Trajectory Optimization and Following for a Three Degrees of Freedom Overactuated Floating Platform

A. Bredenbeck[1, 2, †], S. Vyas[3, 2, ‡], M. Zwick[2], D. Borrmann[1], M.A. Olivares-Mendez[4], A. Nüchter[1] .

***Abstract*— Applications of space robotics, such as Active Space Debris Removal (ASDR), require representative testing before launch. A commonly used approach to emulate the microgravity environment in space are air-bearing based platforms on flat-floors, such as the European Space Agency's Orbital Robotics and GNC Lab (ORGL). This work proposes a control architecture for a floating platform at the ORGL, equipped with eight solenoid-valve-based thrusters and one reaction wheel. The control architecture consists of two main components: a trajectory planner that finds optimal trajectories connecting two states and a trajectory follower that follows any physically feasible trajectory. The controller is first evaluated within an introduced simulation, achieving a 100% success rate at finding and following trajectories to the origin within a Monte-Carlo test. Individual trajectories are also successfully followed by the physical system. In this work, we showcase the ability of the controller to reject disturbances and follow a straight-line trajectory within tens of centimeters.**

I. INTRODUCTION

Space debris is widely recognized as a significant challenge to all future activities in space [1]–[3]. Already in 1978, Kessler et al. [4] noted that the debris forming from satellite collisions could lead to an artificial asteroid belt in popular orbits such as the Low Earth Orbit (LEO) and the Geostationary Orbit (GEO). This belt could severely limit human space activities. In the extreme case, it could entirely prohibit the possibility for safe spaceflight. Becoming known as the "Kessler-Syndrome", the described effect gained traction in recent years for multiple reasons. Firstly, today a significant portion of debris is caused by human activity. Some of the biggest contributions are caused by anti-satellite tests [5], [6] and satellite collisions [7]. Further, planned large constellations, such as SpaceX Starlink, are suspected to influence the space debris situation in the future strongly, possibly endangering space sustainability as indicated by [8]. Despite there being stringent requirements on de-orbiting satellites, many members of the community (cf. [9]–[11]) argue that these are not sufficient to avoid the Kessler-Syndrome mentioned above and campaign for ASDR. Further, in-orbit servicing will play an essential role in reducing the number of malfunctioning satellites by prolonging the lifetime of satellites.

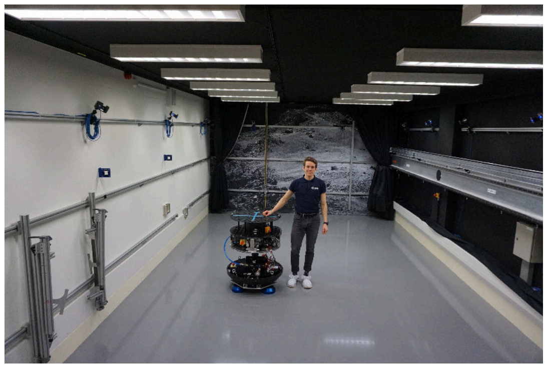
Fig. 1. The first author at the ORGL at ESTEC. An epoxy flat-floor for floating air-bearing platforms, where slight unevenness induces disturbances.

Given the proximity of the servicing/capturing device and the respective client, the missions carry above-average risk. Hence, the flawless operation must be guaranteed as good as possible prior to launch. For this purpose, ground-test facilities that provide system-level evaluation are necessary. The servicing/capturing system will eventually operate in a zero-g environment, while all ground facilities are subjected to Earth's gravity. Hence, precisely replicating the entire operating domain proves to be very difficult.

Air-bearing platforms have turned out to be the most popular type of facility for testing in simulated micro-gravity in academia and industry, with many examples in use [12]–[17].

Figure 1 depicts one of these facilities: the European Space Agency's (ESA) ORGL at ESTEC. One of the testing platforms at the ORGL provides a realistic actuator assembly, with several cold gas thrusters and a reaction wheel. This system functions either as a base platform for testing new technologies or as a dummy target for capturing tests. Thus it is of interest to control the system along desired trajectories as a target or to enable the movement of the tested technology. For all floating platforms, the most limiting factor for the test duration is the amount of cold gas stored onboard, which provides the necessary pressure to keep the platform floating. The compressed breathing air also functions as propellant for the thrusters that controls the position and orientation (pose) of the system. Therefore, a controller that aims at prolonging the test duration must use the thrusters in a propellant-optimal manner along any trajectory. This

[1] Informatics VII, University of Würzburg, Germany
[2] Automation and Robotics Group, ESA, Noordwijk, Netherlands
[3] Robotics Innovation Center (RIC), DFKI Bremen, Germany
[4] SpaceR-SnT, University of Luxembourg, Luxembourg
[†] To abide by the FAIR principles of science, all software created for this work is available as open source at `gitlab.com/anton.bredenbeck/ff-trajectories`
[‡] Author 2 acknowledges the support of Stardust Reloaded project which has received funding from the European Unions Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 813644.

TABLE I

MASS, MOI AND SIZE PROPERTIES OF THE SUBSYSTEMS AND THE OVERALL SUM

Subsystem	Mass	MoI	Height	Radius
ACROBAT	154 kg	10.090 kgm^2	62.5 cm	35 cm
SATSIM	50 kg	1.416 kgm^2	20 cm	35 cm
RECAP	13.66 kg	0.67 kgm^2	20 cm	35 cm
RW	4.01 kg	0.047 kgm^2	–	–
Σ	221.67 kg	12.223 kgm^2	102.5 cm	–

objective is similarly vital for the application on satellites, as depletion of fuel limits the lifetime of a satellite.

This work proposes a controller that first finds optimal trajectories between two points in state-space,then uses optimal control methods combined with a modulation scheme for the thrusters to follow the desired trajectory and validates the proposed method with simulation and experiments.

The rest of the document is structured as follows: After introducing the system model in section II, section III gives an overview of the control architecture. An evaluation in simulation and on the physical system (section IV) is carried out before summarizing and discussing the results in section V.

II. SYSTEM MODEL

The used platform is the combination of three previously existing, modular platforms: Air Cushion Robotic Platform (ACROBAT), Satellite Simulator (SATSIM), and Reaction Control Autonomy Platform (RECAP) [18]. The platforms are stacked to form the overall system, and each provides a different functionality:

- ACROBAT provides the base of the platform. Using three New Way 200 mm air-bearings which are supplied with compressed air at 4.8 bar it enables the micro-gravity behavior on the flat ground.
- SATSIM provides air tanks that supply the ACROBAT air bearing as well as the thrusters that SATSIM uses to provide thrust to the stack. SATSIM combines eight thrusters that are arranged pairwise on each side of the platform to induce forces parallel to the coordinate axes of the local robot coordinate system as indicated in Figure 2
- RECAP provides the Reaction-Wheel (RW) used for yaw control.

Their mass, Moment of Inertia (MoI), and size properties are shown in Table I.

This section characterizes the overall system. In particular, this section identifies the thrust by an individual thruster and the motion model.

A. Thruster Characterization

The thrusters built into this system are a combination of a tank holding pressurized gas, an intermediate pressure regulator to achieve a lower operating pressure, a regulating solenoid valve, and a Laval-nozzle through which the gas escapes. Solenoid valves only allow for the states on and off. Thus, their nominal force when opened is of interest. At a regulated operating pressure of 5.0±0.2 bar the force is measured to be 10.0±0.7 N. The thrusters can hold this peak force for pulses of 100 ms but experience a significant drop for longer pulses in the order of seconds. In the following, this work assumes that the force is constant for the opening duration, and no pulses long enough to induce significant deviation from this model are considered.

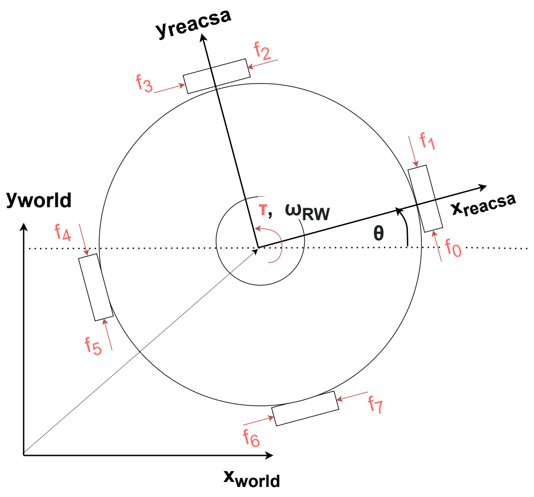

Fig. 2. Coordinate systems and wrenches imposed by the actuators.

B. Dynamic Model

Using the thrust identified above and a torque-based model for the RW the following derives the overall motion model. First, we define all actuators as in Figure 2. The position (x, y) of the system is defined relative to some world coordinate system, and the orientation θ is defined relative to that x axis.
Using the state and control vectors:

$$\mathbf{x} = \begin{bmatrix} x & y & \theta & \dot{x} & \dot{y} & \dot{\theta} & \omega_{RW} \end{bmatrix}^T \tag{1}$$

$$\mathbf{u} = \begin{bmatrix} \tau & f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 \end{bmatrix}^T \tag{2}$$

with the forces of the thrusters f_0 to f_7, the velocity of the reaction wheel ω_{RW} and the torque τ on the RW, the resulting continuous state equation is:

$$\begin{aligned}\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = &\begin{bmatrix} \mathbf{0}^{3\times3} & \mathbf{I}^{3\times3} & 0 \\ & \mathbf{0}^{4\times7} & \end{bmatrix} \mathbf{x} \\ + &\begin{bmatrix} & & & & \mathbf{0}^{3\times7} & & & & \\ 0 & \frac{-s_\theta}{m} & \frac{s_\theta}{m} & \frac{-c_\theta}{m} & \frac{c_\theta}{m} & \frac{s_\theta}{m} & \frac{-s_\theta}{m} & \frac{c_\theta}{m} & \frac{-c_\theta}{m} \\ 0 & \frac{c_\theta}{m} & \frac{-c_\theta}{m} & \frac{-s_\theta}{m} & \frac{s_\theta}{m} & \frac{-c_\theta}{m} & \frac{c_\theta}{m} & \frac{s_\theta}{m} & \frac{-s_\theta}{m} \\ \frac{-1}{I_b} & \frac{r}{I_b} & \frac{-r}{I_b} & \frac{r}{I_b} & \frac{-r}{I_b} & \frac{r}{I_b} & \frac{-r}{I_b} & \frac{r}{I_b} & \frac{-r}{I_b} \\ \frac{1}{I_w} & & & & \mathbf{0}^{1\times6} & & & & \end{bmatrix} \mathbf{u} \ ,\end{aligned} \tag{3}$$

where s_θ and c_θ denote the sine and cosine of the respective angle, m is the system mass, I_w and I_b are the MoI of the RW and the overall system respectively.

III. CONTROLLER

A. Overview

The block diagram in Figure 3 gives an overview of the entire control architecture. The system consists of two

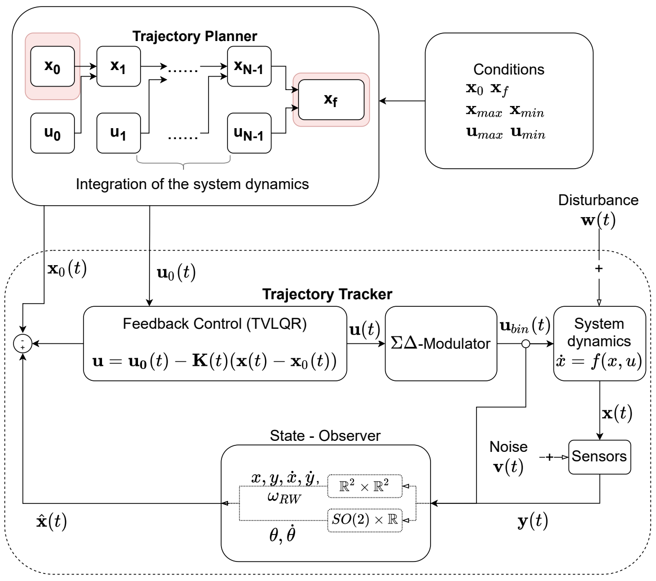

Fig. 3. Block diagram of the control architecture, consisting of a trajectory planner and follower (TVLQR, a $\Sigma\Delta$-Modulator, and an Observer).

main modules: the trajectory planner and the trajectory tracker. The planner computes an optimal trajectory a-priori, given system constraints specified by the user. The resulting trajectory has the form of N knot-points, each comprised of a state and a control vector, which in combination satisfy the dynamics of the system and the specified constraints, if physically feasible. The trajectory tracker then tries to follow the trajectory using three sub-modules: the continuous feedback controller, the modulator, and the observer. The feedback controller computes the optimal, continuous force required to follow the trajectory, using Time-Varying Linear Quadratic Regulator (TVLQR) [19], [20]. The modulator then chooses opening times for the on/off thrusters to match the continuous force best using a $\Sigma\Delta$-Modulation scheme [21]. A motor encoder and a Motion-Capture (Mo-Cap) system provide measurements of the RW velocity and the system pose, respectively. The observer uses the most recent measurements available at the sensors and the commanded control input to estimate the current system state optimally. The state estimates and the torque commands are computed at $100\,\mathrm{Hz}$ while the force command is calculated at a slower $10\,\mathrm{Hz}$ to better abide by the physical limitations of the thrusters. Further, the trajectory follower can follow the previously computed optimal trajectory and any other admissible (physically feasible) trajectory.

B. Trajectory Planner: Optimizing over Discrete States and Control Inputs

1) Cost Function: The cost function J is the main factor determining the shape of the resulting trajectory. The two qualitative criteria that should be optimized are propellant efficiency and trajectory duration. Given the actuator limits, there is a minimal time for the system to reach the final state. This trajectory is called time-optimal. On the other hand, if the desired finishing time is infinite, the most propellant-efficient action is to do nothing. In order to find a trajectory that satisfies both criteria the approach is as follows:

First, find the time-optimal trajectory using the cost function $J = t_f$. In addition, it is enforced that $t_f \geq 0$. It is well known that the time-optimal solution for such a system is a "bang-bang" controller [22] – a controller jumping between maximal and minimal control values. However, it also provides a lower bound for the overall time required to follow the trajectory.

Afterwards, define a desired final time that is a result of the multiplication of a buffer factor α with the time-optimal final time: $t_{f,des} = \alpha \cdot t_f^*$, where the buffer factor α is chosen heuristically to provide slow, smooth movement. The number of tunable heuristics is the main difference between this approach and simultaneously optimizing the final time and actuation costs. Simultaneous optimization requires two separate weights for both cost functions, whereas the separation reduces this to one.

The goal is then to find the trajectory that minimizes the propellant usage, which finishes at the desired time. The force exerted by the thrusters is approximately proportional to the propellant used; therefore, a reasonable proxy for minimal-propellant is minimal force. The cost function is then:

$$J = \sum_{k=1}^{N} \mathbf{u}_k \mathbf{R} \mathbf{u}_k^T \quad , \tag{4}$$

where $\mathbf{R}$ is a diagonal matrix that contains the respective weights for each control value. By choosing a large weight for all thrusters and a small weight for the RW the actuation of the thrusters is minimized.

2) Direct Collocation: To find an optimal trajectory that minimizes the above cost-functions, we use direct collocation [23], [24]. Its key aspect is to discretize the trajectory at N instances of time t_k, which are denoted as knot-points. Each knot-point is subject to constraints that reflect the maximal and minimal state and control values and constraints concerning its neighbors. In particular, the dynamic model (as derived in section II) must be satisfied. Since that model is continuous, we use Hermite-Simpson collocation [23] to integrate its dynamics. Then it is ensured that for any set of subsequent knot-points, the system dynamics is satisfied. In other words, the state and the control applied at some knot-point must result in the state at the next knot-point according to the integrated dynamics. From this it is possible to derive a minimization problem on all $N \times (\mathrm{Dim}(\mathbf{x}) + \mathrm{Dim}(\mathbf{u}))$ variables. For the time-optimal problem the final time adds one more decision variable. However, the optimization problem that incorporates the binary restriction on the actuators is of the class Mixed-Integer Non-Linear Programming (MINLP). These problems are increasingly difficult to solve in that they are NP-hard and combine challenges of handling non-linearities with a combinatorial explosion of integer variables [25]. By relaxing the binary condition on the respective control variables and assuming them to be continuous, the problem reduces to a problem of the class Non-Linear Programming (NLP), which can be solved significantly faster. The resulting optimization problem over all states X and control values U at all knot-points is:

$$\begin{aligned}
&\min_{X,U}\{J(X,U)\} \quad \forall k \in [0, N-1] \text{ s.t.}\\
&\mathbf{x}(0) = \mathbf{x}_{init}, \quad \mathbf{x}(t_f) = \mathbf{x}_{final}\\
&\mathbf{x}_{min} \leq \mathbf{x}_k \leq \mathbf{x}_{max}, \quad \mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max}\\
&\mathbf{x}_{k+1} - \mathbf{x}_k = \frac{\Delta t}{6}(\mathbf{f}_k + 4\mathbf{f}_{k+1/2} + \mathbf{f}_{k+1}) \qquad (5)\\
&\text{where } \mathbf{x}_{k+1/2} = \frac{1}{2}(\mathbf{x}_k + \mathbf{x}_{k+1}) + \frac{\Delta t}{8}(\mathbf{f}_k - \mathbf{f}_{k+1})\\
&\text{and } \mathbf{u}_{k+1/2} = \frac{1}{2}(\mathbf{u}_k + \mathbf{u}_{k+1})
\end{aligned}$$

Readily available, open source solvers, such as IPOPT (Interior Point OPTimiser) [26], using the programming interface provided by Drake [27], find a solution for a zero initial- and final velocity trajectory within ten seconds. However, this relaxation yields trajectories which demand continuous control input which cannot be provided by discrete or binary actuator. Therefore, any trajectory tracking controller needs to consider how to translate the desired control input into the discrete space. This is further discussed in subsection III-C.

C. Trajectory Follower

1) Time Varying Feedback Controller: Once there is an optimal trajectory to be followed, the robot should find the controls that, given the current state, propels the robot along the trajectory. We propose to use a full state feedback controller based on the TVLQR formulation. Solving the Differential Riccati Equation (DRE) by initializing the cost-to-go with its value at the final time and integrating it backward in time yields the respective cost-to-go matrices at different times t and from this, one can compute the feedback gain matrices $\mathbf{K}(t)$ [20]. The resulting control law is:

$$\mathbf{u}(t) = \mathbf{u_0}(t) + \mathbf{K}(t)(\mathbf{x}(t) - \mathbf{x}_0(t)), \qquad (6)$$

where $\mathbf{u}(t)$ is the desired control input, $\mathbf{u_0}(t)$ is the feed-forward control from the pre-computed trajectory, $\mathbf{x}(t)$ is the current state, and $\mathbf{x}_0(t)$ is the desired state according to the trajectory. Note that solving the DRE backward in time implies the controller must pre-compute the gain matrices for a trajectory beforehand, resulting in a few additional seconds of computation time in the initialization.

2) Sigma-Delta-Modulator: Given that there are discrete actuators in the present system, one needs to answer how to modulate the continuous control signal derived in the previous section. One approach introduced in [21] is using a $\Sigma\Delta$-Modulator, a technique commonly used in analog to digital modulation, to modulate the continuous force onto the binary actuators. The basic concept of a $\Sigma\Delta$-Modulator is to trigger a pulse as soon as the integrated error between the desired and current output reaches some specific threshold. The general concept of the $\Sigma\Delta$-Modulator is the following:

1) Sample the continuous signal $u(t)$ at some frequency f_{smpl} using sample and hold.
2) Compute the error $e_{\Sigma\Delta}(t)$ by taking the difference of the current thruster output $y_{Thrust}(t)$ (which is already modulated) to the desired value.
3) Integrate the error by numeric integration where Δt is the time difference between two samples: $w_e(t) = \int_{t_0}^{t} e_{\Sigma\Delta}(\tau)d\tau \approx \Delta t \sum_{i=0}^{T} e_{\Sigma\Delta}(i \cdot \Delta t)$.
4) Once the integrator value surpasses some threshold, i.e. $w_e(t) > \epsilon$, trigger a pulse. The threshold is chosen such that a single pulse resets the error integrator to zero, i.e., has the same area under the curve. This is achieved by choosing the threshold to be the inverse of the control frequency times the nominal force.

Finally, the modulator feeds back the current output value, closing the loop.

3) State-Estimation: In the previous sections, this work assumes that the entire state is available for feedback. This is not the case since the MoCap system only provides pose estimates (x, y, θ), and numerically differentiating those is not sufficient [28]. Further, all available measurements are subject to noise and thus require some filtering process to improve their quality given some underlying system model. The standard for this filter is the Kalman Filter (KF) [29]. By combining the knowledge of the underlying system and the measurements optimally, in the sense of a quadratic estimation error, the filter smoothes the data and rejects extreme outliers that fall significantly outside the distribution of the specified measurement error. This work combines a classical KF for the position, its derivative, and the RW velocity with a KF over the Lie Algebra $SO(2) \times \mathbb{R}$ for the orientation and the angular velocity as given in [30] to achieve full state estimation.

IV. RESULTS

A. Simulation

Before being evaluated on the physical system, this work evaluates the controller in a Gazebo [31] simulation that incorporates sensor noise and the slight unevenness of the floor. The simulated sensors are subjected to Additive White Gaussian Noise (AWGN) with variances that match the measured variances of the real system ($\sigma_x^2 = \sigma_y^2 = 1\times10^{-5}\,\mathrm{m}^2$, $\sigma_\theta^2 = 1\times10^{-5}\,\mathrm{rad}^2$ and $\sigma_{\omega_{RW}}^2 = 1\times10^{-4}\,(\mathrm{rad/s})^2$).

The simulation includes a heightmap representing the last openly available measurements from [32], to incorporate the unevenness of the flat-floor, which has a maximal deviation of $1\,\mathrm{mm}$ over one meter.

1) Stabilization: The system is simulated while being subjected to the instantaneous disturbance consisting of a force and a torque $\mathbf{d} = \begin{bmatrix}\mathbf{f}_d & \tau_d\end{bmatrix}$ and lasting for Δt, where:

$$\mathbf{f}_d = \begin{bmatrix}5000\,\mathrm{N} & 5000\,\mathrm{N}\end{bmatrix}^T, \;\; \tau_d = 1000\,\mathrm{N\,m}, \;\; \Delta t = 0.001\,\mathrm{s}. \qquad (7)$$

The response is shown in Figure 4 and the actuation of the controller in Figure 5. Accompanying each plot of the controlled system response is a plot of the system response to the disturbance without being controlled to highlight the effect of the controller.

As Figure 4 shows, the controller succeeds in bringing the system back to the origin after the initial disturbance. It maintains the system within less than $10\,\mathrm{cm}$ and $6°$ of the origin. The unactuated system, on the other hand, drifts over $1.5\,\mathrm{m}$ and performs more than a full rotation. The offset from

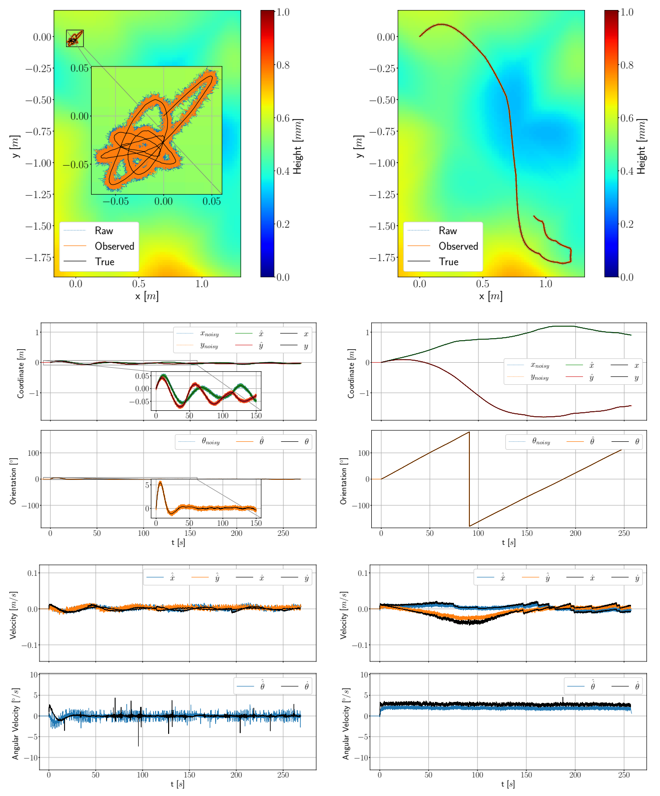

Fig. 4. Ground-track, individual coordinates, and velocities of the system responding to a disturbance (at $T = 0\,\text{s}$) on an uneven floor in simulation. Left: Controller stabilizing the system at the origin. Right: No controller running. An animation of the stabilization process is given at https://youtu.be/KRYcq3VjQUo?t=4.

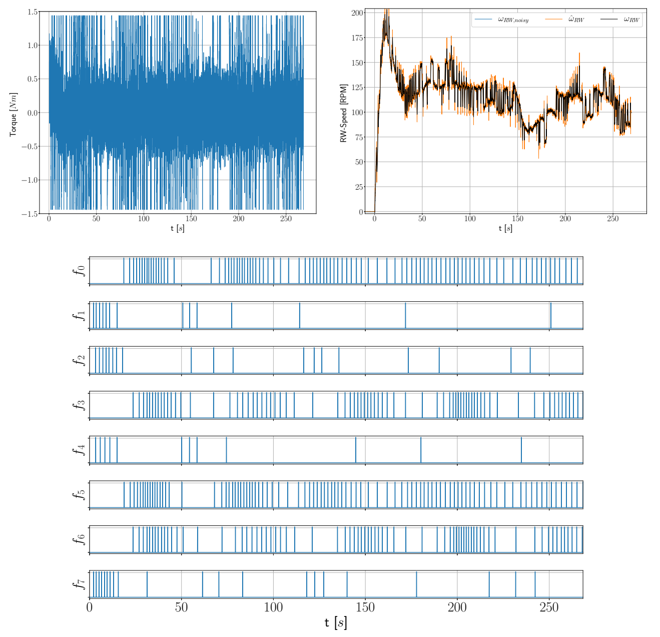

Fig. 5. Actuation required to stabilize the system at the origin on an uneven floor in simulation while being subjected to a disturbance (cf. Figure 4). Top Left: the torque exerted by the motor onto the RW. Top Right: the resulting RW velocity (raw and observed). Bottom: the thruster activity for all eight thrusters.

the origin of the controlled system is attributed to two factors: the binary nature of the thrusters and the unevenness of the floor. The continuous actuation needs to accumulate in the modulator until a firing is triggered, which in turn will propel the system through the origin, and the process repeats in the opposite direction. This well-known behavior for thruster-based systems is referred to as a limit cycle [33]. At the same time, the controller needs to compensate for the disturbances induced by the flat-floor, further adding to the oscillations about the origin. In Figure 5 the torque clearly shows "bang-bang" behavior. Since the intertia of the RWs is two orders of magnitude smaller than that of the system, the controller demands high torques such that it quickly jumps between the maximal and minimal torque. The RW velocity, however, remains within its allowable bounds. Also, in Figure 5, the thrusters 1, 2, 4, and 7 initially show high activity since they are the ones pointed in negative x and y direction, thus counteracting the disturbance. Afterward, the thrusters show alternating behavior corresponding to the small oscillations about the origin. Finally, the thrusters regularly fire to move the system towards positive y direction (Thruster 0 and 5) and slightly less frequently towards positive x direction (Thrusters 3 and 6), which corresponds to the gradient of the floor at the origin.

2) *Monte Carlo:* A Monte Carlo test simulation is executed to demonstrate the generalizability of the controller to different initial locations on the flat-floor. For a large $(n = 100)$ number of episodes, the robot spawns at random initial poses, whereby x, y, and θ result from drawing from uniform random distributions that span the range $[-2, 2]$, $[-4, 4]$, and $[-\pi, \pi]$ respectively. The controller computes the optimal trajectory to the origin and starts following it. An episode is considered successful when the euclidean distance to the origin, the euclidean velocity as well as the angular error and velocity are smaller than the threshold ϵ:

$$\begin{aligned}&\begin{bmatrix}\epsilon_{lin} & \epsilon_{ang} & \epsilon_{lin,vel} & \epsilon_{ang,vel}\end{bmatrix}\\ &= \begin{bmatrix}0.05\,\mathrm{m} & 0.05\,\mathrm{m\,s^{-1}} & 0.05\,\mathrm{rad} & 0.05\,\mathrm{rad\,s^{-1}}\end{bmatrix} \quad (8)\end{aligned}$$

Figure 6 shows the results of the Monte Carlo test. During the Monte Carlo simulation, the controller commands the system to the origin from all 100 tested initial conditions. As seen in Figure 6 the trajectories are mostly straight lines with some minor deviations caused by the uneven ground, in combination with the noise of the system and the binary thrusters. One particular trajectory (purple, bottom right of the ground-track) overshoots a desired state along the trajectory at approximately 38 s. Since the controller only attempts to control the system to the desired state at some time, it returns to a previous location resulting in a loop. This loop also corresponds to the spikes in velocities (purple). All trajectories reach the desired region at the origin and slow down to the desired maximal velocity in less than 140 s.

B. Physical System

1) *Experimental Setup:* The RW experiences stiction at low revolutions per minute. Thus, to avoid issues the RW is spun up to half its rated velocity before each trajectory. From

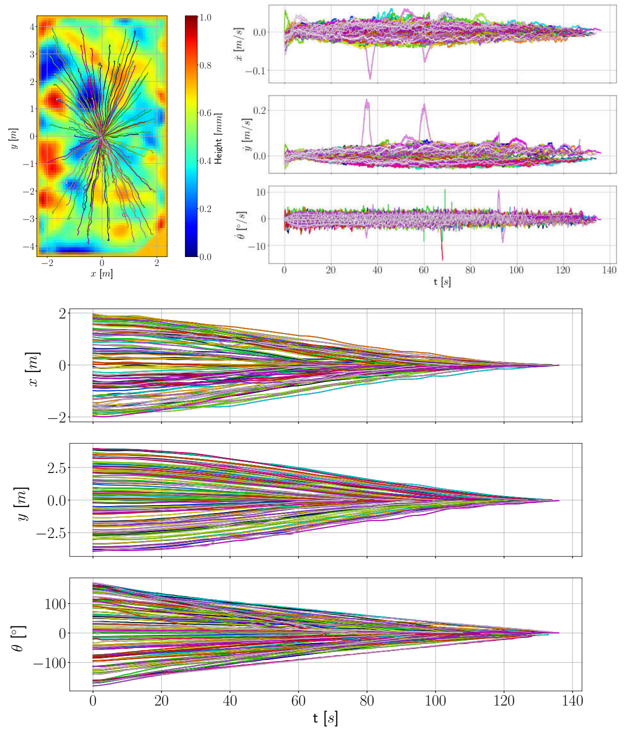

Fig. 6. Monte Carlo simulation of finding and following a trajectory to the origin.

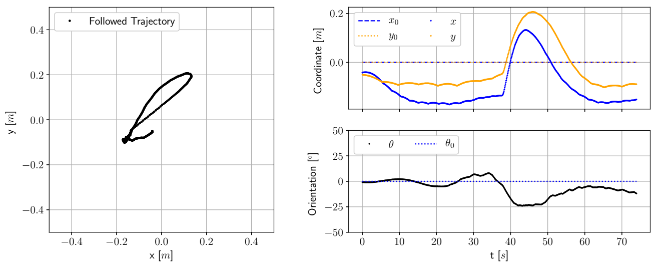

Fig. 7. Ground-track and individual coordinates of the controller stabilizing the physical system at the origin. In the coordinate plots (right) the desired value is indicated as a dashed line. A video of the stabilization process is given at `https://youtu.be/KRYcq3VjQUo?t=168`.

this state, the system is manually placed at the origin of the coordinate system before starting the trajectory follower.

2) Results:

a) Stabilization: First, the stabilization is also tested on the physical system. The disturbance is added by manually pushing the system. The results are shown in Figures 7 and 8.

In all experiments, the general tendency to drift in negative x and y direction is observed. This tendency implies that there is a slope pointing in this direction, which is further supported by the heightmap from [32]. During the stabilization process, the system initially stabilizes within $15\,\mathrm{cm}$ and $15°$ of the origin before the disturbance. The disturbance moves the system about $35\,\mathrm{cm}$ and within $30\,\mathrm{s}$ the system stabilizes in the initial region again. Figure 8 shows that very little thrusting is required to achieve this result, firing no thruster more than once every two seconds during the entire process. The RW, on the other hand,

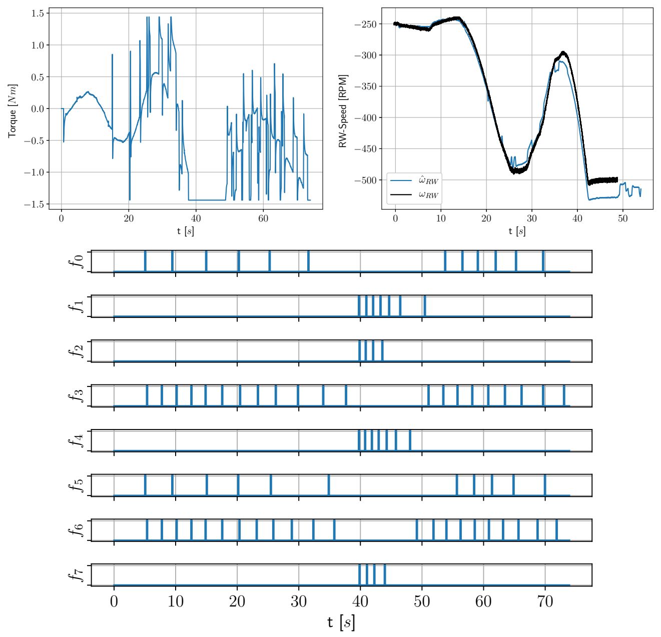

Fig. 8. Actuation of stabilizing the physical system (cf. Figure 7).

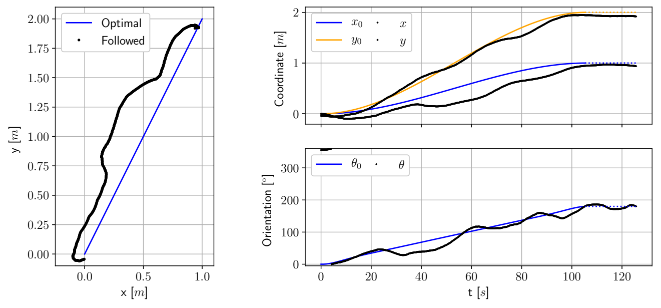

Fig. 9. Ground-track and individual coordinates of the controller following a straight-line trajectory on the physical system. After reaching the final pose of the trajectory plots (right) in the coordinate plots, the desired value is indicated as a dashed line. A video of the trajectory is given at `https://youtu.be/KRYcq3VjQUo?t=193`.

saturates rather quickly after the disturbance. It quickly reaches its maximum rotational velocity as it attempts to absorb the entire angular momentum put into the system by the disturbance. Afterward, the thrusters perform the attitude control by themselves; thus, the pointing accuracy degrades as larger errors are required to trigger a pulse since thruster usage is penalized more than RW acceleration.

b) Straight-Line Trajectory: Further, a straight-line trajectory (similar to the ones from section IV-A.2) is tested on the physical system. The results are shown in Figures 9 and 10.

The average Euclidean and angular error of the physical system to the desired trajectory are $0.325\,\mathrm{m}$ and $23.8°$ respectively, in a trajectory covering $2.2\,\mathrm{m}$ and $180°$. In particular, the error in the x-axis is significant. This error is due to the slope in that region of the flat-floor, with a gradient in the negative x-direction. As in the stabilization case, the heightmap shows a local minimum adjacent to the trajectory.

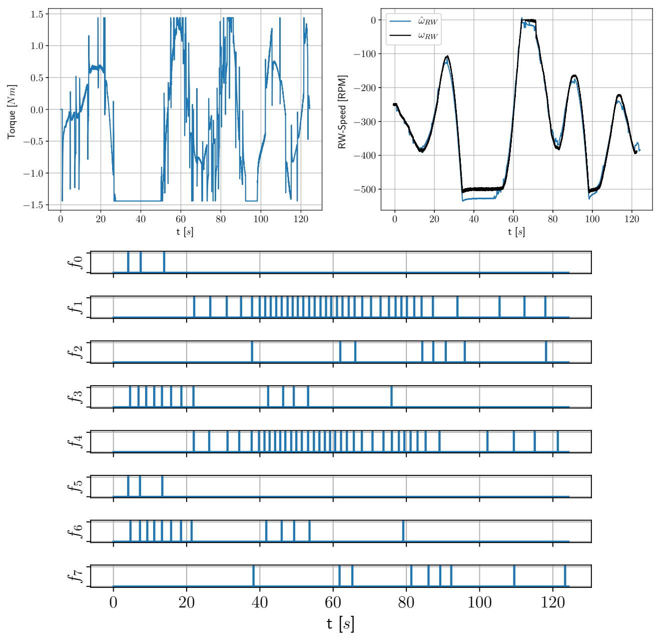

Fig. 10. Actuation of following a straight-line trajectory (cf. Figure 9) with the physical system.

TABLE II

OPTIMAL AND REAL THRUSTER ON-TIME FOR THE STRAIGHT LINE.

Thruster	0	1	2	3	4	5	6	7	Σ
Optimal	0.499 s	0.004 s	0.061 s	0.296 s	0.004 s	0.499 s	0.296 s	0.061 s	1.72 s
Real	0.30 s	3.40 s	0.70 s	1.30 s	3.40 s	0.30 s	1.30 s	0.70 s	11.40 s

Further, the RW saturates at both ends of the allowable range throughout the entire trajectory. At each instance a larger deviation from the desired orientation is observed which implies that large desired changes in angular momentum of the entire system quickly leads to RW saturation. The controller then performs orientation control using only thrusters; thus, losing precision. However, at the end of the trajectory, the RW recovers from saturation, and the pointing accuracy at the final state is within 10° . The thrusters were firing appropriately, never exceeding one fire per second for an individual thruster. Moreover, when considering the thruster on-times, as depicted in Table II, it shows an increase of less than one order of magnitude for the overall thruster on-time compared to the optimal-value. Some individual thrusters (0, 5) even decrease on-time, whereas others increase by one or multiple orders of magnitude, all while compensating for uneven ground. Note that the long on-time for thrusters 1 and 4 stem from compensating the previously mentioned error in x-direction since these thrusters are enacting a force in positive x-direction throughout the middle section of the trajectory.

c) Discussion: The experiments demonstrate that the general behavior of the simulation and the physical system is similar, indicating that the simulation is a realistic representation of the scenario. The controller manages to stabilize the system and follow a pre-computed trajectory. However, the deviations from the set values are more significant on the physical system. Besides shortcomings in the simulation, such as the idealized thrusters, the limited heightmap resolution, and a single ground-contact point, other factors that influence this difference stem from the hardware. The most significant factors that influence the overall system performance are lack of precise system identification and limited control authority. The former is outside the scope of this work; thus, it relies on previous inertial measurements of the system. An offset here contributes to the size of the limit-cycle the system exhibits around the desired orientation. This is especially the case since the control architecture combines the controller with a KF, such that the model errors both accumulate and deteriorate performance [34]. Improving this model in terms of inertial parameters, thrust, and thrust vectors for individual thrusters will significantly improve the system performance.

The latter is a consequence of the initial design of the floating platform. The high mass of the system means that the effects of any unevenness of the floor can be compensated much less by the thrusters. For example, on a slope of $1\,\frac{\mathrm{mm}}{\mathrm{m}}$ the system experiences a constant disturbance force of approximately $2.2\,\mathrm{N}$. In an ideal scenario where two thrusters are exactly aligned with this disturbance force, they can provide approximately $20\,\mathrm{N}$ in the opposing direction. Thus to compensate for the disturbance force, the thrusters would have to have an on-time of at least 10%. This firing rate is already larger than the firing rate observed for the followed straight-line trajectory.

Additionally the RW saturates very fast. Given the inertias of the entire system and the RW it can compensate only for a slight change in angular velocity of approximately $6\,^{\circ}\,\mathrm{s}^{-1}$. Despite that the trajectory planner incorporates this limitation, any disturbance that imposes an equivalent torque on the system quickly saturates the RW and thus degrades the control authority of the system. One source of such disturbance is the uneven floor. Another is an unequal nominal force of individual thrusters. When firing two thrusters that point the same way (e.g., thruster 0 and 5), the exerted wrench should only contain a force pointing in one direction. However, assuming a difference between the two thrusters it also contains a disturbance torque τ_d. If this difference is only 10% and one assumes the 10% thruster on-time to compensate the floor unevenness, this will saturate the RW within $34\,\mathrm{s}$ of operation.

V. CONCLUSION

Three Degrees of Freedom (DoF) floating platforms are a good way of partially emulating microgravity environments as they are present in space applications. This work introduced a controller for one of the heaviest floating platforms in Europe located within the ORGL at ESTEC, ESA.

First, this work develops a dynamic model of the overall system, actuated by eight solenoid-valve thrusters and one RW. The proposed controller consists of two main components: the trajectory planner and the trajectory follower. The former pre-computes optimal trajectories that connect two arbitrary states while minimizing the force exerted by the thrusters. The trajectory follower computes the control actions to follow these and any other physically feasible trajectory using a TVLQR formulation. To abide by the binary nature of the thrusters, it uses a $\Sigma\Delta$- Modulator to modulate the continuous force command computed by the TVLQR onto the thrusters. Finally, a KF estimates the system state, providing feedback for the control architecture.

This work further develops a simulation of the overall system that takes measurement noise and the unevenness of the floor into account. When testing the control architecture in this simulation, the controller achieves at least 16 cm average Euclidean and 5° angular error. The controller proves to be robust to arbitrary initial poses on the flat-floor. In a Monte-Carlo simulation in which the robot is spawned at arbitrary initial poses and tasked with finding and following optimal trajectories to the origin the controller achieves a 100% success rate.

The controller is further evaluated on the physical system. There a straight-line trajectory is also followed successfully but experiences a drop in performance. The average euclidean error approximately doubles, and the angular error increases by an order of magnitude. The increase in error is mostly attributed to a lack of precise system identification and control authority.

A thorough system identification will improve the system performance in the future. The thorough identification includes a more precise characterization of individual thrusters and their thrust vectors. This improved model is helpful in two domains: the trajectory planner would be able to plan trajectories that are more representative of the physical system and the controller would follow those better. Redesigning some system components to increase control authority also helps to improve accuracy. The two most promising options are decreasing the overall weight and inertia of the system and increasing the reaction wheel inertia.

Further work at the ORGL intends to implement, test, and compare multiple controllers based on the software framework developed in this work. One approach would be to solve the trajectory optimization problem online and control the system in a Model Predictive Control (MPC) fashion, increasing resilience to disturbances.

References

[1] R. Crowther, “Space junk–protecting space for future generations,” *Science*, vol. 296, no. 5571, pp. 1241–1242, 2002.

[2] C. J. Newman and M. Williamson, “Space sustainability: Reframing the debate,” *Space Policy*, vol. 46, pp. 30–37, 2018.

[3] D. Mehrholz, L. Leushacke, W. Flury, R. Jehn, H. Klinkrad, and M. Landgraf, “Detecting, tracking and imaging space debris,” *ESA Bulletin(0376-4265)*, no. 109, pp. 128–134, 2002.

[4] D. J. Kessler and B. G. Cour-Palais, “Collision frequency of artificial satellites: The creation of a debris belt,” *Journal of Geophysical Research: Space Physics*, vol. 83, no. A6, pp. 2637–2646, 1978.

[5] G. Neuneck, *China's ASAT test — A warning shot or the beginning of an arms race in space?*, pp. 211–224. Vienna: Springer Vienna, 2008.

[6] L. Crane, “Anti-satellite weapons,” *New Scientist*, vol. 252, no. 3362, p. 15, 2021.

[7] T. Kelso, “Analysis of the iridium 33 cosmos 2251 collision,” 2009.

[8] B. Bastida Virgili, J. Dolado, H. Lewis, J. Radtke, H. Krag, B. Revelin, C. Cazaux, C. Colombo, R. Crowther, and M. Metz, “Risk to space sustainability from large constellations of satellites,” *Acta Astronautica*, vol. 126, pp. 154–162, 2016. Space Flight Safety.

[9] J. Chatterjee, J. N. Pelton, and F. Allahdadi, *Active Orbital Debris RemovalActive orbital debris removaland the Sustainability of Space*, pp. 921–940. Cham: Springer International Publishing, 2015.

[10] S. Peters, H. Fiedler, W. Mai, and R. Förstner, “Research issues and challenges in autonomous active space debris removal,” *Proceedings of the International Astronautical Congress*, 2013.

[11] C. P. Mark and S. Kamath, “Review of active space debris removal methods,” *Space Policy*, vol. 47, pp. 194–206, 2019.

[12] K. Yoshida, “ETS-VII flight experiments for space robot dynamics and control,” *Experimental Robotics VII*, pp. 209–218, 2001.

[13] “Air bearing floor.” `https://www.nasa.gov/centers/johnson/engineering/integrated_environments/air_bearing_floor/index.html`. Accessed: 2021-08-18.

[14] 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.

[15] A. Redah, T. Mikschl, and S. Montenegro, “Physically distributed control and swarm intelligence for space applications,” *Proceedings of the Advanced Space Technologies for Robotics and Automation (ASTRA)*, 2018.

[16] E. Papadopoulos, I. Paraskevas, T. Flessa, K. Nanos, Y. Rekleitis, and I. Kontolatis, “The NTUA space robot simulator: Design & results,” *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[17] S. Wehrmann and M. Schlotterer, “Coordinated orbit and attitude control of a satellite formation in a satellite simulator testbed,” *Proceedings of the 10th International ESA Conference on Guidance, Navigation & Control Systems*, 05 2017.

[18] M. Zwick, I. Huertas, L. Gerdes, and G. Ortega, “ORGL - ESA's test facility for approach and contact operations in orbital and planetary environments,” *Proceedings of the i-SAIRAS*, 2018.

[19] D. Bertsekas, *Dynamic Programming and Optimal Control: Approximate dynamic programming. Volume 2*. Athena Scientific, 2012.

[20] R. Tedrake, “Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for mit 6.832).” Downloaded on 21.12.2021 from http://underactuated.mit.edu/.

[21] R. Zappulla, *Experimental Evaluation Methodology for Spacecraft Proximity Maneuvers in a Dynamic Environment*. PhD thesis, Naval Postgraduate School Monterey United States, 2017.

[22] J. Betts, *Practical Methods for Optimal Control Using Nonlinear Programming, Third Edition*. Advances in Design and Control, Society for Industrial and Applied Mathematics, 2020.

[23] C. HARGRAVES and S. Paris, “Direct trajectory optimization using nonlinear programming and collocation,” *AIAA J. Guidance*, vol. 10, pp. 338–342, 07 1987.

[24] M. Kelly, “An introduction to trajectory optimization: How to do your own direct collocation,” *SIAM Review*, vol. 59, pp. 849–904, 01 2017.

[25] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, “Mixed-integer nonlinear optimization,” *Acta Numerica*, vol. 22, p. 1131, 2013.

[26] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical Programming*, vol. 106, pp. 25–57, 2006.

[27] R. Tedrake and the Drake Development Team, “Drake: Model-based design and verification for robotics,” 2019.

[28] F. Van Breugel, J. N. Kutz, and B. W. Brunton, “Numerical differentiation of noisy data: A unifying multi-objective optimization framework,” *IEEE Access*, vol. 8, pp. 196865–196877, 2020.

[29] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[30] I. Markovic, J. Cesic, and I. Petrovic, “On wrapping the kalman filter and estimating with the SO(2) group,” *CoRR*, vol. abs/1708.05551, 2017.

[31] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” vol. 3, pp. 2149–2154 vol.3, 2004.

[32] H. Kolvenbach and K. Wormnes, “Recent developments on ORBIT, a 3-dof free floating contact dynamics testbed,” *Proceedings of the i-SAIRAS*, 2016.

[33] S. W. Jeon and S. Jung, “Hardware-in-the-loop simulation for the reaction control system using pwm-based limit cycle analysis,” *IEEE Transactions on Control Systems Technology*, vol. 20, no. 2, pp. 538–545, 2012.

[34] J. Doyle, “Guaranteed margins for LQG regulators,” *IEEE Transactions on Automatic Control*, vol. 23, no. 4, pp. 756–757, 1978.