Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

Singularity Maps of Space Robots and their Application to Gradient-based Trajectory Planning

Davide Calzolari[1,2], Roberto Lampariello[2] and Alessandro Massimo Giordano[1,2]
[1]Department of Informatics, Technical University of Munich (TUM), 85748 Garching, Germany.
[2]Institute of Robotics and Mechatronics, German Aerospace Center (DLR),
82234 Weßling, Germany. Email: {davide.calzolari, roberto.lampariello}@dlr.de

***Abstract*—We present a numerical method to compute singularity sets in the configuration space of free-floating robots, comparing two different criteria based on formal methods. By exploiting specific properties of free-floating systems and an alternative formulation of the generalized Jacobian, the search space and computational complexity of the algorithm is reduced. It is shown that the resulting singularity maps can be applied in the context of trajectory planning to guarantee feasibility with respect to singularity avoidance. The proposed approach is validated on a space robot composed of a six degrees-of-freedom (DOF) arm mounted on a body with six DOF.**

I. Introduction

A robot manipulator mounted on a carrier spacecraft is generally referred to as a free-floating robot if the spacecraft is not actuated (see Fig. 1) [37]. In this operational condition, which is favorable for performing proximity tasks such as grasping a tumbling target object, the differential kinematics relationship between the robot end-effector and the robot joints is affected by the dynamic coupling between the manipulator and the spacecraft. In particular, the singularities of the Jacobian matrix, which is referred to as the generalized Jacobian [37], are noticeably modified, when compared to those of the equivalent fixed-based robot [38, 28]. However, their location in the workspace or configuration space of the robot is still today not fully explored. Moreover, their treatment in the context of motion control is still a challenging problem, for which only partial solutions exist.

We present here a methodology which allows to compute configuration space singularity maps for free-floating robots with an open-chain kinematic structure. Furthermore, we demonstrate the usefulness of the map in solving global gradient-based trajectory planning problems with nonlinear programming (NLP), providing guarantees of feasibility with respect to singularity avoidance at the discretization points. Such a framework allows formulating the singularity avoidance problem as a collision avoidance problem.

A. Trajectory Planning

Trajectories resulting from motion planners for free-floating systems are typically defined by smooth continuous functions, e.g. polynomials, Bezier curves, and B-Splines [19, 29, 34]. In the most general setting, a planner optimizes the parameters of such functions, in the attempt to globally minimize a given cost function while satisfying motion constraints. Solving the resulting constrained optimal control problem requires numerical methods [17, Section 55.3.8]. The infinitely dimensional problem is transformed into a finite dimensional NLP, for which the motion constraints are solved at predefined via points along the trajectory. The motion constraints typically include limitations on the robot joint position and velocities, as well as singularity avoidance and other operational constraints. A similar methodology is applied to different robotic applications in [39, 35, 7].

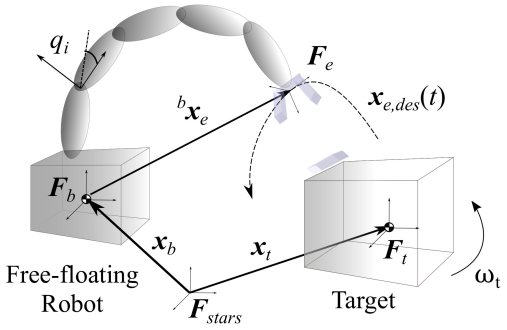


Fig. 1. A free-floating space robot is shown with a target satellite tumbling with angular velocity $\boldsymbol{\omega}_t$. The motion of the target's grasping point (shown on the target on the top left) dictates the desired tracking trajectory of the robot end-effector, $\boldsymbol{x}_{e,des}(t)$, with respect to the inertial frame $\boldsymbol{F}_{stars}$.

An example of a trajectory planning task which requires singularity avoidance, relates to a free-floating robot with n joints which has to track a given Cartesian space trajectory $\boldsymbol{x}_{e,des}(t) \in \mathbb{R}^6, \boldsymbol{\nu}_{e,des}(t) \in \mathbb{R}^6$ (e.g. motion of the grasping point on a tumbling target, see Fig. 1 with the end-effector:

$$\boldsymbol{x}_e(t) = \boldsymbol{x}_b(t) + f_k(\mathbf{q}(t)) = \boldsymbol{x}_{e,des}(t), \tag{1}$$

where $\boldsymbol{x}_e(t) \in \mathbb{R}^6$ is the pose of the end-effector, $\boldsymbol{x}_b \in \mathbb{R}^6$ is the pose of the spacecraft, $\mathbf{q} \in \mathbb{R}^n$ are the joint coordinates, and $f_k(\mathbf{q}) : \mathbb{R}^n \rightarrow \mathbb{R}^6$ is the forward kinematics of the manipulator. The equivalent trajectory in joint space, denoted with $\mathbf{q}_{des}(t) \in \mathbb{R}^n$, can be generated with a differential inverse kinematics approach as

$$\mathbf{q}_{des}(t) = \int_{t_0}^{t_f} \boldsymbol{J}_m^{*-1}(\mathbf{q}(t))\boldsymbol{\nu}_{e,des}(t)dt + \mathbf{q}(t_0), \tag{2}$$

where $\mathbf{q}(t_0) \in \mathbb{R}^n$ is the initial joint configuration, and $\boldsymbol{J}_m^*(\mathbf{q}) \in \mathbb{R}^{6\times n}$ is the generalized Jacobian [37]. Clearly,

$\boldsymbol{J}_m^*(\mathbf{q})$ needs to be always invertible, otherwise the end-effector loses mobility in one or more inertial directions. The singularities of $\boldsymbol{J}_m^*(\mathbf{q})$ are also called dynamic singularities, since they depend on the inertial distribution of the system [28]. At singular configurations, the manipulator cannot physically execute a given maneuver, hence huge tracking errors will result [28]. Additionally, in the neighborhood of a singularity, the required joint rates can grow very quickly, thus, practically, it is not only important to avoid singularities, but also to keep a safe distance from them.

The trajectory planning for the problem above would involve finding a suitable initial spacecraft pose and robot configuration. More generally, in the context of trajectory planning, the problem of singularity avoidance is instead handled by locally maximizing the manipulability at each via-point or alternatively satisfying the joint velocity constraints.

B. Related Work

The preliminary analysis conducted by Papadopoulos and Dubowsky [28] presents the complex problem of handling singularities for free-floating robotic systems, and develops the concepts of the Path Independent Workspace (PIW), i.e., subsets of the workspace in which singularities are not present, and of the Path Depended Workspace (PDW), where a dynamic singularity may be encountered, depending on the path taken by the robot. These concepts are useful for planar manipulators, but we verified that their application loses effectiveness in the spatial case, as the entire workspace can become PDW. This observation can also be drawn from the analysis carried out in Rybus et al. [33] for different robot kinematic chains. In [33], the location of singularity sets are evaluated qualitatively in configuration space for various free-floating space robots, showing that the singularity sets have a complex topology.

Nanos and Papadopoulos [27] proposed an analytical method to find the initial base attitude and joint configuration such that a maneuver between two workspace points can be executed without encountering dynamic singularities. While this method is effective for planar manipulators, the analysis for spatial systems was carried out considering a spherical wrist and a fixed orientation of the end-effector, thus only addressing singularities originating from translational end-effector motions. However, for the problem presented in (1), rotational end-effector motion is necessary. Furthermore, an analytical analysis of all the singularities with the method presented in [27] is not possible, since the relative equations explode in complexity and their manipulation becomes unfeasible. As such, we now turn to alternative numerical methods with which singularities of robot manipulators can be analyzed.

A general, state-of-the-art formal numerical method to locate singularities in the configuration space of mechanisms can be found in Bohigas et al. [6]. This method is effective when applied to mechanisms with loops and the number of DOF is not too high. However, symbolic manipulation of the differential kinematic equations is required, in particular, non-linear terms have to be transformed into sets of inequalities. Unfortunately, in the case of free-floating manipulators, the complexity of the differential kinematic makes it impossible to perform this pre-processing. Other methods rely on numerical approaches which provide formal guarantees on the computations, e.g. Interval Arithmetic (IA). The application of IA is not new to singularity analysis; see for example the work of Merlet [24] and Kotlarski et al. [18] for workspace analysis, in particular for parallel robots. Benoit et al. [4], proposes to analyze the configuration space and workspace singularities using IA. This algorithm requires the symbolic derivative of the determinant and its interval evaluation, which is unfeasible for free-floating systems due to the high complexity of their differential kinematics.

C. Contribution

In this work, we develop a method to compute singularity maps (S-Maps) for free-floating robots in configuration space. These maps define sets which potentially contain singular configurations. The remaining space is guaranteed to be free from singularities. The benefits of using S-Map for singularity avoidance are many. First, a global trajectory planning scheme is possible, since the distance to the singularity surface is readily available. Notice that this feature provides a concrete advantage over using only local information, since it is possible to impose that the solution is always at a predefined minimal distance from the singularities. As such, guarantees can also be provided for a given neighborhood of the planned trajectory, which may include typical tracking control and modeling errors [21]. Second, for a trajectory planner, calculating the expressions that relate to the singularity avoidance constraints and their gradient with respect to the optimization parameters, is computationally cheaper and numerically more robust with an S-Map than with the determinant of the generalized Jacobian or the joint velocities. Currently, there are no other algorithms that are either computationally feasible or that can provide the same guarantees for the systems considered.

The main contributions of this work are the following:

1) A formal numerical method to obtain, for the first time, S-Maps of a 6-DOF free-floating robot with guarantees of completeness using two different criteria, i.e., IA and Taylor Models. To the best of our knowledge, the application of Taylor models for the detection of singularities is a novel approach.
2) A detailed comparison of the two different criteria to compute the map in terms of efficiency and complexity.
3) The definition of a C^1 continuous distance function to the surfaces represented by the maps, and a demonstration of the applicability of the results in the context of gradient-based optimization, addressing the singularity avoidance problem as a collision avoidance problem.

The rest of the paper is organized as follows. Section II presents the kinematics and dynamics of free-floating space robots, and introduces the relevant numerical tools which we make use of in this work. Section III presents the set of algorithms required for the computation of the singularity

maps. In Section IV, essential functions for the utilization of the maps are developed, and an application to a simple motion planning task is given. Section V provides the results of the computation of several singularity maps, as well as of the motion planning task. Finally, Sections VI and VII present the discussion and the conclusion.

II. Preliminaries

Let us consider a space robot composed of a serial open-chain manipulator with n revolute joints, mounted on a spacecraft. The space robot is controlled in free-floating mode.

A. Kinematics and Dynamics of a Free-Floating Robot

The equation of motion can be written as [23]

$$\begin{bmatrix} \mathbf{M}_b(\mathbf{q}) & \mathbf{M}_{bm}(\mathbf{q}) \\ \mathbf{M}_{bm}^T(\mathbf{q}) & \mathbf{M}_m(\mathbf{q}) \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\nu}}_b \\ \ddot{\mathbf{q}} \end{bmatrix} + \boldsymbol{c}(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\nu}_b) = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau} \end{bmatrix}, \quad (3)$$

where $\mathbf{q} \in \mathbb{R}^n$ are the joint coordinates, $\boldsymbol{\nu}_b \in \mathbb{R}^6$ is the spacecraft (or base) body twist, $\mathbf{M}_b \in \mathbb{R}^{6\times6}$ is the system inertia matrix around the base body, $\mathbf{M}_m \in \mathbb{R}^{n\times n}$ is the manipulator inertia matrix, $\mathbf{M}_{bm} \in \mathbb{R}^{6\times n}$ represents the coupling between spacecraft and manipulator, $\boldsymbol{c}(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\nu}_b) \in \mathbb{R}^{6+n}$ are the Coriolis/centrifugal forces, and $\boldsymbol{\tau} \in \mathbb{R}^n$ are the joint torques. The momentum $\boldsymbol{h}_b \in \mathbb{R}^6$ of the system around the base can be written as

$$\boldsymbol{h}_b = \mathbf{M}_b(\mathbf{q})\boldsymbol{\nu}_b + \mathbf{M}_{bm}(\mathbf{q})\dot{\mathbf{q}}. \quad (4)$$

The end-effector velocities $\boldsymbol{\nu}_e \in \mathbb{R}^6$ are related to the space robot base body twist $\boldsymbol{\nu}_b$ and joint rates $\dot{\mathbf{q}}$ as

$$\boldsymbol{\nu}_e = \boldsymbol{J}_b(\mathbf{q})\boldsymbol{\nu}_b + \boldsymbol{J}_m(\mathbf{q})\dot{\mathbf{q}}, \quad (5)$$

where $\boldsymbol{J}_b \in \mathbb{R}^{6\times6}$ is the Jacobian which relates the end-effector wrench to the base body wrench, and $\boldsymbol{J}_m \in \mathbb{R}^{6\times n}$ is the manipulator Jacobian. With the hypothesis of zero momentum (i.e., $\boldsymbol{h}_b = \mathbf{0}$), the conservation of momentum yields the following equation

$$\mathbf{M}_b(\mathbf{q})\boldsymbol{\nu}_b = -\mathbf{M}_{bm}(\mathbf{q})\dot{\mathbf{q}}. \quad (6)$$

When the momentum is zero, (6) can be used to substitute for $\boldsymbol{\nu}_b$ in (5) to obtain

$$\boldsymbol{\nu}_e = \boldsymbol{J}_m^*(\mathbf{q})\dot{\mathbf{q}}, \quad (7)$$

$$\boldsymbol{J}_m^*(\mathbf{q}) = \boldsymbol{J}_m - \boldsymbol{J}_b\mathbf{M}_b^{-1}\mathbf{M}_{bm}, \quad (8)$$

where $\boldsymbol{J}_m^*(\mathbf{q}) \in \mathbb{R}^{6\times n}$ is the generalized Jacobian [37].

Using the inverse-chain approach, first presented in Abiko et al. [1], the equations of motion can be re-written for the same system, however, with an inverted order of the links, such that the base becomes the end-effector and the end-effector the base, as follows:

$$\begin{bmatrix} \mathbf{M}_{ee}(\mathbf{q}) & \mathbf{M}_{em}(\mathbf{q}) \\ \mathbf{M}_{em}^T(\mathbf{q}) & \mathbf{M}_m^{ic}(\mathbf{q}) \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\nu}}_e \\ \ddot{\mathbf{q}} \end{bmatrix} + \boldsymbol{c}_e(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\nu}_e) = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau}^{ic} \end{bmatrix}, \quad (9)$$

where $\mathbf{M}_{ee}(\mathbf{q}) \in \mathbb{R}^{6\times6}$ is the inertia of the system expressed around the end-effector of the original system, $\mathbf{M}_{em}(\mathbf{q}) \in \mathbb{R}^{6\times n}$ are the couplings between manipulator and end-effector dynamics, and $\boldsymbol{c}_e(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\nu}_e) \in \mathbb{R}^{6+n}$ are Coriolis/centrifugal forces. With a little abuse of notation, we denote the joint coordinates with the same symbol $\mathbf{q}$ for both the original and the inverted chain systems.

The momentum equation can now be written with respect to the end-effector frame as

$$\boldsymbol{h}_e = \mathbf{M}_{ee}(\mathbf{q})\boldsymbol{\nu}_e + \mathbf{M}_{em}(\mathbf{q})\dot{\mathbf{q}}. \quad (10)$$

Assuming again that the momentum is zero, (10) yields another factorization for the generalized Jacobian:

$$\boldsymbol{J}_m^*(\mathbf{q}) = -\mathbf{M}_{ee}^{-1}(\mathbf{q})\mathbf{M}_{em}(\mathbf{q}). \quad (11)$$

B. Dynamic Singularities and Relevant Properties

Dynamic singularities occur whenever $\boldsymbol{J}_m^*(\mathbf{q})$ loses rank, i.e., for a non-redundant robot arm

$$\det\left(\boldsymbol{J}_m^*(\mathbf{q})\right) = 0. \quad (12)$$

Common robotic tasks involve trajectories that are defined in the workspace of the robot [27]. Having a map of singularities in the workspace would greatly facilitate the planning and control of the considered systems. Unfortunately, a representation of singularities in the workspace is not meaningful for spatial free-floating robots since, practically, the whole Cartesian space would appear singular, due to the projection of high-dimensional surfaces onto lower dimensions [9]. Moreover, for free-floating space robots, a Cartesian pose may be singular or not depending on the path taken by the manipulator to reach it [28]. This does not happen in the configuration space, where the singularity surfaces are fixed. Since $\boldsymbol{J}_m^*(\mathbf{q})$ depends solely on $\mathbf{q}$, dynamic singularities loci may be found in configuration space by solving (12). Unfortunately, if the number of degrees of freedom of the robot arm is relatively high, the problem becomes analytically intractable. Therefore, considering the aforementioned arguments, the best way to avoid singularities is to analyze and plan trajectories in configuration space using numerical approaches.

The classical formulation of the generalized Jacobian, given in (8), does not allow to simplify the analysis of the determinant. In contrast, the formulation given in (11) using the inverse-chain approach, allows to conclude that all singularities arise by the sole degeneracy of $\mathbf{M}_{em}(\mathbf{q})$, because

$$\det\left(\boldsymbol{J}_m^*(\mathbf{q})\right) = -\det\left(\mathbf{M}_{ee}^{-1}(\mathbf{q})\right)\det\left(\mathbf{M}_{em}(\mathbf{q})\right), \quad (13)$$

where the first term of the product is never zero, since $\mathbf{M}_{ee}(\mathbf{q})$ represents an inertia, thus always invertible. Moreover, in [9], it was shown that singularities do not depend on the last link inertial properties, and, most importantly, that they do not depend on the last joint coordinate. This is very relevant from a computational point of view, since the search space for singularities can be reduced by one dimension.

C. Formal Numerical Methods

In this subsection we introduce the two numerical tools which we make use of in Section III to compute the S-Maps.

Interval Arithmetic: Interval Arithmetic (IA), first introduced by Moore [25], is a model for self-validated numerical analysis, which guarantees bounds on operations involving terms modeled by intervals. The interval model represents the set of possible values for each variable. Formally, we define an interval as the set

$$[a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$$

where a is the infimum and b is the supremum. IA has the correctness property [16]: arithmetic operations (like addition, multiplication, etc.) can be defined mathematically for intervals such that the evaluation of an expression always yields an interval containing all results. For an introduction to IA, refer to [16]. In the scope of this article, we denote an interval quantity with the superscript $(\cdot)^I$, while the operators $mid(x^I)$ and $rad(x^I)$ compute the mid-point and the radius of the interval x^I, respectively. When the operators $mid(\cdot)$ and $rad(\cdot)$ are applied on interval matrices, they operate element-wise and return a matrix.

Taylor models: Evaluating properties of parametric matrices using their interval representation may result in large over-estimation. This is due to the following facts: first, the elements of the matrix may contain long expressions and involve the same variables many times. Evaluating such expressions with IA leads to very conservative results, due to the so called "wrapping effect" [16, 3]. Indeed, this is the case when considering the generalized Jacobian of spatial space robots with several DOF. Second, IA does not account for the dependencies between the elements of the matrix, which is essential to determine invertibility (i.e., if the set of attainable values for the determinant contains zero or not), thus, a substantial degree of conservativeness is introduced. One way to reduce conservativeness is to reduce the size of the inputs, or use higher order methods, like Taylor models introduced by Berz and Hoffstaetter [5]. Taylor models typically use a representation which involves a polynomial plus an interval remainder, allowing to enclose expressions in a more complex way than with simple intervals. In general, IA is computationally fast, but the results can be quite conservative. Instead, Taylor models typically require more computation time, but they can provide much tighter solutions [3].

For the computations performed in this work, we used the interval analysis and Taylor models implementations of the CORA Toolbox for MATLAB [2, 3].

III. S-Map Computation

The formal methods presented in Section II-C require the computation of symbolic expressions. While their computation for the generalized Jacobian may generally be prohibitive, the minimum set of required matrices, i.e., $\boldsymbol{J}_m(\mathbf{q})$, $\boldsymbol{J}_b(\mathbf{q})$, $\mathbf{M}_b(\mathbf{q})$, and $\mathbf{M}_{bm}(\mathbf{q})$ in (8), and $\mathbf{M}_{em}(\mathbf{q})$ in (11), can be readily computed symbolically. In this work, the rigid-body dynamic library in [11] is utilized. The symbolic matrices need to be computed only once, and then stored on a computer in a suitable form. In the following, two different criteria to evaluate the regularity of $\boldsymbol{J}_m^*$ over a set are presented.

A. Criterion 1: Regularity of Interval Matrices

To assure invertibility of a matrix over a set of configurations, we will prove its regularity over this set by means of IA. Consider a set in configuration space defined by an interval box $\mathbf{q}^I$. The interval representation of the generalized Jacobian, denoted by $\boldsymbol{J}_m^{*I}$, can be evaluated with IA as follows, using (8) (see also Section VI):

$$\boldsymbol{J}_m^{*I} = \boldsymbol{J}_m^I(\mathbf{q}^I) - \boldsymbol{J}_b^I(\mathbf{q}^I) \left[\mathbf{M}_b^I(\mathbf{q}^I)\right]^{-1} \mathbf{M}_{bm}^I(\mathbf{q}^I), \quad (14)$$

where the operator $[\cdot]^{-1}$ performs an enclosing of a matrix inverse using the polynomial-time efficient algorithm presented in Rohn and Farhadsefat [32].

In order to check the regularity of $\boldsymbol{J}_m^{*I}$, one may use the sufficient condition presented in Rex and Rohn [31, Corollary 3.2], which we report in the following. Let $\mathbf{A}^I$ be an interval matrix and $mid(\mathbf{A}^I)$ be nonsingular. If

$$\rho(|mid(\mathbf{A}^I)^{-1}|\ rad(\mathbf{A}^I)) < 1, \quad (15)$$

holds, then $\mathbf{A}^I$ is strongly regular. The $\rho(\cdot)$ operator is the spectral radius, while the $|\cdot|$ operator denotes the absolute value performed element-wise. One may wonder if the same conclusion may be drawn evaluating the determinant of an interval matrix using IA and checking if zero is contained. The answer is positive, however using condition (15) is a better choice since it is less conservative. Moreover, condition (15) can be extended in a straightforward way to check regularity for non-square matrices as reported in Shary [36].

B. Criterion 2: Determinant Evaluation using Taylor Models

Following the arguments in Sections II-C, we can expect (15) to be quite conservative when evaluating the regularity of $\boldsymbol{J}_m^{*I}$ for large ranges of $\mathbf{q}^I$. As a consequence, we may be able to prove regularity of $\boldsymbol{J}_m^{*I}$ only for tiny boxes in configuration space, thus leading to an inefficient search.

To solve this problem, we consider the evaluation of the determinant of $\mathbf{M}_{em}(\mathbf{q})$ using Taylor models of 3rd order. To determine regularity of a set in configuration space using Taylor models, consider again an interval box $\mathbf{q}^I$. The interval $\mathbf{q}^I$ is converted to the Taylor models representation, i.e., Taylor variables (or symbols), denoted by $\mathbf{q}^{tm}$, together with Taylor coefficients. Each element of the matrix $\mathbf{M}_{em}(\mathbf{q}^{tm})$ is evaluated using Taylor models arithmetic. The result is a matrix, where each element contains polynomials in $\mathbf{q}^{tm}$ variables up to the 3rd order, plus an interval remainder. Then, the Taylor model of the determinant $\det(\mathbf{M}_{em}(\mathbf{q}^{tm}))$ is computed and denoted by $T_{\det}$. Since Taylor models do not directly provide a range of values, a bound $B(T_{\det})$ on the Taylor model must be computed, for example, using a simple Interval Arithmetic approach [3]. If

$$0 \notin B(T_{\det}), \quad (16)$$

the regularity of $\boldsymbol{J}_m^*$ follows.

In alternative to IA and Taylor models, there exists a further criterion to determine the regularity of parametric interval matrices, proposed in Popova [30, Theorem 7]. This criterion

is attractive since it handles parameters that appear multiple times in the matrix, hence the dependencies between the elements of the Jacobian could be partially accounted for. Unfortunately, this method requires a nontrivial pre-processing of the interval parameters (in our case, the intervals $\cos(q_i^I)$ and $\sin(q_i^I)$ in $\boldsymbol{J}_m^*(\mathbf{q}^I)$), and the generation of specific matrices, which is too complex for the generalized Jacobian of a 6 DOF space robot.

C. Complete Search: Branch and Bound

For a considered portion of configuration space, the computation of the S-Map is carried out using a branch and bound approach described as follows. At first, the search space is bisected iteratively N times and a total of 2^N boxes are obtained. Then, each box is tested for regularity of $\boldsymbol{J}_m^*$ using either condition (15) or condition (16). Whenever a box cannot be proven to be singularity-free, it is bisected along its largest range, in a similar fashion to other numerical approaches, e.g. Bohigas et al. [6], Benoit et al. [4] and Merlet [24]. The bisection proceeds recursively up to a given maximum precision $\epsilon \in \mathbb{R}^+$. If the algorithm cannot conclude regularity of a box at precision ϵ, then the box is considered as a singularity set, and the box data are appended to a list. In case that all bisected sub-boxes of a certain box may contain singularities, the algorithm stores solely the information of the biggest enclosing box.

D. Heuristic Pruning

When applying the bisection method over the search space, in order to isolate singularity sets, the most time consuming part of the algorithm is when checking sets in the neighborhood of the singularity surfaces. The computational time invested in these configuration space locations increases with the search space dimension. An approximation of these close to singularity locations can be used to prune the search space and increase the speed of the algorithm. This method is very useful because it reduces the extent of the search space of the branch and bound algorithm presented in Section III-C.

In the following, we describe one way to compute a point cloud close to singular configurations, and then we explain how to exploit it for the proposed heuristic.

1) Point cloud computation: Starting from many random initial configurations, (12) can be solved locally in order to generate a point cloud of close to singular configurations. In general, these solutions represent hypersurfaces in the manipulator joint space [28]. Solving (12) is equivalent to finding one global minimum of the manipulability $m(\mathbf{q}) = |\det(\boldsymbol{J}_m^*(\mathbf{q}))|$. Here, a simple gradient descent method is used to find robot configurations that locally minimize $m(\mathbf{q})$. Starting from an initial configuration $\mathbf{q}_0 \in \mathbb{R}^n$, the update equation of the k-th iteration to minimize $m(\mathbf{q})$ is

$$\mathbf{q}_{k+1} = \mathbf{q}_k - \alpha \frac{\partial m(\mathbf{q}_k)}{\partial \mathbf{q}}, \tag{17}$$

where $\alpha \in \mathbb{R}^+$ is a small scalar. The numerical gradient $\frac{\partial m(\mathbf{q}_)}{\partial \mathbf{q}}$ is computed analytically. The algorithm is started from multiple randomly uniformly sampled initial configurations (e.g. in the range $[-\pi, \pi]$ for each joint) in order to find a significant number of local minima.

The search for the minimum value of m stops whenever an iteration does not improve the value of the manipulability by a relative percentage, or a number of maximum iterations is reached. Each minimization can be carried out in parallel, thus reducing the computational time. In practice, it may take many iterations and initial configurations in order to obtain a sufficient number of samples with very small manipulability to generate a point cloud that adequately represents the solutions of (12).

It is important that the minimization finds only very close local minima to the initial configuration, otherwise many starting points could converge to "more attractive" surfaces, leaving some singularity sets of configuration space not adequately represented. To improve coverage of the solution hypersurfaces, one can first calculate the manipulability for many random configurations, and following, the minimization algorithm is applied only to the configurations whose corresponding manipulability is lower than a certain threshold. The remaining less promising configurations are discarded.

2) Point cloud utilization: We define S_q as a set containing points with manipulability index inferior to a certain threshold. We propose to use this information in order to improve the efficiency of the branch and bound algorithm by means of the following heuristic: if the volume of the considered set is below a certain threshold and the set contains a point belonging to S_q, then the set is considered singular. In this way, in a neighborhood of singularities, the algorithm does not have to evaluate and bisect all boxes till the maximum precision.

IV. S-Map Utilization

Upon completion, the branch and bound algorithm described in Section III-C returns a list of singularity sets (in form of intervals), which we refer to as S-Map. One of the most useful features of the S-Maps is that they can provide the distance to the nearest singularity set, as well as its location in configuration space. Since calculating the distance of a point to an n-dimensional box can be inefficient, each box of an S-Map is enclosed within a sphere. The midpoint of the i-th interval box is used as the center $\mathbf{q}_{sm}^i \in \mathbb{R}^n$ of the i-th sphere, where $i = 1, \ldots, M$, being M the total number of intervals stored in the S-Map. The radius $r_{sm}^i \in \mathbb{R}$ of the i-th sphere is computed as half of the main diagonal of the i-th box, i.e., the norm of the interval radius vector.

A. Distance Function

Given a configuration $\mathbf{q} \in \mathbb{R}^n$, the i-th element of the vector $\boldsymbol{d} \in \mathbb{R}^M$ of signed euclidean distances to all singularity sets is computed as follows

$$d_i(\mathbf{q}_{sm}^i, r_{sm}^i, \mathbf{q}) = \|\mathbf{q}_{sm}^i - \mathbf{q}\|_2 - r_{sm}^i. \tag{18}$$

The smallest element of $\boldsymbol{d}$, denoted by $d_s \in \mathbb{R}$, is

$$d_s(\mathbf{q}) = \min_i(d_i(\mathbf{q}_{sm}^i, r_{sm}^i, \mathbf{q})), \tag{19}$$

and it represents the signed distance to the closest sphere. The distance measure is signed in the sense that, if a configuration $\mathbf{q}$ is inside a sphere, the corresponding value of d_s is negative. This feature can be used in trajectory planning to detect if a segment connecting two configurations intersects the S-Map, as is explained in the following.

B. Segment Intersection

Given an S-Map and configurations $\mathbf{q}_a$ and $\mathbf{q}_b$, the segment $\mathbf{q}_{ab} = \mathbf{q}_b - \mathbf{q}_a$ is computed. The S-Map is queried to return a list of $j = 1, \ldots, K$ spheres, for which the j-th sphere satisfies

$$d_j(\mathbf{q}_{sm}^j, r_{sm}^j, \frac{1}{2}(\mathbf{q}_a + \mathbf{q}_b)) \leq \frac{1}{2}\|\mathbf{q}_{ab}\|_2. \tag{20}$$

Then, the j-th element of the vector $\boldsymbol{d}^{ab} \in \mathbb{R}^K$ is defined as

$$d_j^{ab} = \|\boldsymbol{v}^j\|_2^2 - (r_{sm}^j)^2, \tag{21}$$

where $\boldsymbol{v}^j = \boldsymbol{a}_p^j - (\boldsymbol{a}_p^j \cdot \boldsymbol{n})\boldsymbol{n}$, with $\boldsymbol{a}_p^j = \mathbf{q}_a - \mathbf{q}_{sm}^j$, and $\boldsymbol{n} = \frac{\mathbf{q}_{ab}}{|\mathbf{q}_{ab}|}$. Here, $\|\boldsymbol{v}^j\|_2$ represents the distance between the center of the j-th sphere and the segment $\mathbf{q}_{ab}$. The necessary and sufficient condition for the segment $\mathbf{q}_{ab}$ to intersect the S-Map is then

$$\min_j(d_j^{ab}) \leq 0. \tag{22}$$

Since the singularity set could consist of relatively thin "walls", we can exploit condition (22) to avoid intersections, as necessary in the following use case.

C. Trajectory Planning

We present a use case for the S-Map which relates to trajectory planning. The distance function d_s presented in (19) may be used as an inequality constraint for a motion planner which minimizes a given cost function, e.g.

$$\begin{aligned} &\min_{\mathbf{q}} \ \gamma(\mathbf{q}) \\ &\text{s.t. } d_s(\mathbf{q}) \geq \Delta, \end{aligned} \tag{23}$$

where $\Delta \in \mathbb{R}^+$ is a small scalar, defining the required minimum distance to the closest singular surface. The distance function enables to safely minimize the cost function $\gamma(\mathbf{q})$ in virtue of a global representation of the singularity sets.

For solving the problem above, we propose the following gradient-based procedure. An initial configuration $\mathbf{q}_0$ is given. At the beginning of each k-th iteration, the distance measure $d_s(\mathbf{q}_k)$ is evaluated. If a step $\beta^* \in \mathbb{R}^+$ in any direction could violate the constraint (23), i.e., if

$$d_s(\mathbf{q}_k) < \beta^* + \Delta, \tag{24}$$

then the optimizer projects the gradient of the cost function $\gamma(\mathbf{q}_k)$ onto the plane tangent to the constraint, represented by the normal vector $\hat{\boldsymbol{h}}(\mathbf{q}_k) \in \mathbb{R}^n$, thus obtaining the projected gradient $\boldsymbol{g}(\mathbf{q}_k) \in \mathbb{R}^n$. The plane is computed as the numerical gradient of the distance function. The next iteration $\mathbf{q}_{k+1}$ is found using the following approach. First, the intermediate step $\mathbf{q}_k^*$ is performed in the direction of the normalized projected gradient $\hat{\boldsymbol{g}}(\mathbf{q})$:

$$\mathbf{q}_{k+1}^* = \mathbf{q}_k - \beta\hat{\boldsymbol{g}}(\mathbf{q}_k), \tag{25}$$

where $\beta \in \mathbb{R}^+$ is the size of the step. The update equation for the k-th iteration is

$$\mathbf{q}_{k+1} = \mathbf{q}_{k+1}^* + \delta\hat{\boldsymbol{h}}(\mathbf{q}_{k+1}^*), \tag{26}$$

$$\delta = \max(0, \Delta - d_s(\mathbf{q}_{k+1}^*)), \tag{27}$$

such that $\mathbf{q}_{k+1}$ is pushed away from the local surface at a minimum distance Δ. The size β is found through line search: from the initial maximum step β_{max}, the step β is halved iteratively until satisfaction of the constraint in (23) at $\mathbf{q}_{k+1}$, thus guaranteeing that the step $\mathbf{q}_k \to \mathbf{q}_{k+1}$ does not intersect the S-Map. The size of the step also ensures that the external boundaries of the map are not violated. The optimization stops either when the projected gradient has a small magnitude or when the feasible step size β falls below the threshold β_{min}.

Note that the approximation of the singularity surfaces with spheres locally presents recurring cusps (due to the overlapping of the spheres). Therefore, this method can incur into numerical inefficiency. In fact, the distance function d_s is a continuous C^1 function, while in general C^2 continuity is preferred in the context of gradient-based optimization. However, in [10], [8], and [35] it was shown that, when searching in high dimensional spaces, having smoother constraint functions translates rather into higher efficiency than better converge properties of the optimizer. A more robust implementation would consist in fitting a mesh on the boundary of the S-Map and retrieving the gradient, for example, with the technique proposed in [22].

V. Results

We present an example of application of the proposed methods for the computation of the S-Maps for a planar and a spatial free-floating robot. Then, we proceed to show the results of a motion planning task utilizing the S-Maps.

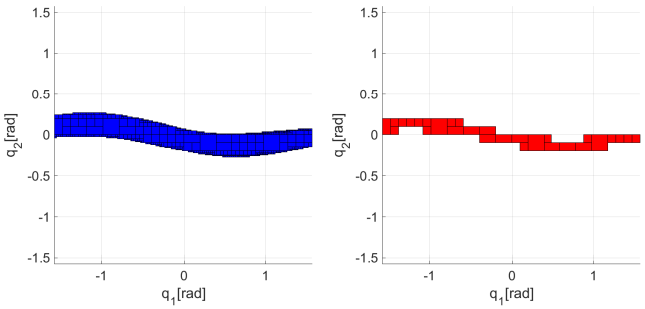
We consider a free-floating system consisting of a 6-DOF manipulator mounted on a spacecraft with mass 322 [kg] and with body inertia

$$\mathbf{I}_b = [46.87, 32.11, 40.52, -1.61, -2.58, -1.29] \ [\text{kg}\cdot\text{m}^2],$$

where the representation $\mathbf{I} = [I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}, I_{yz}]$ is used. The first joint of the manipulator is mounted with an offset $\boldsymbol{r}_0 = \begin{bmatrix}0.2, 0.0, 0.4775\end{bmatrix}^T$ $[m]$ with respect to a reference frame attached at the center of mass of the spacecraft. The kinematic and dynamic parameters of the manipulator can be found in [12], where the third joint is fixed to obtain a 6-DOF robot arm. An additional payload of mass 3 [kg] is attached at the end-effector of the robot.

A. S-Map computation for a planar 3-DOF space robot

We present a comparison of the singularity maps computed for a planar 3-DOF free-floating robot using the two criteria described in Sections III-A and III-B. The robot is obtained by fixing joints 1, 6, and 7 of the arm, and by scaling the mass and inertia of the spacecraft down to 50% of the original values. The algorithms are fed dynamics matrices that are explicit function of all joints (including the last joint, even if it does not play a role for dynamic singularities). The reason behind



(a) Interval Arithmetic (b) Taylor models

Fig. 2. Comparison of the singularity maps of the 3-DOF planar free-floating robot presented in Section V-A computed using the different algorithms presented in Sections III-A and III-B. The S-Maps represent an envelope of the singularity surfaces. The remaining space is guaranteed to be free of singularities. Contrary to fixed-based robots, the singularity curve is not a line. The envelopes resemble the known results for the 2-DOF case, which can be found in the literature [28, 27].

this choice is to highlight the importance of accounting for the dependencies between the elements of the Jacobian in order to conclude its regularity, a feature that only higher order formal methods like Taylor models can provide. The maximum precisions of the algorithms are tuned in order to obtain results of comparable conservativeness. The portion of analyzed search space, denoted by the interval $\mathbf{q}^I_{search}$, is

$$\mathbf{q}^I_{search} = \begin{bmatrix} -\frac{\pi}{2} & \leq & q_1 & \leq & +\frac{\pi}{2} \\ -\frac{\pi}{2} & \leq & q_2 & \leq & +\frac{\pi}{2} \\ -\frac{\pi}{4} & \leq & q_3 & \leq & +\frac{\pi}{4} \end{bmatrix} \text{[rad]},$$

and it is split into an initial set of 2^8 boxes. The computations were performed on a 3.7GHz processor with 6 cores. The resulting S-Maps, projected on the plane q_1, q_2, are depicted in Fig. 2. To notice is that the maps result to be an enclosure of the solutions of 12. The IA-based method required 3300 [sec], while the one based on Taylor models only 225 [sec]. This reflects the fact that Taylor models greatly reduce overestimation of the invertibility of $\boldsymbol{J}^*_m$, thus requiring far fewer bisections. Using Taylor models, the computational time to obtain a map with roughly the same precision is reduced to less than 7% of the time required for IA. Nonetheless, IA has a much simpler implementation, and criterion (15) could straightforwardly handle kinematic redundancy.

B. S-Map computation for a spatial 6-DOF space robot

To compute this S-Map, the IA-based method presented in Section III-A is used together with the heuristic pruning described in Section III-D. We do not present a comparison with Taylor models because they require a performant software implementation which is planned as future work.

For the heuristic, 2 million configurations are randomly generated in the range $[-\pi, +\pi]$ [rad] for each joint. For this case, the portion of analyzed search space is

$$\mathbf{q}^I_{search} = \begin{bmatrix} 0 & \leq & q_1 & \leq & +\frac{\pi}{4} \\ -\frac{\pi}{2} & \leq & q_2 & \leq & -\frac{\pi}{6} \\ +\frac{\pi}{60} & \leq & q_3 & \leq & +\frac{\pi}{2} \\ 0 & \leq & q_4 & \leq & +\frac{\pi}{20} \\ -\frac{\pi}{2} & \leq & q_5 & \leq & -\frac{\pi}{12} \\ & & q_6 & = & 0 \end{bmatrix} \text{[rad]}, \qquad (28)$$

which is initially split into 2^{16} boxes to be checked. The computations were performed on a 2.9GHz processor with 8 cores, considering a maximum precision $\epsilon = 0.0157$ [rad], and taking a total of 170 hours to complete the search. The computational time may seem large, but one must account for the following. First, the curse of dimensionality affects any type of branch and bound algorithm of this kind. Second, we obtained the first S-Maps for a complex free-floating robot. Finally, the algorithm is in fact designed to be performed on a heavily parallelized architecture with an efficient implementation which reduces the computational time by several order of magnitudes. Moreover, given the kinematic and dynamic structure of the space robot, this analysis has to be carried out only once and only for the useful joint ranges. The resulting S-Map defines ≈200k boxes, corresponding to 2.5MBs of compressed data (approximately 18MBs of uncompressed data). Projections of the S-Map are depicted in Fig. 3. The evaluation of the distance function presented in (19)) using this S-Map for random configurations takes 3 [ms] on average, while the detection of an intersection between a segment and the map (condition (22)) takes 30 [ms] in average.

C. Example of trajectory planning using the S-Map

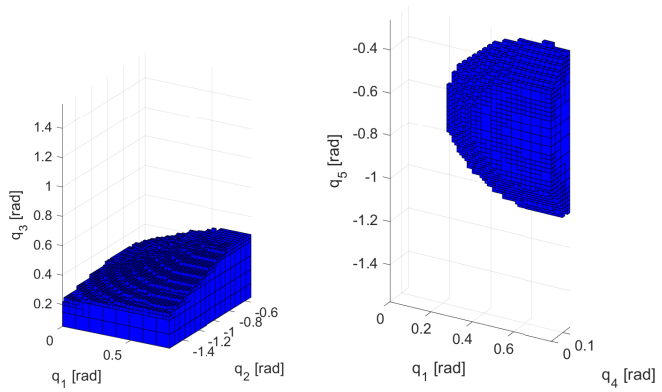
Using the S-Map of the 6-DOF space robot computed in the last section, an example for the algorithm presented in Section IV-C is provided. We consider a problem with cost function $\gamma(\mathbf{q}) = q_3$ to be minimized, and with the constraint presented in (23), which forces the solution to have a minimum distance to the S-Map. For this example, the parameters $\Delta = 0.03$, $\beta^* = 0.15$, $\beta_{min} = 0.001$, and $\beta_{max} = 0.25$ are used. The optimization takes place only for the variables q_1, q_2, and q_3, i.e., the last 3 entries relative to q_4, q_5, and q_6 of the gradient in (27) are forced to be zero. Consider the following initial condition:

$$\mathbf{q}_0 = \left[0.66, -0.72, +1.5, +0.10, -0.5, 0\right]^T \text{[rad]}.$$

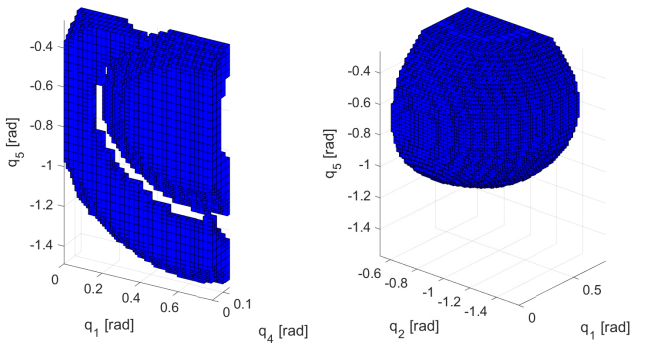
After 11 iterations, the optimizer converges to

$$\mathbf{q}_f = \left[0.00, -1.37, +0.31, +0.10, -0.5, 0\right]^T \text{[rad]}.$$

The optimization steps, together with the corresponding slice of the S-Map, are depicted in Fig. 4. At each iteration, the optimizer performs a step that satisfies the constraints, and that does not violate the boundary of the S-Map. Additionally, Fig. 4 shows that an optimization scheme based on a constraint for the manipulability measure can fail to detect an unfeasible step. Instead, using an S-Map, singularities are avoided.



(a) Slice at $q_4 = 0.10, q_5 = -0.5$. (b) Slice at $q_2 = -1.5, q_3 = 0.27$.



(c) Slice at $q_2 = -1.56, q_3 = 0.23$. (d) Slice at $q_3 = 0.36, q_4 = 0.12$.

Fig. 3. Projections of the S-Map of the spatial 6-DOF robot for the portion of configuration space defined in (28). Note that these figures depict slices of the S-Map of size ± 0.03[rad]. These might not necessarily contain the true singularity surface, which could remain hidden underneath them. The S-Maps represent a conservative envelope of the singularity surfaces. The remaining portion of space is guaranteed to be singularity-free.

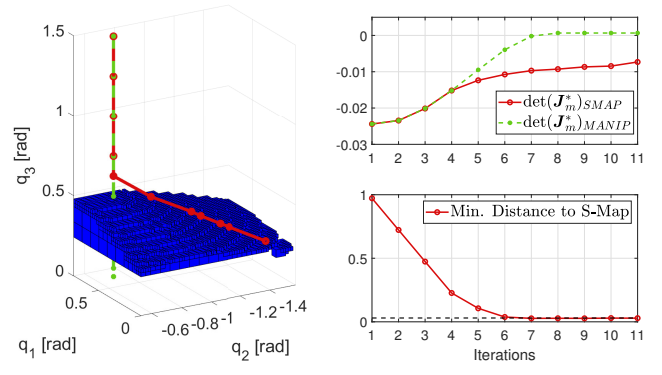


Fig. 4. Minimizing q_3 using the singularity map presented in Fig. 3a as constraint (solid red line). Determinant and distance function shown in the top right and bottom right figures respectively. The dashed black line depicts the required minimum distance. The S-Map is sliced to reveal the steps of the optimization based on a constraint for the manipulability measure (dashed green line), which crosses a singularity (the determinant switches sign), since the steps are performed without global knowledge of the location of all singularities.

VI. Discussion

Formal numerical methods are powerful tools which can find relevant applications in robotics [24]. However, in practice, an efficient application of these methods is not trivial. For example, in the case of free-floating robots, Taylor models are shown to be effective when considering the $\det(\mathbf{M}_{em})$. However, if they were used to compute $\det(\boldsymbol{J}_m^*)$, the resulting algorithm would be inefficient, due to the matrix multiplications necessary to obtain the full matrix in (8), which are especially time consuming for Taylor models. Instead, IA loses effectiveness if applied on $\mathbf{M}_{em}$, since this matrix was found not to be as well-behaved as $\boldsymbol{J}_m^*$, and as such, for the same input ranges, condition 15 is more difficult to be satisfied.

We emphasize that the simple motion planning task in Section IV-C should be interpreted as a typical motion constraint at a via-point along a trajectory [20]. In a general robot trajectory planning setting, such constraint would be applied to each via-point on the trajectory to provide the weak guarantee of feasibility of an NLP (due to the discretization of the optimal control problem). Moreover, the empirical parameter Δ needs to be defined. In order to guarantee feasibility with respect to joint velocities in the vicinity of singularities, a conservative choice for the value might be considered, at the cost of an optimal use of the robot workspace. This choice is justified by the fact that our prime goal is to guarantee feasibility rather than optimality.

Singularity maps may also be effective for online singularity avoidance in free-floating feedback control methods, e.g. [23, 26, 13]. Furthermore, the proposed method may be applicable also for feedback controllers which use base actuation, and whose singularity is determined by $\boldsymbol{J}_m^*$, e.g. [14, 15].

VII. Conclusion and Future Work

We presented a numerical technique to obtain singularity maps of free-floating robots using two different criteria based on formal methods, namely Interval Arithmetic and Taylor models. The results show that the proposed approaches are computationally feasible for a spatial 6-DOF system, and that the computed maps can be used for the scope of trajectory planning with emphasis of feasibility with respect to singularity avoidance. Guarantees are provided in an NLP context. Furthermore, we carried out a comparison of the two criteria, displaying the net superiority of the approach exploiting an alternative formulation of the generalized Jacobian and Taylor models. While this method seems promising, it requires an efficient implementation to express its full potential.

Future research will primarily focus on extending the method to handle kinematic redundancy, and on improving the representation of the singularity sets with a more efficient and smooth enclosure. We expect that, thanks to the global knowledge of the singularity locations, guaranteed singularity-free paths can be planned much more efficiently than with current solutions.

References

[1] S. Abiko, R. Lampariello, and G. Hirzinger. Impedance Control for a Free-Floating Robot in the Grasping of a Tumbling Target with Parameter Uncertainty. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1020–1025, Oct 2006. doi: 10.1109/IROS.2006.281785.

[2] M. Althoff and D. Grebenyuk. Implementation of Interval Arithmetic in CORA 2016. In *Proc. of the 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 91–105, 2016.

[3] M. Althoff, D. Grebenyuk, and N. Kochdumper. Implementation of Taylor models in CORA 2018. In *5th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 54 of *EPiC Series in Computing*, pages 145–173. EasyChair, 2018. doi: 10.29007/zzc7.

[4] R. Benoit, N. Delanoue, S. Lagrange, and P. Wenger. Guaranteed detection of the singularities of 3R robotic manipulators. *Mechanical Sciences*, 7(1): 31–38, 2016. doi: 10.5194/ms-7-31-2016.

[5] M. Berz and G. Hoffstaetter. Computation and Application of Taylor Polynomials with Interval Remainder Bounds. *Reliable Computing*, 4:83–97, February 1998. doi: 10.1023/A:1009958918582.

[6] O. Bohigas, D. Zlatanov, L. Ros, M. Manubens, and J. M. Porta. A General Method for the Numerical Computation of Manipulator Singularity Sets. *IEEE Transactions on Robotics*, 30(2):340–351, April 2014. doi: 10.1109/TRO.2013.2283416.

[7] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone. GuSTO: Guaranteed Sequential Trajectory optimization via Sequential Convex Programming. pages 6741–6747. International Conference on Robotics and Automation (ICRA), 2019. ISBN 978-1-5386-8176-3. doi: 10.1109/ICRA.2019.8794205.

[8] Chong Jin Ong and E. G. Gilbert. Robot path planning with penetration growth distance. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2146–2152 vol.3, May 1994. doi: 10.1109/ROBOT.1994.350965.

[9] F. Cusumano, R. Lampariello, and G. Hirzinger. Development of tele-operation control for a free-floating robot during the grasping of a tumbling target. In *International Conference on Intelligent Manipulation and Grasping*, July 2004.

[10] A. Escande, S. Miossec, M. Benallegue, and A. Kheddar. A Strictly Convex Hull for Computing Proximity Distances With Continuous Gradients. *IEEE Transactions on Robotics*, 30(3):666–678, June 2014. ISSN 1941-0468. doi: 10.1109/TRO.2013.2296332.

[11] G. Garofalo, C. Ott, and A. Albu-Schäffer. On the closed form computation of the dynamic matrices and their differentiations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2364–2359, Nov 2013. doi: 10.1109/IROS.2013.6696688.

[12] C. Gaz, F. Flacco, and A. De Luca. Identifying the dynamic model used by the KUKA LWR: A reverse engineering approach. pages 1386–1392, May 2014. ISSN 1050-4729. doi: 10.1109/ICRA.2014.6907033.

[13] A. M. Giordano, G. Garofalo, M. De Stefano, C. Ott, and A. Albu-Schäffer. Dynamics and control of a free-floating space robot in presence of nonzero linear and angular momenta. In *IEEE 55th Conference on Decision and Control (CDC)*, pages 7527–7534, Dec 2016. doi: 10.1109/CDC.2016.7799432.

[14] A. M. Giordano, G. Garofalo, and A. Albu-Schaffer. Momentum dumping for space robots. In *IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5243–5248, Dec 2017. doi: 10.1109/CDC.2017.8264434.

[15] A. M. Giordano, D. Calzolari, and A. Albu-Schäffer. Workspace fixation for free-floating space robot operations. *IEEE International Conference on Robotics and Automation ICRA*, 2018. doi: 10.1109/icra.2018.8460478. URL https://elib.dlr.de/126555/.

[16] T. Hickey, Q. Ju, and M. H. Van Emden. Interval arithmetic: From principles to implementation. *Journal of the ACM (JACM)*, 48(5):1038–1068, 2001. doi: 10.1145/502102.502106.

[17] O. Khatib and B. Siciliano. *Springer handbook of robotics*. Springer International Publishing, 2016.

[18] J. Kotlarski, R. de Nijs, H. Abdellatif, and B. Heimann. New interval-based approach to determine the guaranteed singularity-free workspace of parallel robots. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1256–1261, May 2009. doi: 10.1109/ROBOT.2009.5152221.

[19] R. Lampariello and G. Hirzinger. Generating feasible trajectories for autonomous on-orbit grasping of spinning debris in a useful time. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5652–5659, Nov 2013. doi: 10.1109/IROS.2013.6697175.

[20] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters. Trajectory planning for optimal robot catching in real-time. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3719–3726, June 2011. doi: 10.1109/ICRA.2011.5980114.

[21] R. Lampariello, H. Mishra, N. Oumer, P. Schmidt, M. De Stefano, and A. Albu-Schäffer. Tracking Control for the Grasping of a Tumbling Satellite With a Free-Floating Robot. 3:3638–3645, 2018. ISSN 2377-3774. doi: 10.1109/LRA.2018.2855799.

[22] C. Luo, I. Safa, and Y. Wang. Approximating Gradients for Meshes and Point Clouds via Diffusion Metric. *Computer Graphics Forum*, 28(5):1497–1508, 2009. doi: 10.1111/j.1467-8659.2009.01526.x.

[23] Y. Masutani, F. Miyazaki, and S. Arimoto. Sensory feedback control for space manipulators. In *Proceedings, International Conference on Robotics and Automation*, pages 1346–1351 vol.3, 1989.

[24] J-P Merlet. Interval analysis and robotics. In *Robotics*

research, pages 147–156. Springer, 2010.

[25] R. E Moore. *Interval analysis*, volume 4. Prentice-Hall Englewood Cliffs, NJ, 1966.

[26] H. Nakanishi and K. Yoshida. Impedance Control for Free-flying Space Robots -Basic Equations and Applications-. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3137–3142, 2006.

[27] K. Nanos and E. Papadopoulos. Avoiding dynamic singularities in Cartesian motions of free-floating manipulators. *IEEE Transactions on Aerospace and Electronic Systems*, 51(3):2305–2318, July 2015. ISSN 2371-9877. doi: 10.1109/TAES.2015.140343.

[28] E. Papadopoulos and S. Dubowsky. *Dynamic Singularities in Free-floating Space Manipulators*, pages 77–100. Springer US, Boston, MA, 1993. ISBN 978-1-4615-3588-1.

[29] E. Papadopoulos, I. Tortopidis, and K. Nanos. Smooth Planning for Free-floating Space Robots Using Polynomials. pages 4272–4277, 2005. ISBN 0-7803-8914-X. doi: 10.1109/ROBOT.2005.1570777.

[30] E. D. Popova. Enclosing the solution set of parametric interval matrix equation A(p)X = B(p). *Numerical Algorithms*, 78(2):423–447, Jun 2018. ISSN 1572-9265. doi: 10.1007/s11075-017-0382-1.

[31] H. G. Rex and J. Rohn. Sufficient Conditions for Regularity and Singularity of Interval Matrices. *SIAM Journal on Matrix Analysis and Applications*, 20, 10 1998. doi: 10.1137/S0895479896310743.

[32] J. Rohn and R. Farhadsefat. Inverse interval matrix: a survey. *Electronic Journal of Linear Algebra*, 22(1):46, 2011.

[33] T. Rybus, J. Lisowski, K. Seweryn, and T. Barciński. Numerical simulations and analytical analyses of the orbital capture manoeuvre as a part of the manipulator-equipped servicing satellite design. In *Proceedings, 17th International Conference on Methods Models in Automation Robotics (MMAR)*, pages 154–159, Aug 2012. doi: 10.1109/MMAR.2012.6347926.

[34] T. Rybus, T. Barciński, J. Lisowski, K. Seweryn, J. Nicolau-Kukliński, J. Grygorczuk, M. Krzewski, K. Skup, T. Szewczyk, and R. Wawrzaszek. Experimental demonstration of singularity avoidance with trajectories based on the Bézier curves for free-floating manipulator. In *9th International Workshop on Robot Motion and Control*, pages 141–146, July 2013. doi: 10.1109/RoMoCo.2013.6614599.

[35] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014. doi: 10.1177/0278364914528132. URL https://doi.org/10.1177/0278364914528132.

[36] S. P. Shary. On full-rank interval matrices. *Numerical Analysis and Applications*, 7(3):241–254, Jul 2014. ISSN 1995-4247. doi: 10.1134/S1995423914030069.

[37] Y. Umetani and K. Yoshida. Resolved motion rate control of space manipulators with generalized Jacobian matrix. *IEEE Transactions on Robotics and Automation*, 5(3):303–314, June 1989. ISSN 2374-958X. doi: 10.1109/70.34766.

[38] Y. Umetani and K. Yoshida. Workspace and Manipulability Analysis of Space Manipulator. *Transactions of the Society of Instrument and Control Engineers*, 26:188–195, 01 1990. doi: 10.9746/sicetr1965.26.188.

[39] A. Werner, R. Lampariello, and C. Ott. Optimization-based generation and experimental validation of optimal walking trajectories for biped robots. pages 4373–4379, 2012. ISBN 978-1-4673-1735-1. doi: 10.1109/IROS.2012.6386154.