Efficient Computation of Interval-Arithmetic-Based Robust Controllers for Rigid Robots

Andrea Giusti and Matthias Althoff
Faculty of Informatics
Technical University of Munich
Garching bei München, Germany
Email: {giusti, althoff}@in.tum.de

***Abstract*—We propose a method for efficient numerical computation of interval-arithmetic-based robust controllers for rigid robot manipulators. The use of interval arithmetic for robust control is the core of a recently proposed approach which allows a user-defined tracking performance to be ultimately met despite uncertain models and input disturbance, without requiring an empirical estimation of bounds of perturbations from uncertain system dynamics. Our proposed algorithm combines a modified recursive Newton-Euler scheme with interval arithmetic computations to automatically obtain formally guaranteed over-approximative estimations of perturbing torques/forces arising from imperfect knowledge of dynamic model parameters. The resulting algorithm has linear computational complexity and can be used online. We validate the applicability of our proposed method with simulations and tests on a commercially available real-time target computer.**

***Keywords*-efficient computation; robust control; interval arithmetic; robot manipulators**

I. Introduction

The vast computational resources available today as well as the effectiveness of modern robot dynamics algorithms [1] make it possible to implement advanced model-based control approaches for robots. The quality of the closed-loop performance of model-based controllers depends on how well the real system dynamics match the corresponding mathematical model. When a sufficiently high quality of the available (nominal) model cannot be ensured, control system designers can resort to robust control techniques for ensuring stability and tracking performance (see e.g. [2], [3]).

Most of the methods proposed for robust control require estimating bounds of perturbation terms arising from uncertain knowledge of the system dynamics (see e.g. [4], [5]). Obtaining such bounds on uncertain model components leads to time consuming estimation phases and does not provide formal guarantees. Indeed, to the best of our knowledge no procedure or methodology has been proposed for formally obtaining those bounds. To remove the above-mentioned limitation and provide a quickly deployable robust controller, we have previously introduced the use of interval arithmetic for passivity-based robust control of rigid robots [6]. The resulting interval-arithmetic-based robust controller is continuous and in principle allows any user-defined tracking performance to be ultimately (in finite time) met and maintained. The price to pay for the benefits introduced mainly relies on the computational complexity introduced by automatically computing the measurement of the worst-case perturbation using interval arithmetic. Indeed, as it will become clearer in Sec. III, the analytical function for computing the worst-case perturbation using interval arithmetic is very large for robots with a large number of joints.

To eliminate the main drawback of the robust control approach proposed in [6], we introduce the integration of interval arithmetic computations within a recursive Newton-Euler (N-E) algorithm [7]. With this approach we enable the efficient numerical computation of guaranteed over-approximative sets of torques/forces arising from uncertain dynamic parameters. In particular, we provide an extension of the modified recursive N-E algorithm for passivity-based control originally proposed in [8] (also to handle prismatic joints), and we describe the introduction of interval arithmetic computations. Our approach is simple since it does not require manipulating any symbolic variable, yet it enables the efficient computation of interval-arithmetic-based robust controllers for a large number of joints with linear computational complexity.

A particularly interesting implication of our approach is that it supports the efficient automatic design of robust controllers for modular robot manipulators [9]. This is achieved by straightforwardly enhancing the framework proposed in [10] by storing uncertainty bounds of dynamical parameters in the modules and applying the algorithm presented here. Additionally, our proposed algorithm can also be used for robust dynamic scaling of trajectories [11].

The remainder of this paper is structured as follows: In Sec. II we recall some preliminary notions on interval arithmetic, and in Sec. III we describe in detail the problem that we address. Subsequently, we describe our proposed solution in Sec. IV and present simulation results in Sec. V.

II. Preliminaries on Interval-Arithmetic

We propose a numerical algorithm based on interval arithmetic to efficiently realize our previously published

robust control idea in [6]. To preserve fluency and clarity of the subsequent description, we recall the following central definitions. The interested reader may refer to [12] for further details.

Definition 1 (Multidimensional interval): A multidimensional interval is a set of real numbers defined as

$$[\mathbf{x}] := [\underline{\mathbf{x}}, \overline{\mathbf{x}}], \quad \underline{\mathbf{x}} \in \mathbb{R}^n, \ \overline{\mathbf{x}} \in \mathbb{R}^n, \ \underline{x}_i \leq \overline{x}_i, \ \text{for all } i = 1, \ldots, n.$$

We denote the scalar case by x instead of $\mathbf{x}$, and we use $\underline{x}$ and $\overline{x}$ to denote the infimum and supremum of an interval $[x]$, respectively.

Definition 2 (Degenerate interval): An interval $[x]$ whose infimum $\underline{x}$ and supremum $\overline{x}$ are equal is called degenerate and will be denoted simply by x hereafter.

Definition 3 (Interval-valued function): Given a function $\mathbf{z} : \mathbb{R}^n \to \mathbb{R}^m$, its interval evaluation over a set $[\mathbf{x}]$ is defined as:

$$\mathbf{z}([\mathbf{x}]) := \{\mathbf{z}(\mathbf{x}) \mid \mathbf{x} \in [\mathbf{x}]\}.$$

We further define the set-based addition, subtraction and multiplication as follows.

Definition 4 (Set-based operations): Let $\mathbb{IR}$ be the set of all scalar intervals. For $[x] \in \mathbb{IR}$ and $[y] \in \mathbb{IR}$, the result of the binary operations $* \in \{+, -, \cdot\}$ is defined as:

$$[x] \circledast [y] := \{x * y \mid x \in [x], y \in [y]\}.$$

We straightforwardly implement the above-mentioned operations as follows (see e.g. [12, App. B]):

$$\begin{aligned} [x] \oplus [y] &= [\underline{x} + \underline{y}, \overline{x} + \overline{y}], \\ [x] \ominus [y] &= [\underline{x} - \overline{y}, \overline{x} + \underline{y}], \\ [x] \odot [y] &= [\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}), \max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y})]. \end{aligned}$$

While set-based addition and subtraction are applied element-wise when multidimensional intervals are involved, we need to further define the interval matrix/scalar-matrix multiplication.

Definition 5 (Interval matrix/scalar-matrix multiplication): Given $[\mathbf{X}] \in \mathbb{IR}^{n \times m}$, $[\mathbf{Y}] \in \mathbb{IR}^{m \times p}$ and $[a] \in \mathbb{IR}$, the results of a matrix and scalar-matrix multiplication are defined respectively as:

$$([\mathbf{X}] \odot [\mathbf{Y}])_{ij} = \bigoplus_{k=1}^{n} \Big([\mathbf{X}]_{ik} \odot [\mathbf{Y}]_{kj} \Big), \quad ([a] \odot [\mathbf{X}])_{ij} = [a] \odot [\mathbf{X}]_{ij},$$

where we denote by $\bigoplus_{k=1}^{n}$ the interval version of the summation symbol, which involves set-based additions. We can now conclude this preliminary section by defining the set-based cross-product which will be required for the robot dynamics.

Definition 6 (Set-based cross-product): Given two interval vectors $[\mathbf{x}] \in \mathbb{IR}^{3\times 1}$ and $[\mathbf{y}] \in \mathbb{IR}^{3\times 1}$, the result of the set-based cross product between them is defined as

$$\begin{aligned} ([\mathbf{x}] \otimes [\mathbf{y}])_1 &= [\mathbf{x}]_2 \odot [\mathbf{y}]_3 \ominus [\mathbf{x}]_3 \odot [\mathbf{y}]_2, \\ ([\mathbf{x}] \otimes [\mathbf{y}])_2 &= [\mathbf{x}]_3 \odot [\mathbf{y}]_1 \ominus [\mathbf{x}]_1 \odot [\mathbf{y}]_3, \\ ([\mathbf{x}] \otimes [\mathbf{y}])_3 &= [\mathbf{x}]_1 \odot [\mathbf{y}]_2 \ominus [\mathbf{x}]_2 \odot [\mathbf{y}]_1. \end{aligned}$$

We constitute that set-based multiplications bind more strongly than additions and subtractions.

III. Problem Description

Throughout this paper we use bold symbols for matrices and vectors. We consider rigid robot manipulators composed of N serially connected links. We denote the mass, the coordinates of the center of mass and the barycentric inertia tensor of the i^{th} link respectively by

$$m_i, \ \mathbf{c}_i^i = \begin{pmatrix} c_{x,i} \\ c_{y,i} \\ c_{z,i} \end{pmatrix} \text{ and } \mathbf{I}_i^i = \begin{pmatrix} I_{xx,i} & I_{xy,i} & I_{xz,i} \\ * & I_{yy,i} & I_{yz,i} \\ * & * & I_{zz,i} \end{pmatrix}.$$

In this paper, the superscript of vectors indicates the link-fixed frame in which they are expressed. The superscript i of the inertia tensor indicates that it is expressed in a barycentric frame oriented as the i^{th} link-fixed frame. We introduce the vector of the dynamical parameters of the manipulator as follows:

$$\begin{aligned} \Delta = (&m_1, \ldots, m_N, c_{x,1}, c_{y,1}, c_{z,1}, \ldots, c_{z,N}, \\ &I_{xx,1}, I_{xy,1}, I_{xz,1}, I_{yy,1}, I_{yz,1}, I_{zz,1}, \ldots, I_{zz,N})^T. \end{aligned}$$

The system dynamics can be modeled as [13, Ch. 7]:

$$\mathbf{M}(\mathbf{q}, \Delta)\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}, \Delta)\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}, \Delta) = \mathbf{u} + \mathbf{d}, \tag{1}$$

where the vector of joint positions, velocities and accelerations are respectively $\mathbf{q}, \dot{\mathbf{q}}$ and $\ddot{\mathbf{q}} \in \mathbb{R}^N$. $\mathbf{M}(\mathbf{q}, \Delta) \in \mathbb{R}^{N \times N}$ is the symmetric and positive definite inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}, \Delta)\dot{\mathbf{q}} \in \mathbb{R}^N$ is the vector of Coriolis and centrifugal terms and $\mathbf{g}(\mathbf{q}, \Delta) \in \mathbb{R}^N$ the vector of gravity terms. The vectors $\mathbf{u}$ and $\mathbf{d} \in \mathbb{R}^N$ respectively represent the actuation and bounded disturbance torques/forces of the joints.

We assume that a nominal value of the dynamical parameters Δ_0 is available and that the uncertainty of each dynamical parameter is known. Friction is not considered for the sake of brevity. This does not cause a loss of generality, since its inclusion is rather straightforward and it does not affect the proposed idea. In principle, within the proposed setting and with sufficiently smooth required trajectories $\mathbf{q}_d$ (at least twice differentiable), any user defined tracking performance can be ultimately met by employing the robust performance control law recently proposed in [6]. In that work, the idea of using interval arithmetic for feedback control is introduced to deploy a robust performance controller automatically (i.e. without the need for estimating bounds of state dependent perturbations from uncertain model terms). We briefly recall this controller in the following.

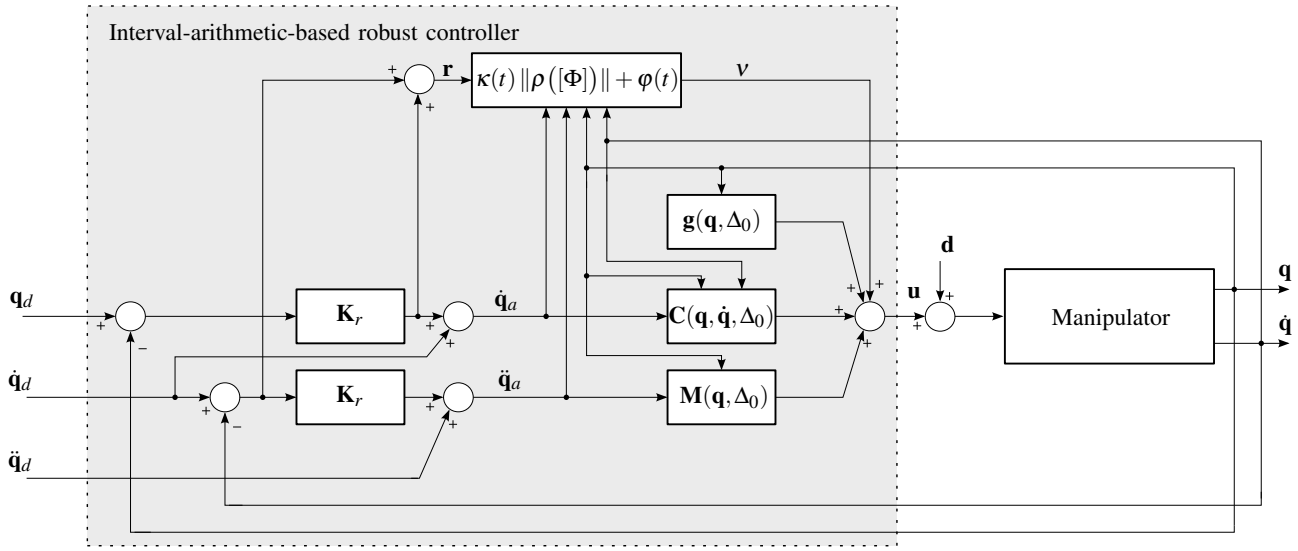

Figure 1. Block diagram of the closed-loop system.

The control method proposed in [6] builds on passivity-based control [3], thus exploiting the property that the matrix $\mathbf{N}(\mathbf{q},\dot{\mathbf{q}},\Delta)=\dot{\mathbf{M}}(\mathbf{q},\Delta)-2\mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\Delta)$ is skew-symmetric for a suitable factorization of $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\Delta)\in\mathbb{R}^{N\times N}$ and thus [13, Section 7.2.1]:

$$\mathbf{x}^T\mathbf{N}(\mathbf{q},\dot{\mathbf{q}},\Delta)\mathbf{x}=0,\quad \forall\mathbf{x}\in\mathbb{R}^N. \tag{2}$$

The above mentioned robust performance controller is composed of the classical passivity-based control relation

$$\mathbf{u}=\mathbf{M}(\mathbf{q},\Delta_0)\ddot{\mathbf{q}}_a+\mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\Delta_0)\dot{\mathbf{q}}_a+\mathbf{g}(\mathbf{q},\Delta_0)+\nu, \tag{3}$$

where

$$\dot{\mathbf{q}}_a=\dot{\mathbf{q}}_d+\mathbf{K}_r\tilde{\mathbf{q}},\ \ \tilde{\mathbf{q}}=\mathbf{q}_d-\mathbf{q}, \tag{4}$$

and $\mathbf{K}_r$ is a diagonal positive definite gain matrix. To properly handle perturbations from external disturbances and modelling errors, this controller introduces the novel use of interval arithmetic with the auxiliary input vector ν. Indeed, by applying the command of (3) to (1), the following closed loop relation is obtained:

$$\mathbf{M}(\mathbf{q},\Delta)\dot{\mathbf{r}}+\mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\Delta)\mathbf{r}=-\nu+\mathbf{w}(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_a,\ddot{\mathbf{q}}_a,\mathbf{d},\Delta,\Delta_0), \tag{5}$$

with $\mathbf{r}=\dot{\tilde{\mathbf{q}}}+\mathbf{K}_r\tilde{\mathbf{q}}$ and where $\mathbf{w}(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_a,\ddot{\mathbf{q}}_a,\mathbf{d},\Delta,\Delta_0)$ denotes the perturbation vector that can be written as follows:

$$\mathbf{w}(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_a,\ddot{\mathbf{q}}_a,\mathbf{d},\Delta,\Delta_0)=\big(\mathbf{M}(\mathbf{q},\Delta)-\mathbf{M}(\mathbf{q},\Delta_0)\big)\ddot{\mathbf{q}}_a+$$
$$\big(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\Delta)-\mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\Delta_0)\big)\dot{\mathbf{q}}_a+\mathbf{g}(\mathbf{q},\Delta)-\mathbf{g}(\mathbf{q},\Delta_0)-\mathbf{d}. \tag{6}$$

The controller is completed by selecting

$$\nu=\Big(\kappa(t)\,\|\rho\big([\Phi]\big)\|+\varphi(t)\Big)\,\mathbf{r}, \tag{7}$$

where $\kappa(t)$ and $\varphi(t)$ are positive increasing functions and where $\rho\big([\Phi]\big)$ is a measure of the worst case disturbance:

$$\rho([\Phi])=\max\Big(|\underline{\Phi}|,|\overline{\Phi}|\Big), \tag{8}$$

where

$$[\Phi]=\mathbf{w}(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_a,\ddot{\mathbf{q}}_a,[\mathbf{d}],[\Delta],\Delta_0). \tag{9}$$

In (9), $\mathbf{w}(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_a,\ddot{\mathbf{q}}_a,[\mathbf{d}],[\Delta])$ is an interval-valued function considering bounded interval vectors of the external disturbances $[\mathbf{d}]$ and of the dynamical parameters $[\Delta]$. A schematic representation of this controller is shown in Fig. 1. All norms in this paper are Euclidean norms.

The price to pay for the benefits gained using this controller is the computational cost for obtaining the measurement of the worst case disturbance $\rho\big([\Phi]\big)$. To the best of our knowledge, no algorithm for its efficient numerical computation has been previously made available. This can be considered as the principal drawback, endangering the general applicability especially for large N. We address the problem of designing an algorithm that allows one to perform the computation of (9) *efficiently* (without the need of any symbolic manipulation) and that can be directly used *on-line*.

IV. Proposed Method

To obtain guaranteed over-approximative sets of joint torques/forces arising from uncertain dynamic parameters in a computationally efficient way, we propose the idea of enhancing recursive N-E algorithms with interval arithmetic computations. In the following we describe the application of this idea in detail.

Thanks to the growing popularity of passivity-based control laws, a modification to the standard recursive N-E algorithm (see e.g. [1], [7]) was introduced in [8] and allows the numerical computation of the classical passivity-based control commands in (3) with linear computational complexity. The algorithm proposed in [8] is described for revolute joints only. Starting from that approach, we

first introduce a small extension to also handle prismatic joints, and we additionally include the use of set-based operations. The resulting interval-arithmetic-based Newton-Euler algorithm denoted by $IANEA_g^*(\mathbf{q}, \dot{\mathbf{q}}, \dot{\mathbf{q}}_a, \ddot{\mathbf{q}}, \mathbf{DHtab}, [\Delta])$ is presented in Alg. 1.

We consider that the link-fixed reference frames are assigned according to the standard Denavit-Hartenberg (D-H) convention [14]. Thus, the orientation of the i^{th} frame with respect to the previous frame can be written using the standard D-H parameters (a_i, θ_i, d_i, α_i) with the following rotation matrix:

$$\mathbf{R}_i^{i-1} = \begin{pmatrix} C_{\theta_i} & -C_{\alpha_i} S_{\theta_i} & S_{\alpha_i} S_{\theta_i} \\ S_{\theta_i} & C_{\alpha_i} C_{\theta_i} & -S_{\alpha_i} C_{\theta_i} \\ 0 & S_{\alpha_i} & C_{\alpha_i} \end{pmatrix}, \tag{10}$$

and the position of its origin from frame $i-1$ expressed in the frame i with the vector:

$$\mathbf{p}_{i-1,i}^i = (\, a_i \;\; d_i S_{\alpha_i} \;\; d_i C_{\alpha_i} \,)^T, \tag{11}$$

where we abbreviate $\sin(\xi)/\cos(\xi)$ with S_ξ/C_ξ. As usual $\theta_i = q_i$ when the joint is revolute and $d_i = q_i$ when it is prismatic.

In Alg. 1, we denote by $\boldsymbol{\omega}_i$ the angular velocity of the i^{th} link, by $\boldsymbol{\omega}_{a_i}$ an auxiliary angular velocity (whose utility is described later), by $\mathbf{c}_i^i$ the position of the center of mass, by $\mathbf{a}_i^i$ the acceleration of the link-fixed frame (and by $\mathbf{a}_{c,i}^i$ the one of the center of mass), by $\mathbf{I}_i^i$ the inertia tensor in the moving frame and finally by $\mathbf{f}_i^i/\mathbf{n}_i^i$ the reaction force/torque with the preceding link. Additionally, we denote the z-axis unit vector by $\mathbf{z}_0 = [0, 0, 1]^T$ and by **DHtab** a matrix containing the D-H parameters and the information for detecting the type of each joint (revolute or prismatic).

Alg. 1 shares the same algorithmic complexity and structure of a standard N-E algorithm (see e.g. [13, Sec. 7.5.2]) being composed of two recursions. While the first recursion (starting at line 11) computes the kinematic relations between subsequent links from the basis to the end effector, the second recursion (starting at line 26) runs backward from the end effector to the basis to compute the balance of forces and torques of the Newton-Euler equations of dynamics.

As it has been shown in the algorithm proposed in [8] for revolute joints, an auxiliary angular velocity vector $\boldsymbol{\omega}_a$ and an auxiliary joint velocity variable $\dot{\mathbf{q}}_a$ are introduced to allow the separation necessary for the Coriolis and centrifugal terms of the model (e.g. to compute (3) for passivity-based control) and to obtain (if necessary) the matrix $\mathbf{C}$ alone. We extend the applicability of the algorithm to prismatic joints by properly introducing the auxiliary angular velocity in line 19 and 21. In particular, in line 21 the quadratic dependency of the angular velocity on the right-hand side has been modified from $2\boldsymbol{\omega}_i^i \times \dot{q}_i \mathbf{z}_i$ of standard algorithms to $\boldsymbol{\omega}_{a,i}^i \times \dot{q}_i \mathbf{z}_i + \boldsymbol{\omega}_i^i \times \dot{q}_{a,i} \mathbf{z}_i$, to enable the above-mentioned separation in the Coriolis and centrifugal model term. It is worth mentioning that friction and approximated rotor inertia effects (that we do not include for the sake of brevity) can be straightforwardly included in line 31 and 33.

The use of set-based operations in Alg. 1 allows one to handle multidimensional interval vectors of the dynamical parameters to directly compute over-approximative sets of joint torques/forces. We are now ready to show in detail

Algorithm 1 Interval-arithmetic-based N-E algorithm: $IANEA_g^*(\mathbf{q}, \dot{\mathbf{q}}, \dot{\mathbf{q}}_a, \ddot{\mathbf{q}}, \mathbf{DHtab}, [\Delta])$.

Input: $\mathbf{q}$, $\dot{\mathbf{q}}$, $\dot{\mathbf{q}}_a$, $\ddot{\mathbf{q}}$, $[\Delta]$, **DHtab** and the gravity vector $\mathbf{g}$
Output: $[\boldsymbol{\tau}]$

1: Initialize number of links N (from **DHtab**)
2: Initialize $\boldsymbol{\omega}_{a,0}^0$, $\boldsymbol{\omega}_0^0$, $\dot{\boldsymbol{\omega}}_0^0$ and $\ddot{\mathbf{p}}_0^0$ to zero and $\mathbf{z}_0 \leftarrow [0,0,1]^T$
3: Include effect of gravity $\ddot{\mathbf{p}}_0^0 \leftarrow \ddot{\mathbf{p}}_0^0 - \mathbf{g}$
4: **for** $i = 1$ to N **do**
5: Set $[\mathbf{c}_i^i]$ and $[\mathbf{I}_i^i]$ using $[\Delta]$
6: $\mathbf{R}_i^{i-1} \leftarrow$ compute (10) using **DHtab** and q_i
7: $\mathbf{R}_{i-1}^i \leftarrow transpose\left(\mathbf{R}_i^{i-1}\right)$
8: $\mathbf{z}_i \leftarrow \mathbf{R}_{i-1}^i \mathbf{z}_0$
9: $\mathbf{p}_{i-1,i}^i \leftarrow$ compute (11) using **DHtab** and q_i
10: **end for**
11: **for** $i = 1$ to N **do** ▷ Start of forward recursion
12: **if** i^{th} joint is revolute **then**
13: $\boldsymbol{\omega}_i^i \leftarrow \mathbf{R}_{i-1}^i \left(\boldsymbol{\omega}_{i-1}^{i-1} + \dot{q}_i \mathbf{z}_0\right)$
14: $\boldsymbol{\omega}_{a,i}^i \leftarrow \mathbf{R}_{i-1}^i \left(\boldsymbol{\omega}_{a,i-1}^{i-1} + \dot{q}_{a,i} \mathbf{z}_0\right)$
15: $\dot{\boldsymbol{\omega}}_i^i \leftarrow \mathbf{R}_{i-1}^i \left(\dot{\boldsymbol{\omega}}_{i-1}^{i-1} + \ddot{q}_i \mathbf{z}_0 + \boldsymbol{\omega}_{a,i-1}^{i-1} \times \dot{q}_i \mathbf{z}_0\right)$
16: $\mathbf{a}_i^i \leftarrow \mathbf{R}_{i-1}^i \mathbf{a}_{i-1}^{i-1} + \dot{\boldsymbol{\omega}}_i^i \times \mathbf{p}_{i-1,i}^i + \boldsymbol{\omega}_i^i \times \left(\boldsymbol{\omega}_{a,i}^i \times \mathbf{p}_{i-1,i}^i\right)$
17: **else** i^{th} joint is prismatic
18: $\boldsymbol{\omega}_i^i \leftarrow \mathbf{R}_{i-1}^i \boldsymbol{\omega}_{i-1}^{i-1}$
19: $\boldsymbol{\omega}_{a,i}^i \leftarrow \mathbf{R}_{i-1}^i \boldsymbol{\omega}_{a,i-1}^{i-1}$
20: $\dot{\boldsymbol{\omega}}_i^i \leftarrow \mathbf{R}_{i-1}^i \dot{\boldsymbol{\omega}}_{i-1}^{i-1}$
21: $\mathbf{a}_i^i \leftarrow \mathbf{R}_{i-1}^i \mathbf{a}_{i-1}^{i-1} + \dot{\boldsymbol{\omega}}_i^i \times \mathbf{p}_{i-1,i}^i + \boldsymbol{\omega}_i^i \times \left(\boldsymbol{\omega}_{a,i}^i \times \mathbf{p}_{i-1,i}^i\right)$
$+ \boldsymbol{\omega}_{a,i}^i \times \dot{q}_i \mathbf{z}_i + \boldsymbol{\omega}_i^i \times \dot{q}_{a,i} \mathbf{z}_i + \ddot{q}_i \mathbf{z}_i$
22: **end if**
23: $[\mathbf{a}_{c,i}^i] \leftarrow \mathbf{a}_i^i \oplus \left(\dot{\boldsymbol{\omega}}_i^i \otimes [\mathbf{c}_i^i]\right) \oplus \left(\boldsymbol{\omega}_i^i \otimes \left(\boldsymbol{\omega}_{a,i}^i \otimes [\mathbf{c}_i^i]\right)\right)$
24: **end for**
25: Initialize $[\mathbf{f}_{N+1}^{N+1}]$, $[\mathbf{n}_{N+1}^{N+1}]$ and $\mathbf{R}_{N+1}^N$
26: **for** $i = N$ to 1 **do** ▷ Start of backward recursion
27: $[\mathbf{F}_i^i] \leftarrow [m_i] \odot [\mathbf{a}_{c,i}^i]$
28: $[\mathbf{f}_i^i] \leftarrow \left(\mathbf{R}_{i+1}^i \odot [\mathbf{f}_{i+1}^{i+1}]\right) \oplus [\mathbf{F}_i^i]$
29: $[\mathbf{n}_i^i] \leftarrow \left(\mathbf{R}_{i+1}^i \odot [\mathbf{n}_{i+1}^{i+1}]\right) \oplus \left([\mathbf{c}_i^i] \otimes [\mathbf{F}_i^i]\right)$
$\oplus \left(\mathbf{p}_{i-1,i}^i \otimes [\mathbf{f}_i^i]\right) \oplus \left([\mathbf{I}_i^i] \odot \dot{\boldsymbol{\omega}}_i^i\right) \oplus \left(\boldsymbol{\omega}_{a,i}^i \otimes \left([\mathbf{I}_i^i] \odot \boldsymbol{\omega}_i^i\right)\right)$
30: **if** i^{th} joint is revolute **then**
31: $[u_i] \leftarrow transpose\left([\mathbf{n}_i^i]\right) \odot \mathbf{z}_i$
32: **else** i^{th} joint is prismatic
33: $[u_i] \leftarrow transpose\left([\mathbf{f}_i^i]\right) \odot \mathbf{z}_i$
34: **end if**
35: **end for**

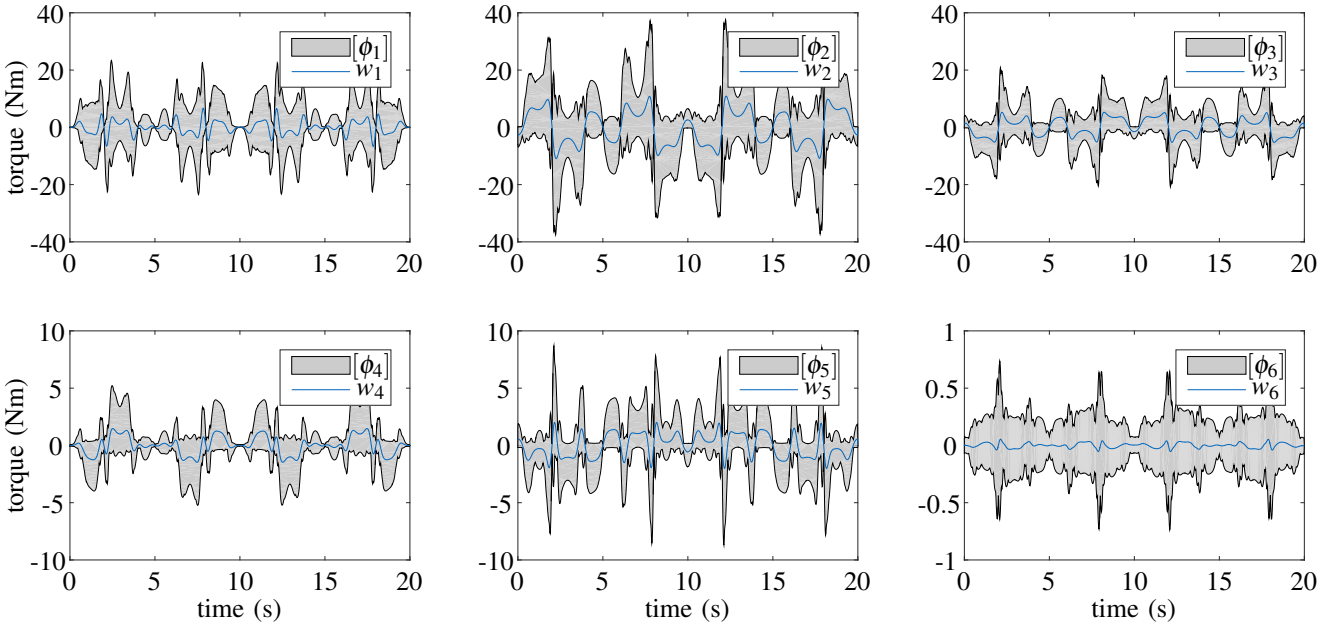

Figure 2. Simulation results showing the perturbation torques and the computed over-approximative sets for scenario 1.

how (9) can be computed efficiently by using the proposed algorithm. Indeed, by considering the following relation:

$$\begin{aligned}
&\mathbf{w}(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_a,\ddot{\mathbf{q}}_a,\mathbf{d},\Delta,\Delta_0) \overset{(6)}{=} \\
&\quad \mathbf{M}(\mathbf{q},\Delta)\ddot{\mathbf{q}}_a + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\Delta)\dot{\mathbf{q}}_a + \mathbf{g}(\mathbf{q},\Delta) \\
&\qquad - \left(\mathbf{M}(\mathbf{q},\Delta_0)\ddot{\mathbf{q}}_a + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\Delta_0)\dot{\mathbf{q}}_a + \mathbf{g}(\mathbf{q},\Delta_0)\right) - \mathbf{d} \\
&\quad = \mathit{IANEA}_g^*(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_a,\ddot{\mathbf{q}}_a,\mathbf{DHtab},\Delta) \\
&\qquad - \mathit{IANEA}_g^*(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_a,\ddot{\mathbf{q}}_a,\mathbf{DHtab},\Delta_0) - \mathbf{d},
\end{aligned}$$

and by introducing the non-degenerate interval for the dynamic parameters, we have

$$\begin{aligned}
&\mathbf{w}(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_a,\ddot{\mathbf{q}}_a,\mathbf{d},\Delta,\Delta_0) \subseteq \\
&\quad [\Phi] = \mathit{IANEA}_g^*(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_a,\ddot{\mathbf{q}}_a,\mathbf{DHtab},[\Delta]) \\
&\qquad \ominus \mathit{IANEA}_g^*(\mathbf{q},\dot{\mathbf{q}},\dot{\mathbf{q}}_a,\ddot{\mathbf{q}}_a,\mathbf{DHtab},\Delta_0) \ominus [\mathbf{d}]. \quad (12)
\end{aligned}$$

Remark 1: We check if the models generated by the proposed extension for prismatic joints allow the property in (2) to be fulfilled, by adopting a similar testing approach that authors in [8] use for revolute joints. We test the fully general symbolic models that can be generated by the algorithm for all combinations of revolute and prismatic joints (up to N=4 for the high computational burden) using the *Symbolic Math Toolbox*® in MATLAB. All tests have been successful.

Remark 2: Since IANEA_g^* has linear computational complexity in the number of links $\mathcal{O}(N)$, the same is preserved for computing (12). This algorithm qualifies for online use, due to its intrinsic computational efficiency and because there is no need for symbolic variable manipulation. Once $[\Phi]$ is efficiently computed, computing the measure of the worst-case disturbance is straightforwardly completed using (8).

V. Simulation Results

We verify the applicability of our proposed algorithm with a realistic simulation-based case study by considering the Schunk LWA 4-P robot. We use the following desired trajectory

$$\mathbf{q}_d(t) = \left(\frac{\pi}{2}, -\frac{\pi}{4}, \frac{\pi}{6}, -\frac{\pi}{6}, \frac{\pi}{4}, -\frac{\pi}{2}\right)^T \sin(\pi t/5)\sin(\pi t/2) \quad (13)$$

and the following initial conditions

$$\mathbf{q}(0) = 2\varepsilon\,(1,1,1,1,1,1)^T, \quad \dot{\mathbf{q}}(0) = (0,0,0,0,0,0)^T.$$

We further assume that the Euclidean norm of the trajectory tracking error is required to be less than $\varepsilon = 0.01\,rad$.

As suggested in [6], for computing (7) we use the following functions:

$$\varphi(t) = \left(\varphi_P + \varphi_I \int_0^t f(\|\tilde{\mathbf{q}}\|)\,d\tau\right),$$
$$\kappa(t) = \left(\kappa_P + \kappa_I \int_0^t f(\|\tilde{\mathbf{q}}\|)\,d\tau\right),$$

with

$$f(\|\tilde{\mathbf{q}}\|) = \begin{cases} 0 & \text{if } \|\tilde{\mathbf{q}}\| < \varepsilon, \\ \|\tilde{\mathbf{q}}\| & \text{otherwise,} \end{cases}$$

and for this application we select the following tuning parameters: $\kappa_P = 2$, $\kappa_I = 100$, $\mathbf{K}_r = 10\mathbf{I}$, $\varphi_P = 1$, $\varphi_I = 200$. We perform the numerical simulations using MATLAB/Simulink 2015b for two different scenarios (see Tab. II) with different (unknown for control design) true robot parameters Δ_a, Δ_b and without input disturbance for brevity. The nominal parameters from CAD data are collected in

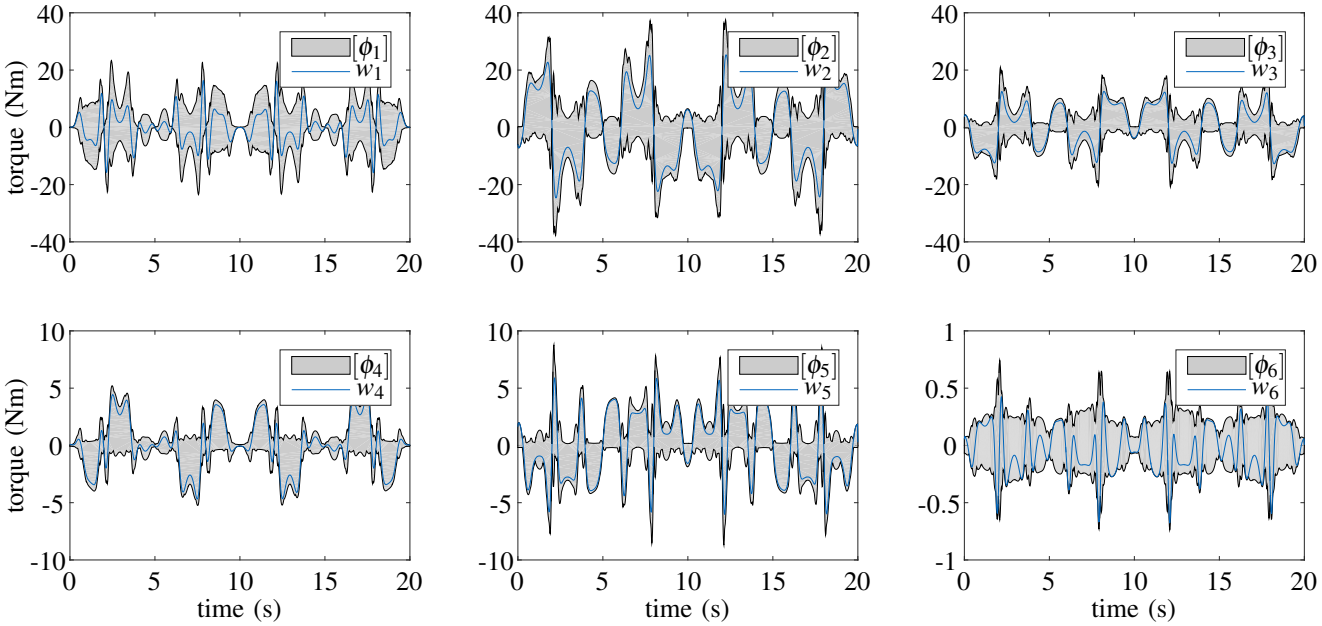

Figure 3. Simulation results showing the perturbation torques and the computed over-approximative sets for scenario 2.

Tab. I. For these simulations, we use the uncertain parameters of the last link and bounds known for control design collected in Tab. II.

We show the tracking performance for the considered scenarios for unmatched initial conditions in Fig. 4, where it can be seen that the desired tracking performance is ultimately met with no sign of chattering or discontinuity in the torque commands. The simulations allow us to show respectively in Fig. 2 and Fig. 3 the evolution over time of $\mathbf{w}$ and the corresponding $[\Phi]$, whose over-approximative nature is visible. As expected, no violation of supremums and infimums have been experienced. In Fig. 3 bounds are particularly tight, allowing us to conclude that our proposed approach is not excessively conservative.

We test the computational efficiency of the proposed approach using Simulink Real Time 2015b and a commercially available rapid control prototyping system[1] equipped with an Intel Core i7-3770K 3.5GHz and 4GB of RAM. We use the input signals to the controller recorded in the above mentioned simulations, and we sample them at $1ms$. We use the sampled signals to feed the overall controller implemented on the real time target machine to directly measure the total execution time on the embedded target machine for processing the control commands. In these tests the maximum experienced execution time is about $1.72 \cdot 10^{-5}\,s$, and the average execution time is $1.53 \cdot 10^{-5}\,s$

[1]SpeedGoat performance real-time target machine.

Table I
DH AND NOMINAL DYNAMIC PARAMETERS.

Link:	1	2	3	4	5	6
$\alpha_i\,(rad)$	$-\pi/2$	π	$-\pi/2$	$\pi/2$	$-\pi/2$	0
$a_i\,(m)$	0	0.35	0	0	0	0
$d_i\,(m)$	0	0	0	0.301	0	0.095
$\theta_i - q_i\,(rad)$	0	$-\pi/2$	$-\pi/2$	0	0	0
$m_i\,(kg)$	3.9	1.62	3.9	1	1.8	1.4
$c_{x,i}\,(m)$	0	-0.175	0	0	0	0
$c_{y,i}\,(m)$	0.018	0	0.018	-0.129	0.012	0
$c_{z,i}\,(m)$	0.019	-0.115	0.019	0.048	-0.012	0.05
$I_{xx,i}$ $(10^{-3}kgm^2)$	13.3	1.43	13.3	1.62	4	1.7
$I_{xy,i}$ $(10^{-3}kgm^2)$	0	0	0	-1.83	0	0
$I_{xz,i}$ $(10^{-3}kgm^2)$	0	0	0	0	0	0
$I_{yy,i}$ $(10^{-3}kgm^2)$	9.78	24.4	9.78	5.86	2.83	3.5
$I_{yz,i}$ $(10^{-3}kgm^2)$	0	0	0	0	0	0
$I_{zz,i}$ $(10^{-3}kgm^2)$	9.78	24.65	9.78	6.6	2.72	3.15

Table II
SCENARIOS OF TRUE (UNKNOWN FOR CONTROL DESIGN) DYNAMIC PARAMETERS AND BOUNDS USED FOR CONTROL DESIGN.

	Scenario 1 (Δ_a)	Scenario 2 (Δ_b)	$\underline{\Delta}$	$\overline{\Delta}$
$m_6\,(kg)$	2.1	2.8	1.4	2.8
$c_{x,6}\,(m)$	0	0.005	-0.005	0.005
$c_{y,6}\,(m)$	0	0.005	-0.005	0.005
$c_{z,6}\,(m)$	0.06	0.1	0.045	0.1
$I_{xx,6}\,(10^{-3}kgm^2)$	2.54	3.4	1.7	3.4
$I_{xy,6}\,(10^{-3}kgm^2)$	0	0	0	0
$I_{xz,6}\,(10^{-3}kgm^2)$	0	0	0	0
$I_{yy,6}\,(10^{-3}kgm^2)$	5.3	7	3.5	7
$I_{yz,6}\,(10^{-3}kgm^2)$	0	0	0	0
$I_{zz,6}\,(10^{-3}kgm^2)$	4.73	6.3	3.15	6.3

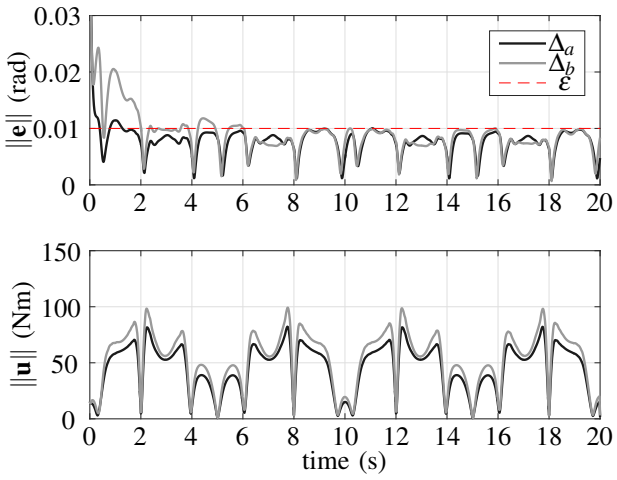

Figure 4. Tracking error and control commands for both simulated scenarios.

with a standard deviation of $2.88 \cdot 10^{-7} s$, which validates the on-line applicability of our proposed method.

VI. Conclusion

We have presented a computationally efficient approach for obtaining formally guaranteed over-approximative sets of joint torques/forces from uncertain dynamic parameters propagated through the system dynamics of rigid robots. Our approach simply combines the use of a recursive N-E algorithm with interval arithmetic operations and enables the on-line computation of interval-arithmetic-based robust controllers, thus eliminating the original computational drawbacks. This makes it possible to have a quick commissioning of robots powered by our modern interval-arithmetic-based robust controller when a large number of joints is also considered. Contrary to other robust control methods, our controller does not require any symbolic variable manipulation or time consuming non-formal steps for estimating bounds of closed-loop perturbations.

The successful application on a commercially available real-time target machine validates the computational effectiveness of our approach. Interesting further developments of this work may involve its application for controlling a real robot, the consideration of joint elasticity and its use for robust scaling of trajectories.

Acknowledgment

The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement number 608022.

References

[1] R. Featherstone and D. Orin, "Robot dynamics: equations and algorithms," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol. 1, 2000, pp. 826–834.

[2] C. Abdallah, D. Dawson, P. Dorato, and M. Jamshidi, "Survey of robust control for rigid robots," *IEEE Control Systems*, vol. 11, no. 2, pp. 24–30, Feb. 1991.

[3] W. Chung, L.-C. Fu, and S.-H. Hsu, *Handbook of Robotics*. Springer, 2008, book section Motion Control, pp. 133–159.

[4] L. Bascetta and P. Rocco, "Revising the robust-control design for rigid robot manipulators," *IEEE Trans. on Robotics*, vol. 26, no. 1, pp. 180–187, Feb. 2010.

[5] S. Zenieh and M. Corless, "Simple robust r - α tracking controllers for uncertain fully-actuated mechanical systems," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 119, no. 4, pp. 821–825, 1997.

[6] A. Giusti and M. Althoff, "Ultimate robust performance control of rigid robot manipulators using interval arithmetic," in *Proc. of the American Control Conference*, 2016, pp. 2995–3001.

[7] J. Luh, M. Walker, and R. Paul, "On-line computational scheme for mechanical manipulators," *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 102, pp. 468–474, 1980.

[8] A. De Luca and L. Ferrajoli, "A modified Newton-Euler method for dynamic computations in robot fault detection and control," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 3359–3364.

[9] C. J. J. Paredis, H. B. Brown, and P. K. Khosla, "A rapidly deployable manipulator system," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol. 2, 1996, pp. 1434–1439.

[10] A. Giusti and M. Althoff, "Automatic centralized controller design for modular and reconfigurable robot manipulators," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sept. 2015, pp. 3268–3275.

[11] J. Hollerbach, "Dynamic scaling of manipulator trajectories," *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 106, pp. 102–106, 1984.

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

[13] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2009.

[14] J. Denavit and R. Hartenberg, "A kinematic notation of lower-pair mechanisms based on matrices," *ASME Journal of Applied Mechanics*, vol. 22, pp. 215–221, 1955.