frontiers in Space Technologies

TYPE Technology and Code
PUBLISHED 11 November 2026
DOI 10.3389/frspt.2026.1955127

Check for updates

OPEN ACCESS

EDITED BY
Ti Chen,
Nanjing University of Aeronautics and Astronautics, China

REVIEWED BY
Selva Kumar Chandrasekar,
Thiagarajar College of Engineering, India
Mingyan Xie,
Nanjing University of Aeronautics and Astronautics, China

*CORRESPONDENCE
Zheng H. Zhu,
gzhu@yorku.ca

RECEIVED 31 July 2026
REVISED 23 August 2026
ACCEPTED 25 August 2026
PUBLISHED 11 November 2026

CITATION
Jahanshahi H and Zhu ZH (2026) space-robot-dq: a Python library for dual quaternion kinematics, dynamics, and control of free-floating space robot manipulators.
Front. Space Technol. 7:1955127.
doi: 10.3389/frspt.2026.1955127

COPYRIGHT
© 2026 Jahanshahi and Zhu. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

space-robot-dq: a Python library for dual quaternion kinematics, dynamics, and control of free-floating space robot manipulators

Hadi Jahanshahi and Zheng H. Zhu*

Department of Mechanical Engineering, York University, Toronto, ON, Canada

This article presents space-robot-dq, an open-source Python library for the kinematics, dynamics, and resolved-rate control of free-floating space robot manipulators using dual quaternion algebra. Unlike ground-based robots, space manipulators operate on platforms with no fixed base. When the arm moves, the spacecraft reacts to conserve momentum, changing the mapping between joint motion and end-effector motion. The library implements Product of Exponentials forward kinematics in dual quaternion arithmetic, numerical and analytical Jacobians, optimization-based inverse kinematics, the Umetani–Yoshida generalized Jacobian for momentum-coupled systems, and closed-loop resolved-rate controllers for a momentum-conserving free-floating plant with base pose propagation. It supports arbitrary N-DOF serial chains of revolute joints through a declarative configuration interface. The implementation is verified by 153 unit tests across four robot architectures and a 19-check benchmark suite. Independent validation covers forward-kinematics agreement below 10^{-11} m against two reference implementations that share no code with the library and first-order-convergent stationarity of the system's center of mass, while construction-level consistency checks confirm momentum residuals at the 10^{-15} level. Two control studies quantify the practical consequences of the free-floating coupling. In Cartesian tracking of a 0.27 m maneuver, a controller using the fixed-base Jacobian accumulates 26.372 mm root mean square (RMS) error even with feedback, whereas the generalized-Jacobian controller tracks at 0.2512 mm RMS (a 105-fold difference that measures the cost of the model mismatch alone, because both controllers run on the same plant). In a capture-approach scenario, the end-effector maintains a full 6-DOF grasp pose against a client tumbling at 5° per second, holding 0.008429 mm and 0.000581° RMS with the generalized Jacobian versus 4.267 mm and 0.22° with the fixed-base model (506-fold and 379-fold differences attributable to the neglected coupling), while the spacecraft base drifts 2.4°, an attitude cost the library quantifies for mission design. All figures and numbers in this article are reproduced by a single, deterministic script included in the repository.

KEYWORDS
dual quaternion, free-floating manipulator, generalized Jacobian, momentum conservation, Python, resolved-rate control, space robotics, tumbling target capture

1 Introduction

Space robot manipulators play a growing role in on-orbit servicing, debris removal, and space station maintenance (Ma et al., 2023; Jahanshahi and Zhu, 2024; Jahanshahi and Zhu, 2025). Unlike terrestrial robots fixed to the ground, space manipulators are mounted on free-floating spacecraft (Figure 1), in which joint angles $\theta_1, \ldots, \theta_n$ and link position vectors define the kinematic chain from the base center of mass to the end-effector, and the base undergoes linear and angular reaction motion in response to arm motion. In the absence of external forces, the total system momentum is conserved, which means that when the arm moves, the base reacts. This dynamic coupling means that the standard fixed-base Jacobian overestimates the manipulator's velocity capability, and controllers designed without accounting for the coupling produce systematic trajectory tracking errors (Umetani and Yoshida, 1989; Dubowsky and Papadopoulos, 1993; Lu and Yue, 2025; Tang et al., 2025; Jahanshahi et al., 2026).

The mathematical framework for handling this coupling is the generalized Jacobian matrix introduced by Umetani and Yoshida (1989), which maps joint velocities to end-effector velocities while accounting for the base reaction:

$$\mathbf{J}_g = \mathbf{J}_m - \mathbf{J}_b \, \mathbf{H}_b^{-1} \, \mathbf{H}_{bm}, \tag{1}$$

where $\mathbf{J}_m$ is the fixed-base manipulator Jacobian, $\mathbf{J}_b$ maps base velocity to end-effector velocity, $\mathbf{H}_b$ is the 6×6 base spatial inertia, and $\mathbf{H}_{bm}$ is the $6{\times}N$ base-manipulator coupling inertia.

In parallel, dual quaternion algebra has become an established representation for rigid-body poses in robotics (Funda et al., 1990; Liendo et al., 2025; Li et al., 2025; Löw et al., 2026). Dual quaternions provide a compact eight-parameter, singularity-free representation that unifies rotation and translation into a single algebraic object and avoids the gimbal lock and discontinuities of Euler-angle parameterizations. Closely related Clifford-algebra representations have also been used for inverse kinematics. Qiao et al. (2010) formulate the general 6R inverse kinematics with double quaternions, which map SE(3) to SO(4) and yield a 16th-degree univariate resultant. Dual quaternion formulations themselves have been applied to kinematic modeling and analysis of industrial manipulators (Ge et al., 2019) and to experimentally validated kinematic modeling of PRR-type robots (Chandrasekar and Gnanasekar, 2025). Hence, the representation has a track record well beyond space robotics.

Both theories are mature; however, no existing Python library provides an integrated, tested implementation that combines dual quaternion kinematics with free-floating dynamics and control for arbitrary serial chains of revolute joints. Table 1 compares the available tools. General-purpose robotics libraries such as the Robotics Toolbox for Python (Corke and Haviland, 2021) and Pinocchio (Carpentier et al., 2019) focus on fixed-base or legged robots; Pinocchio supports floating-base models and momentum computations, but does not provide the Umetani–Yoshida generalized Jacobian or free-floating manipulability measures out of the box. Dual quaternion libraries such as dual_quaternions (Verheye, 2025) provide algebraic primitives but no robot-specific kinematics, dynamics, or momentum conservation. The space-robot-dq library fills this gap and also demonstrates, through two included control studies, why the gap matters: ignoring the coupling costs millimeters of tracking accuracy and tenths of a degree of grasp alignment in realistic maneuvers. Two contributions should be kept distinct throughout: the control-accuracy differences reported in Section 3 stem from the generalized Jacobian, while dual quaternions serve as the representation layer that supplies singularity-free pose algebra and the pose-error metric used by the 6-DOF controller.

Section 2 describes the software architecture and methods, including the new control module. Section 3 presents validation results and four usage studies, including limitations. Section 4 discusses impact, scalability, and future work. All results are reproducible: the repository ships a deterministic script, examples/generate_paper_figures.py, that regenerates every figure and every number quoted in this article.

The contributions of this work are threefold: (1) an open-source, tested implementation that integrates dual quaternion pose algebra, Product of Exponentials (PoE) kinematics, and the Umetani–Yoshida free-floating formulation for arbitrary serial chains of revolute joints, which closes the gap identified in Table 1; (2) a validation methodology that separates independent, physics-based checks from construction-guaranteed consistency checks, together with a fully scripted reproduction of every figure and number in this article; and (3) two quantitative control studies, closed-loop Cartesian tracking and 6-DOF station-keeping against a tumbling target, that measure the cost of neglecting the free-floating coupling and the spacecraft attitude budget consumed by arm motion.

2 Methods

2.1 Software architecture

The library is organized into four modules within the space_robot_dq package (Figure 2). Users declare robot geometry via JointDef and mass properties via LinkProperties, and the four modules form a pipeline from these inputs to end-effector pose, base reaction, manipulability, tracking error, and attitude drift outputs. The dual_quaternion module provides robot-agnostic algebraic primitives: dual quaternion multiplication, conjugation, the logarithmic map, and conversions between dual quaternions, rotation matrices, and screw parameters. The kinematics module contains the N-DOF kinematics engine, built around the RobotConfig and JointDef dataclasses that let users declare any serial chain of revolute joints by each joint's rotation axis and home position; screw axes and the Product of Exponentials chain are derived automatically, with built-in presets for a 7-DOF Spherical-Revolute-Spherical (SRS) space manipulator, a 6-DOF standard arm, and a 3-DOF planar arm. The dynamics module implements free-floating dynamics: per-link mass properties via the LinkProperties dataclass, spatial inertia assembly by the composite rigid body method with link-level Jacobians, momentum conservation, the generalized Jacobian, and a dynamic manipulability measure. The control module implements damped least-squares resolved-rate control for 3-DOF position and full 6-DOF pose tasks, null-space posture control, momentum-conserving base pose propagation, and the simulation drivers used by the studies in Section 3.

Conventions. Throughout the library and this article, all vector quantities are expressed in the inertial world frame; end-effector spatial velocities and Jacobian rows stack angular above linear

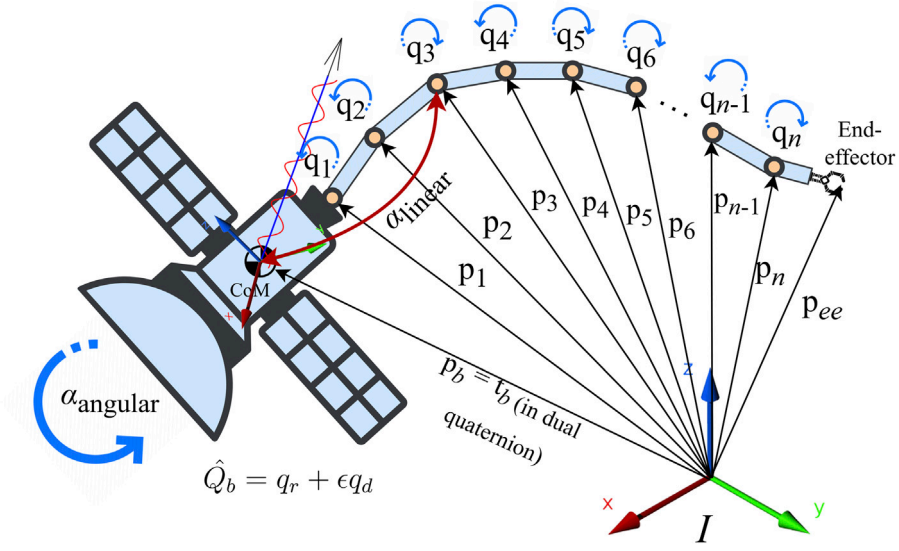


FIGURE 1
Free-floating space robot manipulator with *n* revolute joints.

TABLE 1 Feature comparison with existing tools (Corke and Haviland, 2021; Carpentier et al., 2019; Verheye, 2025).[a]

Feature	This work	RTB-Python	Pinocchio	DQ library
Dual quaternion pose algebra	✓	–	–	✓
Dual quaternion FK for serial chains	✓	–	–	Partial
Arbitrary N-DOF revolute chains	✓	✓	✓	–
Generalized Jacobian (Umetani–Yoshida)	✓	–	–	–
Momentum-conserving base propagation	✓	–	Partial	–
Free-floating manipulability	✓	–	–	–
Free-floating resolved-rate control	✓	–	–	–
Pip-installable	✓	✓	✓	✓

[a]✓: provided out of the box; Partial: the underlying quantities are computable from the library's primitives, but the specific free-floating formulation (Umetani–Yoshida generalized Jacobian, momentum-conserving base propagation) or robot-chain automation is not provided; –: not provided.

components, [ω; v], while the base twist $\mathrm{v_b} = [\mathrm{v_b}; \omega_\mathrm{b}]$ follows the ordering returned by the library's momentum solver; a dual quaternion is written $\hat{\mathbf{q}} = \mathbf{q}_r + \varepsilon\mathbf{q}_d$ with Hamilton quaternion multiplication and scalar-first component ordering; pose composition acts by left multiplication; and the pose error between a target and the end-effector is the left error $\hat{\mathbf{q}}_t \otimes \hat{\mathbf{q}}_{ee}^*$, whose logarithm is expressed in the world frame. Only revolute joints are supported in the present release.

2.2 Forward kinematics via dual quaternion PoE

The forward kinematics computes the end-effector pose using the space-frame PoE formulation (Lynch and Park, 2017), implemented entirely in dual quaternion arithmetic, and is given by Equation 2:

$$\hat{\mathbf{q}}_{ee} = \hat{\mathbf{q}}_b \left(\prod_{i=1}^{N} \exp\left(\hat{\mathbf{S}}_i\, \theta_i\right) \right) \hat{\mathbf{q}}_M, \tag{2}$$

where $\hat{\mathbf{q}}_b$ is the base pose, $\hat{\mathbf{S}}_i$ is the screw axis of joint i encoded as a dual quaternion exponential, θ_i is the joint angle (the product runs over joints $i = 1, \ldots, N$ in order), and $\hat{\mathbf{q}}_M$ is the home configuration. Because the screw axes are captured in the base frame at construction, updating the base pose at run time yields correct world-frame kinematics for a manipulator on a moving spacecraft, the property the control module exploits to propagate the base during simulation. The module supports position-only and full 6-DOF outputs.

2.3 Jacobians and inverse kinematics

Two Jacobian implementations are provided, both producing the $6 \times N$ matrix mapping joint velocities to the spatial end-effector

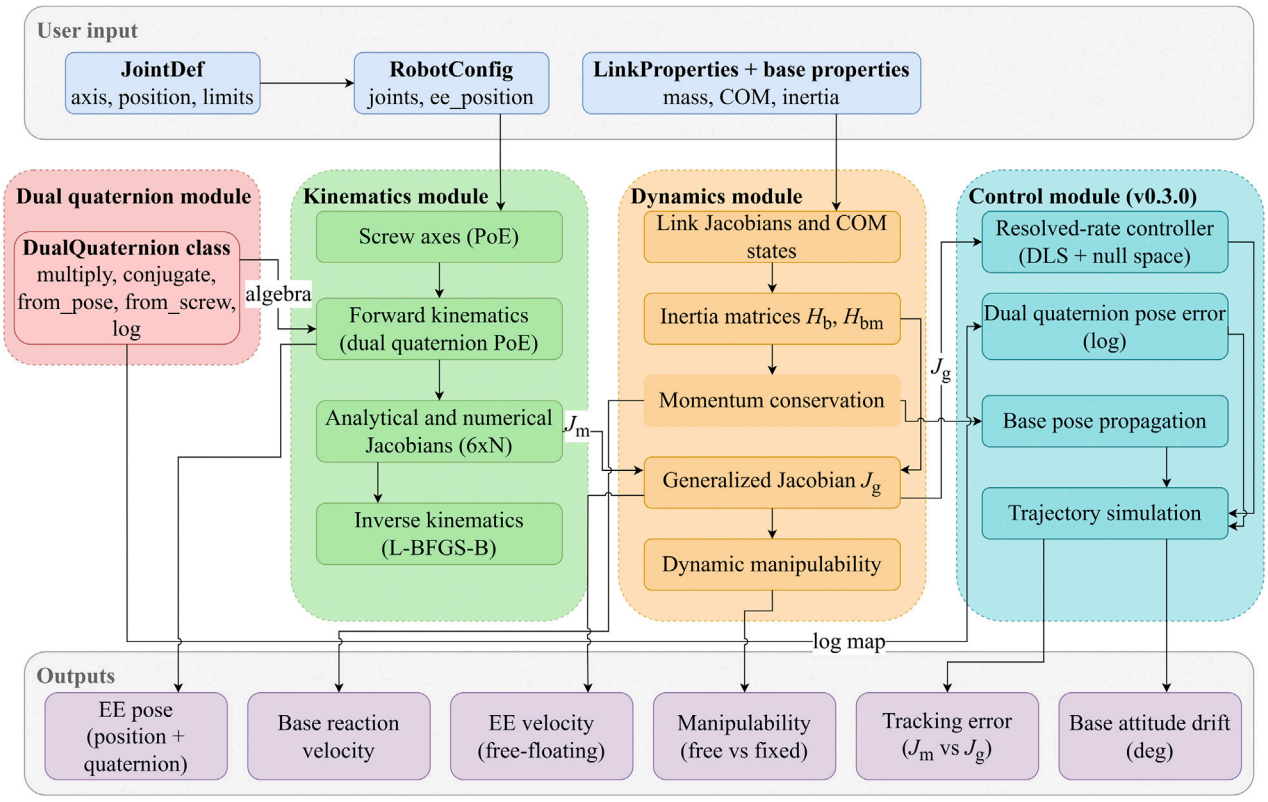


FIGURE 2
Software architecture of space-robot-dq.

velocity [**ω**; **v**]. The numerical geometric Jacobian applies central finite differences to the dual quaternion forward kinematics (FK) and is consistent with the FK by construction. The analytical geometric Jacobian is derived from the PoE by transforming each screw axis through the cumulative product of preceding joint motions; it agrees with the numerical Jacobian to approximately 5×10^{-9} (the finite-difference limit), is roughly an order of magnitude faster, and is therefore the default inside the simulation loops. Position-only and full 6-DOF inverse kinematics are provided via L-BFGS-B optimization, with joint-limit constraints, configurable position and orientation weights, a warm start from position-only inverse kinematics (IK), and a five-restart multi-start strategy.

2.4 Momentum conservation and the generalized Jacobian

For a free-floating system with zero initial momentum (Umetani and Yoshida, 1989), momentum conservation ties the base motion to the joint motion. The base motion is described by the twist $\boldsymbol{\nu}_b = [\boldsymbol{v}_b; \boldsymbol{\omega}_b]$, the stacked instantaneous linear and angular velocity of the base in the world frame. A twist is used because no globally defined pose vector exists on SE(3) whose time derivative could serve this role. The conservation constraint then reads

$$\mathbf{H}_b \boldsymbol{\nu}_b + \mathbf{H}_{bm} \dot{\mathbf{q}} = 0. \tag{3}$$

The library solves this for the base twist and assembles the generalized Jacobian of Equation 1. The spatial inertia matrices are built from per-link mass, center-of-mass, and inertia-tensor properties. A dynamic manipulability measure $w = \sqrt{\det(\mathbf{J}\mathbf{J}^{\mathrm{T}})}$, evaluated with either the generalized or the fixed-base Jacobian, quantifies velocity capability under free-floating versus fixed-base assumptions. Throughout this article, w is evaluated on the $3{\times}N$ position rows of the respective Jacobian, and the ratio $w_{\mathrm{free}}/w_{\mathrm{fixed}}$ compares the two at the same joint configuration.

2.5 Resolved-rate control on the free-floating plant

The control module closes the loop between the kinematic and dynamic layers. All controllers run against the same truth model: at every step, joint rates are commanded, the base twist follows from Equation 3, and a first-order exponential update propagates the base pose, so the plant is always the momentum-conserving free-floating system, regardless of what the controller believes. Two controller models are compared throughout: a naive controller using the fixed-base Jacobian, and the correct controller using the generalized Jacobian. The controllers are standard damped least-squares resolved-rate laws in the lineage of the generalized Jacobian formulation (Umetani and Yoshida, 1989). The module contributes a

TABLE 2 Summary of validation results, grouped by epistemic status.

Independent and physics-based validation		
Check	Metric	Result
FK vs. matrix exponential (25 configs)	Max position error	1.22×10^{-12} m
FK vs. elementary transforms (25 configs)	Max position error	1.22×10^{-12} m
FK orientation vs. both references	Max error	$<2 \times 10^{-6}$ deg
Numerical vs. analytical Jacobian (15 configs)	Max error	3.36×10^{-9}
Jacobian vs. independent reference	Max error	4.58×10^{-9}
Center-of-mass stationarity (moving-base simulation)	Max drift at dt = 0.01 s; convergence order	3.35×10^{-5} m; 1.00 (first order)
Heavy-base limit (10,000:1, unit test)	$\max\lvert\mathbf{J}_g - \mathbf{J}_m\rvert$	3.1×10^{-4}
IK success (100 reachable targets)[b]	Error <50 mm	93%
IK median (100 reachable targets)[b]	Median error	<0.01 mm
Consistency checks (satisfied by construction)		
Momentum (2 s trajectory, 200 steps)	Max $\lVert\mathbf{h}\rVert$	5.09×10^{-16}
Generalized Jacobian identity	Residual	$<10^{-15}$
Demonstration studies		
Resolved-rate tracking (generalized)	RMS error	0.2512 mm (105× lower than fixed-base model)[c]
Pose tracking, 5°/s tumble (generalized)	Pos/ori RMS	0.008429 mm/0.000581°

[b]Targets generated as forward kinematics of joint vectors sampled uniformly within joint limits and hence guaranteed reachable (Section 3.1).
[c]The factor quantifies the cost of neglecting the free-floating coupling (model mismatch); it is not a comparison between software tools (Section 3.5).

validated, open implementation of this free-floating baseline. Advanced schemes, such as the learning-based, statics-compensating cooperative controllers developed for fixed-base flexible-payload transport (Xie et al., 2025), can then be extended to the free-floating case and benchmarked against it.

For a 3-DOF Cartesian position task, the commanded velocity combines feedforward and proportional feedback, $\mathbf{v}_{cmd} = \mathbf{v}_{des} + k_p\,(\mathbf{p}_{des} - \mathbf{p}_{ee})$, and is mapped to joint rates by damped least squares:

$$\dot{\mathbf{q}} = \mathbf{J}^{\mathrm{T}}\left(\mathbf{J}\mathbf{J}^{\mathrm{T}} + \lambda\mathbf{I}\right)^{-1}\mathbf{v}_{cmd}, \tag{4}$$

with damping λ for singularity robustness, an element-wise joint-rate limit, and, for the redundant position task, a null-space posture term $(\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\,k_0\,(\mathbf{q}_{ref} - \mathbf{q})$ that keeps the arm away from singular configurations without disturbing the Cartesian task.

For the full 6-DOF pose task, the pose error is computed with dual quaternion algebra. Given the target pose $\hat{\mathbf{q}}_t$ and end-effector pose $\hat{\mathbf{q}}_{ee}$, the error dual quaternion and its logarithm,

$$\boldsymbol{\xi} = \log\left(\hat{\mathbf{q}}_t \otimes \hat{\mathbf{q}}_{ee}^{*}\right), \tag{5}$$

yield a screw whose rotational part is used as the orientation feedback, so the commanded spatial velocity is $[\boldsymbol{\omega}_t + k_o\,\boldsymbol{\xi}_{rot}\,;\ \mathbf{v}_t + k_p\,(\mathbf{p}_t - \mathbf{p}_{ee})]$, mapped through the full $6 \times N$ Jacobian by Equation 4. Using the dual quaternion logarithm as the pose-error metric puts the algebraic core of the library to work in the control layer and in the kinematics.

2.6 Implementation and availability

The library is written in Python (3.9 or later) and is platform-independent; the results in this article were verified independently on Windows 11 (Python 3.12.6, NumPy 2.5.1) and Ubuntu 24.04 (Python 3.12, NumPy 2.x). Runtime dependencies are NumPy and SciPy only; PyTorch tensor support is an optional extra. The code is released under the MIT license with no restrictions on non-academic use. Project links: https://github.com/HJahanshahi/space-robot-dq (development) and DOI 10.5281/zenodo.21969007 (archived release v0.3.1). Machine-readable citation metadata is provided in CITATION.cff.

3 Results

3.1 Validation

The library is validated through 153 unit tests (pytest) parametrized across four robot configurations (7-DOF SRS, 6-DOF standard, 3-DOF planar, and a custom 4-DOF robot), together with a 19-check benchmark suite. The benchmark suite compares the library against two independent implementations that share no code with the dual quaternion module, one based on the matrix exponential with the Rodrigues formula, and one based on elementary transforms. All benchmark checks are seeded and deterministic. Table 2 summarizes the results obtained on the Windows platform above; the Linux run reproduces every value to within last-digit floating-point

differences. The checks fall into two epistemically distinct groups. The first group contains independent, physics-based validation, that is, checks that could fail if the implementation were wrong. It includes cross-validation of the forward kinematics against the two independent implementations, agreement of the analytical Jacobian with finite differences, and convergence of the generalized Jacobian to the fixed-base Jacobian in the heavy-base limit. For the moving-base simulation, it also includes stationarity of the system's center of mass, which zero linear momentum requires but which nothing in the code enforces. The second group contains consistency checks that verify identities that the implementation satisfies by construction, such as the momentum residual of Equation 3 and the defining identity of the generalized Jacobian. These checks confirm numerical health at the 10^{-15} level, not physical correctness, and Table 2 labels them as such.

Forward kinematics agrees with both independent references below 10^{-11} m (picometer level) across all tested configurations, confirming the screw-theoretic formulation, the dual quaternion arithmetic, and the PoE composition. The three Jacobian implementations agree to approximately 5×10^{-9}, the limit of finite-difference precision at step size 10^{-7}. Momentum is conserved to the 10^{-16} level over trajectories, the generalized Jacobian satisfies its defining identity to better than 10^{-15}, and it converges to the fixed-base Jacobian in the heavy-base limit. Two invariants validate the moving-base simulation specifically: momentum stays below 10^{-12} throughout closed-loop runs, and the system's center of mass, which zero linear momentum requires to be stationary (a property not enforced by construction), drifts only at the level of the first-order integrator error and converges with decreasing step size. Quantitatively, halving the step size from 0.02 s to 0.0025 s reduces the maximum center-of-mass drift from 6.71×10^{-5} m to 8.34×10^{-6} m, with observed convergence orders of 1.01, 1.00, and 1.00 across the three successive halvings, matching the first-order accuracy of the exponential base-pose integrator.

The inverse-kinematics statistics in Table 2 follow a stated protocol: the 100 targets are generated as the forward kinematics of joint vectors sampled uniformly within the joint limits and are therefore reachable by construction; success is defined as a final position error below 50 mm. The error distribution is strongly bimodal: 93 of 100 runs converge to sub-0.01 mm accuracy, so the median error is effectively zero, while the remaining runs stall in optimizer local minima at tens of millimeters despite the five-restart multi-start strategy; the 50 mm threshold therefore separates converged from non-converged runs rather than grading intermediate accuracy.

3.2 Example of use: Defining a custom robot

The library supports arbitrary serial chains through a declarative interface. The following complete example defines a custom 4-DOF robot and computes its forward kinematics and generalized Jacobian; users specify only each joint's rotation axis and home position, plus the end-effector location.

```
from space_robot_dq import (
  RobotConfig, JointDef,
  SpaceRobotKinematics,
  SpaceRobotDynamics)
import numpy as np
config = RobotConfig(
  joints=[
    JointDef(axis=[0,0,1],
      position=[0,0,0],
      name="yaw"),
    JointDef(axis=[0,1,0],
      position=[0,0,0.3],
      name="pitch1"),
    JointDef(axis=[0,1,0],
      position=[0,0,0.6],
      name="pitch2"),
    JointDef(axis=[1,0,0],
      position=[0,0,0.9],
      name="roll"),
  ],
  ee_position=[0, 0, 1.1],
  name="Custom 4-DOF Arm")
kin = SpaceRobotKinematics(config)
dyn = SpaceRobotDynamics(
  kinematics=kin, base_mass=50.0)
q = np.array([0.5, 0.3, 0.8, 0.2])
pos, quat = kin.forward_kinematics_6dof(q)
J_g, J_m, _ = (
  dyn.compute_generalized_jacobian(q))
```

3.3 Momentum coupling along a trajectory

To demonstrate the central physical effect, a 2 s sinusoidal joint trajectory $\mathbf{q}(t) = \mathbf{q}_0 + \mathbf{A}\sin(\boldsymbol{\omega} t)$ is simulated for the built-in 7-DOF SRS manipulator with a 100 kg base and 20.67 kg total arm mass (the preset's auto-generated link properties; all trajectory parameters are stated in the reproduction script). At each of 200 timesteps, the base reaction follows from momentum conservation, the base pose is propagated, and the end-effector speed is evaluated with both the fixed-base and the generalized Jacobian. Figure 3 presents the results: panel (a) compares the end-effector speed predicted by the fixed-base Jacobian (blue solid) with the true free-floating value (red dashed), the shaded region being the velocity lost to dynamic coupling; panel (b) shows the linear part of the base reaction twist $\boldsymbol{\nu}_b$, which peaks at 0.030 m/s; and panel (c) shows the total system momentum, which remains at approximately 10^{-15} throughout, a consistency check of the momentum-constraint implementation (Section 3.1), with independent validation provided by the center-of-mass invariant.

At $t = 0$, the fixed-base model predicts an end-effector speed of 0.843 m/s while the actual free-floating speed is 0.643 m/s, a 24% overestimation, and the loss ranges from 10% to 24% along the trajectory. The spacecraft base also accumulates 6.9° of attitude excursion over the 2 s motion. The design implication is direct: velocity commands and feasibility analyses based on the fixed-base Jacobian systematically

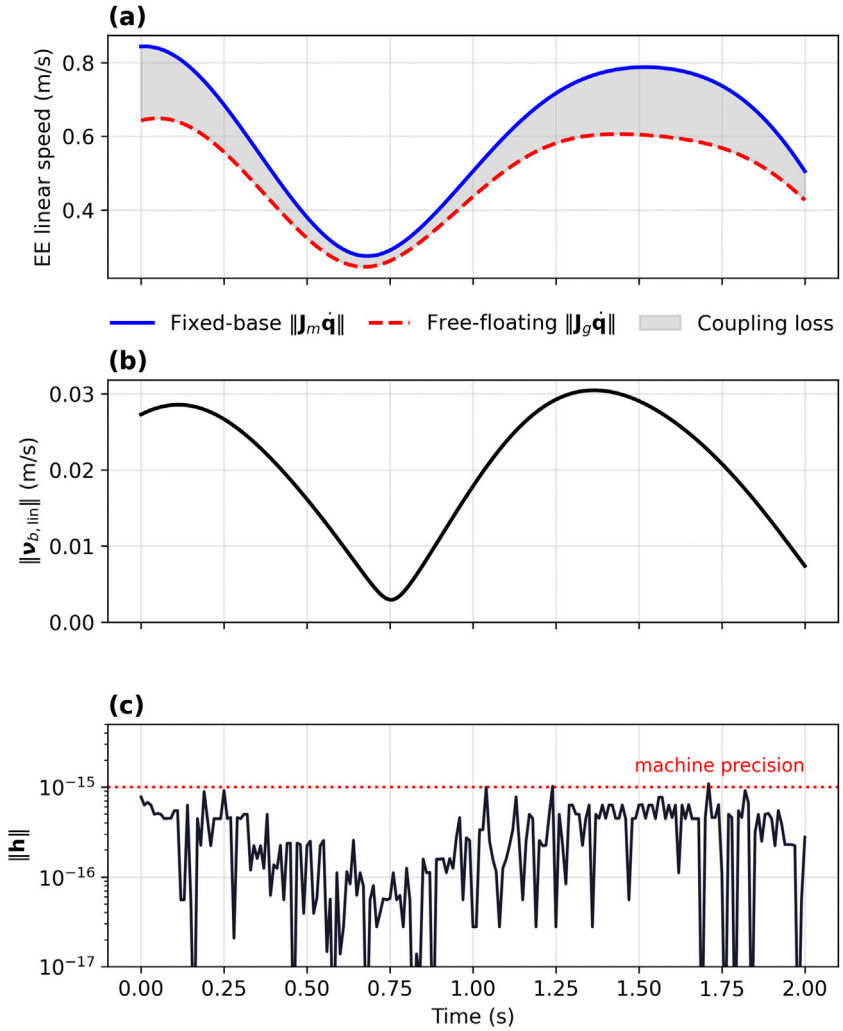


FIGURE 3
Trajectory simulation over 2 s: **(A)** end-effector linear speed, **(B)** base reaction twist $\boldsymbol{\nu}_b$, and **(C)** total system momentum.

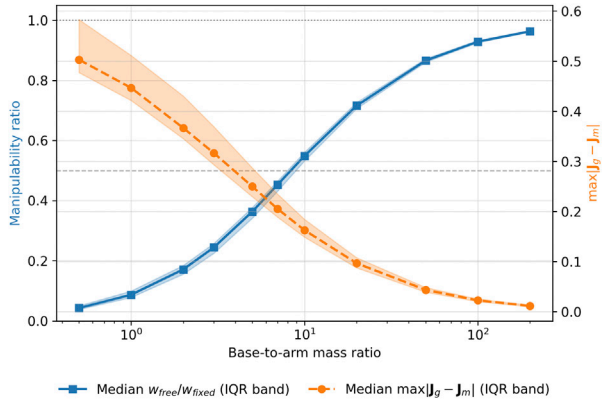


FIGURE 4
Base-to-arm mass ratio versus manipulability ratio (left axis) and $\max|\mathbf{J}_g - \mathbf{J}_m|$ (right axis); medians with interquartile bands over 20 configurations.

overpromise, and the generalized Jacobian provides the correct mapping.

3.4 Effect of the base-to-arm mass ratio

The severity of the coupling depends on the base-to-arm mass ratio (Figure 4). To assess configuration dependence, the sweep is evaluated over 20 random non-degenerate configurations (sampled uniformly within ±1.2 rad per joint, retaining configurations with $w_{\mathrm{fixed}} > 0.05$; seed 2026). Figure 4 reports medians with interquartile bands. The left axis gives the manipulability ratio $w_{\mathrm{free}}/w_{\mathrm{fixed}}$, and the right axis gives the maximum element-wise difference $\max|\mathbf{J}_g - \mathbf{J}_m|$ between the generalized and fixed-base Jacobians. The dotted and dashed horizontal guide lines mark unity and 50% retention on the left axis. At 0.5:1 (arm heavier than base), the median retained fraction is only 4.3% of the fixed-base velocity capability; at 5:1, it is 36%; at 7:1, it is 45%; at 10:1, it is 55%; and at 100:1, it

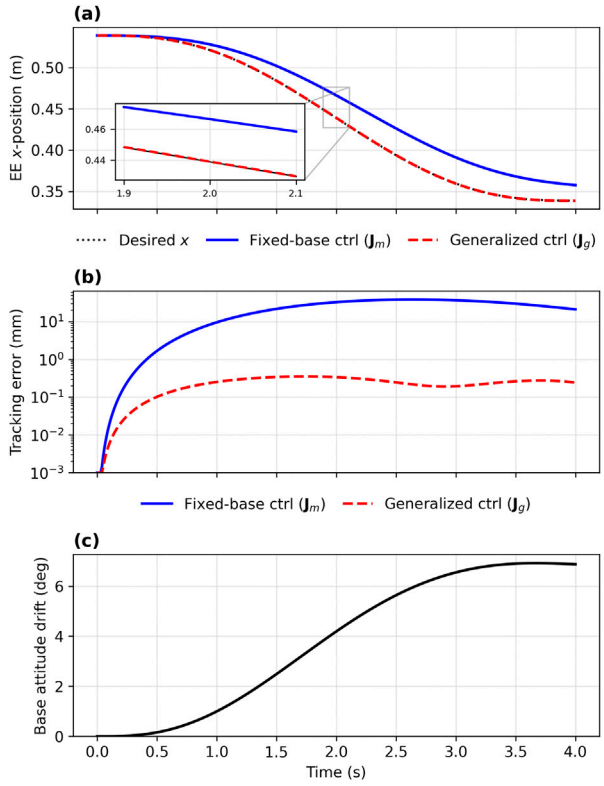


FIGURE 5
Closed-loop resolved-rate tracking of a 0.27 m line move: **(a)** end-effector x-position, **(b)** tracking error, and **(c)** base attitude drift.

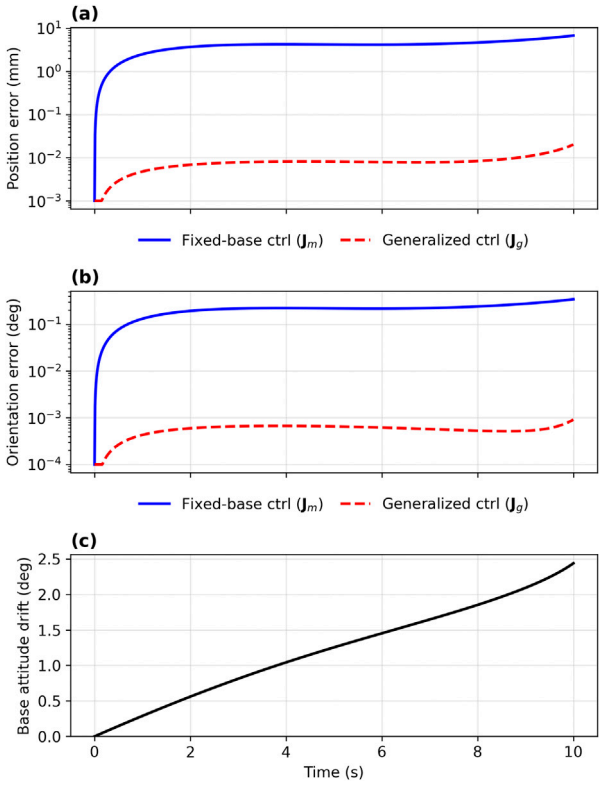


FIGURE 6
Six-DOF station-keeping against a client tumbling at 5°/s: **(a)** position error, **(b)** orientation error, and **(c)** base attitude drift.

reaches 93%. The interquartile spread is small throughout (typically ±0.02), and it shrinks with increasing mass ratio, so for this manipulator geometry, the trend is robust to configuration, although the thresholds remain geometry-specific. As a practical guideline for this manipulator, below roughly 8:1, less than half of the fixed-base capability is available (median 45% at 7:1), and the free-floating coupling dominates its behavior. Above roughly 50:1, the fixed-base model becomes an increasingly reasonable approximation (median 87% at 50:1). This quantitative relationship lets mission designers estimate the minimum spacecraft mass needed for acceptable manipulator performance.

3.5 Closed-loop resolved-rate tracking

The first control study quantifies what the coupling costs a controller. The end-effector must follow a 0.27 m straight-line Cartesian path with quintic rest-to-rest timing over 4 s (k_p = 1 s^{-1}, dt = 10 ms). Both controllers run on the same free-floating plant; they differ only in the Jacobian used to map commanded velocity to joint rates. Figure 5 presents the results: panel (a) shows the end-effector x-position against the desired path (dotted black), with the generalized-Jacobian controller (dashed red) and the fixed-base controller (solid blue); an inset magnifies a mid-trajectory window in which the fixed-base trajectory is visibly offset while the generalized-Jacobian trajectory is indistinguishable from the desired one (the desired path is drawn as a thin solid line inside the inset for visibility); panel (b) shows the tracking-error histories; and panel (c) shows the spacecraft base attitude drift.

The fixed-base controller misses by 26.372 mm root mean square (RMS) and 21 mm at the terminal point, even with proportional feedback, because the model error, the unmodeled recoil of the base, regenerates at every instant of motion. The generalized-Jacobian controller tracks at 0.2512 mm RMS, a 105-fold difference that measures what neglecting the coupling costs the controller model, with momentum conserved to 3×10^{-16} throughout. This single 27-cm arm motion also rotates the spacecraft by 6.9°, an attitude cost that must be budgeted in reaction-wheel sizing or recovered after the maneuver, and which is essentially identical for both controllers because it is dictated by conservation, not by controller choice.

3.6 Six-DOF pose tracking of a tumbling target

The second study is a capture-approach scenario. The end-effector must maintain a full 6-DOF grasp-ready pose relative to a client object tumbling at 5° per second, a realistic rate for debris and defunct satellites. The pose is held for 10 s, during which the target accumulates 50° of rotation. The pose error is computed by the dual quaternion logarithm of Equation 5, and the gains are $k_p = k_o = 1\ s^{-1}$. Figure 6 presents the results. Panels (a) and (b) show the position and orientation errors on logarithmic scales.

The initial vertical rise occurs because the error starts at exactly zero, which lies below the plotting floor of the logarithmic axes. The mild upturn near $t = 10$s marks the point where the target's accumulated rotation approaches the boundary of the arm's orientation workspace for this configuration. Panel (c) shows the spacecraft base attitude drift.

The generalized-Jacobian controller holds the grasp pose at 0.008429 mm and 0.000581° RMS, so the end-effector is, for practical purposes, rigidly attached to the tumbling client. The fixed-base controller wanders at 4.267 mm and 0.220°. In a capture context, a few millimeters of unmodeled sway at the gripper separates a clean grasp from an uncontrolled contact event. The spacecraft pays for the 10 s of station-keeping with 2.4° of attitude drift, again dictated by momentum conservation. As in Section 3.5, the 506-fold and 379-fold ratios quantify the model mismatch, that is, the penalty for controlling a free-floating system with a fixed-base model. Extending the same scenario to 15 s characterizes the operating envelope. Errors remain at the micrometer level through 50° of accumulated target rotation (0.020 mm at 50°). Degradation begins near 55° (1.17 mm, 0.086°), reaches 2.6 mm at 60°, and tracking is lost beyond about 65° (48 mm, 8.7°) as the joints approach their $\pm\pi$ limits (maximum joint excursion 3.11 rad). The envelope is thus bounded by the arm's orientation workspace from this configuration rather than by the controller.

3.7 Limitations

The plant model is kinematic-plus-momentum. Joint rates are commanded directly, and the base follows the momentum constraint, which is the standard formulation for resolved-rate studies of free-floating arms (Umetani and Yoshida, 1989). Joint-torque dynamics, actuator models, flexibility, and contact forces are not modeled. Base propagation uses a first-order exponential integrator, whose error is monitored through the center-of-mass invariant and converges with decreasing step size. Task feasibility is bounded by the arm's workspace. In the tumbling scenario, sufficient accumulated target rotation takes the commanded pose outside the reachable orientation workspace, and both controllers degrade. Section 3.6 quantifies the onset near 55° and the loss of tracking beyond about 65° for the studied configuration. The simulation reports this degradation explicitly. The library currently supports only revolute joints. The validation is numerical; no hardware experiments are included. The plant also assumes exact knowledge of the joint and base states. In flight, the base pose and rate must be estimated from noisy, intermittently available measurements (Jahanshahi and Zhu, 2026).

4 Discussion

The space-robot-dq library addresses a gap in the Python ecosystem for space robotics (Table 1). The control studies of Sections 3.5 and 3.6 demonstrate the engineering consequences of that gap. In realistic maneuvers, the difference between the fixed-base and generalized Jacobians amounts to two to three orders of magnitude of tracking accuracy, reflecting the cost of the model mismatch. The library serves three audiences. Researchers developing control algorithms for on-orbit servicing obtain a validated kinodynamic model of the free-floating coupling, and a working closed-loop baseline, without re-implementing the Umetani–Yoshida formulation. For educators, it is a transparent implementation that students can inspect, with every figure of this article regenerable by one script. Mission designers can use the mass-ratio analysis of Figure 4 and the attitude-drift accounting of Figures 5, 6 for spacecraft sizing and attitude-budget decisions. To be clear, the performance differences arise entirely from the generalized Jacobian, that is, from modeling the momentum coupling. The dual quaternion layer supplies the singularity-free pose representation and the logarithmic pose-error metric of Equation 5.

The library scales across arm sizes. The declarative interface models any serial chain, from a 2-DOF testbed to a Canadarm-class system, without modifying library code. All algorithms are $O(N)$ or $O(N^2)$ in the joint count, with small constants. On commodity hardware, dual quaternion FK evaluates in approximately 0.2 ms, the analytical Jacobian in approximately 0.4 ms, and the generalized Jacobian in approximately 8 ms (dominated by the numerical-Jacobian default; approximately 1 ms with the analytical option used by the control loops). These timings support hundreds of simulation steps per second for the 7-DOF studies shown. In our codebase, the dual quaternion FK is approximately three times slower than the plain matrix-exponential FK. This is an implementation-level comparison against a PoE baseline that shares the same Python-level structure, and it is not a property of the algebra, whose operation counts favor quaternion composition. Reported dual-quaternion speedups elsewhere are measured against exponential-product FK (Ge et al., 2019) or DH-based homogeneous-matrix composition (Chandrasekar and Gnanasekar, 2025), so timings do not transfer across implementations. The representation is chosen for its singularity-free algebra and its role in the pose-error metric rather than for raw speed. Both implementations ship in the benchmark suite for comparison.

A reaction null-space term, using kinematic redundancy to reduce base disturbance (Nenchev et al., 1999), was prototyped during this study. It reduced attitude drift by only about 15% for the studied configuration and task, and it is left for a dedicated treatment with momentum-decomposition methods. Further directions include multi-arm cooperative manipulation, torque-level dynamics with actuator models, hardware-in-the-loop validation, and prismatic joints, whose uniform treatment alongside revolute pairs is a noted advantage of quaternion-based formulations (Qiao et al., 2010; Chandrasekar and Gnanasekar, 2025).

5 Conclusion

This article presented space-robot-dq, an open-source, pip-installable Python library that unifies dual quaternion kinematics, free-floating dynamics, and resolved-rate control for serial-chain space manipulators with revolute joints. The implementation was verified against two independent reference implementations and

physics-based invariants. The checks show picometer-level forward-kinematics agreement, base propagation that conserves momentum and converges at first order, and a stationary system center of mass. Two included studies quantified the engineering cost of neglecting the free-floating coupling. In Cartesian tracking, a fixed-base controller model accumulates tens of millimeters of error, while the generalized-Jacobian controller holds sub-millimeter accuracy. In the capture-approach scenario, the generalized-Jacobian controller maintains a 6-DOF grasp pose against a 5°/s tumbling client at micrometer and millidegree levels, while the spacecraft base drifts 2.4°. The mass-ratio analysis provides a configuration-robust design guideline for this class of manipulator, and the operating-envelope characterization delimits where the approach applies. All figures and quoted numbers are regenerated by a single deterministic script in the permanently archived repository. The result is a validated, reproducible baseline for research, education, and mission analysis in on-orbit servicing.

Data availability statement

The complete source code, the 153-test suite, the 19-check benchmark suite, and the script examples/generate_paper_figures.py that deterministically reproduces every figure and number in this article are openly available at https://github.com/HJahanshahi/space-robot-dq. The exact version described here (v0.3.1) is permanently archived at Zenodo under DOI 10.5281/zenodo.21969007; the originally submitted version (v0.3.0, DOI 10.5281/zenodo.21433100) remains available under the same concept record. The software is written in Python (3.9 or later), is platform independent, and is released under the MIT license with no restrictions on non-academic use.

Author contributions

HJ: Conceptualization, Formal Analysis, Funding acquisition, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. ZHZ: Conceptualization, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Writing – review and editing.

Funding

The author(s) declared that financial support was received for this work and/or its publication. This work was supported by the Discovery Grant (RGPIN-2024-06290) and Collaborative Research and Training Experience Program Grant (555425-2021) of the Natural Sciences and Engineering Research Council of Canada, awarded to ZHZ. HJ acknowledges support from the Natural Sciences and Engineering Research Council of Canada (NSERC) through a Postgraduate Scholarship–Doctoral (PGS D), reference number 601200. HJ remercie le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG) de son soutien par l'entremise d'une bourse d'études supérieures du CRSNG–doctorat (ES D), numéro de référence 601200.

Conflict of interest

The author(s) declared that this work was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declared that generative AI was not used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Carpentier, J., Saurel, G., Buondonno, G., Mirabel, J., Lamiraux, F., Stasse, O., et al. (2019). "The Pinocchio C++ library: a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *2019 IEEE/SICE International Symposium on System Integration (SII)*, 614–619. doi: 10.1109/SII.2019.8700380

Chandrasekar, S. K., and Gnanasekar, R. (2025). A dual quaternion approach to kinematic modeling and evaluation of PRR-type robotic manipulators. *Prog. Eng. Sci.* 2, 100174. doi: 10.1016/j.pes.2025.100174

Corke, P., and Haviland, J. (2021). *Not your Grandmother's toolbox–the Robotics Toolbox Reinvented for Python*. IEEE International Conference on Robotics and Automation ICRA, 11357–11363. doi: 10.1109/ICRA48506.2021.9561366

Dubowsky, S., and Papadopoulos, E. (1993). The kinematics, dynamics, and control of free-flying and free-floating space robotic systems. *IEEE Trans. Robotics Automation* 9, 531–543. doi: 10.1109/70.258046

Funda, J., Taylor, R. H., and Paul, R. P. (1990). On homogeneous transforms, quaternions, and computational efficiency. *IEEE Trans. Robotics Automation* 6, 382–388. doi: 10.1109/70.56658

Ge, W., Chen, L., Wang, X., Xing, E., and Zielinska, T. (2019). "Kinematics modeling and analysis of manipulator using the dual quaternion," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, 750–755. doi: 10.1109/ICMA.2019.8816603

Jahanshahi, H., and Zhu, Z. H. (2024). Review of machine learning in robotic grasping control in space application. *Acta Astronaut.* 220, 37–61. doi: 10.1016/j.actaastro.2024.04.012

Jahanshahi, H., and Zhu, Z. H. (2025). A comprehensive review of tactile sensing technologies in space robotics. *Chin. J. Aeronautics* 38, 103423. doi: 10.1016/j.cja.2025.103423

Jahanshahi, H., and Zhu, Z. H. (2026). Uncertainty propagation networks for neural ordinary differential equations. *Neurocomputing* 677, 133134. doi: 10.1016/j.neucom.2026.133134

Jahanshahi, H., Alizadeh, M., and Zhu, Z. H. (2026). Hardware-in-the-loop emulation for 6-DOF free-floating spacecraft using active gravity compensation with a collaborative robot. *Acta Astronaut.* 247, 782–794. doi: 10.1016/j.actaastro.2026.05.058

Li, X., Lu, Q., Huang, Z., and Li, K. (2025). Robust torque-computed control for a robot manipulator with unit dual quaternion. *IEEE Trans. Industrial Electron.* 72, 10728–10738. doi: 10.1109/TIE.2025.3552269

Liendo, F., Hernández, C., and Galez, C. (2025). An improved dual quaternion dynamic movement primitives-based algorithm for robot-agnostic learning and execution of throwing tasks. *J. Mech. Robotics* 17, 091012. doi: 10.1115/1.4068555

Löw, T., de Paula Assis Fonseca, M., Pruks, V., Deacon, G., Konstantinova, J., and Calinon, S. (2026). Dual-arm admittance control using conformal geometric algebra. *Front. Robot. AI* 13, 1807613. doi: 10.3389/frobt.2026.1807613

Lu, Y., and Yue, B. (2025). Dynamic modeling of liquid-filled free-floating space robot and joint trajectory planning with considering liquid positioning. *Aerosp. Sci. Technol.* 161, 110133. doi: 10.1016/j.ast.2025.110133

Lynch, K. M., and Park, F. C. (2017). *Modern Robotics*. Cambridge University Press.

Ma, B., Jiang, Z., Liu, Y., and Xie, Z. (2023). Advances in space robots for On-Orbit servicing: a comprehensive review. *Adv. Intell. Syst.* 5, 2200397. doi: 10.1002/aisy.202200397

Nenchev, D. N., Yoshida, K., Vichitkulsawat, P., and Uchiyama, M. (1999). Reaction null-space control of flexible structure mounted manipulator systems. *IEEE Trans. Robotics Automation* 15, 1011–1023. doi: 10.1109/70.817666

Qiao, S., Liao, Q., Wei, S., and Su, H.-J. (2010). Inverse kinematic analysis of the general 6R serial manipulators based on double quaternions. *Mech. Mach. Theory* 45, 193–199. doi: 10.1016/j.mechmachtheory.2009.05.013

Tang, R., Liu, Y., Yang, J., Ma, X., and Yan, W. (2025). Explicit-time trajectory tracking for a state-constraint continuum free-floating space robot with smooth joint-path and low input. *Appl. Sci.* 15, 2730. doi: 10.3390/app15052730

Umetani, Y., and Yoshida, K. (1989). Resolved motion rate control of space manipulators with generalized Jacobian matrix. *IEEE Trans. Robotics Automation* 5, 303–314. doi: 10.1109/70.34766

Verheye, A. (2025). Dual_quaternions. Available online at: https://github.com/Achllle/dual_quaternions (Accessed July 18, 2026).

Xie, M., Chen, T., Ni, S., and Feng, C. (2025). Flexible payload transportation using cooperative space manipulators with statics compensation. *ISA Trans.* 157, 329–339. doi: 10.1016/j.isatra.2024.12.001