Robotic Space Simulator: Controls Implementation for Auxiliary Axes and Zero-G Dynamics

Eddie Hilburn, Adam Pettinger, Emily Wilkinson, Ian Lansdowne, Robert Ambrose
Robotics and Automation Design Lab, *Texas A&M University*, College Station, TX 77843

***Abstract*—The Robotic Space Simulator was developed as a physical simulation for in-space manipulation tasks. It incorporates external inputs to its dynamics simulation via force/torque sensors mounted to the 2 6-DoF Stewart platforms which compose its primary structure. Each platform is augmented with an additional degree of freedom in the form of an auxiliary axis - one in translation and one in rotation. Previous work has not effectively included the additional workspace provided by these auxiliary axes. Additionally, it limited the use of external force/torque inputs to the case of platform translation only because the external forces/torques due to platform motion and gravitational force were not removed from the sensor inputs prior to inclusion in the dynamic simulation. In this work, we address each of these limitations. We develop and test two methods of auxiliary axis control: Cartesian Workspace and Joint Cost-Function, and find that both methods are an improvement over the existing system. Additionally we develop and test a method for calculating the mass properties of hardware mounted to the force/torque sensors and a dynamics compensation method for this hardware. Using this technique we are able to effectively compensate for gravitational force in different platform orientations, and achieve zero-g behavior of the system.**

I. Introduction

Simulations are a critical component of developing in-space robotics, especially for remote proximity operations where two spacecraft will be in contact. Recent work at NASA [1] has developed a software simulation for this case, but the physical simulation options are limited. Traditional options include air-bearing floors [2], underwater robotics [3], or robotic manipulators [4]–[9]. Air-bearing floors are generally limited to planar motion, and have limited runtime. Underwater platforms require waterproofing of the test articles and include viscous friction not present in space. Using parallel manipulators as in [5]–[7] is a popular choice for their relatively high payload and good manipulability, but their workspace size may be small. Serial manipulators are also a popular choice [4], [8], [9], and have a much larger workspace but lower payloads.

The Robotic Space Simulator (RSS) is a newly-developed 14-DoF physical simulation with additional manipulators for zero-g interactions between two spacecraft. The system consists of two parallel 6-DoF Stewart platforms modified to include an additional 7th DoF (Fig. 1). The first platform (hereafter, the "slide platform") rides on a $6\,\mathrm{m}$ track, providing a significant increase in its workspace along one translational axis (x-axis). This additional DoF is intended to support simulating the final approach to an objective spacecraft prior to contact. The second platform (hereafter, the "rotary platform") adds two rotations in the yaw- axis, allowing increased range for manipulation tasks during contact. In this paper we extend the RSS capabilities by addressing two simplifications employed during initial validation of the system (Fig. 2).

The first improvement we make is better control of each platform's auxiliary (aux) axis. The Stewart platform workspaces are smaller than the aux workspaces, so it is critical to use the aux axes efficiently. This paper describes two methods to improve the effective control for each aux axis, one in terms of workspace allocation (Section II-A) and the other in terms of a joint-based cost function (Section II-B).

The second improvement is dynamics compensation for simulation hardware (Section III). The RSS utilizes 6-axis force/torque sensors (FTS) as inputs to the spacecraft dynamics simulation. These sensors are mounted beneath a structure with appreciable mass and read changes in the gravitational load of the structure as the platform moves. As a result, we restricted the platforms to translation only. In this paper we describe the methods utilized to implement zero-g dynamics with full SE(3) control. These include calculating the inertial properties of the sensor mounted hardware and removing the gravitational force from the spacecraft simulation dynamics.

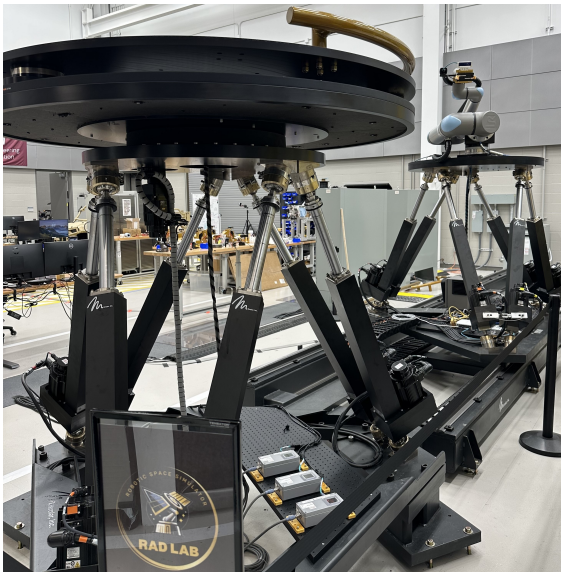

Fig. 1: The RSS rotary platform (front) and slide platform (back)

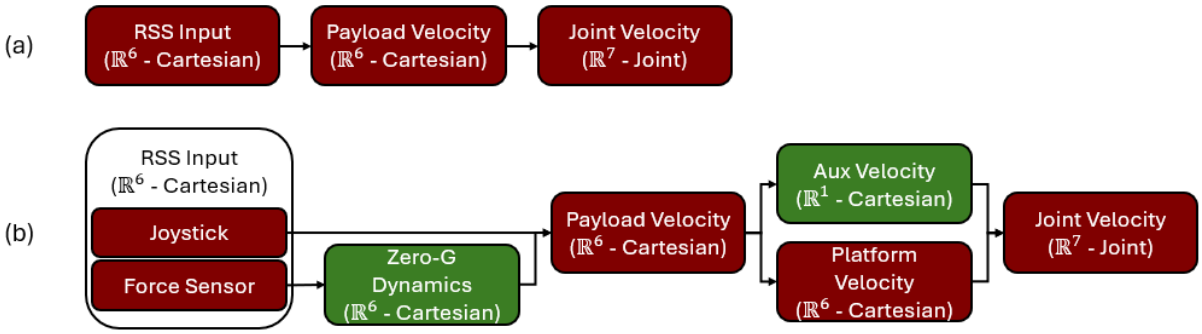

Fig. 2: RSS Dynamics Simulation: (a) Initial System Validation and (b) Component improvements included in this work

These developments continue to improve the fidelity of the RSS system and its capability to provide a high fidelity terrestrial space robotic manipulation testbed.

II. Auxiliary Axis Control

The RSS dynamics simulation processes force and torque inputs in Cartesian space, and outputs encoder counts to the proprietary low-level control computers for each platform. Inputs are provided by an operator simulating spacecraft attitude system controls and force/torque measurements from sensors mounted on the platform surface.

In the current system configuration, control of each platform is implemented independently. Each platform implements a Jacobian-based feed-forward velocity controller using dual quaternion kinematics [10]. The inputs are combined and applied to the simulated spacecraft's center of mass, then converted to a velocity using a mass-spring-damper model,

$$\mathcal{F} = M\ddot{x} - B\dot{x} + Kx \tag{1}$$

where $F \in \mathbb{R}^6$ combines the externally applied wrench and input attitude controls, M the satellite's desired inertial properties, and $\ddot{x} \in \mathbb{R}^6$ the satellite's acceleration. Note that for in-space applications, we are particularly interested in the case with no damping or spring $B = K = 0$.

Integrating (1) gives the spacecraft's velocity $\dot{x}$. We use $\dot{x}$ and the Jacobian of the platform to calculate joint commands

$$\dot{q} = J^{\dagger}\dot{x} \tag{2}$$

where $\dot{q} \in \mathbb{R}^7$ is the velocity of the platform axes including the extra axis, and $J^{\dagger}$ is the Moore-Penrose inverse [11] of the combined platform and aux axis Jacobian. The Jacobian for strictly the platform is calculated as in [10], and J in (2) is modified due to the additional DoF. In prior work, motion along the respective aux axis was split equally between itself and the platform. The limits of the platform actuators were reached prior to the limits of the aux axes, limiting the available workspace to the system. In the following section, we describe the development of a control system to take advantage of the additional workspace provided by the aux axes.

The desired system behavior is to prefer motion in the aux axes. To achieve this, we have implemented and tested two methods: a Cartesian workspace based method and a joint cost-function based method. In both cases, we modify the above description of the RSS control system as follows:

1) Compute aux axis velocity based on methods described in either of the following subsections.
2) Remove the computed aux axis velocity from the relevant Cartesian platform velocity axis.
3) Change Eq (2) to calculate platform leg velocities only, where J becomes the Jacobian of the platform only

A. Cartesian Workspace Method

The workspace for a Stewart platform is dictated by its geometry with respect to the relative positions of the leg attachment points and the leg length limits [12]. For the purposes of controlling the aux axes, we are interested in the workspace in the x-axis of the slide platform and the yaw-axis of the rotary platform. For the remainder of this paper, unless otherwise specified, we restrict our description to the slide platform only. Without a loss of generality, the methodology holds for the rotary platform as well.

We implement a position controller with the desired position as a function of the available workspace in the aux axis and its platform corollary. We define the following terms:

ws_{plat}: percentage of workspace remaining in the relevant platform axis (i.e., x- or yaw-axis)

ws_{aux}: percentage of aux axis workspace remaining

and the error to be controlled

$$e = ws_{plat} - ws_{aux}. \tag{3}$$

We then calculate a control wrench for the aux axis

$$\mathcal{F}_{aux} = K_d\dot{e} + K_p e \tag{4}$$

where K_p is a tunable proportional gain set to 1000 for the testing reported in this work, $K_d = 2\sqrt{K_p M}$ is a critically damped derivative gain based on the mass of the simulated satellite (M), and $\dot{e}$ is the time derivative of (3) obtained by finite difference method. Integrating (4) gives the aux axis velocity. This velocity is subtracted from the platform velocity such that **the overall payload velocity is conserved**, effectively using the redundancy of the system.

The primary limit to this control method is the ability to compute the workspace of the platform in near real time. The challenges and a promising solution are described in [13], but the solution was not pursued within the scope of this paper.

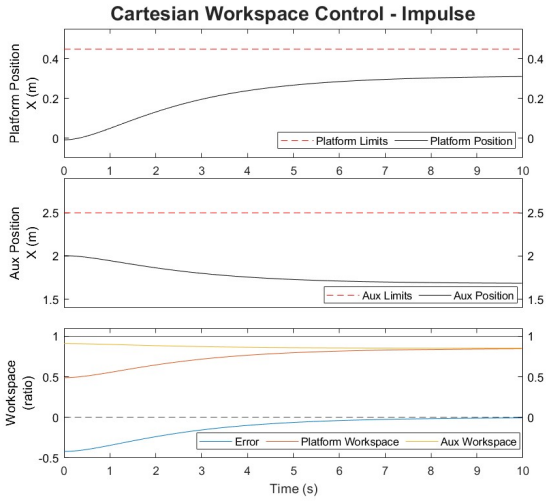

Fig. 3: Impulse response using workspace control

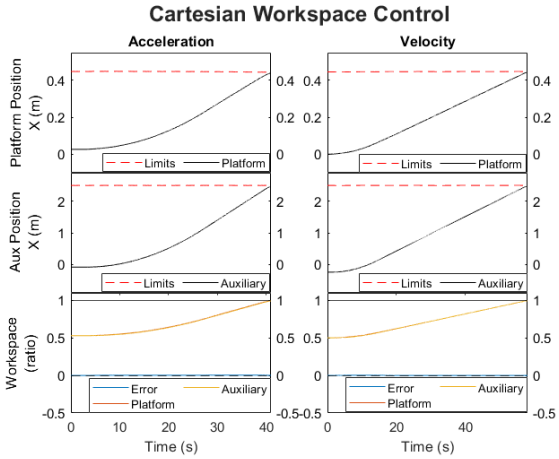

Fig. 4: Constant velocity (right) and acceleration (left) responses using workspace control

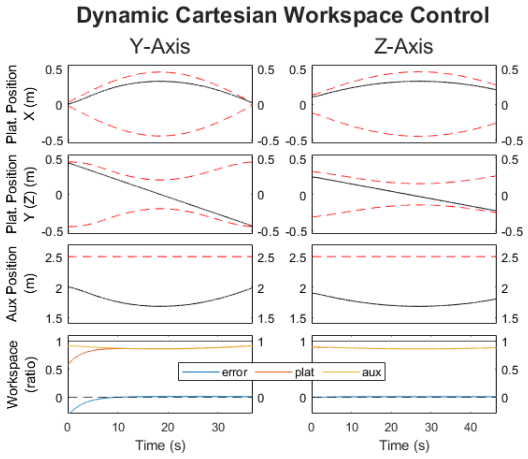

Fig. 5: Controller performance during dynamic sizing of platform workspace

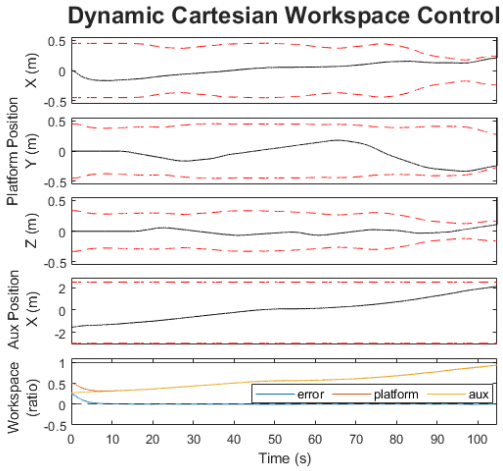

Fig. 6: Workspace aux controller performance during simulated spacecraft approach

As a proof of concept, the controller was tested on the slide platform with a simplified spherical workspace calculation ignoring platform orientation. To calculate this workspace, the x-axis workspace boundary is defined as

$$x_{bound} = plat_{x,max}(cos\frac{\pi}{2}\frac{plat_y}{plat_{y,max}})(cos\frac{\pi}{2}\frac{plat_z}{plat_{z,max}}) \tag{5}$$

where $plat_{max} = \pm[0.45\,\mathrm{m}, 0.45\,\mathrm{m}, 0.30\,\mathrm{m}]$ defines a reduced volume from the true platform workspace. The controller in (4) now operates with a dynamically sized platform workspace.

Initial testing of the controller was accomplished via impulse by moving the aux axis to a location away from its neutral position ($+2.0\,\mathrm{m}$) while keeping the platform fixed in its neutral position. Once established, the controller was enabled, and the aux axis and platform established an equal relative position within their respective workspaces (Fig. 3).

The primary benefit of this controller is to maximize the total workspace so the most relevant testing region is near the workspace limits. The desired behavior is the platform and the aux axes approaching their limits in tandem. This aspect of the controller was tested first by inputting a constant x-axis velocity and allowing the system to slowly approach its limit. The system maintained near-zero controller error for the entire operation, and was able to achieve maximum workspace utilization in both the aux and the artificially reduced platform axes (Fig. 4). Next, we repeated this test with constant x-axis acceleration, again driving the system to its axis limit, and again maintaining near-zero error.

The above test results are given for neutral platform y and z positions, keeping the x-axis workspace constant. To evaluate the performance of the dynamic workspace sizing, we established a similar position to the impulse testing performed previously, except in this case we alternatingly initiated testing by fully extending the platform in the y or z directions. Once the controller was enabled, we moved the platform from upper to lower limits of y and z axes, observing the change of available workspace in the x axis. The changing workspace limits of the platform's x axis required the controller to reposition the aux axis in order to maintain a common ratio between it and the platform x axis available workspace. The controller performed as expected

over the entire sequence, maximizing available workspace by effective positioning of the aux axis (Fig. 5). Finally, we performed an end-to-end test, with an operator varying the motion in a non-scripted manner through full range of motion in a simulated spacecraft approach scenario. Fig. 6 shows the dynamic sizing of the platform as well as the controller performance throughout this test.

B. Joint-Based Cost Function Method

In order to minimize the complexity of computing the workspace of a Stewart platform as a part of the RSS control loop, we have implemented an alternative control method for the RSS aux axis based on a cost function of each joint position. We define a parabolic cost function

$$\mathcal{C} = \sum_{i=1}^{7} \alpha_i (q_i - q_{0,i})^2 \tag{6}$$

with q a 7-element vector of actuator positions (including the aux axis), q_0 a 7-element vector of actuator neutral positions, and α a vector of weights for each joint. The weight value for each leg of the platform is set to 1, and the weight for the aux axis is a tunable parameter, set to $\alpha_{aux} = 0.30$ as this value was found to give good results during testing. We seek to minimize this cost function via its derivative. In the case of the slide platform

$$\frac{\partial \mathcal{C}}{\partial x} = \sum_{1}^{7} 2\alpha_i (q_i - q_{0,i}) \frac{\partial q_i}{\partial x} = 0 \tag{7}$$

The partial derivative with respect to x of each joint is a component of the platform Jacobian, and the partial derivative of the aux axis with respect to x is -1 because it requires the platform to move in the opposite direction to maintain an EE position. We then define the error function e to be (7) and calculate a control wrench for the aux axis as

$$\mathcal{F}_{aux} = K_d \frac{\partial \dot{\mathcal{C}}}{\partial x} + K_p \frac{\partial \mathcal{C}}{\partial x} = K_d \dot{e} + K_p e \tag{8}$$

with $K_p = 200$ and $K_d = 2\sqrt{K_p M}$ as in Section II-A.

We tested this controller similarly to the workspace controller, with results shown in Figures 7-10. In cases where motion is restricted to the x-axis, and all other axes in their neutral position, the controller behaved as expected, maximizing the aux axis workspace. Moving the platform along the y- and z-axes, yielded less straightforward results (Fig. 9). In the case of motion in y, the changing leg lengths which compose the cost function offset each other resulting in near-zero aux axis motion. Had the aux axis moved forward, it would have provided additional reach for the platform in the y-axis by minimizing the platform extension in the x direction. In the case of motion in z, the aux axis did not center itself beneath the platform at full extension and moved further from center at full retraction. In both cases this behavior artificially limited the z-axis workspace.

C. Auxiliary Axis Control Results

Both methods of aux axis control are improvements over previous work. However, both are limited in different ways. The Cartesian Workspace method provides a high degree of control and effectively utilizes the available workspace of the combined system. To fully realize the benefits of this method we need to address the problem of calculating the true workspace boundary during runtime. The Cost Function method is simple to compute and is agnostic to translations or rotations of the platform. The cost function chosen for this analysis needs to be improved to avoid limitations on axes other than the aux-aligned platform axis. This could mean increasing the cost more aggressively as each actuator approaches its limit (e.g., exponentially) or defining a different cost function for the aux and platform actuators.

III. Dynamics Compensation

The RSS is intended to be a physical simulation of two contacting orbital bodies. Accurately measuring the contact forces between the two platforms is critical for safety and simulation fidelity. The weight and dynamics of any hardware mounted above the FTS must be compensated for to measure contact forces.

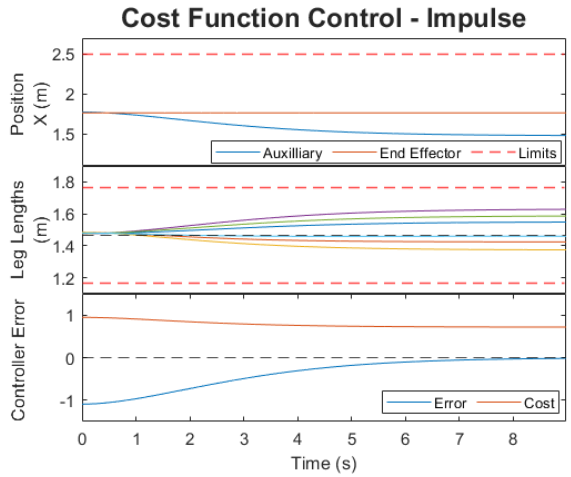

Fig. 7: Impulse response using cost function control

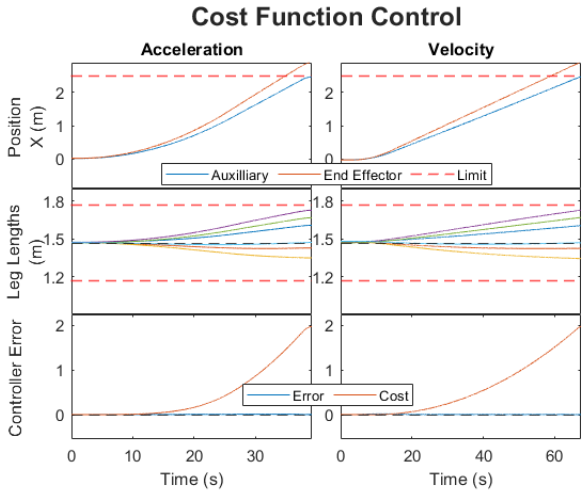

Fig. 8: Constant velocity response using cost function control

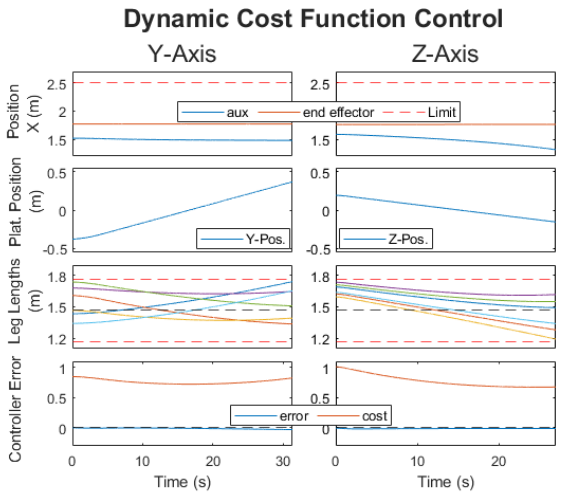

Fig. 9: Cost function controller response to Y- and Z- axis translation

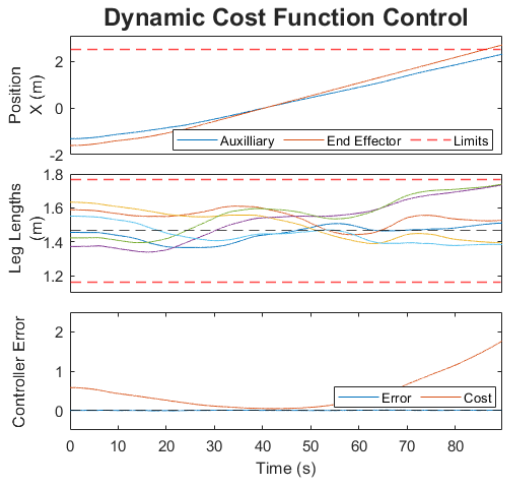

Fig. 10: Cost function controller performance during simulated spacecraft approach

A. Payload Dynamics and Compensation

The FTS outputs the 3 reaction forces and 3 reaction torques measured at its output surface. The signal comprises of components from

- the weight and inertia of the object mounted above it,
- internal dynamics of the object if it is not static, and
- external loads applied to the object.

For the purposes of this work, we are focused on accurately measuring the externally applied loads, and only consider static mounted objects with no internal dynamics. Following the notation in [14], the external load is

$$\mathcal{F}_{ext} = \begin{bmatrix}\tau_{ext} & f_{ext}\end{bmatrix}^T = \mathcal{F}_S - \mathcal{F}_{dyn} \tag{9}$$

with $\mathcal{F} \in \mathbb{R}^6$ comprising of torques τ and forces f. The sensor measurement is $\mathcal{F}_S$, and $\mathcal{F}_{dyn}$ is the wrench caused by the dynamics (including gravity) of the mounted object, expressed in the *sensor* frame.

We can calculate $\mathcal{F}_{dyn}$ at each time given the object's mass properties and the platform position, velocity, and acceleration. The mass properties include the mass, inertial matrix, and center of mass location. Note that these are very likely to be different from the *simulated* mass properties in (1). We define the following frames:

- $\{0\}$ the spatial frame where gravity is fixed
- $\{P\}$ the body frame attached to the moving platform's top surface
- $\{D\}$ the body frame attached to the mounted object's center of mass. This is fixed with respect to $\{P\}$
- $\{S\}$ the frame of the FTS, fixed with respect to $\{P\}$

We calculate the dynamics of these frames with

$$\mathcal{V}_0 = \begin{bmatrix}\omega \\ v\end{bmatrix} = 0, \quad \dot{\mathcal{V}}_0 = \begin{bmatrix}0 \\ -g\end{bmatrix} \tag{10}$$

where $\mathcal{V}_0 \in \mathbb{R}^6$ is the twist of frame $\{0\}$ with angular velocity ω and linear velocity v, and $\dot{\mathcal{V}}_0$ is the acceleration of $\{0\}$ using gravity vector g. Assuming we know the position of $\{P\}$ with respect to $\{0\}$, T_{0P}, and the $\{P\}$ frame's velocity $\mathcal{V}_P$ and acceleration $\dot{\mathcal{V}}$ given in $\{P\}$, we calculate

$$\dot{\mathcal{V}}_P = \dot{\mathcal{V}} + [Ad_{P0}]\dot{\mathcal{V}}_0 + [ad_{\mathcal{V}P}]\mathcal{V}_P \tag{11}$$

where $[Ad]$ and $[ad]$ are the adjoints of homogeneous transforms and twists respectively, as given in [14]. The velocity and acceleration of the mounted object is then

$$\mathcal{V}_D = [Ad_{DP}]\mathcal{V}_P, \quad \dot{\mathcal{V}}_D = [Ad_{DP}]\dot{\mathcal{V}}_P \tag{12}$$

The dynamics for frame $\{D\}$ are given as

$$\mathcal{F}_D = \mathcal{G}_D\dot{\mathcal{V}}_D - [ad_{\mathcal{V}D}]\mathcal{G}_D\mathcal{V}_D \tag{13a}$$

$$\mathcal{G} = \begin{bmatrix}\mathcal{I} & 0 \\ 0 & mI\end{bmatrix} \in \mathbb{R}^{6\times6} \tag{13b}$$

as in [14] with inertia $\mathcal{I} \in \mathbb{R}^{3\times3}$ and mass m. The wrench at the sensor is found with a change of reference

$$\mathcal{F}_{dyn} = -[Ad_{DS}]^T\mathcal{F}_D \tag{14}$$

where the -1 multiplier is included because the sensor measures the *reaction* force. Eqs (9) and (14) allow us to calculate the externally applied wrench that is used as the input for the dynamic simulation in (1).

B. Payload Mass Properties Identification

The above section details how we remove the mounted hardware's weight and dynamics from the FTS signal. It assumes known mass properties of the equipment. We can use the FTS itself to identify those mass properties. We break this into three steps:

1) Calculate the mass of the mounted hardware
2) Calculate the center of mass of the mounted hardware
3) Calculate the inertia of the mounted hardware

To calculate the mass, we begin with the relationship

$$f_S = f_{bias} + mR^Tg \tag{15}$$

where f_S is the sensed force, f_{bias} is some unknown bias of the sensor with respect to force, m is the mass of the hardware, R is the rotation matrix of the sensor frame $\{S\}$, and g is gravity as in (10). Given data collected for i samples,

the values of m and f_{bias} can be estimated with a least-squares regression of the form $\mathbf{Ax} = \mathbf{b}$ using the parameters

$$\mathbf{A} = \begin{bmatrix} I & R_0^T g \\ \vdots & \vdots \\ I & R_i^T g \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} f_{S,0} \\ \vdots \\ f_{S,i} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} f_{bias} \\ m \end{bmatrix} \tag{16}$$

To calculate the center of mass, we use

$$\tau_S = \tau_{bias} + r \times f_{weight} = \tau_{bias} - [mR^T g]r \tag{17}$$

where τ_S is the sensed torque, τ_{bias} is some unknown bias of the sensor with respect to torque, r is the vector locating the center of mass with respect to the FTS, and $f_{weight} = mR^T g$ is the gravitational force calculated from the regressed mass in (16). The bracket notation [14] denotes the $3x3$ skew-symmetric representation of the cross-product. We again perform a least-squares regression to solve for

$$\mathbf{x} = \begin{bmatrix} \tau_{bias} & r \end{bmatrix}^T \tag{18}$$

by which we obtain the location of the center of mass and the sensor bias with respect to torque.

In order to obtain the inertia components of the object, we assume it is a rigid body and begin with the dynamics given by (13a). This is regressed similarly to (16) and (18) to solve for the spatial inertia components

$$\mathbf{x} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} & I_{yy} & I_{yz} & I_{zz} \end{bmatrix}^T \tag{19}$$

C. Dynamics Compensation Results

To test both the dynamics compensation and the mounted hardware mass identification, we implemented a script to move the Stewart platform through a repeatable set of poses and orientations. The orientations include roll, pitch, yaw angles of $[-30°, -15°, 0°, 15°, 30°]$ and permutations of each. The position of the table T_{0P} and the sensor reading $\mathcal{F}_S$ are recorded at $250\,\mathrm{Hz}$ throughout the test.

To identify the object mass properties, the table position is fit to a cubic spline, which yields the table velocity and acceleration. These are used with the sensor data in the regressions in (16), (18), and (19) sequentially to respectively find the object mass, center of mass, and inertia matrix.

We tested the regression with a statically-mounted UR20 manipulator (Fig. 1). The arm and its mounting hardware are known to be $79.5\,\mathrm{kg}$, and the regression calculated a mass of $83.2\,\mathrm{kg}$. The RMSE for the mass, center of mass, and inertia regressions are $3.50\,\mathrm{kg}$, $0.550\,\mathrm{m}$, and $79.1\,\mathrm{kg\,m^2}$ respectively.

During the data collection for the regression, the dynamics compensation described in Section III was running with a payload mass of $0\,\mathrm{kg}$, which effectively disables the dynamic compensation. In this case, the external wrench from (9) is inaccurately calculated as just the measured wrench from the sensor. Fig. 11 (top) shows the calculated external wrench for this case, motivating dynamic compensation. The FTS is biased at the start of the test and starts near 0. However, as the platform moves to new orientations, there is a constant non-zero offset in both force and torque. If this was used as an input to the platform control, it would result in large undesirable motions and ruin the simulation fidelity.

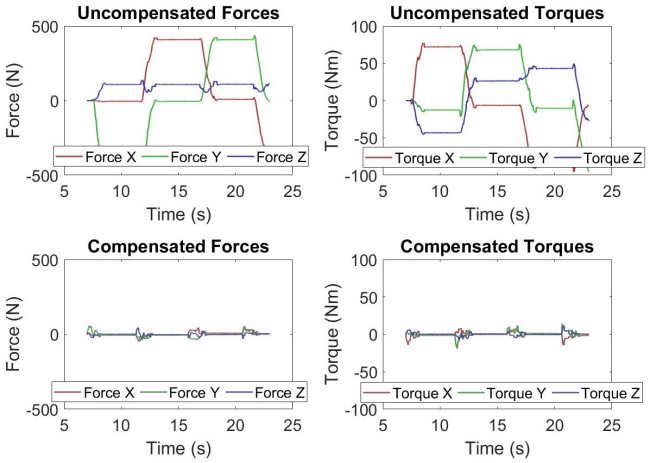

Fig. 11: Comparison of the uncompensated (top) and compensated (bottom) external wrench during the same platform motion

Using the regressed mass properties with the dynamic compensation results in an external wrench as shown in Fig. 11 (bottom) for the same platform motion profile. The nominal output is 0 wrench through the test as no external load was applied. In Fig. 11, the steady state error is reduced when using dynamic compensation, with error remaining only during platform acceleration. Note the external wrench shown in the bottom of Fig. 11 is calculated using 0 platform velocity and acceleration in (11). These are noisy to calculate at run-time and introduce spikes in the calculated $\mathcal{F}_{dyn}$ during platform acceleration. Faster and smoother estimations of the velocity and acceleration should improve the dynamic compensation, but calculating those is left to future work. The RMSE for the external wrench calculation was $282\,\mathrm{N}$ and $59\,\mathrm{N\,m}$ before the dynamics compensation was applied. Using the compensation reduced the error to $16\,\mathrm{N}$ and $4\,\mathrm{N\,m}$, which is more than a 90% reduction for both force and torque.

IV. Conclusion

In this work, we introduced two major improvements to the Robotic Space Simulator, dramatically increasing its usable workspace. We introduced two methods for aux axis control, and demonstrated both work to balance workspace constraints between the Stewart platforms and their aux axes. Key results include using the full range of aux motion and accurately tracking the target EE pose. We also compensated for the dynamics and gravitational load of test hardware mounted on the simulator, allowing for full SE(3) control during simulation runs. We showed the ability to accurately measure test articles using the FTS, then compensate for their dynamics, reducing the error in the measured forces by 90+%. Together, these improvements increase the capability of the RSS to include 3D motion simulations within an approximate volume of $8\,\mathrm{m} \times 1\,\mathrm{m} \times 0.6\,\mathrm{m}$ including roll and pitch angles up to $\pm 30°$.

REFERENCES

[1] A. Garcia, "Basic orbital and robotic simulation (boars)," *Dynamics Operations Training Systems*, 2023. [Online]. Available: https://www.nasa.gov/general/generic-robotics-on-orbit-trainer/

[2] T. Rybus and K. Seweryn, "Planar air-bearing microgravity simulators: Review of applications, existing solutions and design parameters," *Acta Astronautica*, vol. 120, pp. 239–259, 2016.

[3] W. Whitacre, "An autonomous underwater vehicle as a spacecraft attitude control simulator," in *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005, p. 137.

[4] F. Aghili, "Pre-and post-grasping robot motion planning to capture and stabilize a tumbling/drifting free-floater with uncertain dynamics," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 5461–5468.

[5] J. D. Mitchell, S. P. Cryan, K. Baker, T. Martin, R. Goode, K. W. Key, T. Manning, and C.-H. Chien, "Integrated docking simulation and testing with the johnson space center six-degree-of-freedom dynamic test system," in *AIP Conference Proceedings*, vol. 969, no. 1. American Institute of Physics, 2008, pp. 709–716.

[6] S. Dubowsky, W. Durfee, A. Kuklinski, U. Müller, I. Paul, and J. Pennington, "The design and implementation of a laboratory test bed for space robotics: The ves mod ii," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 12860. American Society of Mechanical Engineers, 1994, pp. 99–108.

[7] L. Hernández, E. Izaguirre, E. Rubio, O. Urquijo, and J. Guerra, "Kinematic task space control scheme for 3 dof pneumatic parallel robot," *Croacia: InTech*, pp. 67–84, 2011.

[8] C. Carignan, N. Scott, and S. Roderick, "Hardware-in-the-loop simulation of satellite capture on a ground-based robotic testbed," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, vol. 6, 2014.

[9] F. Aghili and M. Namvar, "Scaling inertia properties of a manipulator payload for 0-g emulation of spacecraft," *The International Journal of Robotics Research*, vol. 28, no. 7, pp. 883–894, 2009.

[10] S. Montgomery-Smith and C. Shy, "Using lie derivatives with dual quaternions for parallel robots," *Machines*, vol. 11, no. 12, p. 1056, 2023.

[11] J. C. A. Barata and M. S. Hussein, "The moore–penrose pseudoinverse: A tutorial review of the theory," *Brazilian Journal of Physics*, vol. 42, pp. 146–165, 2012.

[12] C. Gosselin, "Determination of the workspace of 6-DOF parallel manipulators," vol. 112, no. 3, pp. 331–336. [Online]. Available: https://doi.org/10.1115/1.2912612

[13] O. Bohigas, M. Manubens, and L. Ros, "A linear relaxation method for computing workspace slices of the stewart platform," vol. 5, no. 11005. [Online]. Available: https://doi.org/10.1115/1.4007706

[14] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge, UK: Cambridge University Press, 2017.