Dynamics-based Algorithm for Reliable Assembly Mode Tracking in Parallel Robots

Adrien Koessler[1], Alexandre Goldsztejn[2], Sébastien Briot[2] and Nicolas Bouton[1]

***Abstract*—Finding the current pose of the end-effector of a parallel robot is a problem, since its forward geometric model generally has several solutions. Current methods to address this problem operate mainly under the assumption that the robot never changes its assembly mode nor gets close to Type 2 singularities.**

Nonetheless, recent works proved that a parallel robot can change its assembly mode thanks to dedicated trajectory generation and control. Such a feature allows increasing the operational workspace of such manipulators. Hence tracking correctly the end-effector pose while crossing Type 2 singularities, is mandatory for a practical usage of this workspace enhancement method. However, on Type 2 singularities several solutions of the forward geometric model merge, making current tracking methods ineffective.

To fill this gap, we propose a two-step pose-tracking methodology: First, a differential inclusion based on kinematics and dynamics is solved. Second, joint measurements are used to tighten resulting enclosures. The effectiveness of this method is discussed thanks to experimental data gathered on a planar parallel robot.

***Index Terms*—Parallel Robots, Assembly Modes, Robot Dynamics, Singularity Crossing, Interval Analysis**

I. Introduction

A. Methods for Assembly Mode Change

Parallel robots have gained popularity in industry thanks to recent developments. Compared to serial robots, their low cycle time and high payload-to-weight ratio are acknowledged. Yet, such advantages are balanced with an important drawback: They have a small workspace compared to serial robots, which generally contains singularities.

In particular, Type 2 singularities [1], [2] generally split the workspace into several subsets that can be defined as *generalized aspects* [3] and correspond each to one or several *assembly modes*. In these configurations, the manipulator gains uncontrollable degrees of freedom. Usual motion control strategies like PID regulation do not take this into account and compute unpredictable voltage setpoints, making any attempt at crossing Type 2 singularities hazardous. Hence, aspects cannot be interconnected in a trivial manner. This explains why joint motion range on industrial robots is often software-limited to stay away from Type 2 singularities.

[1] Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, 63000 Clermont-Ferrand, France {Adrien.Koessler, Nicolas.Bouton}@sigma-clermont.fr

[2] Centre National de la Recherche Scientifique (CNRS), Laboratoire des Sciences du Numérique de Nantes (LS2N), UMR CNRS 6004, Nantes, France {Alexandre.Goldsztejn, Sebastien.Briot}@ls2n.fr

Numerous solutions to enlarge the operational workspace of parallel robots have been proposed to solve this problem. They are mainly dependent on the design of the robot:

- Optimal design allows to find the best geometric parameters given a desired workspace volume [4]. Yet, this method does not address workspace splitting.
- Design of singularity-free robots avoids the problem of workspace splitting [5], [6]. Proposed architectures may have other drawbacks such as reduced stiffness.
- Actuation redundancy [7], [8] and modified actuation scheme [9], [10] permit to avoid the impact of singularities, but induce control complexity and supplementary design costs.
- Some architectures do meet geometric requirements for a non-singular assembly mode change, either with cusp points [11], [12] or Type 1 singularities [13], but this is not applicable in the general case.

As opposed to these solutions, researches on Type 2 singularity crossing emerged, as shown by early works [14], [15], [16]. They were continuated by a more recent approach based on optimal trajectory planning and dedicated control. Under the condition that the wrench exerted on a robot's end-effector is purely reciprocal to the directions of uncontrolled motion, it is proven that the dynamic model of the robot does not degenerate even in singularities [17], [18]. This result can be used in computed torque control (CTC) schemes with the following strategy [19]:

1) Plan a crossing trajectory that respects the former condition;
2) Enforce the condition by computing torques from a non-degenerating inverse dynamic model when the singularity is crossed.

This method proved its efficiency in most cases, though the trajectory itself plays a critical role in the success of the assembly mode change. Tracking errors, which are unavoidable, have a strong impact near singular configurations and may lead to a failure in following the desired trajectory. Consequently, in addition to trajectory planning and control, there is a specific need for validation of the assembly mode change, which is tackled by the present paper. The new algorithm presented in this paper intends to catch failures by tracking the pose of the end-effector together with its assembly mode during the Type 2 singularity crossing, so that the robot can operate in an autonomous and safe way.

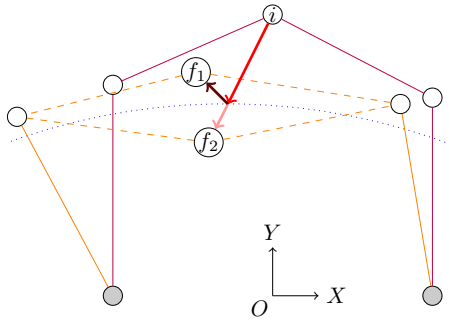


Fig. 1. Ambiguity in assembly mode change. The robot starts in configuration i and reaches a Type 2 singularity (blue dots). Is the final pose really f_2 or rather f_1?

B. Pose and Assembly Mode Tracking Algorithms

It is well known that solving the Forward Kinematic Model (FKM) of a robot is usually problematic, since the solution is not unique in most cases. In the context of assembly mode changing, the robot switches between different solutions to the FKM, see Fig. 1: It passes through a double solution of the FKM, and both trajectories f_1 and f_2 are compatible with the active joint coordinates. The desired trajectory f_2 can be implemented only dynamically relying on the inertia of the robot, and the pose tracking algorithm needs to track the assembly mode as well, i.e., decide which trajectory among f_1 and f_2 is actually performed by the robot.

Several methods based on exteroception already exist to compute the current assembly mode of a parallel robot. Introducing sensors in passive joints to create measurement redundancy has been extensively investigated [20] and is still used for new robot architectures [21]. Drawbacks are the lack of generality in additional sensor placement, impossibility to implement sensors in some joints (eg. spherical) and necessity for mechanical redesign. External measurement with vision [22] can answer partly this problem, for instance by tracking a target placed on the end-effector, but sensor precision is too low to distinguish different assembly modes in the vicinity of singularities, especially if depth estimation is needed. Such an approach comes at an extra cost and require heterogeneous sensor data fusion.

It is therefore desirable to design a pose tracking algorithm, relying only on encoder information, which is robust to assembly mode change. Such pose tracking algorithms solve the FKM for successive active joint measurements $\mathbf{q}_{ak}$ coming from encoders to compute successive poses $\mathbf{x}_k$. Since it generally has several solutions , it is crucial to solve the FKM locally to compute the only solution that corresponds to the actual pose. Usually, either the assembly mode is fixed by the design of the robot and one can use an explicit expression of the correct FKM solution, or one performs a Newton iteration starting at the previous pose $\mathbf{x}_{k-1}$ to solve the Loop Closure Equation (LCE)

$$\mathbf{f}(\mathbf{x}, \mathbf{q}_{ak}) = \mathbf{0}, \tag{1}$$

which is foreseen to converge very quickly to the correct new pose $\mathbf{x}_k$. However, this latter simple and efficient strategy may fail and converge to a wrong solution, in particular in the vicinity of singularities.

Merlet [23] investigated the resolution of the FKM, and proposed a certified pose tracking algorithm based on interval analysis, which operates in two steps. The first step consists in computing a crude interval domain $[\mathbf{x}^C]_k$ for $\mathbf{x}_k$ by solving the following simple update equation, which requires the knowledge of a maximal end-effector velocity v_{max}:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \int_0^{t_s} \dot{\mathbf{x}}(t)\, dt \in [\mathbf{x}]_{k-1} + t_s[\mathbf{v}_{max}] =: [\mathbf{x}^C]_k, \tag{2}$$

where the velocity domain $[\mathbf{v}_{max}]$ encloses all possible velocities by fixing $[\mathbf{v}_{max}] := [-\mathbf{v}_{max}, \mathbf{v}_{max}]$. The second step consists in obtaining a sharp enclosure $[\mathbf{x}]_k$ by solving the LCE (1) using an interval Newton operator and the crude enclosure $[\mathbf{x}^C]_k$ as an initial domain. This pose tracking algorithm has very interesting properties: Provided that a correct velocity upper bound is used, the strict contraction of the interval Newton operator proves that the correct pose has been enclosed. This can be checked numerically, while the computational effort is compatible with real time. However, two drawbacks are related to the present paper.

1) The upper bound $\mathbf{v}_{max}$ on the pose velocity is difficult to compute formally: While the desired pose velocity is often known from the trajectory planing, the impact of tracking errors on the velocity is difficult to assess, in particular in the vicinity of Type 2 singularities.
2) As pointed out in [23], several solutions to the FKP are enclosed in the crude domain $[\mathbf{x}^C]_k$ in the vicinity of Type 2 singularities, which prevents the interval Newton operator to strictly contract and this pose-tracking algorithm to succeed. Hence, FKM solutions cannot be separated anymore once singularity is reached.

Therefore, the pose tracking algorithm proposed in [23] cannot be used to correctly track the pose and the assembly mode when crossing singularities. Furthermore, an underestimated maximal velocity could lead this algorithm to capture the wrong pose in the vicinity of Type 2 singularities, then continuing to track the pose in the wrong assembly mode without any prompt, a behavior that we wish absolutely to avoid.

The second drawback mentioned above was recently addressed by the authors of the present paper in [24] by tracking the end-effector velocity together with its pose:

$$\dot{\mathbf{x}}_k = \dot{\mathbf{x}}_{k-1} + \int_0^{t_s} \ddot{\mathbf{x}}(t)\, dt \in [\dot{\mathbf{x}}]_{k-1} + t_s[\mathbf{a}_{max}] =: [\dot{\mathbf{x}}^C]_k, \tag{3}$$

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \int_0^{t_s} \dot{\mathbf{x}}(t)\, dt \in [\mathbf{x}]_{k-1} + t_s[\dot{\mathbf{x}}^C] =: [\mathbf{x}^C]_k, \tag{4}$$

which now requires an enclosure $[\mathbf{a}_{max}] = [-\mathbf{a}_{max}, \mathbf{a}_{max}]$ on the end-effector acceleration. Using (3) instead of $[\mathbf{v}_{max}]$ describes more accurately the set of admissible velocities, and in particular it provides an information on the direction in which the end-effector is moving. This additional information has been proved to allow separating the different solutions to the FKM when crossing singularities in [24]. As in [23], resulting crude enclosures are sharpened to $[\dot{\mathbf{x}}]_k$ and $[\mathbf{x}]_k$ using the LCE and now the first order Forward Kinematic Model

(FKM1) respectively. It was shown in [24] that for correctly planned trajectories, tracking the velocity using (3) and using $[\dot{\mathbf{x}}^C]$ instead of $[\mathbf{v}_{max}]$ in (4) actually allows tracking the pose and its assembly mode when crossing a singularity. However, requiring an upper bound $\mathbf{a}_{max}$ on the end-effector acceleration presents the same drawbacks as requiring an upper bound on its velocity: It is difficult to assess, while too large overestimation leads the algorithm to diverge and underestimation may lead to an incorrect tracking of the assembly mode. Indeed, a too small acceleration upper bound may incorrectly enforce the pose tracking algorithm to track a wrong trajectory passing smoothly through the singularity with small pose acceleration while the true trajectory reaches the singularity locus but does not cross it, appearing to bounce on the singularity with high end-effector acceleration. Details are not given here since the present paper supersedes this previous work of the authors.

C. Proposed methodology and contributions

In order to avoid requiring an upper-bound on the end-effector acceleration, the proposed pose and assembly mode tracking algorithm computes the crude enclosures of the end-effector velocity and pose by simulating a differential inclusion [25]. This differential inclusion allows computing an enclosure of the end-effector acceleration by using both the 2^{nd}-order Forward Kinematic Model (FKM2), which is accurate away of Type 2 singularities but degenerates in their vicinity, and an adequate Direct Dynamic Model (DDM), which is less accurate but does not degenerate in Type 2 singularities. Due to the diverging nature of differential inclusion resolution along time, at-time step contraction of enclosures should be provided based on encoder measurements. Assembly mode detection would then be achieved if no Type 2 singular configuration belongs to the contracted pose enclosure.

This approach requires both an expression of the direct dynamics of parallel robot and a method to solve differential inclusions. The resolution of the differential inclusion requires enclosures of the joint values, rates, accelerations between time-steps, as well as of the torques and the uncertainties on the DDM. Although meaningful enclosures are available, they cannot be rigorously enclosed. Most importantly however, underestimated enclosures that lead to a wrong assembly mode tracking will now make the differential inclusion inconsistent, hence preventing continuing to track the pose in the wrong assembly mode. This is therefore a definite advantage with respect to the usage of a maximal end-effector acceleration.

This strategy is used to create an algorithm for parallel robot pose estimation without additional sensing, which is the main contribution of the paper and comprises:

- a new expression of the DDM which is shown to be consistent in Type 2 singularities
- the concurrent usage of kinematic and dynamic models to reliably enclose end-effector acceleration even around singularities
- interleaving of the differential inclusion enclosure with the at-time step contraction using encoder information
- theoretical results and practical algorithm for solving differential inclusions
- exhaustive experimental results showing the worth and the limits of the proposed algorithm.

D. Outline of the paper

The article is organized as follows: Kinematic and dynamic models of parallel robots will be presented in Section II. Crucially, a new formulation for the direct dynamic model is given that does not degenerate in Type 2 singularities. Using these models as constraints by the means of differential inclusion techniques, a generic algorithm able to track the pose of the end-effector and its assembly mode is presented in Section III. The methodology is instantiated to a 2-degrees of freedom planar five-bar parallel robot called DexTAR in Section IV. Experiments are carried out on the real robot and the conditions for assembly mode detection success are investigated in Section V.

II. Kinematic and Dynamic Modeling

A. Parallel Robot Kinematic Modeling

In this section, the general equations for kinematic analysis of parallel robots are recalled. They are all demonstrated in [26]. For concision matters, we will only consider fully parallel manipulators, that are n-dof robots driven by n actuators, but the following can be extended to other types of parallel architectures. Kinematic analysis establishes relations between:

- the vector of active joint coordinates $\mathbf{q}_a \in \mathbb{R}^n$ and its derivatives $\dot{\mathbf{q}}_a, \ddot{\mathbf{q}}_a \in \mathbb{R}^n$;
- the vector of end-effector pose coordinates $\mathbf{x} \in \mathbb{R}^n$ and its derivatives $\dot{\mathbf{x}}, \ddot{\mathbf{x}} \in \mathbb{R}^n$;
- the vector of constant geometrical parameters $\boldsymbol{\xi} \in \mathbb{R}^{n_g}$.

We also introduce $\mathbf{q}_d$ which is the vector of passive joint coordinates and its derivatives $\dot{\mathbf{q}}_d$, $\ddot{\mathbf{q}}_d$. They are used further on for dynamic modeling.

Parallel mechanisms usually have a LCE [26] that can be written as

$$\mathbf{f}(\mathbf{x}, \mathbf{q}_a, \boldsymbol{\xi}) = \mathbf{0}. \tag{5}$$

Through derivation of the former, the 1^{st}-order kinematics can be deduced as

$$\mathbf{A}(\mathbf{x}, \mathbf{q}_a, \boldsymbol{\xi})\,\dot{\mathbf{x}} + \mathbf{B}(\mathbf{x}, \mathbf{q}_a, \boldsymbol{\xi})\,\dot{\mathbf{q}}_a = \mathbf{0} \tag{6}$$

where $\mathbf{A}$ and $\mathbf{B}$ are $(n \times n)$ Jacobian matrices of the robot. The kinematic Jacobian, which links joint rates to end-effector velocity, is defined as $\mathbf{J} = -\mathbf{A}^{-1}\mathbf{B}$. Another derivation step leads to the equation for 2^{nd}-order kinematics, that is

$$\mathbf{A}(\mathbf{x}, \mathbf{q}_a, \boldsymbol{\xi})\,\ddot{\mathbf{x}} + \mathbf{B}(\mathbf{x}, \mathbf{q}_a, \boldsymbol{\xi})\,\ddot{\mathbf{q}}_a = \mathbf{b}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{q}_a, \dot{\mathbf{q}}_a, \boldsymbol{\xi}) \tag{7}$$

The two former models can be used to compute respectively $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$. (6) and (7) are the aforementionned FKM1 and FKM2. In Type 2 singular configurations, matrix $\mathbf{A}$ becomes rank-deficient [1] and FKMs are indeed not defined. Therefore, $\ddot{\mathbf{x}}$ cannot be computed and another equation is necessary: the DDM presented in next section.

Briefly speaking, it is also possible to relate the passive joint coordinates to the end-effector pose and active joint coordinates by a relation of the form [27]

$$\mathbf{h}(\mathbf{x}, \mathbf{q}_a, \mathbf{q}_d) = \mathbf{0}. \tag{8}$$

leading to the following kinematic relationship

$$\mathbf{J}_{td}\dot{\mathbf{q}}_d = \mathbf{J}_{tr}\dot{\mathbf{x}} - \mathbf{J}_{ta}\dot{\mathbf{q}}_a \tag{9}$$

where $\mathbf{J}_{td}$ is a $(n \times n_d)$ matrix, $\mathbf{J}_{tr}$ and $\mathbf{J}_{ta}$ $(n \times n)$ matrices. This relation is needed for further developments. The way to obtain them is not recalled here and the reader should refer to [27].

B. Parallel Robot Dynamic Modeling

The goal of this section is to establish a model that allows to compute $\ddot{\mathbf{x}}$ in Type 2 singularities, where FKM2 (7) is not defined. A classic expression of the Direct Dynamic Model of parallel robots computed from input torques $\boldsymbol{\tau}$ can be found in [27] under the form

$$\ddot{\mathbf{x}} = \mathbf{M}_{rob}^{-1}(\mathbf{x})\left(\mathbf{J}_{inv}^T\boldsymbol{\tau} - \mathbf{c}_{rob}(\mathbf{x},\dot{\mathbf{x}})\right), \tag{10}$$

where $\mathbf{M}_{rob}$ is a $(n \times n)$ matrix, $\mathbf{c}_{rob}$ a $(n \times 1)$ vector and $\mathbf{J}_{inv} = -\mathbf{B}^{-1}\mathbf{A}$ is the inverse Jacobian matrix. In (10), all variables related to joint measurements have been substituted by their expression in $\mathbf{x}$, $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$. Thus, expression (10) could be used to compute end-effector acceleration thanks to enclosures on $\mathbf{x}$, $\dot{\mathbf{x}}$ (that are initially deduced from joint measurement). However, this method is prone to overestimation , up to the point that infinite enclosures are computed around Type 2 singularities. Numerical evidence is provided in appendix A. This is caused by the multiple occurrences of pose and velocity intervals. Consequently, this is not suited for our application and we need to express DDM in another way, where possible inputs are active joint values, rates and accelerations.

We start back from the classical Inverse Dynamic Model (IDM) of parallel robots as given in [27]:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_a + \mathbf{J}^T\mathbf{w}_p + \mathbf{J}_d^T\boldsymbol{\tau}_d, \tag{11}$$

where $\mathbf{J}$ is the Jacobian Matrix defined after (6), $\mathbf{J}_d$ is the matrix that links active joint rates to passive joint rates (defined in what follows), $\boldsymbol{\tau}$ is the vector of input torques and $\boldsymbol{\tau}_a$, $\boldsymbol{\tau}_d$, $\mathbf{w}_p$ are defined next.

As proposed in [27], virtual efforts $\boldsymbol{\tau}_a$, $\boldsymbol{\tau}_d$ in the legs are expressed from the Lagrangian of the robot L as a function of active joint values $\mathbf{q}_a$ and rates $\dot{\mathbf{q}}_a$, passive joint values $\mathbf{q}_d$ and rates $\dot{\mathbf{q}}_d$ (of size $(n_d \times 1)$), end-effector pose $\mathbf{x}$ and velocity $\dot{\mathbf{x}}$:

$$\boldsymbol{\tau}_a = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\mathbf{q}}_a}\right)^T - \left(\frac{\partial L}{\partial \mathbf{q}_a}\right)^T = \mathbf{M}_{aax}\ddot{\mathbf{q}}_a + \mathbf{M}_{dax}\ddot{\mathbf{q}}_d + \mathbf{c}_{ax} \tag{12a}$$

$$\boldsymbol{\tau}_d = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\mathbf{q}}_d}\right)^T - \left(\frac{\partial L}{\partial \mathbf{q}_d}\right)^T = \mathbf{M}_{adx}\ddot{\mathbf{q}}_a + \mathbf{M}_{ddx}\ddot{\mathbf{q}}_d + \mathbf{c}_{dx}. \tag{12b}$$

with mass matrices $\mathbf{M}_{aax}$, $\mathbf{M}_{dax}$, $\mathbf{M}_{adx}$, $\mathbf{M}_{ddx}$ and vectors of Coriolis, centrifugal and gravity effects $\mathbf{c}_{ax}$, $\mathbf{c}_{dx}$. Meanwhile, the virtual efforts $\mathbf{w}_p$ exerted on the platform, derived from the Lagrangian L, depend only on end-effector pose parameters and can be written as

$$\mathbf{w}_p = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\mathbf{x}}}\right)^T - \left(\frac{\partial L}{\partial \mathbf{x}}\right)^T = \mathbf{M}_p(\mathbf{x})\ddot{\mathbf{x}} + \mathbf{c}_p(\mathbf{x},\dot{\mathbf{x}}) \tag{13}$$

where $\mathbf{M}_p$ is the mass matrix of the end-effector and $\mathbf{c}_p$ a vector of Coriolis, centrifugal, gravity and friction effects.

We now seek to eliminate the dependance on passive joint variables. The dependance on $\mathbf{q}_d$ can be removed thanks to (8). Using (9) allows to get rid of passive joint velocities in equations, despite matrix $\mathbf{J}_{td}$ not being invertible in some configurations involving Leg Passive Joint Twist Singularities (LPJTS) [2]. In the following, we assume that the robot is not in such a configuration as they are fairly rare and only exist on mechanisms with complex limbs. Passive joint rates are then computed as $\dot{\mathbf{q}}_d = \mathbf{J}_d\dot{\mathbf{q}}_a$ where $\mathbf{J}_d = \mathbf{J}_{td}^{-1}(\mathbf{J}_{tr}\mathbf{J} - \mathbf{J}_{ta})$ is the matrix necessary in equation (9). Since passive joint values can always be computed thanks to active joint values and end-effector pose, all mass matrices and vectors in equations (12) and (13) do not rely on passive joint variables. It is important to notice that those mass matrices are always full-rank [27]. Differentiating the relation (9) with respect to time yields a formula for passive joint accelerations:

$$\ddot{\mathbf{q}}_d = \mathbf{J}_{td}^{-1}\left(\mathbf{J}_{tr}\ddot{\mathbf{x}} - \mathbf{J}_{ta}\ddot{\mathbf{q}}_a + \mathbf{d}_d\right). \tag{14}$$

with $\mathbf{d}_d$ only depending on $\mathbf{x}$, $\dot{\mathbf{x}}$, $\mathbf{q}_a$ and $\dot{\mathbf{q}}_a$ too.

Integrating (12) (where passive accelerations are computed with (14)) and (13) in the IDM (11) yields a result of the form:

$$\boldsymbol{\tau} = \mathbf{M}_x\ddot{\mathbf{x}} + \mathbf{M}_q\ddot{\mathbf{q}}_a + \mathbf{c}_{xq} \tag{15}$$

with

$$\mathbf{M}_x = \mathbf{M}_{dax}\mathbf{J}_{td}^{-1}\mathbf{J}_{tr} + \mathbf{J}^T\mathbf{M}_p + \mathbf{J}_d^T\mathbf{M}_{ddx}\mathbf{J}_{td}^{-1}\mathbf{J}_{tr} \tag{16}$$

$$\mathbf{M}_q = \mathbf{M}_{aax} - \mathbf{M}_{dax}\mathbf{J}_{td}^{-1}\mathbf{J}_{ta} + \mathbf{J}_d^T\mathbf{M}_{adx} - \mathbf{J}_d^T\mathbf{M}_{ddx}\mathbf{J}_{td}^{-1}\mathbf{J}_{ta} \tag{17}$$

$$\mathbf{c}_{xq} = \mathbf{M}_{dax}\mathbf{J}_{td}^{-1}\mathbf{d}_d + \mathbf{c}_{ax} + \mathbf{J}^T\mathbf{c}_p + \mathbf{J}_d^T\left(\mathbf{M}_{ddx}\mathbf{J}_{td}^{-1}\mathbf{d}_d + \mathbf{c}_{dx}\right). \tag{18}$$

The former expressions cannot be evaluated in Type 2 singular configurations because of the kinematic Jacobian $\mathbf{J}$. Left-multiplying them by $\mathbf{J}_{inv}^T = -\mathbf{A}^T\mathbf{B}^{-T}$ allows to get rid of matrix $\mathbf{J}$ in (16) and (18). The resulting expression is

$$\mathbf{J}_{inv}^T\boldsymbol{\tau} = \mathbf{M}_{ix}\ddot{\mathbf{x}} + \mathbf{M}_{iq}\ddot{\mathbf{q}}_a + \mathbf{c}_{ixq}, \tag{19}$$

with

$$\mathbf{M}_{ix} = \mathbf{J}_{inv}^T\mathbf{M}_{dax}\mathbf{J}_{td}^{-1}\mathbf{J}_{tr} + \mathbf{M}_p + \mathbf{J}_{inv}^T\mathbf{J}_d^T\mathbf{M}_{ddx}\mathbf{J}_{td}^{-1}\mathbf{J}_{tr} \tag{20}$$

$$\mathbf{M}_{iq} = \mathbf{J}_{inv}^T\mathbf{M}_{aax} - \mathbf{J}_{inv}^T\mathbf{M}_{dax}\mathbf{J}_{td}^{-1}\mathbf{J}_{ta} + \mathbf{J}_{inv}^T\mathbf{J}_d^T\mathbf{M}_{adx} - \mathbf{J}_{inv}^T\mathbf{J}_d^T\mathbf{M}_{ddx}\mathbf{J}_{td}^{-1}\mathbf{J}_{ta} \tag{21}$$

$$\mathbf{c}_{ixq} = \mathbf{J}_{inv}^T\mathbf{M}_{dax}\mathbf{J}_{td}^{-1}\mathbf{d}_d + \mathbf{J}_{inv}^T\mathbf{c}_{ax} + \mathbf{c}_p + \mathbf{J}_{inv}^T\mathbf{J}_d^T\left(\mathbf{M}_{ddx}\mathbf{J}_{td}^{-1}\mathbf{d}_d + \mathbf{c}_{dx}\right). \tag{22}$$

The terms $\mathbf{M}_{ix}$, $\mathbf{M}_{iq}$ and $\mathbf{c}_{ixq}$ can be computed in Type 2 singular configurations, since none involve the computation of $\mathbf{J}$, i.e. inversion of matrix $\mathbf{A}$.

Consequently, the Direct Dynamic Model (DDM) computing end-effector acceleration can be obtained as

$$\ddot{\mathbf{x}} = \mathbf{M}_{ix}^{-1}\left(\mathbf{J}_{inv}^T\boldsymbol{\tau} - \mathbf{M}_{iq}\ddot{\mathbf{q}}_a - \mathbf{c}_{ixq}\right), \tag{23}$$

which is a generic expression of the Direct Dynamic Model of a parallel manipulator that can be used even in Type 2 singularities. It is noteworthy that given this expression,

DDM (23) cannot be computed in Type 1 singularities, but this is not a problem since FKM2 (7) is well-conditioned in those configurations. When combined, FKM2 and DDM allow to compute the acceleration of the end-effector in every configuration, provided that Type 2 and LPJTS singularities do not coincide.

Invertibility analysis of matrix $\mathbf{M}_{ix}$ is complex because of its structure, so it is out of the scope of this paper. However, this matrix is always invertible for the case-study presented in Section IV.

III. Pose Tracking Algorithm

This section presents the pose tracking algorithm. It is organized as follows: Subsection III-A provides some brief background on interval analysis. Subsection III-B actually describes the the pose tracking algorithm. Finally, Subsection III-C discusses the expected behavior of the algorithm, the uncertainties on its inputs and its overall reliability.

A. Background on Interval Analysis

Interval analysis [28], [29] (IA) relies on interval arithmetic, which is the extension of usual operators like addition, subtraction, multiplication, division and elementary functions like sin, cos, etc., to interval arguments. For example, the addition between two intervals is defined by $[\underline{x},\overline{x}]+[\underline{y},\overline{y}]=[\underline{x}+\underline{y},\overline{x}+\overline{y}]$ and their multiplication by $[\underline{x},\overline{x}]\times[\underline{y},\overline{y}]=[\min\{\underline{x}\underline{y},\overline{x}\underline{y},\underline{x}\overline{y},\overline{x}\overline{y}\},\max\{\underline{x}\underline{y},\overline{x}\underline{y},\underline{x}\overline{y},\overline{x}\overline{y}\}]$. Functions compound of these operations can then be evaluated for interval arguments, allowing the computation of image intervals that contain the range of the function over the interval arguments. For example, the interval evaluation $[x]^2-[x]$ for the interval $[x]=[-1,1]$ is $[-1,1]^2-[-1,1]=[-1,2]$, which is a superset of the corresponding function range $\{x^2-x : x\in[-1,1]\}=[-\frac{1}{4},2]$. Vector and matrix operations are also extended accordingly, e.g., $[\mathbf{x}]^T[\mathbf{y}]\supseteq\{\mathbf{x}^T\mathbf{y} : x\in[\mathbf{x}],y\in[\mathbf{y}]\}$.

IA has been used in a wide range of applications in robotics, e.g., mobile robot localization [30], trajectory planning [31], continuous solution of the inverse geometric model [32], robust control [33], [34], parameter estimation [35], [36], tolerance analysis and synthesis [37], [38], [39] or workspace computation [40], [41].

In our context, IA is used to:

- Enclose the trajectories of a differential inclusion in Subsection III-B1. The detailed description of this algorithm is given in Appendix B.
- Contract the pose and velocity domains for given joint coordinates domains using the geometric model of the robot in Subsection III-B2. To this end, numerical constraint programming techniques are used to solve the systems of equations of the LCE (5) and FKM1 (6). Contraction of pose coordinates intervals is ensured by the composition of a classical Forward-Backward method, based on the classical algorithm HC4revise [42]) and a polytope contractor based rigorous linearizations called X-Taylor [43]. Contractions of velocity enclosures is obtained using classical Gauss-Seidel iteration (see [29], chap. 7).

B. Description of the algorithm

The algorithm inputs are:

- An enclosure of the initial pose coordinate $[\mathbf{x}]_0$;
- Joint coordinates measurements $[\mathbf{q}_a]_k$ at time-steps $t_k=kt_s$.

The proposed pose tracking algorithm works in two steps: First, a differential inclusion that models the dynamic of the robot is simulated between the last time step t_{k-1} and the current time-step t_k, leading to a crude enclosure $[\mathbf{x}]_{k-1,k}$ of the pose coordinates between time-steps t_{k-1} and t_k, and a crude enclosure $[\mathbf{x}^C]_k$ at time t_k. This process is described in Subsection III-B1. Second, the crude enclosure $[\mathbf{x}^C]_k$ is contracted using numerical constraint programming by solving the LCE (5) for the joint coordinates measurement $[\mathbf{q}_a]_k$, leading to a sharp pose coordinates enclosure $[\mathbf{x}]_k$ at time-step t_k, see Subsection III-B2.

1) A differential inclusion for the between-time-step integration of the pose coordinates: The two models FKM2 (7) and DDM (23) define two ordinary differential equations

$$\ddot{\mathbf{x}}(t) = \text{FKM2}(\dot{\mathbf{x}}(t),\mathbf{x}(t),\mathbf{q}_a(t),\dot{\mathbf{q}}_a(t),\ddot{\mathbf{q}}_a(t)) \text{ and } (24)$$

$$\ddot{\mathbf{x}}(t) = \text{DDM}(\dot{\mathbf{x}}(t),\mathbf{x}(t),\mathbf{q}_a(t),\dot{\mathbf{q}}_a(t),\ddot{\mathbf{q}}_a(t),\boldsymbol{\tau}(t)). \quad (25)$$

Since expressions of $\mathbf{q}_a(t)$, $\dot{\mathbf{q}}_a(t)$ and $\ddot{\mathbf{q}}_a(t)$, $\boldsymbol{\tau}(t)$ are not available, they are replaced by interval domains giving rise to two differential inclusions (DI)

$$\ddot{\mathbf{x}}(t) \in \text{FKM2}(\dot{\mathbf{x}}(t),\mathbf{x}(t),[\mathbf{q}_a],[\dot{\mathbf{q}}_a],[\ddot{\mathbf{q}}_a]) \text{ and } (26)$$

$$\ddot{\mathbf{x}}(t) \in \text{DDM}(\dot{\mathbf{x}}(t),\mathbf{x}(t),[\mathbf{q}_a],[\dot{\mathbf{q}}_a],[\ddot{\mathbf{q}}_a],[\boldsymbol{\tau}]). \quad (27)$$

Since $\ddot{\mathbf{x}}$ needs to belong to these two DIs, we obtain a single one by intersecting them two:

$$\ddot{\mathbf{x}}(t)\in[\text{FKM2}](\mathbf{x}(t),\dot{\mathbf{x}}(t))\cap[\text{DDM}](\mathbf{x}(t),\dot{\mathbf{x}}(t)), \quad (28)$$

where parameter domains are dropped for compactness. This has several advantages: In Type 1 singularities FKM2 becomes unbounded but DDM remains bounded, while in Type 2 singularities DDM becomes unbounded but FKM2 remains bounded. Therefore performing their intersection allows obtaining a meaningful DI in both cases (in Type 3 singularity however the DI (28) is unbounded and therefore useless). Furthermore, far from Type 1 singularities FKM2 is more accurate than DDM allowing computing a sharper between-time-step pose coordinates enclosure. This process allows taking advantage of both models, without relying on a threshholded multi-model that would switch between the two models. Finally, emptiness of the intersection indicates the two models are inconsistent so the pose and assembly mode were not tracked correctly. This means that either one of the models has not been written correctly, or that measurement and parameter uncertainty was not set in a reliable fashion, which will be the subject of section III-C2.

We use this DI to enclose the pose coordinates between the time-steps t_{k-1} and t_k: Using initial domains $[\mathbf{x}]_{k-1}\ni\mathbf{x}(t_{k-1})$ and $[\dot{\mathbf{x}}]_{k-1}\ni\dot{\mathbf{x}}(t_{k-1})$ and parameter domains $[\mathbf{q}_a]_{k-1,k}\ni\mathbf{q}_a(t)$, $[\dot{\mathbf{q}}_a]_{k-1,k}\ni\dot{\mathbf{q}}_a(t)$, $[\ddot{\mathbf{q}}_a]_{k-1,k}\ni\ddot{\mathbf{q}}_a(t)$ and $[\boldsymbol{\tau}]_{k-1,k}\ni\boldsymbol{\tau}(t)$ for all $t\in[t_{k-1},t_k]$, the aim is to use the DI (28) to compute domains $[\mathbf{x}]_{k-1,k}\ni\mathbf{x}(t)$, $[\dot{\mathbf{x}}]_{k-1,k}\ni\dot{\mathbf{x}}(t)$ and $[\ddot{\mathbf{x}}]_{k-1,k}\ni\ddot{\mathbf{x}}(t)$ for all

$t \in [t_{k-1}, t_k]$. Such trajectories of differential inclusions cannot be enclosed by usual interval methods dedicated to ODEs, which rely on automatic differentiation of the ODE, while DI cannot be differentiated. Up to our knowledge, the only attempt to enclose DI trajectories is [44], but it is dedicated to the longtime simulation of differential inclusion with small uncertainties and the crude enclosure step needed here is not actually provided. We propose Algorithm 1 in Appendix B, which relies on rigorously bounded first order Taylor series given in Theorem 1 to enclose DI trajectories. In addition to compute the between-time-step crude enclosure $[\mathbf{x}]_{k-1,k}$, a tighter crude enclosure $[\mathbf{x}^C]_k$ for the pose coordinate $\mathbf{x}(t_k)$ is computed, which is used in the next subsection.

2) At-time-step contraction of pose and velocity enclosures: The crude enclosure $[\mathbf{x}^C]_k$ of the pose coordinates computed using the DI (28) contains the true pose coordinate $\mathbf{x}(t_k)$. Therefore, the pose coordinate $\mathbf{x}(t_k)$ satisfies

$$\mathbf{x}(t_k) \in [\mathbf{x}^C]_k, \tag{29}$$

$$\exists \mathbf{q}_a \in [\mathbf{q}_a]_k, \exists \boldsymbol{\xi} \in [\boldsymbol{\xi}], \mathbf{f}(\mathbf{x}(t_k), \mathbf{q}_a, \boldsymbol{\xi}) = 0, \tag{30}$$

where (30) is the LCE (1). This is a square system of equations with variables $\mathbf{x}(t_k)$ and uncertain parameters $\mathbf{q}_a$. Provided that both the variable domain $[\mathbf{x}^C]_k$ and the parameter domain $[\mathbf{q}]_k$ are small enough, this system can be solved efficiently by standard contracting techniques borrowed to numerical constraint programming. In particular, the joint coordinates measurement $[\mathbf{q}_a]_k$ is expected to be very accurate, and provided that $[\mathbf{x}^C]_k$ is accurate enough to contain only one solution of the direct kinematic problem for each $\mathbf{q}_a \in [\mathbf{q}_a]_k$, solving the constraint problem (29) and (30) allows reducing the domain $[\mathbf{x}^C]_k$ to a new domain $[\mathbf{x}]_k$. The computed domain $[\mathbf{x}]_k$ is a sharp enclosure of the pose coordinates corresponding to the measured joint coordinates $[\mathbf{q}_a]_k$, independently of both the uncertainties of the DI and the pessimism of the trajectory enclosing algorithm. Velocity crude enclosure $[\dot{\mathbf{x}}^C]_k$ is sharpened in a similar fashion using the following constraints:

$$\dot{\mathbf{x}}(t_k) \in [\dot{\mathbf{x}}^C]_k, \tag{31}$$

$$\exists \mathbf{q}_a \in [\mathbf{q}_a]_k, \exists \dot{\mathbf{q}}_a \in [\dot{\mathbf{q}}_a]_k, \exists \boldsymbol{\xi} \in [\boldsymbol{\xi}],$$
$$\mathbf{A}(\mathbf{x}(t_k), \mathbf{q}_a, \boldsymbol{\xi})\, \dot{\mathbf{x}}(t_k) + \mathbf{B}(\mathbf{x}(t_k), \mathbf{q}_a, \boldsymbol{\xi})\, \dot{\mathbf{q}}_a = \mathbf{0}, \tag{32}$$

where (32) is the FKM1 (6). Solving the problem (31) and (32) allows reducing the domain $[\dot{\mathbf{x}}^C]_k$ to a new domain denoted $[\dot{\mathbf{x}}]_k$.

C. Overall reliability of the algorithm

1) Expected behavior of the algorithm: As explained previously, the DI (28) contains overestimated uncertainties, and the long-term enclosures $[\mathbf{x}]_{k-1,k}$ and $[\dot{\mathbf{x}}]_{k-1,k}$ of its solutions by an interval algorithm shall diverge. Preventing the enclosures to diverge relies on the contraction performed by solving the problem (29)-(32) using joint coordinates measurements $[\mathbf{q}_a]_k$ and joint velocities measurement $[\dot{\mathbf{q}}_a]_k$. Far from Type 2 singularities, the contraction process is efficient and can compensate the divergence of the DI enclosure. However, this contraction process becomes inoperative in the vicinity of Type 2 singularities since FKM1 is badly conditioned. As a consequence, the computed enclosures $[\mathbf{x}]_k$, $[\dot{\mathbf{x}}]_k$ will result only in the DI enclosing algorithm and will diverge during the Type 2 singularity crossing. When the robot leaves this area, two cases arise:

- The enclosures $[\mathbf{x}]_k$, relying only on the DI during the singularity crossing, have not diverged too much, so that numerical constraint techniques can solve (29) and (30) efficiently when leaving the vicinity of Type 2 singularities. In this case, the tracking algorithm succeeds in tracking the pose during the singularity crossing, and will continue tracking the pose correctly.
- The enclosures $[\mathbf{x}]_k$ have diverged too much and numerical constraint techniques cannot solve (29) and (30) efficiently. Then the enclosure will keep growing even when leaving the vicinity of the Type 2 singularities and their width will quickly grow to infinity hence the failure of the pose tracking algorithm.

Therefore, the success of the pose tracking algorithm is expected to depend on the time spent in the vicinity of the singularity, and therefore on the velocity of the platform during the singularity crossing. This is verified experimentally in Section V.

2) Errors on models and measurements: Both the kinematic model and the dynamic model are used in the algorithm. Enclosures must be given on kinematic parameters $\boldsymbol{\xi}$ and dynamic parameters $\boldsymbol{\psi}$, taking into account both uncertainties on real parameters values and uncertainties due to unmodeled phenomena. Also, in order to enclose trajectories of the DI, enclosures of $\mathbf{q}_a$, $\dot{\mathbf{q}}_a$, $\ddot{\mathbf{q}}_a$ and $\boldsymbol{\tau}$ between two time steps are needed.

Interval methods for kinematic calibration have been investigated [36], [45] and give reliable enclosures $[\boldsymbol{\xi}]$ of the kinematic parameters. In the context of singularity crossing, it is important to use near-singularity configurations in the process. This ensures that the kinematic model to be used is also valid around the singularity, where clearances usually have an impact on positioning.

Instead of searching for an enclosure of the dynamic parameters $\boldsymbol{\psi}$, which seems unreliable and difficult to validate as evidenced by [46], a global uncertainty $\boldsymbol{\varepsilon}_{\boldsymbol{\tau}}$ on the value of $\boldsymbol{\tau}$ is used, which also accounts for unmodeled phenomena. The choice of the bound $[\boldsymbol{\varepsilon}_{\boldsymbol{\tau}}]$ of this uncertainty is heuristic and based on the expert's knowledge of robot dynamics. It is typically a percentage of the maximal admissible torque τ_{max}. Its impact will be assessed experimentally in Section V.

For most encoders, an error bound is available for the measure at time steps: $\mathbf{q}_a(t_k)$ is bounded to belong to $\mathbf{q}_{ak} + [\boldsymbol{\varepsilon}_{\mathbf{q}}] =: [\mathbf{q}_a]_k$, where $\mathbf{q}_{ak}$ is the output of the encoder at time-step k. Enclosures of $\mathbf{q}_a$, $\dot{\mathbf{q}}_a$ and $\ddot{\mathbf{q}}_a$ between two time steps are not readily available. Computing such meaningful enclosures from the measurements $[\mathbf{q}_a]_k$ relies on the hypothesis that the signal $\mathbf{q}_a(t)$ is correctly sampled, i.e., that the sampling frequency is sensibly higher than the signal bandwidth. Under this hypothesis, one can use

$$\mathbf{q}_a(t) \in \square([\mathbf{q}_a]_{k-1} \cup [\mathbf{q}_a]_k) \tag{33}$$

where $\square$ is the interval hull operator. Finite differences provide meaningful domains for the between-time-step joint velocity and acceleration:

$$\dot{\mathbf{q}}_a(t) \in \frac{[\mathbf{q}_a]_{k+1}-[\mathbf{q}_a]_{k-1}}{2t_s} = \frac{\mathbf{q}_{ak+1}-\mathbf{q}_{ak-1}}{2t_s} + [\boldsymbol{\varepsilon}_{dq}], \quad (34)$$

$$\ddot{\mathbf{q}}_a(t) \in \frac{[\mathbf{q}_a]_{k+1}-2[\mathbf{q}_a]_k+[\mathbf{q}_a]_{k-1}}{t_s^2} = \frac{\mathbf{q}_{ak+1}-2\mathbf{q}_{ak}+\mathbf{q}_{ak-1}}{t_s^2} + [\boldsymbol{\varepsilon}_{ddq}], \quad (35)$$

with $[\boldsymbol{\varepsilon}_{dq}] = \frac{2[\boldsymbol{\varepsilon}_{\mathbf{q}}]}{2t_s}$ and $[\boldsymbol{\varepsilon}_{ddq}] = \frac{4[\boldsymbol{\varepsilon}_{\mathbf{q}}]}{t_s^2}$. By the mean-value theorem, these intervals actually contain at least one value of the joint velocity and acceleration. Under the correct sampling hypothesis, they shall be domain for the between-time-steps quantities.

The sampling frequency, the controller frequency and the pose-tracking algorithm frequency may differ. In particular, for a fixed encoder precision $[\boldsymbol{\varepsilon}_{\mathbf{q}}]$, a too high sampling frequency makes the finite difference errors $[\boldsymbol{\varepsilon}_{dq}]$ and $[\boldsymbol{\varepsilon}_{ddq}]$ too large. By reducing the pose-tracking algorithm frequency, under the constraint that it remains sensibly larger than the bandwidth of $\mathbf{q}(t)$, one will decrease the finite difference errors. Furthermore, such a reduced pose-tracking frequency allows computing these finite differences on a sliding window, hence providing a more reliable enclosure of the between-time-step joint velocity and acceleration.

3) Success and failure of the algorithm: As discussed previously, enclosures of the pose coordinates may either converge or diverge after crossing the singularity locus. The divergence of the enclosures indicates that the pose could not be tracked correctly and therefore that the robot task should be halted. If error bounds on the dynamical model and the between-time-step joint coordinates are under-estimated then the pose tracking algorithm may identify an inconsistency and return an empty set for the pose enclosure. In this situation, error bounds may be increased, or the robot task halted.

Most importantly, the algorithm should not result in a false positive tracking, where the enclosures converge to a wrong assembly mode. Such a false positive output would imply that error bounds have been under-estimated. However, under-estimated errors and a wrong assembly mode shall make the dynamical and kinematic models, which are merged in the DI (28), inconsistent hence leading to an empty-set evaluation of the DI and to the failure of the algorithm. The situation where under-estimated errors make the DI (28) compatible with a wrong assembly mode but incompatible with the correct assembly mode, which is the only situation that would lead to a false positive tracking, cannot happen if the dynamic model has been correctly designed.

IV. Definition of the Benchmark: Planar Five-Bar Mechanism

To carry out simulations and experiments, the case of a five-bar planar parallel mechanism is used. Its workspace is struck by Type 2 singularities. The robot used is the DexTAR, which is manufactured by the company *Mecademic*. The design of the robot, which is optimal in regards to reachable workspace and nonsingular assembly mode change, is described in [4].

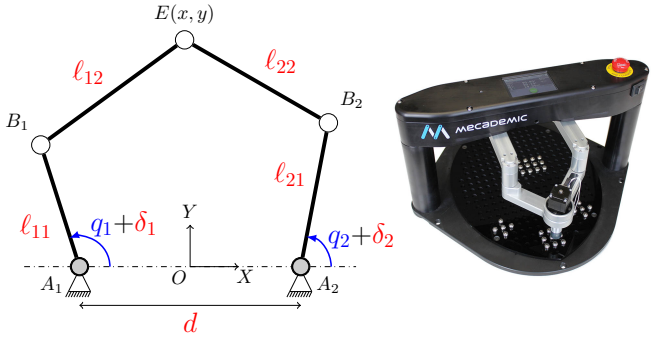


Fig. 2. DexTAR geometry

TABLE I
Geometrical Parameters Values

ξ_i	unit	$[\xi_i]$	nominal
ℓ_{11}	mm	[89.902,90.068]	90
ℓ_{21}	mm	[89.921,90.091]	90
ℓ_{12}	mm	[90.019,90.201]	90
ℓ_{22}	mm	[89.899,90.059]	90
d	mm	[117.956,118.192]	118
δ_1	rad	[-0.004,0.004]	0
δ_2	rad	[-0.004,0.004]	0

A. Robot Kinematic Analysis

The geometry of the robot is described in Fig. 2, alongside with a picture of the DexTAR. The two motors are located at points A_1 and A_2, with active joint inputs $\mathbf{q}_a = (q_1\, q_2)^T$. Seven geometrical parameters are taken into account to model the geometry, that is five bar lengths ℓ_{11}, ℓ_{12}, ℓ_{21}, ℓ_{22}, d and two angular offsets δ_1, δ_2 added for the needs of kinematic calibration. Those seven parameters are grouped in the vector $\boldsymbol{\xi}$. The output is the pose $\mathbf{x} = (x\, y)^T$ of the end-effector E.

The system of loop-closure equations (36) is $\mathbf{f}(\mathbf{x}, \mathbf{q}_a) = (f_1\, f_2)^T$, where for $i = 1, 2$:

$$f_i = (x \pm \frac{d}{2} - \ell_{i1}\cos(q_i))^2 + (y - \ell_{i1}\sin(q_i))^2 - \ell_{i2}^2 = 0. \quad (36)$$

Matrices $\mathbf{A}$ and $\mathbf{B}$ to be used in FKM1, FKM2 and DDM can be formally expressed using the relations

$$\mathbf{A} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right) \quad \text{and} \quad \mathbf{B} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{q}_a}\right). \quad (37)$$

The DexTAR is calibrated using a laser tracker. 50 optimal calibration poses were chosen using the adapted DETMAX algorithm described in [47]. Using a least-square routine [48], an estimate $\hat{\xi}_i$ of each geometric parameter is computed, as well as associated standard deviation $\hat{\sigma}_i$ assuming that the measurement noise is white.

For every parameter ξ_i, the interval $[\xi_i] = [\hat{\xi}_i - 5\hat{\sigma}_i; \hat{\xi}_i + 5\hat{\sigma}_i]$ was consequently set. It was remarked by Daney [36] that using an interval FK solver allows for verification of the intervals $[\xi_i]$. If the loop-closure equations (36) are verified, the solver never returns an empty set, thus bracketings on kinematic parameters are correct. We checked that it was never the case (in singular configurations especially), what accounts for validation of the parameters presented in Table I.

TABLE II
DYNAMIC PARAMETERS VALUES

ψ_i	unit	apparent value
zz_{11}	kg·m^2	$1.34{\cdot}10^{-2}$
zz_{21}	kg·m^2	$1.42{\cdot}10^{-2}$
m_p	kg	$5.24{\cdot}10^{-1}$
f_{s11}	N·m	$5.34{\cdot}10^{-1}$
f_{s21}	N·m	$5.55{\cdot}10^{-1}$
off_{11}	N·m	$5.19{\cdot}10^{-2}$
off_{21}	N·m	$5.48{\cdot}10^{-2}$

B. Robot Dynamic Analysis

Contrary to kinematic analysis where a full model was used to describe the robot, we seek a simplified version of the DDM, but known to capture the main features of the dynamics. For the DexTAR, only seven parameters can be identified [49], forming the vector $\boldsymbol{\psi}$ whose components are:

- zz_{11} and zz_{21}, apparent lumped inertias of proximal elements around axes $(A_1,\vec{Z})$ and $(A_2,\vec{Z})$ respectively
- m_p, apparent lumped mass of the end-effector at point E
- f_{s11}, f_{s21}, apparent coefficients of Coulomb friction in the active joints
- off_{11} and off_{21}, apparent offset terms on motor torques

Consequently, the IDM (11) of the robot can be written as given in [50]:

$$\boldsymbol{\tau} = \mathbf{ZZ}\ddot{\mathbf{q}}_a + \mathbf{F}_s \text{sign}(\dot{\mathbf{q}}_a) + \mathbf{Off} + m_p \mathbf{J}^T \ddot{\mathbf{x}} \tag{38}$$

with $\mathbf{ZZ} = \text{diag}(zz_{11}, zz_{21})$, $\mathbf{F}_s = \text{diag}(f_{s11}, f_{s21})$ and $\mathbf{Off} = \text{diag}(off_{11}, off_{21})$. Multiplying by $\mathbf{J}_{inv}^T = -\mathbf{B}^{-1}\mathbf{A}$ and rearranging the terms gives the expression to be used for the DDM (23), which is well-defined in Type 2 singularities:

$$\ddot{\mathbf{x}} = \frac{1}{m_p}\mathbf{J}_{inv}^T(\boldsymbol{\tau} - \mathbf{ZZ}\ddot{\mathbf{q}}_a - \mathbf{F}_s \text{sign}(\dot{\mathbf{q}}_a) - \mathbf{Off}) \tag{39}$$

The values of the parameters $\boldsymbol{\psi}$ are given in Table II. It is interesting to notice that computation of passive joint values, velocities or accelerations is not needed.

To translate this identified model into a certified one, the approach presented in section III-C2 is used with a global uncertainty $\boldsymbol{\varepsilon}_{\boldsymbol{\tau}}$ set via trial-and-error is added to the input torque $\boldsymbol{\tau}$.

C. Trajectory Generation and Control

As the goal is to perform Type 2 singularity crossing, dedicated algorithms for trajectory generation and control are needed. It was shown that the IDM does not degenerate in Type 2 singularities if the following criterion is respected: the wrench exerted on the end-effector must be reciprocal to the direction of the uncontrolled motion of the end-effector [17]. This constraint was integrated in a dedicated trajectory generator [19].

Since the computation of torque setpoints is the problem around Type 2 singularity, a multi-model Computed Torque Control scheme has been proposed by Pagis [19]. It uses the full IDM away from Type 2 singularities but a modified, nondegenerating version of the IDM near them, in order to ensure the criterion is respected. This control scheme proved reliable, but as said in the introduction, the tracking errors can lead to crossing failure on particular trajectories. Further improvements to this control scheme can be found in [51], [52].

Without notable modification, it is implemented on our DexTAR through a *Matlab/Simulink* development environment and a *Quanser Q-PID* board.

D. Setting up Uncertainties and Intervals

The subject of this section is finding numerical values for the different uncertainties that are used to create the intervals needed by the tracking algorithm. Uncertainties on geometrical parameters $\boldsymbol{\xi}$ were already given in section IV-A.

The initial guesses on pose and velocity are not problematic as long as the motion starts in a configuration that is not near singularities. The homing process of the DexTAR allows setting initial guesses for pose and velocity enclosures rigorously. Expected homing pose is $\mathbf{x}_0 = (0.09\ 0.068)$ (m) with null velocity. Intervals $[\mathbf{x}]_0 = [0.08, 0.10] \times [0.06, 0.08]$ (m) and $[\dot{\mathbf{x}}]_0 = [-0.1, 0.1] \times [-0.1, 0.1]$ (m/s) are set consequently.

To obtain a very fine sampling, which benefits to the differential inclusion procedure, sample time t_s is set to 4 ms. The cutoff frequency of the controller is f_1=15.9 Hz, which is an order of magnitude smaller than the sample rate (250 Hz). This should allow us to use the hull union approximation between two time steps, as explained in Section III-C2.

Joint values are measured using optical encoders with 34,000 counts per revolution on the DexTAR, which means that $[\boldsymbol{\varepsilon}_{\mathbf{q}}]$ can be set accordingly to $[-\frac{2\pi}{34000}, \frac{2\pi}{34000}]^2$ (rad).

Using formulas given in section III-C2, the uncertainty on joint velocity amounts to $[\boldsymbol{\varepsilon}_{dq}] \subset [-9.2 \cdot 10^{-2}, 9.2 \cdot 10^{-2}]^2$ (rad/s), while those on acceleration measurement are $[\boldsymbol{\varepsilon}_{ddq}] \subset [-46.2, 46.2]^2$ (rad/s^2), which is still an order of magnitude smaller than the acceleration itself. Thus this way of computing the uncertainty is relevant for acceleration measurements.

Overall, $\boldsymbol{\varepsilon}_{\boldsymbol{\tau}}$ is tricky to determine formally. As a workaround, we propose to test several values for this uncertainty that amount to a percentage of the maximal torque $\tau_{max} = \pm 8$ Nm. The maximal relative error on torque computation in the identification process is typically 15% of the torque amplitude attained during the identification trajectory, but maximal torque values are never reached during this process.

V. EXPERIMENTS ON DEXTAR ROBOT

To analyze the output and the behavior of the algorithm, experiments need to be carried out on DexTAR robot. The purpose is not only to check if it succeeds in detecting assembly mode change, but also to prove that the method we used to set uncertainties is viable, while keeping the computations reliable. All algorithms are implemented in C++ using *Filib++* for low-level computations and *IBEX* for contractor programming [53].

A. Behaviour of the Algorithm

The first experiment is carried out on the robot and consists in analyzing a case of successful detection. The desired

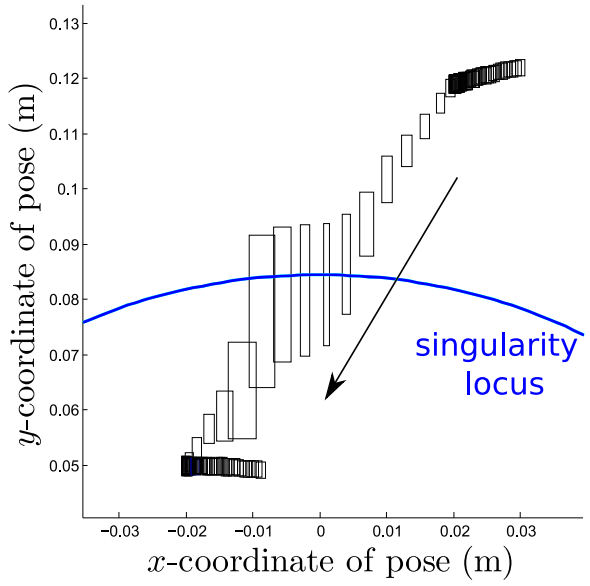


Fig. 3. Drawing of the enclosures on end-effector pose along a crossing trajectory. The blue curve is the locus of Type 2 singularities.

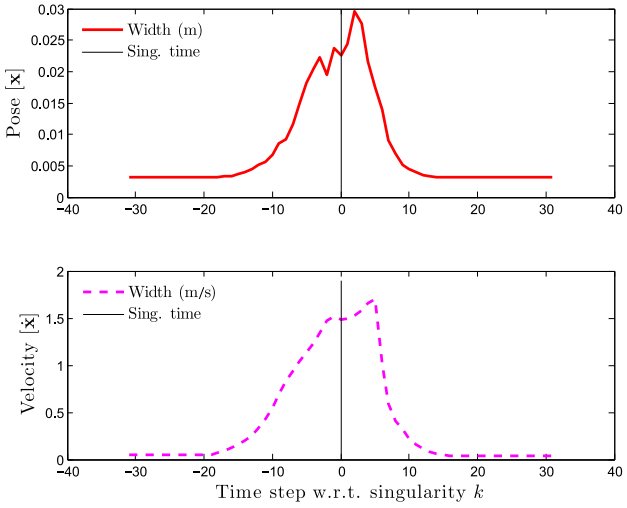


Fig. 4. Pose and velocity intervals width

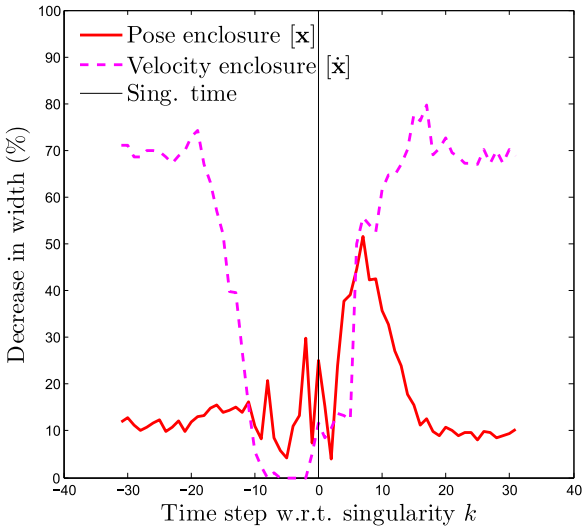


Fig. 5. Contraction efficiency along the crossing trajectory

trajectory consists in crossing the singularity locus at the desired point $\mathbf{x}_s = (0, 84.4)^T$ (mm), with an angle of 60 deg to it and with desired velocity $\dot{\mathbf{x}}_s = (0.390, 0.675)^T$ (m/s). It has been remarked that such a trajectory usually never fails in changing the assembly mode, since crossing velocity and angle are good. Of course, measurement is performed on a trajectory that effectively changes the assembly mode. As a first attempt, uncertainty $\boldsymbol{\varepsilon}_{\boldsymbol{\tau}}$ is set to $\pm 10\%$ of τ_{max}.

Resulting boxes on end-effector pose are represented on Fig. 3, evolving from top to bottom. It can be seen that their size is quite small far from the singularities, and grows afterwards. During the crossing, some boxes fall in both assembly modes, showing that the algorithm can only yield its result on assembly mode detection with a delay. Eventually, boxes shrink and belong to only one side of the singularity, translating the fact that the final assembly mode is known.

Width of pose and velocity boxes during the crossing phase can be seen on Fig. 4. Enclosures on both $\mathbf{x}$ and $\dot{\mathbf{x}}$ are growing near the singularity, then shrinking happens suddenly. Growth of intervals was expected since contractors should work poorly in this zone.

Efficiency of the contraction phase is the subject of Fig. 5. We propose to measure it as the relative difference in box width before and after contraction. While some gains can be obtained on the precision of $[\mathbf{x}]$ near the singularity, the Gauss-Seidel method used to contract $[\dot{\mathbf{x}}]$ is of no use since the FKM1 is not well defined. For pose boxes, a peak efficiency is reached shortly after the crossing, corresponding to brutal contraction of the enclosures. It results in a better definition of the FKM1, allowing efficient velocity contraction again.

B. Uncertainties and Operating Range

While being successful, results obtained in last section are not fully reliable since it cannot be stated if the error on dynamic modeling did truly belong to uncertainty $\boldsymbol{\varepsilon}_{\boldsymbol{\tau}}$. Sadly, we have no means to verify this statement, and can only see the influence of the values set for uncertainties. We would also like to run the algorithm on different trajectories. Consequently, we introduce two parameters characterizing the desired velocity at singularity: its norm v_{sing} and its angle with the singularity locus α (see Fig. 6). For each trajectory, the algorithm is run with varying uncertainty levels on $\boldsymbol{\tau}$ from 1% to 30%. When recording the trajectories, all proved to change efficiently the assembly mode of the robot.

Results are presented in Table III. A checkmark (✓) denotes a success in final assembly mode detection, a cross (×) denotes a failure caused by diverging enclosures and an empty sign (∅) denotes a failure due to empty set computation. These cases have been discussed in section III-C3. It can be seen that slow crossings tend to result in detection failure, while higher velocities tend to be successfully detected. The explanation is that higher desired velocities make up for sharper enclosures which eventually exclude the parasite solution, while slower trajectories remain in the neighborhood of singularities where

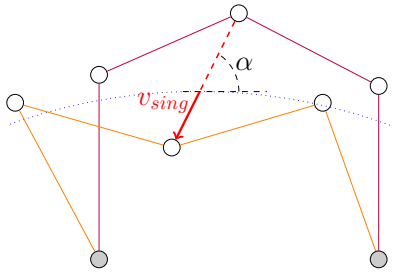


Fig. 6. Crossing trajectory parametrization

TABLE III
ALGORITHM OPERATING RANGE

v_{sing}=0.584 m/s

α (deg) \ ϵ_τ	1%	2%	5%	10%	15%	20%	25%	30%
30	×	×	×	×	×	×	×	×
45	×	×	×	×	×	×	×	×
60	✓	✓	×	×	×	×	×	×
75	✓	✓	✓	✓	×	×	×	×
90	✓	✓	✓	✓	×	×	×	×
105	✓	✓	✓	✓	×	×	×	×
120	✓	✓	×	×	×	×	×	×
135	×	×	×	×	×	×	×	×
150	×	×	×	×	×	×	×	×

v_{sing}=0.779 m/s

α (deg) \ ϵ_τ	1%	2%	5%	10%	15%	20%	25%	30%
30	×	×	×	×	×	×	×	×
45	×	×	×	×	×	×	×	×
60	✓	✓	✓	✓	×	×	×	×
75	✓	✓	✓	✓	✓	✓	✓	×
90	✓	✓	✓	✓	✓	✓	✓	✓
105	✓	✓	✓	✓	✓	✓	✓	×
120	✓	✓	✓	✓	×	×	×	×
135	✓	×	×	×	×	×	×	×
150	×	×	×	×	×	×	×	×

v_{sing}=0.974 m/s

α (deg) \ ϵ_τ	1%	2%	5%	10%	15%	20%	25%	30%
30	∅	∅	∅	×	×	×	×	×
45	∅	∅	✓	×	×	×	×	×
60	✓	✓	✓	✓	✓	✓	✓	✓
75	∅	∅	✓	✓	✓	✓	✓	✓
90	∅	∅	✓	✓	✓	✓	✓	✓
105	∅	∅	✓	✓	✓	✓	✓	✓
120	∅	✓	✓	✓	✓	✓	✓	✓
135	✓	✓	✓	×	×	×	×	×
150	✓	×	×	×	×	×	×	×

contraction is inefficient.

Compared to former works [24], the operating range was drastically increased in terms of crossing velocity. Trajectories down to 0.6 m/s can now be successfully detected, while the limit was 1.2 m/s for the former algorithm.

In addition to this, it can be seen that trajectories that are normal to the singularity locus are easy to detect, while those with an angle α closer to 0 or 180 deg are almost never detected. In the latter case, the pose of the robot is nearing the singularity for a long time and the algorithm cannot contract boxes, eventually failing to detect the change in assembly mode.

It seems that high velocity and orthogonality to the singularity locus are the key components of a successful crossing, but it should be tested (at least in simulation) on other types of robots to be proven. The DexTAR has only 2-dof in translation, and it is not clear what would happen with a robot that can achieve rotations of the end-effector.

In terms of uncertainty management, the results obtained for the trajectory with highest crossing speed contain some cases of empty set evaluations ($\emptyset$). As the velocity is higher, the possibility of empty boxes appears for small uncertainties, showing that values of $\boldsymbol{\epsilon_\tau}$ inferior to 10% do not account correctly for unmodeled dynamic effects. Such values should be avoided because they do not ensure the reliability of the algorithm.

C. Trying to Create False Positives

It has been observed experimentally that when the robot does not cross the singularity, it appears to bounce on it. It can happen if the trajectories are slow, too close from singularities or if tracking errors are too high. Such a failure to cross the singularity can be reproduced by skewing computed torque values so that they do not respect the non-degeneracy criterion for singularity crossing [17]. The desired trajectory consists in crossing the singularity locus at the desired point $\mathbf{x}_s = (0, 84.4)^T$ (mm), with an angle $\alpha_{sing} = 75$ deg and a velocity $v_{sing} = 0.58$ m/s. As expected, the robot fails to change its assembly mode.

The recorded trajectory is fed to the algorithm with uncertainty $\boldsymbol{\epsilon_\tau}$ kept to $\pm 10\%$ of τ_{max}. It can be seen on Fig. 7 that pose enclosures expand in both directions before an interruption is cast by the differential inclusion solver. Hence, the result is good since the actual FK solution is enclosed. No false positive is generated. Repeating the experiment on several other trajectories (not reported in this paper), we never achieved to get a false positive.

Overall, results tend to prove the reliability of the proposed algorithm. The conditions to get a successful detection were highlighted. A supplementary experiment with multiple singularity crossings is shown in a video attached with this paper.

VI. Conclusion and Perspectives

Parallel robots suffer from a lack of reconfigurability and a reduced operational workspace. Among proposed solutions to this problem, crossing Type 2 singularities seems general and versatile. However, this method to change assembly mode needs to gain reliability to be used in an industrial context.

In order to know what the current assembly mode of a parallel robot is, a reliable algorithm based on robot kinematics and dynamics is proposed. Reliability is ensured through the use of interval arithmetic. This could be achieved by finding a new formula for the acceleration of the robot's end-effector which holds even in Type 2 singularities. The formula

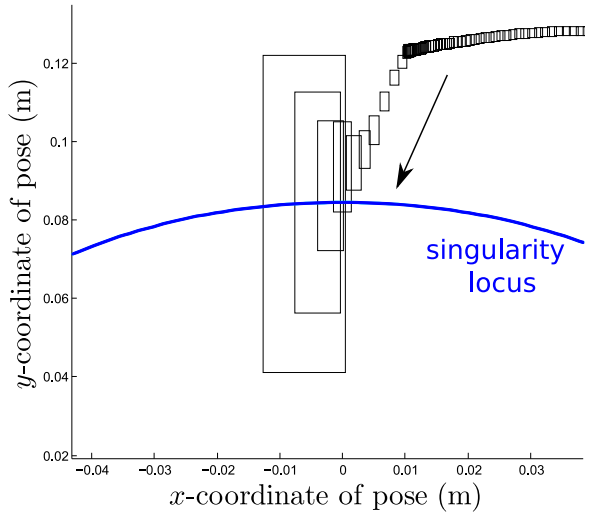


Fig. 7. Drawing of the enclosures on end-effector pose along a bouncing trajectory. The blue curve is the locus of Type 2 singularities.

was successfully used in the proposed solver for differential inclusions, that is a contribution of this paper too.

Experiments were carried on a 2-dof planar parallel robot named DexTAR, which is capable of changing its assembly mode thanks to specific motion generator and control scheme. They proved the reliability of the algorithm and showed which type of trajectories favor detection of singularity crossing. A persistent limit is the absence of a method for computing bounds on dynamic model uncertainties, though a trivial solution consists in choosing voluntarily pessimistic bounds.

An interesting future work would be to use such an algorithm during motion generation rather than task execution. This would allow to create a reliable trajectory planner that ensures successful assembly mode change.

APPENDIX A
NON-SINGULAR DIRECT DYNAMIC MODEL

In order to get a fully operational algorithm, we want to check that the proposed DDM (23) gives thinner enclosures on end-effector acceleration than other formulas such as (10), as stated in section II-B.

For this purpose, the equation (39) is tested in two cases for the five-bar mechanism presented in section IV:

1) as proposed in the paper, so that it needs joint acceleration values. It is thus named DDM-acc
2) where joint acceleration is substituted thanks to second-order inverse kinematic model, so that joint acceleration values are not needed. It is then named DDM-vel.

As described in section III-C, computing sharp end-effector acceleration enclosures is crucial for the success of the differential inclusion procedure, more so that the algorithm cannot rely on local solvers to sharpen enclosures in Type 2 singularity. In consequence, the preciseness of both DDM formulations has to be compared in the context of singularity, where end-effector pose and velocity intervals show their maximal width (see Fig. 4).

TABLE IV
END-EFFECTOR ACCELERATION COMPUTATION

			max of width($[\ddot{\mathbf{x}}]$)	
case	width($[\mathbf{x}]$)	width($[\dot{\mathbf{x}}]$)	DDM-acc	DDM-vel
	mm	m/s	m/s^2	
tight	1	0.1	36.82	27.08
loose	10	0.8	56.38	1420.24
maximal	50	4.5	361.4	∞

Based on data gathered from former paper [24], enclosures on end-effector acceleration are computed in three cases: when pose and velocity intervals are tight, then loosen up, then reach their maximal width.

From Table IV, it can be noticed that DDM-vel gives sharp results when input bracketings on $\mathbf{x}$ and $\dot{\mathbf{x}}$ are tight, but poor results when those enclosures have loosened up. The overestimation brought by multiple occurrences of the wide intervals $[\mathbf{x}]$ and $[\dot{\mathbf{x}}]$ in DDM-vel results in acceleration values growing towards infinity, forcing a failure of the differential inclusion procedure around Type 2 singularities. In contrast, DDM-acc is slightly less accurate far from singularities, but crucially more precise near them. This allows to conclude that DDM-acc, which was introduced formerly, is needed for the success of the presented algorithm. This also means that a measurement of joint acceleration is needed for the algorithm to operate.

APPENDIX B
DIFFERENTIAL INCLUSION FLOW ENCLOSURE

We consider a second order differential inclusion (DI)

$$\ddot{\mathbf{x}}(t) \in [F](\mathbf{x}(t), \dot{\mathbf{x}}(t)), \tag{40}$$

where $[F] : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{IR}^n$, and interval vector enclosures of the initial position $[\mathbf{x}_0] \in \mathbb{IR}^n$ and velocity $[\dot{\mathbf{x}}_0] \in \mathbb{IR}^n$. A solution of the DI for a duration $h > 0$ is a twice differentiable function $x : [0,h] \to \mathbb{R}^n$ that satisfies (40) for all $t \in [0,h]$. A solution of the corresponding initial value problem (IVP) is a solution of the DI that furthermore satisfies $\mathbf{x}(0) \in [\mathbf{x}_0]$, $\dot{\mathbf{x}}(0) \in [\dot{\mathbf{x}}_0]$. The aim of this section is to provide an algorithm that computes interval vector enclosures $[\mathbf{x}_h], [\dot{\mathbf{x}}_h] \in \mathbb{IR}^n$ of all IVP solutions position and velocity at time h.

A. Guessed and updated enclosures

We now furthermore consider interval vectors $[\mathbf{x}^G], [\dot{\mathbf{x}}^G] \in \mathbb{IR}^n$, which are guessed enclosures for the time period $[0,h]$, and define the following updated interval vectors

$$[\mathbf{x}^U] := [\mathbf{x}_0] + [0,h][\dot{\mathbf{x}}_0] + \tfrac{[0,h]^2}{2}[F^G] \tag{41}$$
$$[\dot{\mathbf{x}}^U] := [\dot{\mathbf{x}}_0] + [0,h][F^G], \tag{42}$$

where $[F^G] = [F]([\mathbf{x}^G], [\dot{\mathbf{x}}^G])$. The following theorem provides sufficient conditions for the IVP solution to actually belong to guessed and updated enclosures.

Theorem 1. *The following three statements hold:*

1) *Every IVP solution that belongs to the guessed enclosures also belongs to the updated enclosure.*

2) *If the updated enclosures are subsets of the interior of the guessed enclosure then every IVP solution belong to updated enclosures.*
3) *Every IVP solution that belongs to the updated enclosures also satisfies*

$$\mathbf{x}(h) \in [\mathbf{x}_0]+h[\dot{\mathbf{x}}_0]+\tfrac{h^2}{2}[F^U] \quad (43)$$
$$\dot{\mathbf{x}}(h) \in [\dot{\mathbf{x}}_0]+h[F^U], \quad (44)$$

where $[F^U]=[F]([\mathbf{x}^U],[\dot{\mathbf{x}}^U])$.

Remark 1. Statement 2 proves that updated enclosures enclose the trajectories provided that they are strictly included inside the guessed enclosures. These enclosures are rough in general, and Statement 3 provides sharper enclosures for time h.

Proof. We prove a slightly stronger statement than Statement 1: Given an arbitrary $t^* \in [0,h]$, we prove that every IVP solution that belong to the guessed enclosures for the time interval $[0,t^*]$ also belongs to the updated enclosures for this time period. Statement 1 follows with $t^*=h$. Considering an arbitrary IVP solution $\mathbf{x}(t)$ and both its first order Taylor expansion as well as the zeroth order Taylor expansion of $\dot{\mathbf{x}}(t)$, with the integral form of the remainder, one obtains

$$\mathbf{x}(t^*) \in \mathbf{x}(0)+t^*\dot{\mathbf{x}}(0)+\int_{\xi=0}^{t^*}(t^*-\xi)\ddot{\mathbf{x}}(\xi)d\xi \quad (45)$$
$$\subseteq \mathbf{x}(0)+t^*\dot{\mathbf{x}}(0)+\int_{\xi=0}^{t^*}(t^*-\xi)[F^G]d\xi \quad (46)$$
$$\dot{\mathbf{x}}(t^*) \in \dot{\mathbf{x}}(0)+\int_{\xi=0}^{t^*}\ddot{\mathbf{x}}(\xi)d\xi, \quad (47)$$
$$\subseteq \dot{\mathbf{x}}(0)+\int_{\xi=0}^{t^*}[F^G]d\xi, \quad (48)$$

where the hypothesis that $\ddot{\mathbf{x}}(\xi)\in[F](\mathbf{x}(\xi),\dot{\mathbf{x}}(\xi))\subseteq[F^G]$ for all $\xi\in[0,t^*]$ is used. Constant lower and upper bounds $[F^G]$ can be moved out of the integral. One eventually obtains $\mathbf{x}(t^*)\in[\mathbf{x}^U]$ and $\dot{\mathbf{x}}(t^*)\in[\dot{\mathbf{x}}^U]$ noting that $\int_{\xi=0}^{t^*}(t-\xi)d\xi=\frac{1}{2}t^{*2}$ and $\int_{\xi=0}^{t^*}d\xi=t^*$, and replacing t^* by its enclosure $[0,h]$.

We prove Statement 2 by contradiction. Suppose that there exists an IVP solution $\mathbf{x}(t)$ such that either $\mathbf{x}(t)$ leaves $[\mathbf{x}^G]$ or $\dot{\mathbf{x}}(t)$ leaves $[\dot{\mathbf{x}}^G]$. Obviously $[\mathbf{x}_0]\subseteq[\mathbf{x}^U]\subseteq\text{int}[\mathbf{x}^G]$ and $[\dot{\mathbf{x}}_0]\subseteq[\dot{\mathbf{x}}^U]\subseteq\text{int}[\dot{\mathbf{x}}^G]$, so there exists $t^*\in]0,h[$ such that $\mathbf{x}(t)\in[\mathbf{x}^G]$ and $\dot{\mathbf{x}}(t)\in[\dot{\mathbf{x}}^G]$ holds for all $t\in[0,t^*]$ and either $\mathbf{x}(t^*)$ is on the boundary of $[\mathbf{x}^G]$ or $\dot{\mathbf{x}}(t^*)$ is on the boundary of $[\dot{\mathbf{x}}^G]$ (roughly speaking, t^* is the time just before $\mathbf{x}(t)$ or $\dot{\mathbf{x}}(t)$ first leaves the guessed enclosures). However, we proved above that for the time period $[0,t^*]$ this trajectory is included in the updated enclosure, which are supposed to be included in the interior of the guessed enclosure, a contradiction.

For Statement 3, we use the Taylor expansions with the Lagrange form of the remainder for each component of $\mathbf{x}(t)$:

$$\mathbf{x}_i(h) \in \mathbf{x}_i(0)+h\dot{\mathbf{x}}_i(0)+\tfrac{h^2}{2}F_i(\mathbf{x}(\xi_i),\dot{\mathbf{x}}(\xi_i)) \quad (49)$$
$$\dot{\mathbf{x}}_i(h) \in \dot{\mathbf{x}}(0)+hF_i(\mathbf{x}(\xi_i'),\dot{\mathbf{x}}(\xi_i')), \quad (50)$$

with $\xi_i,\xi_i'\in[0,h]$. Since by hypothesis $\mathbf{x}(t)\in[\mathbf{x}^U]$ and $\dot{\mathbf{x}}(t)\in[\dot{\mathbf{x}}^U]$ for all $t\in[0,h]$, we have both $F_i(\mathbf{x}(\xi_i),\dot{\mathbf{x}}(\xi_i))\in[F_i^U]$ and $F_i(\mathbf{x}(\xi_i'),\dot{\mathbf{x}}(\xi_i'))\in[F_i^U]$, which proves Statement 3. □

Algorithm 1: Enclosure of the trajectories of a second order differential inclusion. Typical values for parameters are $k_{\max}=10$, $\delta=1.1$ and $\varepsilon=10^{-10}$.

Input: $[F]:\mathbb{R}^n\times\mathbb{R}^n\to\mathbb{IR}^n$; $[\mathbf{x}_0],[\dot{\mathbf{x}}_0]\in\mathbb{IR}^n$; $h>0$;
1 $[\mathbf{x}^G]\leftarrow[\mathbf{x}_0]$; $[\dot{\mathbf{x}}^G]\leftarrow[\dot{\mathbf{x}}_0]$; $k\leftarrow1$; success $\leftarrow$ **false**;
2 **repeat**
3 **if not** success **then**
4 $[\mathbf{x}^G]\leftarrow\text{mid}[\mathbf{x}^G]+\delta([\mathbf{x}^G]-\text{mid}[\mathbf{x}^G])+\varepsilon[-1,1]$;
5 $[\dot{\mathbf{x}}^G]\leftarrow\text{mid}[\dot{\mathbf{x}}^G]+\delta([\dot{\mathbf{x}}^G]-\text{mid}[\dot{\mathbf{x}}^G])+\varepsilon[-1,1]$;
6 **end**
7 $[\mathbf{x}^U]\leftarrow(41)$; $[\dot{\mathbf{x}}^U]\leftarrow(42)$;
8 **if** $[\mathbf{x}^U]\subseteq\text{int}[\mathbf{x}^G]$ **and** $[\dot{\mathbf{x}}^U]\subseteq\text{int}[\dot{\mathbf{x}}^G]$ **then** success $\leftarrow$ **true**;
9 $[\mathbf{x}^G]\leftarrow[\mathbf{x}^U]$; $[\dot{\mathbf{x}}^G]\leftarrow[\dot{\mathbf{x}}^U]$;
10 $k\leftarrow k+1$;
11 **until** $k>k_{\max}$;
12 **if** success **then return** (43) *and* (44);
13 **else return false** ;

B. Enclosing algorithm

The following algorithm implements Theorem 1 in a similar way as in the case of the interval simulation of ODEs. It is provided here for the sake of completeness. It returns the enclosures of position and velocity at time h in case of success, or false in case of failure.

Line 4 and Line 5 perform a static inflation of the enclosures. This static inflation is necessary to obtain the strict contraction, but slightly degrades the enclosures, and is therefore not applied anymore once the strict inclusion is observed. The rest of the algorithm iteratively applies Theorem 1 until the maximum number of iterations is reached.

REFERENCES

[1] C. Gosselin and J. Angeles, "Singularity analysis of closed loop kinematic chains," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 3, pp. 281–290, 1990.
[2] M. Conconi and M. Carricato, "A New Assessment of Singularities of Parallel Kinematic Chains," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 3–12, 2009.
[3] D. Chablat and P. Wenger, "Working modes and aspects in fully parallel manipulators," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation*, vol. 3, May 1998, pp. 1964–1969.
[4] A. Figielski, I. A. Bonev, and P. Bigras, "Towards development of a 2-DOF planar parallel robot with optimal workspace use," in *IEEE International Conference on Systems, Man and Cybernetics*, November 2007, pp. 1562–1566.
[5] G. Gogu, "Structural synthesis of fully-isotropic translational parallel robots via theory of linear transformations," *European Journal of Mechanics, A/Solids*, vol. 23, no. 6, pp. 1021–1039, 2004.
[6] M. Carricato, "Singularity-free fully-isotropic translational parallel manipulators," Ph.D. dissertation, 2001.
[7] J.-P. Merlet, "Redundant parallel manipulators," *Laboratory Robotics and Automation*, vol. 8, no. 1, pp. 17–24, 1996.
[8] A. Müller, "Redundant actuation of parallel manipulators," *Parallel Manipulators, towards new applications.*, no. April, 2008.
[9] V. Arakelian, S. Briot, and V. Glazunov, "Increase of singularity-free zones in the workspace of parallel manipulators using mechanisms of variable structure," *Mechanism and Machine Theory*, vol. 43, no. 9, pp. 1129–1140, 2008.
[10] N. Rakotomanga, D. Chablat, and S. Caro, "Kinetostatic performance of a planar parallel mechanism with variable actuation," *Advances in Robot Kinematics: Analysis and Design*, pp. 311–320, 2008.

[11] P. R. McAree and R. W. Daniel, "An Explanation of Never-Special Assembly Changing Motions for 3-3 Parallel Manipulators," *The International Journal of Robotics Research*, vol. 18, no. 6, pp. 556–574, 1999.

[12] M. Zein, P. Wenger, and D. Chablat, "Singular Curves in the Joint Space and Cusp Points of 3-RPR Parallel Manipulators," *Robotica*, vol. 25, no. 6, pp. 717–724, 2007.

[13] F. Bourbonnais, P. Bigras, and I. A. Bonev, "Minimum-time trajectory planning and control of a pick-and-place five-bar parallel robot," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 740–749, 2015.

[14] D. N. Nenchev, S. Bhattacharya, and M. Uchiyama, "Dynamic Analysis of Parallel Manipulators under the Singularity-Consistent Parameterization," *Robotica*, vol. 15, no. 4, pp. 375–384, 1997.

[15] J. Hesselbach, M. Helm, and S. Soetebier, "Connecting assembly modes for workspace enlargement," *Advances in Robot Kinematics*, pp. 347–356, 2002.

[16] C. K. Kevin Jui and Q. Sun, "Path Tracking of Parallel Manipulators in the Presence of Force Singularity," *Journal of Dynamic Systems, Measurement, and Control*, vol. 127, no. 4, p. 550, 2005.

[17] S. Briot and V. Arakelian, "Optimal Force Generation in Parallel Manipulators for Passing through the Singular Positions," *The International Journal of Robotics Research*, vol. 27, no. 8, pp. 967–983, 2008.

[18] S. K. Ider, "Inverse dynamics of parallel manipulators in the presence of drive singularities," *Mechanism and Machine Theory*, vol. 40, no. 1, pp. 33–34, 2005.

[19] G. Pagis, N. Bouton, S. Briot, and P. Martinet, "Enlarging parallel robot workspace through Type-2 singularity crossing," *Control Engineering Practice*, vol. 39, pp. 1–11, 2015.

[20] R. Vertechy and V. Parenti-Castelli, "Robust, fast and accurate solution of the direct position analysis of parallel manipulators by using extra-sensors," in *Parallel Manipulators, towards new applications.*, 2008, pp. 133–154.

[21] H. Saafi, M. A. Laribi, and S. Zeghloul, "Forward kinematic model improvement of a spherical parallel manipulator using an extra sensor," *Mechanism and Machine Theory*, vol. 91, pp. 102–119, 2015.

[22] R. Dahmouche, N. Andreff, Y. Mezouar, and P. Martinet, "Efficient high-speed vision-based computed torque control of the orthoglide parallel robot," in *Proc of the IEEE International Conference on Robotics and Automation*, 2010, pp. 644–649.

[23] J. P. Merlet, "Solving the Forward Kinematics of a Gough-Type Parallel Manipulator with Interval Analysis," *The International Journal of Robotics Research*, vol. 23, no. 3, pp. 221–235, 2004.

[24] A. Koessler, A. Goldsztejn, S. Briot, and N. Bouton, "Certified Detection of Parallel Robot Assembly Mode under Type 2 Singularity Crossing Trajectories," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 6073–6079.

[25] J.-P. Aubin and A. Cellina, *Differential Inclusions*. Springer-Verlag Berlin Heidelberg, 1984.

[26] J.-P. Merlet, *Parallel Robots (2nd Ed.)*, G. M. L. Gladwell, Ed. Dordrecht Springer, 2006.

[27] S. Briot and W. Khalil, *Dynamics of Parallel Robots: from Rigid Bodies to Flexible Elements*, M. Ceccarelli, Ed. Dordrecht Springer, 2015.

[28] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis*, Springer, Ed. Springer, 2001.

[29] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, 2009.

[30] F. Le Bars, A. Bertholom, S. Jan, and L. Jaulin, "Interval SLAM for underwater robots. A new experiment," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2010, pp. 42–47.

[31] L. Jaulin, "Path planning using intervals and graphs," *Reliable Computing*, vol. 7, no. 1, pp. 1–15, 2001.

[32] B. Martin, A. Goldsztejn, C. Jermann, and L. Granvilliers, "Certified Parallelotope Continuation for One-Manifolds," *SIAM Journal On Numerical Analysis*, vol. 51, no. 6, pp. 3373–3401, 2013.

[33] J. Vehí, I. Ferrer, and M. Á. Sainz, "a Survey of Applications of Interval Analysis To Robust Control," in *IFAC Proceedings Volumes*, vol. 35, no. 1, 2002, pp. 389–400.

[34] L. Jaulin and F. Le Bars, "An Interval Approach for Stability Analysis; Application to Sailboat Robotics," *IEEE Transactions on Robotics*, vol. 29, no. 1, pp. 282–287, 2013.

[35] L. Jaulin and E. Walter, "Set inversion via interval analysis for nonlinear bounded-error estimation," *Automatica*, vol. 29, no. 4, pp. 1053–1064, 1993.

[36] D. Daney, "Interval methods for certification of the kinematic calibration of parallel robots," in *IEEE Int. Conf. on Robotics and Automation*, April 2004, pp. 1913–1918.

[37] M. R. Pac and D. O. Popa, "Interval analysis for robot precision evaluation," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2012, pp. 1087–1092.

[38] M. Tannous, S. Caro, and A. Goldsztejn, "Sensitivity analysis of parallel manipulators using an interval linearization method," *Mechanism and Machine Theory*, vol. 71, pp. 93–114, 2014.

[39] A. Goldsztejn, S. Caro, and G. Chabert, "A three-step methodology for dimensional tolerance synthesis of parallel manipulators," *Mechanism and Machine Theory*, vol. 105, pp. 213–234, 2015.

[40] D. Oetomo, D. Daney, B. Shirinzadeh, and J.-P. Merlet, "An Interval-Based Method for Workspace Analysis of Planar Flexure-Jointed Mechanism," *Journal of Mechanical Design*, vol. 131, no. 1, pp. 1–14, 2009.

[41] S. Caro, D. Chablat, A. Goldsztejn, D. Ishii, and C. Jermann, "A branch and prune algorithm for the computation of generalized aspects of parallel robots," *Artificial Intelligence*, vol. 211, no. 1, pp. 34–50, 2014.

[42] F. Benhamou, F. Goualard, L. Granvilliers, and J. Puget, "Revising hull and box consistency," in *Proc. of the 16th Int. Conf. on Logic Programming*, 1999, pp. 230–244.

[43] I. Araya, G. Trombettoni, and B. Neveu, "A contractor based on convex interval Taylor," in *Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems*, 2012, pp. 1–16.

[44] T. Kapela and P. Zgliczyński, "A Lohner-type algorithm for control systems and ordinary differential inclusions," *Discrete and Continuous Dynamical Systems - Series B*, vol. 11, no. 2, pp. 365–385, 2008.

[45] D. Daney, N. Andreff, G. Chabert, and Y. Papegay, "Interval method for calibration of parallel robots: Vision-based experiments," *Mechanism and Machine Theory*, vol. 41, no. 8, pp. 929–944, 2006.

[46] P. Poignet, N. Ramdani, and O. Andres Vivas, "Robust Estimation of Parallel Robot Dynamic Parameters with Interval Analysis," in *IEEE Conference on Decision and Control*, 2003, pp. 6503–6508.

[47] A. Joubair and I. A. Bonev, "Comparison of the efficiency of five observability indices for robot calibration," *Mechanism and Machine Theory*, vol. 70, pp. 254–265, 2013.

[48] W. Khalil and E. Dombre, *Modélisation, identification et commande des robots*. Hermès Science, 1999.

[49] M. Gautier and S. Briot, "Global Identification of Joint Drive Gains and Dynamic Parameters of Parallel Robots," *Multibody System Dynamics*, vol. 33, no. 1, pp. 3–26, 2013.

[50] S. Briot, N. Bouton, and P. Bigras, "Controlling parallel robots during singular assembly mode changing," in *International Conference on Multibody System Dynamics*, 2016, pp. 116–128.

[51] D. Six, S. Briot, A. Chriette, and P. Martinet, "A Controller Avoiding Dynamic Model Degeneracy of Parallel Robots During Singularity Crossing," *Journal of Mechanisms and Robotics*, vol. 9, no. 5, pp. 1–8, 2017.

[52] A. Koessler, N. Bouton, S. Briot, B. C. Bouzgarrou, and Y. Mezouar, "Linear Adaptive Computed Torque Control for Singularity Crossing of Parallel Robots," in *22nd CISM IFToMM Symposium on Robot Design, Dynamics and Control*, 2018 (Accepted), pp. 1–8.

[53] G. Chabert and L. Jaulin, "Contractor programming," *Artificial Intelligence*, vol. 173, no. 11, pp. 1079–1100, 2009.

Adrien Koessler recieved his engineer degree from Institut Français de Mécanique Avancée and his masters degree in robotics from Université Blaise Pascal, Clermont-Ferrand, France in 2015. He also recieved his Ph.D. in Engineering Science from Université Clermont Auvergne, France in 2018. He is currently postdoctoral fellow in FactoLab, Institut Pascal.

His research interests include kinematic and dynamic modeling of robots, as well as control law synthesis.

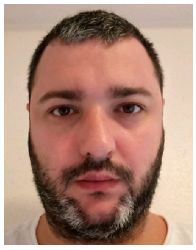
Alexandre Goldsztejn received his engineer degree from the Institut Supérieur d'Electronique et du Numérique, Lille, France, in 2001, and his Ph.D. in Computer Science from the University of Nice-Sophia Antipolis in 2005. He has spent one year as a postdoctoral fellow in the University of Central Arkansas and the University of California Irvine. He is full time CNRS researcher since 2007.

His research interests include interval analysis and its application to constraint satisfaction, nonlinear global optimization, robotics and control.

Sébastien Briot received the B.S. and M.S. degrees in Mechanical Engineering in 2004 from the National Institute of Applied Sciences (INSA) of Rennes (France). Then, he began a PhD thesis, under the supervision of Prof. Vigen Arakelian, at the INSA of Rennes and received the PhD degree in 2007. He worked at the Ecole de Technologie Suprieure of Montreal (Canada) with Prof. Ilian Bonev as a postdoctorate fellow in 2008. Since 2009, he is a full-time CNRS researcher at the LS2N (ex-IRCCyN Lab.) in Nantes (France). Since 2017, he is the head of the ARMEN research team at LS2N.

His research fields concern the design optimization of robots and the analysis of their dynamic performance. He also studies the impact of sensor-based controllers on the robot performance. He is the author of 40 referred journal papers, 2 books and 3 inventions.

Dr. Briot received the Award of the Best Ph.D. Thesis in Robotics from the French CNRS for year 2007. In 2011, he received two other awards: the Award for the Best Young Researcher from the French Region Bretagne and the Award for the Best Young Researcher from the French Section of the American Society of Mechanical Engineering (SF-ASME).



Nicolas Bouton received his engineer degree from Institut Français de Mécanique Avancée, Clermont-Ferrand, France in 2006. He also received his masters degree in robotics from Université Blaise Pascal, Clermont-Ferrand, France in 2006 and his Ph.D. in Engineering Sciences and Mechatronics from Unier-sité Blaise Pascal of Clermont-Ferrand in 2009. He is associate professor at SIGMA Clermont since 2010.

His research interests include automatic, nonlinear control theory, advanced control law algorithms and observability and their applications to robotics, mechatronics, machines and systems.