Proceedings of the 12th International Conference of Control Systems, and Robotics (CDSR 2025)
July 13, 2025 - July 15, 2025 | Imperial College London Conference Center, London, United Kingdom
Paper No. 126
DOI: 10.11159/cdsr25.126

A Computational Tool for On-Site Customization of Space Robotic Manipulators through Dynamic Analysis

Ethan White[1], Md Assad-Uz-Zaman[1], Md Rasedul Islam [1]
[1]Mechanical Engineering Department, University of Wisconsin-Green Bay
2420 Nicolet Drive, Green Bay, WI 54311, USA
white30@uwgb.edu; uzm@uwgb.edu; islamm@uwgb.edu

Abstract –Space missions often face unexpected challenges requiring the rapid deployment of robotic manipulators. Given the dynamic nature, evolving priorities, and extraterrestrial environments of these missions, the ability to develop robotic manipulators on-site is crucial for enhancing mission flexibility. A key aspect of this development is the analysis of manipulator dynamics, which is essential for optimizing design, control, and payload handling. To address this need, in this research, we developed a computational tool for real-time dynamic analysis of robotic manipulators. This tool will enable astronauts to swiftly customize manipulator designs and assess their performance on-site, allowing for adaptive problem-solving and efficient modification to meet mission objectives. By providing an accessible and responsive analytical resource, our approach enhances the adaptability and efficiency of space operations, ensuring that robotic manipulators can be tailored to evolving mission demands. This tool takes the modified DH parameters, lengths, mass and inertia as input and outputs joint torques, M, V and G function. The tool has been tested with two degrees of freedom planar robot for known values and been able to successfully compute dynamics.

Keywords**:** Robot, Manipulator, Dynamics, Computational Tool

1. Introduction

In modern space exploration, robotic systems have become indispensable for performing a wide array of complex operations, including docking, refueling, repairing, upgrading, transportation, rescue missions, and orbital debris removal [1, 2]. Over the past four decades, a variety of robotic manipulators have been designed, tested, and deployed in space missions, particularly for on-orbit servicing and debris mitigation [3-5]. While these robots undergo extensive ground-based verification in simulated space environments before deployment, the rapidly evolving landscape of space missions now demands greater adaptability and responsiveness in robotic systems [6,7]. This increasing demand underscores the necessity for on-site development capabilities, enabling real-time adjustments to robotic operations in response to unforeseen challenges and mission-specific conditions. However, the current limitations in on-site customization significantly hinder the ability of robotic manipulators to navigate dynamic extraterrestrial environments and execute tasks efficiently.

A fundamental aspect of on-site customization for space robotic manipulators is the development and analysis of their dynamic behavior. Although extensive research has been conducted on modeling and formulating the dynamics of space robots, these studies have primarily focused on static analysis and pre-mission programming [8,9]. While such approaches have provided valuable insights into the design and control of robotic manipulators, they often fall short in addressing real-time challenges encountered during actual space missions. In space, unpredictable environmental factors and mission-specific requirements necessitate robotic systems capable of adaptive on-site modifications—a capability that conventional pre-programmed methodologies fail to offer [10].

Motivated by the vision of enhancing space mission autonomy and efficiency, this research proposes a computational tool designed to compute the dynamics of robotic manipulators given their kinematic and inertial parameters. This tool aims to bridge the gap between pre-mission static analysis and real-time dynamic adaptability, enabling on-site customization of robotic behavior in response to evolving mission conditions. This study makes the following key contributions:

I. Development of a generalized dynamic model for an n-Degree of Freedom (n-DoF) manipulator using an iterative recursive Newton-Euler formulation.
II. Creation of a computational framework to compute essential dynamic quantities, including joint velocities, accelerations, forces, and torques, while also accounting for external interaction forces and torques exerted by the environment.

By introducing a computationally efficient, adaptable, and mission-responsive approach to dynamic analysis, this research aims to augment the capabilities of space-based robotic systems, ensuring enhanced performance and reliability in challenging extraterrestrial conditions.

2. Kinematics of n-DoF robotic manipulator

The kinematic configuration of an n-Degree of Freedom (DoF) robotic manipulator is illustrated in Fig. 1. An orthogonal Cartesian coordinate system is assigned to each link, with $\hat{Z}-axes$ passing through the manipulator joints, $\hat{X}-axes$ aligned along the common normal between two successive joint axes, and $\hat{Y}-axes$ determined using the right-hand screw rule. Subsequently, coordinate frames for each link and joint are established based on the modified Denavit-Hartenberg (DH) convention [11,12]. This leads to the calculation of four modified DH parameters for each frame, namely link length, link twist, joint angle, and link offset. The position and orientation of the i-th link frame with respect to its preceding frame (i-1) th can be computed using the determined modified DH parameters as follows [12]:

$$T_{i,\ i-1} = \begin{bmatrix} R_{i,\ i-1} & P_{i,\ i-1} \\ 0_{1\times3} & 1_{1\times1} \end{bmatrix} \tag{1}$$

Where,

$$T_{i,\ i-1} \in \mathbb{R}^{4\times4} = \text{Description of Frame } \{\text{i}\} \text{ with respect to frame } \{\text{i}-1\}$$
$$R_{i,\ i-1} \in \mathbb{R}^{3\times3} = \text{Rotaion matrix of Frame } \{\text{i}\} \text{ with respect to frame } \{\text{i}-1\}$$
$$P_{i,\ i-1} \in \mathbb{R}^{3\times1} = \text{Rotaion matrix of Frame } \{\text{i}\} \text{ with respect to frame } \{\text{i}-1\}$$

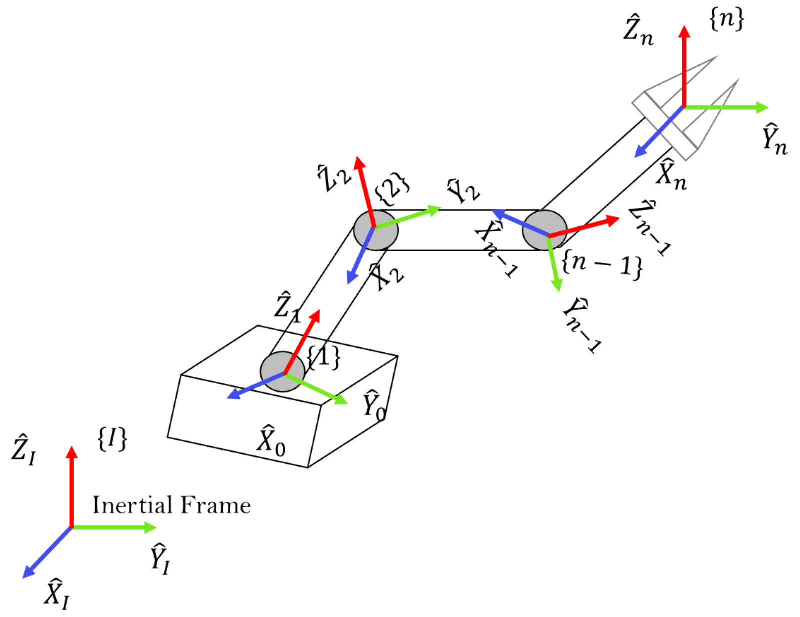

Fig. 1: Kinematic configuration of a n-DoF space robotic manipulator

The kinematic position and orientation of the robotic manipulator's end-effector relative to the inertial (reference frame) can be determined by applying transformation matrices to the assigned coordinate frames to the robot’s links. This information is expressed through the successive multiplication of transformation matrices [13].

$$T_{n,\ I} = T_{I,\ 0} * T_{0,\ 1} * T_{2,\ 1} * \dots \dots * T_{n-1,\ n-2} * T_{n,\ n-1} \tag{2}$$

Note that $T_{n,\ I}$ is a function of joint variables, meaning that it yields the description of the end-effector when provided with values for the joint variables.

3. Dynamics of n-DoF robotic manipulator

The initial phase in constructing a dynamic model for a manipulator involves computing the velocities and accelerations generated at each joint, with the understanding that these values will propagate from one link to the next. The velocity vectors of adjacent links are illustrated in Fig.2. By leveraging the information pertaining to the description of each link, the velocity of each link can be computed and expressed within that particular link. The outward iteration to compute angular and linear velocities is outlined as follows [11,13]:

$$\omega_{i+1,\ i+1} = R_{i,\ i+1}\,\omega_{i,\ i} + \dot{\theta}_{i+1}\,\hat{Z}_{i+1,\ i+1} \tag{3}$$

$$v_{i+1,\ i+1} = R_{i,\ i+1}\left(v_{i,\ i} + \omega_{i,\ i} \times P_{i,\ i+1}\right) + \dot{d}_{i+1}\,\hat{Z}_{i+1,\ i+1} \tag{4}$$

Where,

$\omega_{i+1,\ i+1}$ = Angular velocity of frame $\{i+1\}$ described in $\{i+1\}$

$\omega_{i,\ i}$ = Angular velocity of frame $\{i\}$ described in $\{i\}$

$v_{i+1,\ i+1}$ = Linear velocity of frame $\{i+1\}$ described in $\{i+1\}$

$v_{i,\ i}$ = Linear velocity of frame $\{i\}$ described in $\{i\}$

$\dot{\theta}_{i+1}$ = Joint rate at i + 1 th revolute joint; 0 if joint is prismatic

$\dot{d}_{i+1}$ = Joint rate at i + 1 th prismatic joint; 0 if joint is revolute

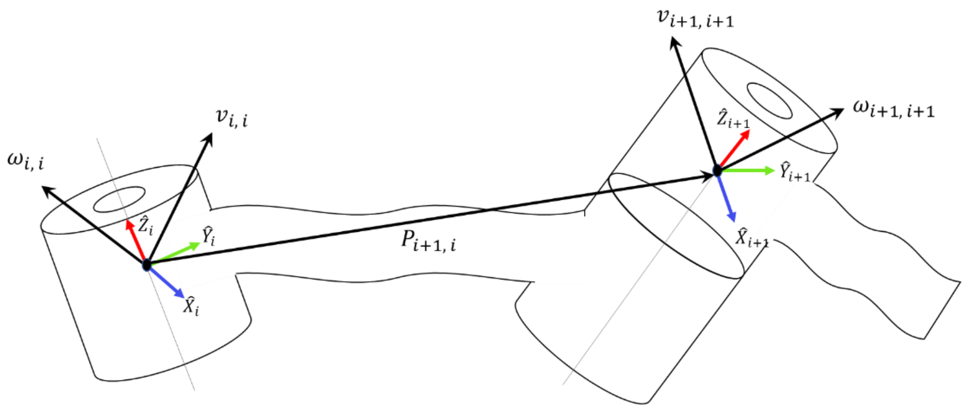

Fig. 2: Velocity vectors of neighboring links

Utilizing Equations (3) and (4), the velocities for all frames (i.e., n frames in total) can be determined. Likewise, the propagation of angular acceleration and linear acceleration within link frames can be calculated as follows:

$$\dot{\omega}_{i+1,\ i+1} = R_{i,\ i+1}\,\omega_{i,\ i} + R_{i,\ i+1}\,\omega_{i,\ i} \times \dot{\theta}_{i+1}\,\hat{Z}_{i+1,\ i+1} + \ddot{\theta}_{i+1}\,\hat{Z}_{i+1,\ i+1} \tag{5}$$

$$\dot{v}_{i+1,\ i+1} = R_{i,\ i+1}\left[\dot{v}_{i,\ i} + \dot{\omega}_{i,\ i} \times P_{i,\ i+1} + \omega_{i,\ i} \times \left(\omega_{i,\ i} \times P_{i,\ i+1}\right)\right] + 2\,\omega_{i+1,\ i+1} \times \dot{d}_{i+1}\,\hat{Z}_{i+1,\ i+1} + \ddot{d}_{i+1}\,\hat{Z}_{i+1,\ i+1} \tag{6}$$

Where,

$\ddot{\omega}_{i+1,\ i+1}$ = Angular acceleration of frame $\{i+1\}$ described in $\{i+1\}$

$\ddot{v}_{i+1,\ i+1}$ = Linear acceleration of frame $\{i+1\}$ described in $\{i+1\}$

$\ddot{\theta}_{i+1}$ = Joint acceleration at i + 1 th revolute joint; 0 if joint is prismatic

$\ddot{d}_{i+1}$ = Joint acceleration at i + 1 th prismatic joint; 0 if joint is revolute

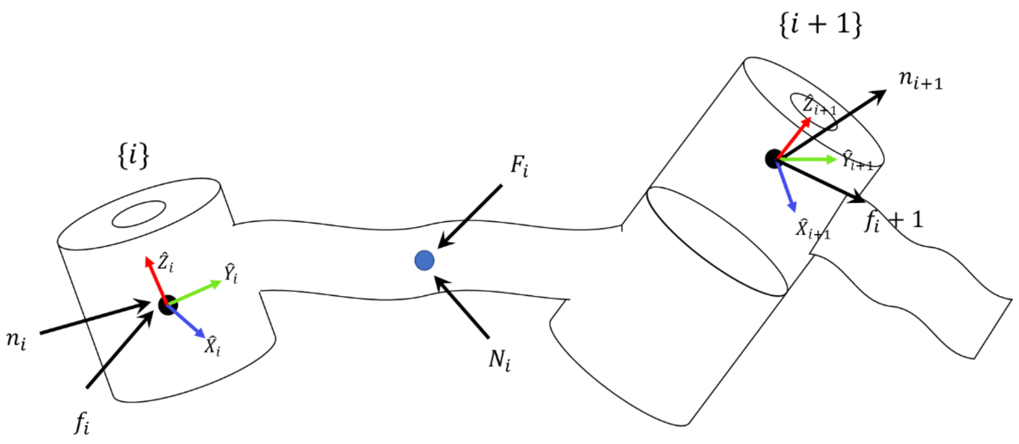

Fig. 3: The force balance, including inertial forces, for neighboring links.

The propagated linear and angular accelerations, along with velocities, will result in an acceleration at the center of mass, expressed as follows:

$$\dot{v}_{C_{i,\ i}} = \dot{\omega}_{i,\ i} \times P_{C_{i,\ i}} + \omega_{i,\ i} \times \left(\omega_{i,\ i} \times P_{C_{i,\ i}}\right) + \dot{v}_{i,\ i} \qquad (7)$$

The Force $(F_{i,\ i})$ and Moment $(N_{i,\ i})$ generated by this acceleration, as illustrated in Fig. 3, can be calculated using the Newton-Euler iterative formulation provided below. This formulation ultimately yields the joint forces $(f_{i,\ i})$ and moments $(n_{i,\ i})$.

$$F_{i,\ i} = m_i\, \dot{v}_{C_{i,\ i}}$$

$$N_{i,\ i} = C_i I\, \dot{\omega}_{i,\ i} + \omega_{i,\ i} \times C_i I\, \omega_{i,\ i}$$

$$f_{i,\ i} = R_{i+1,\ i}\, f_{i+1,\ i+1} + F_{i,\ i}$$

$$n_{i,\ i} = N_{i,\ i} + R_{i+1,\ i}\, n_{i+1,\ i+1} + P_{C_{i,\ i}} \times F_{i,\ i} + P_{i+1,\ i} \times R_{i+1,\ i}\, f_{i+1,\ i+1}$$

Now, the torque for each joint can be determined using the following equations, which are based on joint types:

$$\tau_i = n_{i,\ i}^T\, \hat{Z}_{i,\ i} \quad \text{(Joint is revolute)} \quad or \quad f_{i,\ i}^T\, \hat{Z}_{i,\ i} \quad \text{(Joint is prismatic)}$$

The dynamic formulation can be expressed in the following form, incorporating friction and interaction forces [14].

$$\tau = M(q)\ddot{q} + V(q,\dot{q}) + G(q) + F(q,\dot{q}) + J^T(q)\, F_{external}$$

Where, $q \in \mathbb{R}^{n\times 1}$ is the vector of joint coordinates, $q^{\cdot} \in \mathbb{R}^{n\times 1}$ is vector of joint rates, $\ddot{q} \in \mathbb{R}^{n\times 1}$ is the vector of joint accelerations, $M(q) \in \mathbb{R}^{n\times n}$ is inertia matrix, $V(q,\dot{q}) \in \mathbb{R}^{n\times 1}$ is vector of the centrifugal and Coriolis terms, $G(q) \in \mathbb{R}^{n\times 1}$ is the vector of micro or zero gravity, $F(q,\dot{q}) \in \mathbb{R}^{n\times 1}$ is the friction vector considering dry friction and $J^T(q) \in \mathbb{R}^{n\times 6}$ is the transpose of Jacobian of angular and linear velocities.

4.0 Computational Framework and MATLAB app development

Employing the equations outlined in section 3 enables the computation of link velocities, accelerations, acceleration of the mass center, forces and moments acting at the mass center, forces and moments at each link frame, and ultimately the joint torque. The algorithm detailed in Table-1 for the developed computation tool has been used to calculate the dynamics of an n-Degree of Freedom (DoF) robotic manipulator.

Table-1: Algorithm for the developed computation tool to compute dynamics of n-DoF space robot

```
**Algorithm**: Dynamic Analysis of n-DoF space robot

**Input**: DoF $n$, Joint Variables $q, \dot{q}, \ddot{q}$, Robot Parameters (Link Length, Mass, Inertia, Mass Center, micro gravity)
**Output**: Joint Torques $\tau$, $M(q)$, $V(q, \dot{q})$ and $G(q)$

for each joint $i$ ($i = 1\ to\ n$) do  // Kinematics
    Compute Homogenous Transformation matrices
    Find Rotation matrices
    Find Position vector
end for

for each joint $i$ ($i = 0\ to\ n$) do  // Outward iterations:
    Compute linear and angular velocities
    Compute linear and angular accelerations
    Compute Forces and Moments developed at mass center
end for

for each joint $i$ ($i = n\ to\ 1$) do  //Inward iterations:
    Compute Joint Forces and Moments developed
    Joint Torque
end for
Extract M, V and G
```

The algorithm has been implemented in MATLAB R2024a and used to develop an app, as shown in Fig. 4, to compute the dynamics of robotic manipulators. This app enables users to input modified Denavit-Hartenberg (DH) parameters, inertia data, and link length information, and it calculates joint torques while displaying them as output. The app is designed to accommodate various robotic configurations by allowing users to specify the degrees of freedom (DoF) of the manipulator. Using callback functions and iterative loops, the app computes key kinematic and dynamic parameters, including joint positions, velocities, accelerations, forces, and torques. It formulates the necessary dynamic torque equations for each joint using the Recursive Newton-Euler Algorithm, ensuring accurate computation of manipulator dynamics. Additionally, the app automates the generation of MATLAB functions to express the mass matrix (M), Coriolis and centrifugal forces (V), and gravity forces (G), which are essential for modeling the robot's dynamic behavior. The implementation utilizes MATLAB’s Symbolic Math Toolbox to derive closed-form dynamic equations, enhancing its usability for real-time simulations and control applications. To further aid users, the app provides graphical output, allowing for better visualization and analysis of computed joint torques.

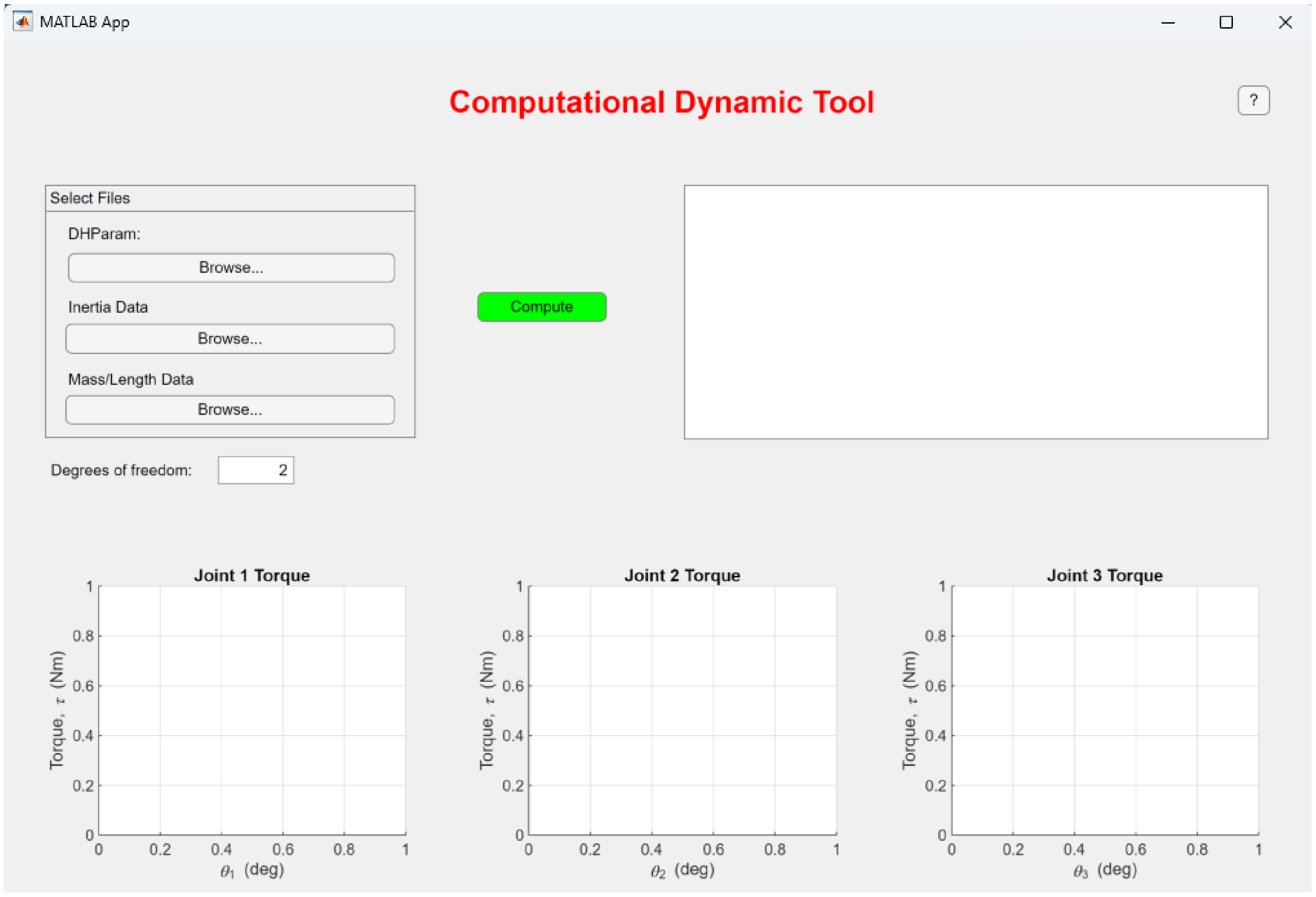

Fig. 4: The Graphical User Interface (GUI) of the developed computational tool

5.0 Validation of the developed Computational Tool

The developed MATLAB app was tested with two degrees-of-freedom planar robotic manipulator (2R planar robot) that has two revolute joints.

5.1 Kinematic Validation

The kinematic validation of the developed app was performed using a 2-DoF planar robot with $L_1 = 0.5$ m, $L_2 = 0.48$ m, and various joint positions. Figure 5 illustrates the kinematic results generated by the developed algorithm for three different joint configurations. For example, when all joints remain at their initial angles ($\theta_1 = 0, \theta_2 = 0$), the end-effector position is given by $[L_1 + L_2, 0, 0]$, which evaluates to $[0.98, 0, 0]$.

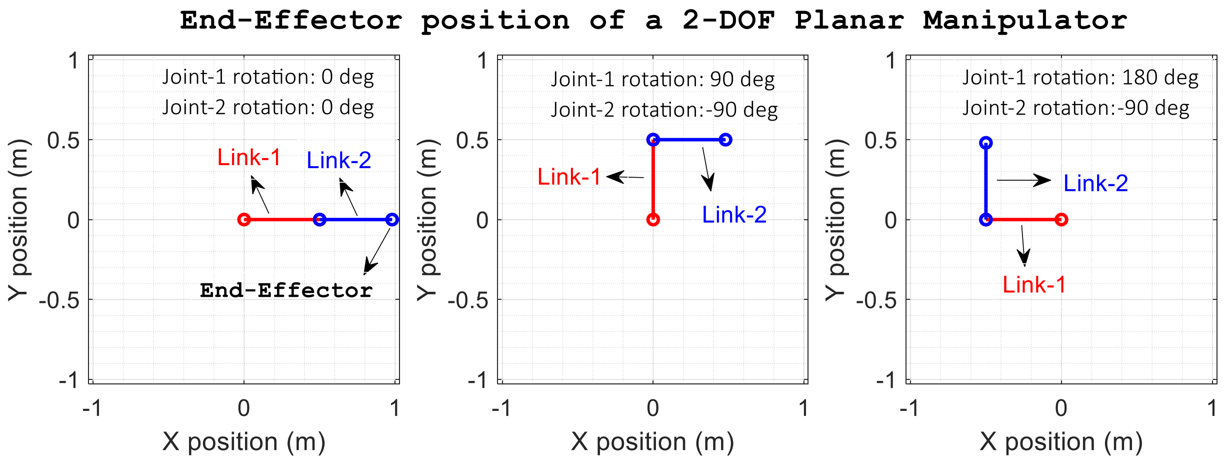

Fig. 5: Algorithm Validation for Computing Kinematics across Various Joint Configurations

5.2 Dynamic Validation

To construct the dynamic model, as mentioned before, the iterative Newton-Euler method was used to formulate both the linear and angular velocities and accelerations for each link, including their centers of mass. Using the link masses and center-of-mass accelerations, the forces acting at each center of mass were calculated. Angular velocities, accelerations, and link inertias were then used to compute the moments about the assigned coordinate axes. Finally, joint torques were calculated using the corresponding rotation matrices, forces, and moments. From these, the inertia matrix, centrifugal and Coriolis terms, and additional components were generated as a user-defined function in MATLAB.

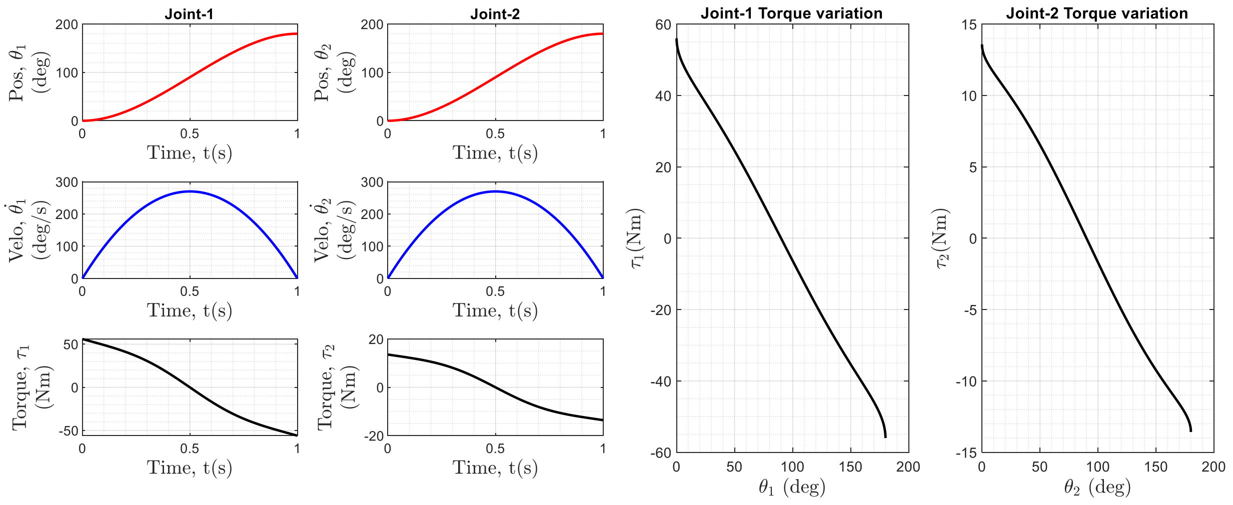

Fig. 6: Algorithm Validation for Computing Dynamics for given trajectories

Figure 6 shows the torque requirement for both joints based on the joint rotation and velocity. Torques are computed for each joint's rotation from 0° to 180°, with the other joint held fixed. The two rightmost plots illustrate the variation in torque with respect to the corresponding joint angles. As observed, both joints shows zero torque at 90°, which is expected. In this configuration, assuming no external forces at the end-effector, the only acting forces are due to the weight of the links, which act downward. Since the line of action of these forces goes through the joints, they do not generate any torque, resulting in a net torque of zero at this specific joint position.

6. Conclusion

This research introduces a computational tool for real-time dynamic analysis of robotic manipulators, enabling on-site customization in space missions. Utilizing modified DH parameters and an iterative Newton-Euler formulation, the tool calculates joint torques, velocities, accelerations, and interaction forces. Developed in MATLAB, it automates the generation of dynamic equations and essential functions for mass (M), Coriolis (V), and gravity (G) matrices. Validation with a two-degree-of-freedom planar robot demonstrated its accuracy and reliability. By eliminating complex programming requirements, the tool simplifies user interaction while enhancing the adaptability of robotic systems in space. Future work will focus on expanding its capabilities for more complex robotic configurations and predictive modeling.

Acknowledgements

This research/material is based upon work supported by NASA under Award No. ESI25_2-1 issued through Wisconsin Space Grant Consortium, and 2) any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration.

References

[1] Schenker, P. S., 2020. "Intelligent robotics for space applications." In Intelligent Robotic Systems, pp. 545-592. CRC Press.
[2] Flores-Abad, A, Ou Ma, Khanh Pham, and Steve Ulrich. "A review of space robotics technologies for on-orbit servicing." Progress in aerospace sciences 68 (2014): 1-26.
[3] Ellery, A., 2019. Tutorial review on space manipulators for space debris mitigation. Robotics, 8(2), p.34.
[4] Yoshida, K., Wilcox, B., Hirzinger, G. and Lampariello, R., 2016. Space robotics. Springer Handbook of Robotics, pp.1423-1462.
[5] Laryssa, P.; Lindsay, E.; Layi, O.; Marius, O.; Nara, K.; Aris, L.; Ed, T. International space station robotics: A comparative study of ERA, JEMRMS and MSS. In Proceedings of the 7th ESAWorkshop on Advanced Space Technologies for Robotics and Automation, Noordwijk, The Netherlands, 19–21 November 2002
[6] Post, M.A., Yan, X.T. and Letier, P., 2021. Modularity for the future in space robotics: A review. Acta Astronautica, 189, pp.530-547.
[7] 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. https://doi.org/10.1002/aisy.202200397
[8] Xu, W., Peng, J., Liang, B. and Mu, Z., 2016. Hybrid modeling and analysis method for dynamic coupling of space robots. IEEE Transactions on Aerospace and Electronic Systems, 52(1), pp.85-98.
[9] Zarafshan, P. and Moosavian, S.A.A., 2013. Dynamics modelling and hybrid suppression control of space robots performing cooperative object manipulation. Communications in Nonlinear Science and Numerical Simulation, 18(10), pp.2807-2824.
[10] Mattila, J., Koivumäki, J., Caldwell, D.G. and Semini, C., 2017. A survey on control of hydraulic robotic manipulators with projection to future trends. IEEE/ASME Transactions on Mechatronics, 22(2), pp.669-680.
[11] Lynch, K.M. and Park, F.C., 2017. Modern robotics. Cambridge University Press.
[12] Craig, J., 2018. Introduction to Robotics: Mechanics and Control, 4th edition, Pearson.
[13] Niku, S.B., 2020. Introduction to robotics: analysis, control, applications. John Wiley & Sons.
[14] Sciavicco, L., Siciliano, B., Villani, L. and Oriolo, G., 2011. Robotics: Modelling, Planning and Control, ser. Advanced Textbooks in Control and Signal Processing.