, DC: National Aeronautics and Space Administration; 2018 [updated 2018 Aug 25]. Available from: https://www.nasa.gov/feature/nasa-seeks-ideas-for-commercial-uses-of-gateway.
3. NASA Image Use Policy [Internet]. Washington, DC: National Aeronautics and Space Administration. Available from: https://gpm.nasa.gov/image-use-policy.
4. Bucchioni G, Innocenti M. Rendezvous in cis-lunar space near rectilinear halo orbit: Dynamics and control issues. Aerospace 2021; 8(3): 68. doi: 10.3390/aerospace8030068.
5. Bando M, Namati H, Akiyama Y, Hokamoto S. Formation flying along libration point orbits using chattering attenuation sliding mode control. Frontiers in Space Technologies 2022; 3: 919932. doi: 10.3389/frspt.2022.919932.
6. Colombi F, Colagrossi A, Lavagna M. Characterization of 6DOF natural and controlled relative dynamics in cislunar space. Acta Astronautica 2022; 196: 369–379. doi: 10.1016/j.actaastro.2021.01.017.
7. Sands T. Flattening the curve of flexible space robotics. Applied Sciences 2022; 12(6): 2992. doi: 10.3390/app12062992.
8. Sands T. Optimization provenance of whiplash compensation for flexible space robotics. Aerospace 2019; 6(9): 93. doi: 10.3390/aerospace6090093.
9. Jones A. Chinese space station robot arm tests bring amazing views from orbit [Internet]. New York: Space.com; 2022 [updated 2022 Aug 9]. Available from: https://www.space.com/china-space-station-wentian-robot-arm-test.
10. Jones A. See a large robotic arm "crawl" across China's space station [Internet]. New York: Space.com; 2022 [published 2022 Feb 16]. Available from: https://www.space.com/china-space-station-robot-arm-video.
11. Lafarge N, Miller D, Howell K, Linares R. Autonomous closed-loop guidance using reinforcement learning in a low-thrust, multi-body dynamical environment. Acta Astronautica 2021; 186: 1–23. doi: 10.1016/j.actaastro.2021.05.014.
12. Albee K, Oestreich C, Specht C, *et al.* A robust observation, planning, and control pipeline for autonomous rendezvous with tumbling targets. Frontiers in Robotics and AI 2021; 8. doi: 10.3389/frobt.2021.641338.
13. Mehta PM, Linares R. A new transformative framework for data assimilation and calibration of physical ionosphere-thermosphere models. Space Weather 2018; 16(18): 1086–1100. doi: 10.1029/2018SW001875.
14. Oestreich CE, Linares R, Gondhalekar R. Autonomous six-degree-of-freedom spacecraft docking with rotating targets via reinforcement learning. Journal of Aerospace Information Systems 2021; 18(7): 417–428. doi: 10.2514/1.1010914.
15. Avendaño M, Arnas D, Linares R, Lifson M. Efficient search of optimal flower constellations. Acta Astronautica 2021; 179: 290–295. doi: 10.1016/j.actaastro.2020.10.026.
16. Arnas D, Lifson M, Linares R, Avendaño M. Definition of low earth orbit slotting architectures using 2D lattice flower constellations. Advances in Space Research 2021; 67(11): 3696–3711. doi: 10.1016/j.asr.2020.04.021.
17. Oestreich C, Espinoza A, Todd J, *et al.* On-orbit inspection of an unknown, tumbling target using NASA's Astrobee robotic free-flyers. In: 2021 IEEE/CVF Conference on Computer Vision and

Pattern Recognition Workshops (CVPRW); 2021 Jun 20–25; Nashville. New York: Institute of Electrical and Electronics Engineers (IEEE); 2021. p. 2039–2047.

18. Roberts TG. Geosynchronous satellite maneuver classification and orbital pattern anomaly detection via supervised machine learning [Master's thesis]. Massachusetts (MA): Massachusetts Institute of Technology; 2021.

19. Ekal M, Albee K, Coltin B, *et al*. Online information-aware motion planning with inertial parameter learning for robotic free-flyers. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2021 Sep 27–Oct 1; Prague. New York: Institute of Electrical and Electronics Engineers (IEEE); 2021. p. 8766–8773. doi: 10.1109/IROS51168.2021.9636325.

20. Cassinis LP, Park TH, Stacey N, *et al*. Leveraging neural network uncertainty in adaptive unscented Kalman Filter for spacecraft pose estimation. Advances in Space Research 2023; 71(12): 5061–5082. doi: 10.1016/j.asr.2023.02.021.

21. Park TH, D'Amico S. Robust multi-task learning and online refinement for spacecraft pose estimation across domain gap. Advances in Space Research 2023. In Press. doi: 10.1016/j.asr.2023.03.036.

22. Park TH, D'Amico S. Adaptive neural network-based unscented Kalman Filter for robust pose tracking of noncooperative spacecraft. New York: arXiv; 2022. doi: 10.48550/arXiv.2206.03796.

23. Zhang K, Pan B. Control design of spacecraft autonomous rendezvous using nonlinear models with uncertainty. Journal of Zhejiang University (Engineering Science) 2022; 56(4): 833–842. doi: 10.3785/j.issn.1008-973X.2022.04.024.

24. Smeresky B, Rizzo A, Sands T. Optimal learning and self-awareness versus PDI. Algorithms 2020; 13(1): 23. doi: 10.3390/a13010023.

25. Slotine J, Li W. Applied nonlinear control. Wilmington, DE: Prentice-Hall, Inc.; 1991. p. 392–436.

26. Sands T. Development of deterministic artificial intelligence for unmanned underwater vehicles (UUV). Journal of Marine Science and Engineering 2020; 8(8): 578. doi: 10.3390/jmse8080578.

27. Sandberg A, Sands T. Autonomous trajectory generation algorithms for spacecraft slew maneuvers. Aerospace 2022; 9(3): 135. doi: 10.3390/aerospace9030135.

28. Raigoza K, Sands T. Autonomous trajectory generation comparison for de-orbiting with multiple collision avoidance. Sensors 2022; 22(18): 7066. doi: 10.3390/s22187066.

29. Wilt E, Sands T. Microsatellite uncertainty control using deterministic artificial intelligence. Sensors 2022; 22(22): 8723. doi: 10.3390/s22228723.

Appendix A

The MATLAB® code used in this manuscript is pasted below. The program utilizes the ode45 solver to simulate the response of the overall system combining the controller and the controlled system.

```
clc; clear; close all
%% DAI Matrix Derivation
% ====================
% Derive the matrix of known w.r.t. vector of nuknown using symbolic
toolbox,
% and turn it into a matlab function.
% P is the matrix of known.
% th := [Ixx Ixy Ixz Iyy Iyz Izz]' is the vector of unknown.
% ====================
syms fwx(t) fwy(t) fwz(t)
syms Ixx Ixy Ixz Iyy Iyz Izz real
syms wx wy wz dwx dwy dwz ddwx ddwy ddwz real
w = [fwx;fwy;fwz];
wT = [fwx fwy fwz];
dw = diff(w,t);
ddw = diff(dw,t);
I = [Ixx Ixy Ixz; Ixy Iyy Iyz; Ixz Iyz Izz];
PhTh = I*dw + cross(w,I*w);
Peq = PhTh == 0;
[P, ~] = equationsToMatrix(Peq, [Ixx Ixy Ixz Iyy Iyz Izz]);
dP = diff(P,t);
sP = subs(P, [diff(diff(wT,t),t) diff(wT,t) fwx fwy fwz], [ddwx ddwy ddwz dwx
dwy dwz wx wy wz]);
sdP = subs(dP, [diff(diff(wT,t),t) diff(wT,t) fwx fwy fwz], [ddwx ddwy ddwz
dwx dwy dwz wx wy wz]);
sfP = symfun(sP, [wx wy wz dwx dwy dwz ddwx ddwy ddwz]);
sfdP = symfun(sdP, [wx wy wz dwx dwy dwz ddwx ddwy ddwz]);

%% Derive matrix K for the modified learning --- specific
% ====================
% Derive the K matrix, mentioned in equation 9, used in specific learning,
% and turn it into a matlab function.
% i is the unknown vector error, i := I_real - I_estimate.
% w is the state error, w := w_desired - w_real.
% ====================
syms wdx wdy wdz dwdx dwdy dwdz real
syms ixx ixy ixz iyy iyz izz real
w = [wx;wy;wz];
wd = [wdx;wdy;wdz];
dwd = [dwdx dwdy dwdz]';
i = [ixx ixy ixz; ixy iyy iyz; ixz iyz izz];
```

```
Ki = i'*dwd + cross(wd,i*wd) - cross(wd,i*w);
Keq = Ki == [0;0;0];
[K, sbz] = equationsToMatrix(Keq, [ixx ixy ixz iyy iyz izz]);
fK = symfun(K, [wx wy wz wdx wdy wdz dwdx dwdy dwdz]);

%% Simulation: ODE45
% ===================
% Define simulation parameters and simulate
% ===================
%
% ===== <Parameter definition> =====
p.J = [1 0.2 0.3; 0.2 2 0.4; 0.3 0.4 1]; % Example in section 3_2
p.dwd = [1 1 1]';
p.P = matlabFunction(sfP);
p.dP = matlabFunction(sfdP);
p.K = matlabFunction(fK);
p.G = 3*eye(6);
p.pinvTol = 1e-3;
Jt = [p.J(1,1);p.J(1,2);p.J(1,3);p.J(2,2);p.J(2,3);p.J(3,3)];
%
% ===== <Simulation time> =====
deltat = 0.01;
tfinal = 3;
t = 0:deltat:tfinal;% for evaluating solution
%
% ===== <Simulation: ODE45> =====
z0 = [1 0 0 0 0 0 0 0.02 0.03 0.01 1.06 0.21 0.31 1.90 0.41 1.15]'; % 3_2 0.08
0.08 004
options = odeset('absTol',1e-10,'relTol',1e-10);
% The simulation for the general version of modified learning method
[t_dai, z_dai] = ode45(@(t,z)DAI_modified_general(t,z,p), t, z0, options);
% The simulation for the specific version of modified learning method
[t_dmd, z_dmd] = ode45(@(t,z)DAI_modified_specific(t,z,p), t, z0, options);
% The simulation for the original version of learning method
[t_dor, z_dor] = ode45(@(t,z)DAI_original(t,z,p), t, z0, options);

%% Plot parameter estimations and state trajectories
figure()
plot(t_dai, z_dai(:,11),t_dai, z_dai(:,14),t_dai, z_dai(:,16))
legend('Ixx', 'Iyy', 'Izz');
title('Vector of unknown Estimation of DAI Modification General Version');
figure()
plot(t_dmd, z_dmd(:,11),t_dmd, z_dmd(:,14),t_dmd, z_dmd(:,16))
legend('Ixx', 'Iyy', 'Izz');
title('Vector of unknown Estimation of DAI Modification Specific Version');
```

```
figure()
plot(t_dor, z_dor(:,11),t_dor, z_dor(:,14),t_dor, z_dor(:,16))
legend('Ixx', 'Iyy', 'Izz');
title('Vector of unknown Estimation of Original DAI');
figure()
plot(t_dai, z_dai(:,7),t_dor, z_dor(:,7),t_dmd, z_dmd(:,7),t_dai, z_dai(:,10))
legend('Modified 1', 'Original', 'Modified 2', 'Desired');
title('Angular Velocity Tracking of DAI Modification General Version');
figure()
plot(t_dai, z_dai(:,6),t_dor, z_dor(:,6),t_dmd, z_dmd(:,6),t_dai, z_dai(:,9))
legend('Modified 1', 'Original', 'Modified 2', 'Desired');
title('Angular Velocity Tracking of DAI Modification Specific Version');
figure()
plot(t_dai, z_dai(:,5),t_dor, z_dor(:,5),t_dmd, z_dmd(:,5),t_dai, z_dai(:,8))
legend('Modified 1', 'Original', 'Modified 2', 'Desired');
title('Angular Velocity Tracking of Original DAI');

%% Analysis the norm rates
% inertia norm ratio: Equation 15
J_dai = z_dai(:,11:16);
J_dmd = z_dmd(:,11:16);
J_dor = z_dor(:,11:16);
n = length(t_dai);
nJ_dai = vecnorm(J_dai'-Jt*ones(1,n))/norm(Jt);
nJ_dmd = vecnorm(J_dmd'-Jt*ones(1,n))/norm(Jt);
nJ_dor = vecnorm(J_dor'-Jt*ones(1,n))/norm(Jt);
figure()
plot(t_dai, nJ_dai, t_dmd, nJ_dmd, t_dor, nJ_dor);
legend('DAI modified General Version', 'DAI modified Specific Version', 'DAI
original');
title('Convergence of the parameter error norm ratio');
xlabel('time (s)');
ylabel('Parameter error norm ratio');
% state error norm ratio: Equation 15
dw_dai = z_dai(:,5:7)-z_dai(:,8:10);
nw_dai = vecnorm(dw_dai')./vecnorm(z_dai(:,8:10)');
dw_dmd = z_dmd(:,5:7)-z_dmd(:,8:10);
nw_dmd = vecnorm(dw_dmd')./vecnorm(z_dmd(:,8:10)');
dw_dor = z_dor(:,5:7)-z_dor(:,8:10);
nw_dor = vecnorm(dw_dor')./vecnorm(z_dor(:,8:10)');
figure()
plot(t_dai, log(nw_dai), t_dmd, log(nw_dmd), t_dor, log(nw_dor));
legend('DAI modified General Version', 'DAI modified Specific Version', 'DAI
original');
title('Convergence of the state error norm ratio');
```

```
xlabel('time (s)');
ylabel('State error norm ratio');

%% mean and deviation: Table 4
clc
% DAI modified General Version
ParamErrorMeanGeneral = mean(mean(J_dai-(Jt*ones(1,length(J_dai)))'))
ParamErrorStdrGeneral = mean(std(J_dai-(Jt*ones(1,length(J_dai)))'))
StateErrorMeanGeneral = mean(mean(dw_dai))
StateErrorStdrGeneral = mean(std(dw_dai))
% DAI modified Specific Version
ParamErrorMeanSpecific = mean(mean(J_dmd-(Jt*ones(1,length(J_dmd)))'))
ParamErrorStdrSpecific = mean(std(J_dmd-(Jt*ones(1,length(J_dmd)))'))
StateErrorMeanSpecific = mean(mean(dw_dmd))
StateErrorStdrSpecific = mean(std(dw_dmd))
% DAI Original
ParamErrorMeanOriginal = mean(mean(J_dor-(Jt*ones(1,length(J_dor)))'))
ParamErrorStdrOriginal = mean(std(J_dor-(Jt*ones(1,length(J_dor)))'))
StateErrorMeanOriginal = mean(mean(dw_dor))
StateErrorStdrOriginal = mean(std(dw_dor))

%% Test the performance of Specific Version Modified DAI in different G
% G = 0.5
p.G = 0.5*eye(6);
[t_dmd_h, z_dmd_h] = ode45(@(t,z)DAI_modified_specific(t,z,p), t, z0, op-
tions);
n = length(t_dmd_h);
J_dmd_h = z_dmd_h(:,11:16);
nJ_dmd_h = vecnorm(J_dmd_h'-Jt*ones(1,n))/norm(Jt);
dw_dmd_h = z_dmd_h(:,5:7)-z_dmd_h(:,8:10);
nw_dmd_h = vecnorm(dw_dmd_h')./vecnorm(z_dmd_h(:,8:10)');
% G = 1
p.G = 1*eye(6);
[t_dmd_1, z_dmd_1] = ode45(@(t,z)DAI_modified_specific(t,z,p), t, z0, op-
tions);
J_dmd_1 = z_dmd_1(:,11:16);
nJ_dmd_1 = vecnorm(J_dmd_1'-Jt*ones(1,n))/norm(Jt);
dw_dmd_1 = z_dmd_1(:,5:7)-z_dmd_1(:,8:10);
nw_dmd_1 = vecnorm(dw_dmd_1')./vecnorm(z_dmd_1(:,8:10)');
% G = 2
p.G = 2*eye(6);
[t_dmd_2, z_dmd_2] = ode45(@(t,z)DAI_modified_specific(t,z,p), t, z0, op-
tions);
J_dmd_2 = z_dmd_2(:,11:16);
nJ_dmd_2 = vecnorm(J_dmd_2'-Jt*ones(1,n))/norm(Jt);
```

```
dw_dmd_2 = z_dmd_2(:,5:7)-z_dmd_2(:,8:10);
nw_dmd_2 = vecnorm(dw_dmd_2')./vecnorm(z_dmd_2(:,8:10)');
% G = 4
p.G = 4*eye(6);
[t_dmd_4, z_dmd_4] = ode45(@(t,z)DAI_modified_specific(t,z,p), t, z0, op-
tions);
J_dmd_4 = z_dmd_4(:,11:16);
nJ_dmd_4 = vecnorm(J_dmd_4'-Jt*ones(1,n))/norm(Jt);
dw_dmd_4 = z_dmd_4(:,5:7)-z_dmd_4(:,8:10);
nw_dmd_4 = vecnorm(dw_dmd_4')./vecnorm(z_dmd_4(:,8:10)');
% Original
[t_dor, z_dor] = ode45(@(t,z)DAI_original(t,z,p), t, z0, options);
J_dor = z_dor(:,11:16);
nJ_dor = vecnorm(J_dor'-Jt*ones(1,n))/norm(Jt);
dw_dor = z_dor(:,5:7)-z_dor(:,8:10);
nw_dor = vecnorm(dw_dor')./vecnorm(z_dor(:,8:10)');
%
% Plot the result
rnge = 200;
figure();
plot(t_dmd_h(1:rnge), nJ_dmd_h(1:rnge),...
      t_dmd_1(1:rnge), nJ_dmd_1(1:rnge),...
      t_dmd_2(1:rnge), nJ_dmd_2(1:rnge),...
      t_dmd_4(1:rnge), nJ_dmd_4(1:rnge),...
      t_dor(1:rnge), nJ_dor(1:rnge));
legend('DAI modified r = 0.5',...
        'DAI modified r = 1',...
        'DAI modified r = 2',...
        'DAI modified r = 4',...
        'DAI original');
title('Convergence of the parameter error norm ratio');
xlabel('time (s)');
ylabel('Parameter error norm ratio');
figure();
plot(t_dmd_h(1:rnge), log(nw_dmd_h(1:rnge)), ...
      t_dmd_1(1:rnge), log(nw_dmd_1(1:rnge)), ...
      t_dmd_2(1:rnge), log(nw_dmd_2(1:rnge)), ...
      t_dmd_4(1:rnge), log(nw_dmd_4(1:rnge)), ...
      t_dor(1:rnge), log(nw_dor(1:rnge)));
legend('DAI modified r = 0.5',...
        'DAI modified r = 1',...
        'DAI modified r = 2',...
        'DAI modified r = 4',...
        'DAI original');
title('Convergence of the state error norm ratio');
```

```
xlabel('time (s)');
ylabel('State error norm ratio');

%% mean and deviation: Table 6
% G = 0.5
ParamErrorMeanGHlf = mean(mean(J_dmd_h(1:rnge,:)-(Jt*ones(1,rnge))'))
ParamErrorStdrGHlf = mean(std(J_dmd_h(1:rnge,:)-(Jt*ones(1,rnge))'))
StateErrorMeanGHlf = mean(mean(dw_dmd_h(1:rnge,:)))
StateErrorStdrGHlf = mean(std(dw_dmd_h(1:rnge,:)))
% G = 1
ParamErrorMeanGOne = mean(mean(J_dmd_1(1:rnge,:)-(Jt*ones(1,rnge))'))
ParamErrorStdrGOne = mean(std(J_dmd_1(1:rnge,:)-(Jt*ones(1,rnge))'))
StateErrorMeanGOne = mean(mean(dw_dmd_1(1:rnge,:)))
StateErrorStdrGOne = mean(std(dw_dmd_1(1:rnge,:)))
% G = 2
ParamErrorMeanGTwo = mean(mean(J_dmd_2(1:rnge,:)-(Jt*ones(1,rnge))'))
ParamErrorStdrGTwo = mean(std(J_dmd_2(1:rnge,:)-(Jt*ones(1,rnge))'))
StateErrorMeanGTwo = mean(mean(dw_dmd_2(1:rnge,:)))
StateErrorStdrGTwo = mean(std(dw_dmd_2(1:rnge,:)))
% G = 4
ParamErrorMeanGFor = mean(mean(J_dmd_4(1:rnge,:)-(Jt*ones(1,rnge))'))
ParamErrorStdrGFor = mean(std(J_dmd_4(1:rnge,:)-(Jt*ones(1,rnge))'))
StateErrorMeanGFor = mean(mean(dw_dmd_4(1:rnge,:)))
StateErrorStdrGFor = mean(std(dw_dmd_4(1:rnge,:)))
% Original DAI
ParamErrorMeanOrig = mean(mean(J_dor(1:rnge,:)-(Jt*ones(1,rnge))'))
ParamErrorStdrOrig = mean(std(J_dor(1:rnge,:)-(Jt*ones(1,rnge))'))
StateErrorMeanOrig = mean(mean(dw_dor(1:rnge,:)))
StateErrorStdrOrig = mean(std(dw_dor(1:rnge,:)))

%% Function
function zdot = DAI_modified_general(t, z, p)
    % Orientation of the rigid body as quaternion
    q = z(1:4);
    % Angular velocity
    w = z(5:7);
    % Desired velocity
    wd = z(8:10);
    % Vector of unknown: Inertia
    th = z(11:16); % theta hat
    Jh = [th(1) th(2) th(3); th(2) th(4) th(5); th(3) th(5) th(6)];
    % Generate trajectory to be followed
    [sdwd,sddwd] = traj_gen(t,wd,p);
    % Generate feed forward control torque
    tau = Jh*sdwd + cross(wd', Jh*wd)';
```

```
        % Update the dynamic of the system
        dw = p.J\(tau-cross(w', p.J*w)');
        dq = 0.5*quatmultiply([0 w'],q');
        % Update the parameter estimation
        ddw = [0;0;0];
        % Matrix of known w.r.t. desired angular velocity
        Pd                                                                          =
p.P(wd(1),wd(2),wd(3),sdwd(1),sdwd(2),sdwd(3),sddwd(1),sddwd(2),sddwd(3));
        % Matrix of known w.r.t. actual angular velocity
        P = p.P(w(1),w(2),w(3),dw(1),dw(2),dw(3),0,0,0);
        % Time derivative of the Matrix of known w.r.t. actual angular velocity
        dP = p.dP(w(1),w(2),w(3),dw(1),dw(2),dw(3),ddw(1),ddw(2),ddw(3));
        % Equation 6: General version of modified learning in DAI
        ph = Pd-P;
        A = dP*pinv(P,p.pinvTol)*pinv(P' + pinv(P,p.pinvTol),p.pinvTol);
        B = (P' + pinv(P,p.pinvTol))'*p.G;
        dth = (th'*ph'*(A+B))';
        % Print simulation time every 1 second
        if abs(t-round(t)) < 0.001
            disp(t)
        end
        zdot = [dq';dw;sdwd;dth];
end

function zdot = DAI_modified_specific(t, z, p)
        % Orientation of the rigid body as quaternion
        q = z(1:4);
        % Angular velocity
        w = z(5:7);
        % Desired velocity
        wd = z(8:10);
        % Vector of unknown: Inertia
        th = z(11:16); % theta hat
        Jh = [th(1) th(2) th(3); th(2) th(4) th(5); th(3) th(5) th(6)];
        % Generate trajectory to be followed
        [sdwd,~] = traj_gen(t,wd,p);
        % Generate feed forward control torque: Equation 8
        tau = Jh*sdwd + cross(wd', Jh*w)';
        % Update the dynamic of the system
        dw = p.J\(tau-cross(w', p.J*w)');
        dq = 0.5*quatmultiply([0 w'],q');
        % Update the parameter estimation
        P = p.P(w(1),w(2),w(3),dw(1),dw(2),dw(3),0,0,0);
        % Equation 9/12: Specific version of modified learning in DAI
        ew = wd-w;
```

```
	K = p.K(ew(1),ew(2),ew(3),wd(1),wd(2),wd(3),sdwd(1),sdwd(2),sdwd(3));
	Dth = pinv(P,p.pinvTol)*(tau-P*th);
	dth = K'*ew+p.G*Dth;
	% Monitor the Lyapunov Candidate function to see if it is decreasing
	% TDth = [p.J(1,1);p.J(1,2);p.J(1,3);p.J(2,2);p.J(2,3);p.J(3,3)]-th;
	% V = ew'*p.J*ew+TDth'*TDth
	zdot = [dq';dw;sdwd;dth];
end

function zdot = DAI_original(t, z, p)
	% Orientation of the rigid body as quaternion
	q = z(1:4);
	% Angular velocity
	w = z(5:7);
	% Desired velocity
	wd = z(8:10);
	% Vector of unknown: Inertia
	th = z(11:16); % theta hat
	Jh = [th(1) th(2) th(3); th(2) th(4) th(5); th(3) th(5) th(6)];
	% Generate trajectory to be followed
	[sdwd,~] = traj_gen(t,wd,p);
	% Generate feed forward control torque
	tau = Jh*sdwd + cross(wd', Jh*wd)';
	% Update the dynamic of the system
	dw = p.J\(tau-cross(w', p.J*w)');
	dq = 0.5*quatmultiply([0 w'],q');
	% Update the parameter estimation: Equation 3-a, 3-b
	P = p.P(w(1),w(2),w(3),dw(1),dw(2),dw(3),0,0,0);
	dth = 1.5*pinv(P,p.pinvTol)*(tau-P*th);
	zdot = [dq';dw;sdwd;dth];
end

function [dwd,ddwd] = traj_gen(t,wd,p)
	tau = [5;2;-2];
	if t>7
		tau = [0;0;0];
	end
	dwd = p.J\(tau-cross(wd', p.J*wd)');
	ddwd = [0;0;0];
end
```