Feedback control over constrained robotic systems through the Udvadia-Kalaba approach

Ilia Milioshin

May 20, 2024

Contents

Chapter 1

Introduction

The advancement of microelectronics and sensor technologies has catalyzed a widespread integration of robotic systems into various domains. Manipulators, delivery robots, and flying drones have become ubiquitous, presenting developers and researchers with novel challenges in terms of robustness, adaptiveness, and synchronization. Among these challenges, synchronization stands out as a critical issue, especially given the proliferation and increasing complexity of such systems. In real-world applications, synchronization of robotic systems is crucial across various industries. For instance, in manufacturing assembly lines, synchronized robots ensure smooth production flow and maximize throughput. Similarly, in warehouse logistics, coordinated robotic systems optimize order fulfillment times and enhance overall productivity. Furthermore, collaborative robotics scenarios in construction projects benefit from synchronized robotic systems for tasks like concrete pouring and steel beam placement, improving efficiency and minimizing delays. A common challenge in this realm involves effectively controlling robots constrained by a shared object. This paper proposes a straightforward methodology to address such problems through the Udvadia-Kalaba[1] approach.

This proposed methodology gains particular significance within the context of modern physics simulation, derivation, and optimization libraries. These tools offer substantial computational speed, enabling efficient problem-solving. In this article, we leverage MuJoCo[2], Pinocchio[3], and ProxSuite[4] for simulation, robotic dynamics computation, and optimization, respectively. Pinocchio, in particular, emerges as a key tool for computing the dynamics of robotic systems. However, its limitation to open-loop physics models poses challenges for controlling and synchronizing multiple robots.

Previous solutions have often involved constructing models with pre-existing constraints or employing the KKT (Karush-Kuhn-Tucker) approach. However, these methods suffer from computational complexity and issues with constraint prioritization. The proposed methodology combines the strengths of both approaches, integrating physical grounding from the former and simplicity of application from the latter. Notably, it allows for manual fine-tuning of constraint priorities and boasts superior computational speed through auto code generation.

The implementation of the proposed methodology demonstrates significant advancements in the control and synchronization of robotic systems constrained by shared objects. Through the integration of robust physics simulation, precise robotic dynamics computation, and efficient optimization techniques, the method achieves remarkable outcomes across various simulation experiments. By leveraging advanced simulation, computation, and optimization techniques, the method has improved levels of efficiency, accuracy, and adaptability in robotic operations, paving the way for further advancements in automation and robotics technology.

In subsequent chapters, we delve deeper into various aspects of the proposed method. Chapter 2 provides an exhaustive review of recent literature, highlighting the existing landscape of solutions and their limitations. Chapter 3 elucidates the

methodology underlying the proposed approach, offering insights into its theoretical underpinnings. Implementation details and code snippets are presented in Chapter 4, demonstrating the practical application of the method. Chapter 5 undertakes a comparative analysis, pitting technique against established methods to gauge its efficacy. Finally, Chapter 6 summarizes findings, discusses implications, and outlines avenues for future research.

Chapter 2

Literature Review

The control of the interacting physical systems is a challenging task. To achieve high efficiency and precision in the said task, three problems should be solved, namely: the choice of the most appropriate mathematical model, the unified methodology for defining the interaction, and the well-defined virtual constraint. The first element is necessary to cover a wide range of physical systems. The second one is crucial to work with different mechanical connections. The last one is needed for stability and robustness analysis.

This literature review covers all these subproblems and explores them from the point of view of the work of Firdaus Udwadia and Robert Kalaba [1]. Section 2.1 considers a mathematical model was used in the recent studies. Section 2.2 reviews a rigorous methodology to define how physical systems can affect each other. Section 2.3 explores previous research in virtual constraint defining and analysis of different manifolds. Last Section 2.4 concludes this chapter and outlines the techniques that were chosen for further investigation.

The selection of the papers for this review aims to provide solid grounds for the choice of the methods in this study. Some of the papers were chosen as a

reference for theoretical background. Other papers were reviewed to study the existing techniques.

2.1 The mathematical model

This section contains a review of existing mathematical models for physical systems. It also compares the models in terms of numerical integration convenience and simplicity of defining the initial conditions. It is necessary to clarify that from now on this study considers only systems of rigid bodies with only inner stiffness. The detailed explanation of the reasons behind such a choice is presented later in Chapter 3.

The most popular relevant model is usually referred as canonical manipulator equation [1], [5], and [6]. Usually, it is called the canonical manipulator equation. The model can be formulated as

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q}) = \boldsymbol{\tau} \tag{2.1}$$

where $M(\mathbf{q})$ is known as inertia matrix, $C(\mathbf{q}, \dot{\mathbf{q}})$ is Centrifugal-Coriolis matrix, $g(\mathbf{q})$ is a gradient of potential forces, and $\mathbf{q}$, $\boldsymbol{\tau}$ are generalized coordinates and torques respectively. This equation is explored in detail in next Chapter 3.

Conversely, Udwadia [7] utilized a Hamiltonian view of the problem. This equation also manipulates deals with generalized coordinates but additionally it introduces generalized momentum and Hamiltonian. The choice of such variables makes this approach more convenient in terms of numerical integration.

The aforementioned equations are proven to be equivalent. Thus, both models can be utilized to achieve the main goal. However, the crucial difference

lies in the other plane.

In the vast majority of cases, both differential equations can be solved analytically. Therefore, it is necessary to use the numerical methods. Equation (2.1) has a second-time derivative. It enforces the construction of proper state variables for numerical integration. Conversely, the Hamiltonian equations have only the first derivative. Hence, they are more simple to use. Although the definition of initial conditions is harder for Hamiltonian equations, this problem has already been solved. So, the choice of the model fully depends on the specific discussed task.

Most of the reviewed papers rely on the canonical model (2.1). Therefore, for our project I adhere to the same equation.

2.2 The methodology to defining coupling

The second crucial problem in this paper is the definition of coupling between systems. In this section I show how this problem can be addressed and which methodology is preferable in the context of the discussed question. Importantly, here I only consider interaction through rigid bodies. The schematic representation of such coupling is shown in Fig. 2.1. Chapter 3 presents a reasons why such coupling is chosen.

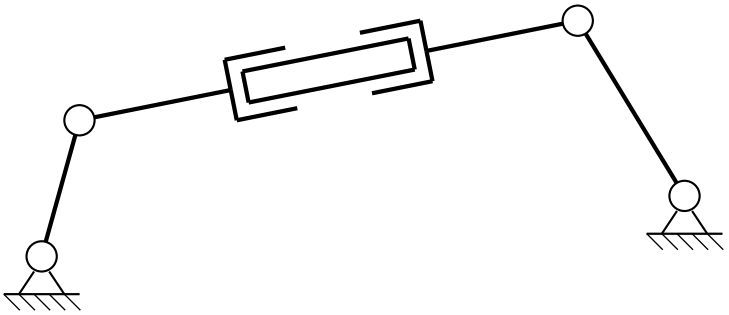

Fig. 2.1. Rigid body coupling

The first straightforward solution of this problem is deliberately include interaction in the mathematical model. In the discussed case it can be achieved by deriving $M(\mathbf{q})$, $C(\mathbf{q}, \dot{\mathbf{q}})$, and $g(\mathbf{q})$ in equation (2.1) as formulation of closed-loop system. It can be achieved if it is possible to formulate and analytically solve the following equation:

$$\varphi(\mathbf{q}, t) = 0 \tag{2.2}$$

From 2.2, the minimal set of generalized coordinates can be constructed, which implies rewriting the aforementioned parts of the canonical manipulator equation. Thus, the advantage of this method is the absence of the need for stabilization, because it is guaranteed by the model itself. Furthermore, this formulation can be used with a great range of control techniques. However, [8] demonstrates a lower computation speed of terms in comparison to open-loop algorithms [3]. Moreover, proposed closed-loop version requires a predefined description of whole system and works well only with systems that have a small number of DoFs. Thus, it cannot be used in real time in a dynamic environment.

The second discussed approach is analyzed in [9] and [10]. These articles propose to use a force cone to define a contact between a physical system and a solid surface. The actuation force can be formulated as

$$\boldsymbol{\lambda} = \sum_{i=1}^{N_d} \beta_i(\mathbf{n} + \mu\, \mathbf{d}_i),\ \beta_i \geqslant 0 \tag{2.3}$$

where $\mathbf{n}$ is a normal force, $\mathbf{d}_i$ is a tangent to contact vector, μ is the Coulomb friction coefficient, and N_d is the amount of used tangent vectors. This contact force later can be translated to joint space via respective Jacobian - $J^T(\mathbf{q})$.

This approach allows to emulate an interaction between physical systems via

a rigid body. It can be achieved by defining the motion of the connecting body through the force acting on it. However, this method cannot guarantee stability. Moreover, constructing a feedback loop in this case is a complex task.

Finally, the third approach to solve the discussed problem is defining the right constraint. In this study the impact of the rigid body on connected systems can be described by holonomic constraint, which is described by equation (2.2). Further, this equation can be used in the KKT (Karush-Kuhn-Tucker) technique to define a system with imposed constraints. Moreover, the equation (2.2) facilitates the construction of a stabilization mechanism. For instance, Baumgarte's approach can be used with the KKT method to achieve this goal. Nevertheless, this study does not rely on the aforementioned technique (KKT), instead, I utilized the Udwadia-Kalaba approach, which rewrites generalized accelerations obtained from equation (2.2) as affine transform. Details are shown in Chapter 3.

2.3 Virtual constraint

The last crucial subproblem is the rigorous definition of virtual constraints to ensure stabilization. It is especially important in the context of the chosen type of interaction. Therefore, this section reviews the ways to calculate the discrepancy between the current system state and the desired one in the form of equation (2.2) and explains the ways to to guarantee its convergence in the context of the form proposed by the Udwadia-Kalaba approach.

In the initial study [1] Ferdaus Udwadia and Robert Kalaba state that the straightforward differentiation of the holonomic constraint (2.2) produces instability during numerical integration (constraint drift). Thus the authors recommend to use Baumgarte's stabilization [11] via rewriting the linear second order differ-

ential equation with φ in the proposed matrix form.

Nevertheless, further in this study I show that it is hard to use this technique in the case of attitude tracking. The problem mostly is caused by the fact that the linear differential equations cannot capture the topology of the $\mathrm{SE}(3)$. Therefore, I was forced to use non-linear ones by applying the right pose difference formulation. Studies [12]–[16] describe methods to achieve this. The majority [13]–[16] of source explores the problem through utilizing rotations matrices. On the other hand, [12] uses quaternions for achieving the goal. Let's compare these solutions.

The study [12] states the advantage of quaternions over the rotations matrix. This research shows that the proposed method achieves the *exponential asymptotic stability* against *almost global stability* of techniques from [13], [14]. Furthermore, the quaternion-based method in the discussed study [12] presents the superiority of the analogous ones. The authors prove that their technique avoids the unwinding phenomenon, i.e., error converges to zero by the shortest path in $\mathbb{S}^3$.

The solution via utilizing the rotations matrices is presented in [13]–[16]. However, the aforementioned quaternion-based approach shows that basing on such structure methods has disadvantages. The naive error computation via matrices is a calculation of dot product between base vectors of current and desired orientation. As mentioned above, this approach can achieve only *almost global stability*. Nevertheless, studies [15], [16] rely on deep topology analysis of $SE(3)$ group. These research use Lie theory to construct a right discrepancy between attitudes. In Chapter 3 the *exponential convergence* of the method based on the studies presented in [15] and [16] is proven.

To conclude, in this study the rotation matrices are chosen to define an attitude error. My method draws upon the approach presented in [15] and [16]. The main reason for such a choice is the convenience of work in the context of

framework [3], which I use for numerical experiments.

2.4 Conclusion

In this chapter, the solutions to the necessary aforementioned subproblems were reviewed. The canonical manipulator equation (2.1) was chosen as the base for investigations because it is most widely used in previous studies. To define coupling between systems, I decided to use the Udwadia-Kalaba approach and the form of coupling proposed in the latter. Finally, for building a stable virtual constraint the discrepancy based on Lie theory was taken.

Chapter 3

Methodology

This chapter will present a detailed explanation of the principles underlying the proposed methodology. The section 3.1 offers a concise overview of rigid-body systems and constraints that can be imposed on them. The next section 3.2 introduces the Udwadia-Kalaba approach and physical principles that stays behind it. Further in Section 3.3, this technique is applied to solve the discussed problem. Then Section 3.4 use the Udwadia-Kalaba approach to construct a control law. Finally, Section 3.5 contains thoughts about prioritization constraint task over control one.

3.1 Rigid-body systems and constraints

The most common class of physical systems that can be observed in the real life cases is the rigid-body one. Some examples of such systems are demonstrated in Figure 3.1. The key feature of rigid-body systems is non-deformable parts that forms it. With these limitations only an inner stiffness can be described.

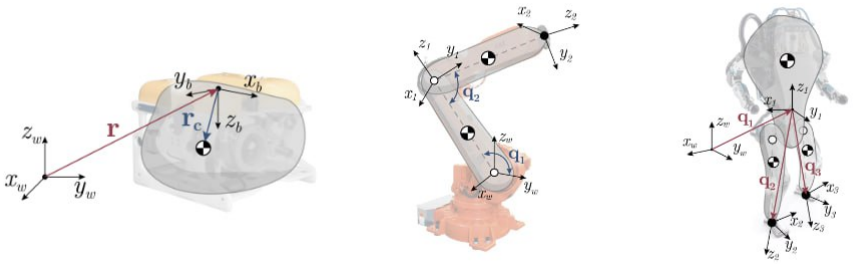

Fig. 3.1. Examples of rigid-body systems

Nevertheless, the mathematical tools to work with rigid-body systems was introduced in XVIII-th century by Newton and Euler works. Using this approach it is easy to formulate a set f deferential equations that describes the system behavior. However, in this study the another (more convenient) technique is utilized. It is called the least action principle. The formulation is the following equation

$$\min_{\mathbf{q}} \quad \int_{t_1}^{t_2} L(\mathbf{q}(t), \mathbf{v}(t), t)dt \tag{3.1}$$

where L is Lagrangian of the system, and $\mathbf{q}$, $\mathbf{v}$ are functions of the generalized coordinates and velocities respectively. The solution of the variational problem (3.1) is the Euler-Lagrange differential equations. The solution through utilizing the Lagrangian can be easily generalized to all rigid-body systems. This generalization is aforementioned the canonical manipulator equation (2.1).

In this chapter the convenient representation of this equation is used. The main idea of rewriting is combination of the inertial forces with non-inertial ones.

$$M\dot{\mathbf{v}} = \mathbf{Q} \tag{3.2}$$

In the above equation $\mathbf{Q} = -C(\mathbf{q}, \mathbf{v})\dot{\mathbf{q}} - g(\mathbf{q}) + \boldsymbol{\tau}$, $M \succcurlyeq 0$ is inertia

matrix, C is Centrifugal-Coriolis matrix, g is gradient of conservative forces. The dependency from coordinates and velocities is amended for convenience. In the most generalized case $\dim \mathbf{q} = n_q \neq n_v = \dim \mathbf{v}$.

The aforementioned description is applicable for open and closed loop systems. The closed one is described in Figure 2.1. However, as mentioned above utilizing only equation 3.2 to handle such system is not efficient. Thus, the approach via constraints described by equation (2.2) is more preferable. It can be inserted in (3.1), which leads to the following variational problem,

$$\min_{\mathbf{q}} \quad \int_{t_1}^{t_2} [L(\mathbf{q}(t), \mathbf{v}(t), t) - \boldsymbol{\lambda}^T \varphi(\mathbf{q}(t), t)] dt \tag{3.3}$$

The solution the above equation can be expressed in terms of the KKT matrix in the following way,

$$\begin{bmatrix} M & J^T \\ J & 0 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{v}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \\ -\dot{J}\mathbf{v} \end{bmatrix}, \; J = \frac{\partial \boldsymbol{\varphi}}{\partial \mathbf{q}} \tag{3.4}$$

The downside of the equation (3.4) is that it does not support non-holonomic constraints generally. This type of constraints can be defined as

$$\varphi(\mathbf{q}, \mathbf{v}, t) = 0 \tag{3.5}$$

Therefore, the dependency from generalized velocity makes the KKT approach not applicable. The handle of such constraints is crucial for defining a control. Hence, it is necessary to use another technique.

3.2 The Udwadia-Kalaba approach

The main differences of the Udwadia-Kalaba approach from the method mentioned above are the another view on constraints, and utilizing different physical principle.

Let's start from reviewing the constraints form in the discussed technique. It can be derived from holonomic and non-holonomic type by differentiating over time, and transforming to an affine form. The general equation is the following

$$A(\mathbf{q}, \mathbf{v}, t)\dot{\mathbf{v}} = \mathbf{b}(\mathbf{q}, \mathbf{v}, t) \tag{3.6}$$

The above equation can be used in the Gauss least constraint principle. For further convenience the arguments of matrix functions is amended. It leads to the following optimization problem:

$$\begin{aligned} \min_{\dot{\mathbf{v}}} \quad & [\dot{\mathbf{v}} - \mathbf{a}]^T M [\dot{\mathbf{v}} - \mathbf{a}] \\ \text{s.t.} \quad & A\dot{\mathbf{v}} = \mathbf{b} \\ & \mathbf{a} = M^{-1}\mathbf{Q} \end{aligned} \tag{3.7}$$

This optimization can be solved analytically as Ferdaus Udwadia and Robert Kalaba demonstrated [1]. This solution is shown below.

$$\dot{\mathbf{v}} = \mathbf{a} + M^{-1/2}(AM^{-1/2})^{+}(\mathbf{b} - AM^{-1}\mathbf{Q}) \tag{3.8}$$

where $[\cdot]^{+}$ is the Moore-Penrose inverse, and $M^{\pm 1/2} = W\Lambda^{\pm 1/2}W^T$, W is the orthogonal matrix of eigen vectors.

In case of positive semi-definiteness of M the equation (3.8) is not applicable because M^{-1}, $M^{-1/2}$ cannot exist. The Firdaus Udwadia and Aaron Schutte

demonstrates [17] a methodology to bypass this problem by replacing M, and $\mathbf{a}$ by the following quantities respectively

$$\begin{aligned} M_A &= M + A^+ A \succ 0 \\ \mathbf{a}_A &= M_A^{-1}\mathbf{Q} \end{aligned} \tag{3.9}$$

Utilizing the mentioned quantities it is possible to define constraints force over the wide range of rigid-body systems. These forces can be expressed in the following manner

$$\mathbf{Q}_C = M^{1/2}(AM^{-1/2})^+(\mathbf{b} - AM^{-1}\mathbf{Q}) \tag{3.10}$$

Hence, the Udwadia-Kalaba approach is applicable for emulation a of rigid body constraint in computation easily. Only the problem with rigorous defining A and $\mathbf{b}$ remains.

3.3 Defying rigid body constraint

As mentioned above the discussed coupling between physical systems is created by some rigid body. This type of mathematical object can be described by the following identity for any two points

$$\|\mathbf{r}_A(t) - \mathbf{r}_B(t)\| \equiv \text{const} \tag{3.11}$$

From the above equation kinematic laws can be directly derived. For further investigations it is necessary to introduce a point velocity of the rigid body

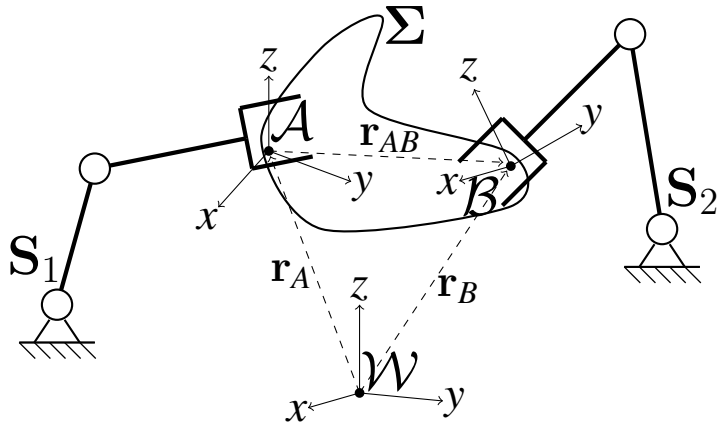

Fig. 3.2. Rigid body ($\boldsymbol{\Sigma}$) coupling with frames

$$\begin{aligned} \boldsymbol{v}_B &= \boldsymbol{v}_A + \boldsymbol{\omega}_A \times \mathbf{r}_{AB} \\ \boldsymbol{\omega}_A &= \boldsymbol{\omega}_B = \boldsymbol{\omega}_\Sigma \end{aligned} \tag{3.12}$$

Equation (3.12) describes motion of points, which shown in the Fig. 3.2. $\boldsymbol{\omega}_X$ denotes the angular velocity of some *fixed frame* with the origin at X. Further in this chapter the below notation is used for for attitude (member of $\mathrm{SE}(3)$) description:

$$T_{\mathcal{W}}^{\mathcal{A}} = \begin{bmatrix} R_{\mathcal{W}}^{\mathcal{A}} & \mathbf{p}_{\mathcal{A}} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathrm{SE}(3) \tag{3.13}$$

where $R_{\mathcal{W}}^{\mathcal{A}} \in \mathrm{SO}(3)$ represents basis of $\mathcal{A}$ in $\mathcal{W}$, and $\mathbf{p}_{\mathcal{A}}$ is origin of $\mathcal{A}$ in terms of $\mathcal{W}$.

Now it is possible to define a holonomic constraint that emulates a coupling through a rigid body. Let E_1, E_2 are some *fixed frames* of S_1, S_2 respectively, see Fig. 3.2; A, B are some points of the rigid body Σ. Also, let's suppose that for any $t \geq 0$, A and E_1 correspond. Thus,

$$\varphi(\mathbf{q}) = \mathbf{e}_{\mathrm{SE}(3)}(T_{\mathcal{W}}^{\mathcal{E}_1} \cdot T_A^B, T_{\mathcal{W}}^{\mathcal{E}_2}) = \mathbf{0} \tag{3.14}$$

where $\mathbf{e}_{\mathrm{SE}(3)} : \mathrm{SE}(3) \times \mathrm{SE}(3) \to \mathbb{R}^6$ is defined as

$$\mathbf{e}_{\mathrm{SE}(3)}(T_1, T_2) = \begin{bmatrix} \mathbf{p}_2 - \mathbf{p}_1 \\ \log(R_1^T R_2) \end{bmatrix} \tag{3.15}$$

$\log$ in 3.15 is a matrix logarithm defined in APPENDIX REF TODO. The choice of such non trivial function is inspired by [15] and [16].

Equation (3.14) should be differentiated twice to obtain A and $\mathbf{b}$. However, even if matrix logarithm had a simple second derivative. the Udwadia-Kalaba approach cannot guarantee a stability. As Ferdaus Udwadia and Robert Kalaba showed [18] the straightforward application of technique suffers from constraint drift during numerical integration. Hence, some stabilization method should be used. The study [18] offers the Baumgarte's technique. For holonomic constraint it looks as

$$A\dot{\mathbf{v}} = \mathbf{b} - K_d\dot{\varphi} - K_p\varphi; \quad K_p, K_d \in \mathbb{R} \tag{3.16}$$

Although the Baumgarte's stabilization is relatively simple it is also not applicable in the context of equation (3.14). Therefore, the more advance method should be used.

Udwadia REF demonstrated that if some stable differential equation can be converted to (3.6) form, then the Udwadia-Kalaba approach with such constraint is stable. Thus, it is enough to find some stable kinematic differential equation. Let's consider the following equation

$$\begin{bmatrix} \dot{\boldsymbol{\upsilon}}_{E_2} - \dot{\boldsymbol{\upsilon}}_B \\ \dot{\boldsymbol{\omega}}_{E_2}^{\mathcal{E}_2} - \dot{\boldsymbol{\omega}}_B^{\mathcal{E}_2} \end{bmatrix} + K_d \begin{bmatrix} \boldsymbol{\upsilon}_{E_2} - \boldsymbol{\upsilon}_B \\ \boldsymbol{\omega}_{E_2}^{\mathcal{E}_2} - \boldsymbol{\omega}_B^{\mathcal{E}_2} \end{bmatrix} + K_p \cdot \mathrm{e}_{\mathrm{SE}(3)}(T_{\mathcal{W}}^{\mathcal{E}_1} \cdot T_A^B, T_{\mathcal{W}}^{\mathcal{E}_2}) = 0, \quad K_p, K_d \in \mathbb{R} \tag{3.17}$$

where $\boldsymbol{\omega}_X^{\mathcal{Y}}$ states the angular velocity of some *fixed frame* at origin X expressed in the frame $\mathcal{Y}$ coordinates. For simplicity let $\boldsymbol{\omega}_{E_2}^{\mathcal{E}_2} - \boldsymbol{\omega}_B^{\mathcal{E}_2} = \boldsymbol{\omega}_d^{\mathcal{E}_2}$, $\boldsymbol{\upsilon}_{E_2} - \boldsymbol{\upsilon}_B = \boldsymbol{\upsilon}_d$, $R_{\mathcal{X}}^{\mathcal{Y}} \equiv R_X$ represents basis of $\mathcal{Y}$ in frame $\mathcal{X}$ at origin X. Hence, equation (3.17) transforms to

$$\begin{bmatrix} \dot{\boldsymbol{\upsilon}}_d \\ \dot{\boldsymbol{\omega}}_d^{\mathcal{E}_2} \end{bmatrix} + K_d \begin{bmatrix} \boldsymbol{\upsilon}_d \\ \boldsymbol{\omega}_d^{\mathcal{E}_2} \end{bmatrix} + K_p \begin{bmatrix} \mathbf{r}_d \\ \log(R_B^T R_{E_2}) \end{bmatrix} = 0 \tag{3.18}$$

Theorem 1. *Let $\boldsymbol{\omega}^{\mathcal{D}}$ is an angular velocity of some frame ($\mathcal{X}$) represented in desired frame $\mathcal{D}$, $\boldsymbol{\omega}_D^{\mathcal{D}}$ is an angular velocity of desired frame $\mathcal{D}$, R is orientation of frame $\mathcal{X}$, and R_d is orientation of frame $\mathcal{D}$. R and R_d are described in the world frame $\mathcal{W}$. Therefore,*

$$(\dot{\boldsymbol{\omega}}_D^{\mathcal{D}} - \dot{\boldsymbol{\omega}}^{\mathcal{D}}) + K_d(\boldsymbol{\omega}_D^{\mathcal{D}} - \boldsymbol{\omega}^{\mathcal{D}}) + K_p \log(R^T R_d) = 0, \quad K_p, K_d > 0 \tag{3.19}$$

is asymptotic stable. $\log X$ is a matrix logarithm.

Proof. Let $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_D^{\mathcal{D}} - \boldsymbol{\omega}^{\mathcal{D}}$, $\mathbf{e} = \log \tilde{R}$, and $\tilde{R} = R^T R_d$. Hence, equation 3.19 can be rewritten in

$$\dot{\tilde{\boldsymbol{\omega}}} + K_d \tilde{\boldsymbol{\omega}} + K_p \mathbf{e} = 0 \tag{3.20}$$

From REF TO APPENDIX

$$\frac{d}{dt}\log\tilde{R} = J_r^{-1}(\mathbf{e})[\tilde{R}^T\dot{\tilde{R}}]^\vee \tag{3.21}$$

where $J_r^{-1}(\mathbf{e})$ and $[.]^\vee$ are defined in REF TO APPENDIX. Proof of (3.21) can be found in Lemma 2 of [16].

Let's simplify equation (3.21)

$$\begin{aligned}
\dot{\mathbf{e}} &= J_r^{-1}(\mathbf{e})[(R^TR_d)^T(\dot{R}^TR_d + R^T\dot{R}_d)]^\vee = \\
&= J_r^{-1}(\mathbf{e})[R_d^TR\dot{R}^TR_d + R_d^T\dot{R}_d]^\vee = \\
&= J_r^{-1}(\mathbf{e})[R_d^T\hat{\boldsymbol{\omega}}^TR_d + \hat{\boldsymbol{\omega}}_D^{\mathcal{D}}]^\vee = \\
&= J_r^{-1}(\mathbf{e})[\hat{\boldsymbol{\omega}}_D^{\mathcal{D}} - \hat{\boldsymbol{\omega}}^{\mathcal{D}}]^\vee = \\
&= J_r^{-1}(\mathbf{e})\tilde{\boldsymbol{\omega}}
\end{aligned} \tag{3.22}$$

where $\hat{[.]}$ (or $[.]^\wedge$) is described in REF TO APPENDIX.

Let the Lyapunov candidate is $V = \frac{1}{2}\tilde{\boldsymbol{\omega}}^T\tilde{\boldsymbol{\omega}} + \frac{K_p}{2}\mathbf{e}^T\mathbf{e}$. Therefore,

$$\begin{aligned}
\dot{V} &= \tilde{\boldsymbol{\omega}}^T\dot{\tilde{\boldsymbol{\omega}}} + K_p\mathbf{e}^T\dot{\mathbf{e}} = \\
&= \tilde{\boldsymbol{\omega}}^T[-K_d\tilde{\boldsymbol{\omega}} - K_p\mathbf{e}] + K_p\mathbf{e}^TJ_r^{-1}(\mathbf{e})\tilde{\boldsymbol{\omega}} = \\
&= -K_D\tilde{\boldsymbol{\omega}}^T\tilde{\boldsymbol{\omega}} - K_p\tilde{\boldsymbol{\omega}}^T\mathbf{e} + K_p\mathbf{e}^T\tilde{\boldsymbol{\omega}} = \\
&= -K_D\tilde{\boldsymbol{\omega}}^T\tilde{\boldsymbol{\omega}} < 0
\end{aligned} \tag{3.23}$$

The transition to the third line of (3.23) is conditioned by corollary of Lemma 3 from [16]. It states that $\mathbf{e}^TJ_r^{-1}(\mathbf{e}) = \mathbf{e}^T$.

Let $\mathcal{I} = \{\begin{bmatrix}\tilde{\boldsymbol{\omega}}^T & \mathbf{e}^T\end{bmatrix}^T : \dot{V} = 0\}$. If $\tilde{\boldsymbol{\omega}} = \mathbf{0}$ and $\mathbf{e} \neq \mathbf{0}$, then $\dot{\tilde{\boldsymbol{\omega}}} \neq \mathbf{0}$ by equation (3.20). Hence, $\mathcal{I}$ does not contain non-trivial system trajectories. By applying LaSalle's Invariance Theorem differential equation (3.19) is asymptotic stable.

□

Equation (3.18) can be decomposed to two independent differential equations. The linear part is

$$\dot{\boldsymbol{v}}_d + K_d \boldsymbol{v}_d + K_p \mathbf{r}_d = 0 \tag{3.24}$$

The above equation (3.24) is asymptotic stable for any K_p and K_d such that the characteristics equation has a negative roots (real part). The best convergence can be achieved if $K_p = \omega^2$ and $K_d = 2\omega$. The angular part is

$$\dot{\boldsymbol{\omega}}_d^{\mathcal{D}} + K_d \boldsymbol{\omega}_d^{\mathcal{D}} + K_p \log(R_B^T R_{E_2}) = 0 \tag{3.25}$$

It is asymptotic stable due to Th. 1.

Now I can formulate A and $\mathbf{b}$ from equation (3.17). Firstly, let's define a velocities

$$\begin{aligned} \begin{bmatrix} \boldsymbol{v}_{E_2} \\ \boldsymbol{\omega}_{E_2}^{\mathcal{E}_2} \end{bmatrix} &= \begin{bmatrix} J_{E_2,v} \\ R_{E_2}^T J_{E_2,\omega} \end{bmatrix} \mathbf{v} \\ \begin{bmatrix} \boldsymbol{v}_B \\ \boldsymbol{\omega}_B^{\mathcal{E}_2} \end{bmatrix} &= \begin{bmatrix} J_{E_1,v} \\ R_{E_2}^T J_{E_1,\omega} \end{bmatrix} \mathbf{v} + \begin{bmatrix} \boldsymbol{\omega}_{E_1} \times \mathbf{r}_{AB} \\ \mathbf{0} \end{bmatrix} \end{aligned} \tag{3.26}$$

where $J_{X,v}$, $J_{X,\omega}$ are linear and angular velocity of frame $\mathcal{X}$ with origin at X respectively. The equation above can be simplified to

$$\begin{bmatrix} \boldsymbol{v}_d \\ \boldsymbol{\omega}_d^{\mathcal{E}_2} \end{bmatrix} = \begin{bmatrix} J_{d,v} + [R_{E_1} \mathbf{r}_{AB}^{\mathcal{A}}]^{\wedge} J_{E_1,\omega} \\ R_{E_2}^T J_{d,\omega} \end{bmatrix} \mathbf{v} = \begin{bmatrix} \tilde{J}_{d,v} \\ \tilde{J}_{d,\omega} \end{bmatrix} \mathbf{v},\ J_{d,*} = J_{E_2,*} - J_{E_1,*} \tag{3.27}$$

Now it is easy to express A and b by differencing equation (3.27) and adding PD part:

$$
\begin{aligned}
A &= \begin{bmatrix} \tilde{J}_{d,v} \\ \tilde{J}_{d,\omega} \end{bmatrix} \\
b &= -\begin{bmatrix} \dot{\tilde{J}}_{d,v} \\ \dot{\tilde{J}}_{d,\omega} \end{bmatrix} \mathbf{v} - K_d \begin{bmatrix} \boldsymbol{\upsilon}_d \\ \boldsymbol{\omega}_d^{\mathcal{E}_2} \end{bmatrix} - K_p \mathbf{e}_{\mathrm{SE}(3)}(T_{\mathcal{W}}^{\mathcal{E}_1} \cdot T_{\mathcal{A}}^{\mathcal{B}}, T_{\mathcal{W}}^{\mathcal{E}_2})
\end{aligned} \tag{3.28}
$$

where $\dot{\tilde{J}}_{d,v}$ is

$$
\begin{aligned}
\dot{\tilde{J}}_{d,v} &= \dot{J}_{d,v} + \left(\frac{d}{dt} [R_{E_1} \mathbf{r}_{AB}^{\mathcal{A}}]^{\wedge} \right) J_{E_1,\omega} + [R_{E_1} \mathbf{r}_{AB}^{\mathcal{A}}]^{\wedge} \dot{J}_{E_1,\omega} = \\
&= \dot{J}_{d,v} + \hat{\boldsymbol{\omega}}_{E_1} \hat{\mathbf{r}}_{AB} J_{E_1,\omega} + \hat{\mathbf{r}}_{AB} \dot{J}_{E_1,\omega}
\end{aligned} \tag{3.29}
$$

and $\dot{\tilde{J}}_{d,\omega}$ is

$$
\begin{aligned}
\dot{\tilde{J}}_{d,\omega} &= \dot{R}_{E_2}^T J_{d,\omega} + R_{E_2}^T \dot{J}_{d,\omega} = \\
&= [\hat{\boldsymbol{\omega}}_{E_2} R_{E_2}]^T J_{d,\omega} + R_{E_2}^T \dot{J}_{d,\omega} = \\
&= R_{E_2}^T [\dot{J}_{d,\omega} - \hat{\boldsymbol{\omega}}_{E_2} J_{d,\omega}]
\end{aligned} \tag{3.30}
$$

Now it is possible to define full algorithm. Let $\mathbf{S}_c$ is combined physical system. It's union of two rigid-body systems, namely: $\mathbf{S}_1$ and $\mathbf{S}_2$. For reference, see Fig. 3.2. The dynamics of $\mathbf{S}_c$ is

$$
\begin{bmatrix} M_1 & \mathbf{0} \\ \mathbf{0} & M_2 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{v}}_1 \\ \dot{\mathbf{v}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \end{bmatrix} \tag{3.31}
$$

If $\mathbf{q} = \begin{bmatrix} \mathbf{q}_1^T & \mathbf{q}_2^T \end{bmatrix}^T$ and $\mathbf{v} = \begin{bmatrix} \mathbf{v}_1^T & \mathbf{v}_2^T \end{bmatrix}^T$ than A and $\mathbf{b}$ can be found by the following algorithm

Algorithm 3.3.1 Computing A and $\mathbf{b}$ for rigid body constraint

function GETRIGIDBODYCONSTRAINT(combined system $\mathbf{S}_c$, $\mathbf{q}$, $\mathbf{v}$, E_1, E_2, $T_{\mathcal{A}}^{\mathcal{B}}$) :
    Compute attitudes $T_{\mathcal{W}}^{\mathcal{E}_1}$ and $T_{\mathcal{W}}^{\mathcal{E}_2}$ for $\mathbf{S}_c$ at state $\mathbf{q}$
    Compute velocities $\boldsymbol{v}_{E_1}$, $\boldsymbol{v}_{E_2}$, $\boldsymbol{\omega}_{E_1}$, and $\boldsymbol{\omega}_{E_2}$ for $\mathbf{S}_c$ at state $\mathbf{q}$, $\mathbf{v}$
    Compute Jacobians $J_{E_1,v}$, $J_{E_1,\omega}$, $J_{E_2,v}$, and $J_{E_2,\omega}$ for $\mathbf{S}_c$ at state $\mathbf{q}$
    Compute Jacobians time derivatives $\dot{J}_{E_1,v}$, $\dot{J}_{E_1,\omega}$, $\dot{J}_{E_2,v}$, and $\dot{J}_{E_2,\omega}$ for $\mathbf{S}_c$ at state $\mathbf{q}$, $\mathbf{v}$
    Obtain $J_{d,*} = J_{E_2,*} - J_{E_1,*}$ and $\dot{J}_{d,*} = \dot{J}_{E_2,*} - \dot{J}_{E_1,*}$, where $*$ is v or ω
    Calculate $\tilde{J}_{d,v}$ and $\tilde{J}_{d,\omega}$ by equation (3.27)
    Calculate $\dot{\tilde{J}}_{d,v}$ and $\dot{\tilde{J}}_{d,\omega}$ by equation (3.29) and (3.30) respectively
    Compute A and $\mathbf{b}$ by equation (3.28)
    return A, $\mathbf{b}$
end function

3.4 Control through the Udwadia-Kalaba approach

As shown above, the Udwadia-Kalaba approach can be used to impose both holonomic and non-holonomic constraints. Therefore, it can be applied for constructing controls. Let's consider two possible scenarios: controlling a system frame, or controlling the fixed frame of a rigid body connecting systems.

For both cases the desired law of motion described by the following quantities: $T_{\mathcal{W}}^{\mathcal{D}}$, $\begin{bmatrix} \boldsymbol{v}_D^T & [\boldsymbol{\omega}_D^{\mathcal{D}}]^T \end{bmatrix}^T$, $\begin{bmatrix} \mathbf{a}_D^T & [\boldsymbol{\epsilon}_D^{\mathcal{D}}]^T \end{bmatrix}^T$ are desired attitude, velocity and acceleration respectively. These values should not be a contradictory for sure.

Now it is possible to define control for some fixed frame $\mathcal{E}$ of system. The holonomic is

$$\begin{bmatrix} \mathbf{a}_D^{\mathcal{D}} - \dot{\boldsymbol{v}}_E \\ \boldsymbol{\epsilon}_D^{\mathcal{D}} - \dot{\boldsymbol{\omega}}_E^{\mathcal{D}} \end{bmatrix} + K_d \begin{bmatrix} \boldsymbol{v}_D - \boldsymbol{v}_E \\ \boldsymbol{\omega}_D^{\mathcal{D}} - \boldsymbol{\omega}_E^{\mathcal{D}} \end{bmatrix} + K_p \mathbf{e}_{\mathrm{SE}(3)}(T_{\mathcal{W}}^{\mathcal{E}}, T_{\mathcal{W}}^{\mathcal{D}}) = \mathbf{0} \tag{3.32}$$

It can be simplified by substitution $\boldsymbol{v}_d = \boldsymbol{v}_D - \boldsymbol{v}_E$ and $\boldsymbol{\omega}_d^{\mathcal{D}} = \boldsymbol{\omega}_D^{\mathcal{D}} - \boldsymbol{\omega}_E^{\mathcal{D}}$ to

$$\begin{aligned} &\begin{bmatrix} -J_{E,v} \\ -R_D^T J_{E,\omega} \end{bmatrix} \dot{\mathbf{v}} + \begin{bmatrix} -\dot{J}_{E,v} \\ -R_D^T [\dot{J}_{E,\omega} - \hat{\boldsymbol{\omega}}_D J_{E,\omega}] \end{bmatrix} \mathbf{v} + \\ &+ \begin{bmatrix} \mathbf{a}_D^{\mathcal{D}} \\ \boldsymbol{\epsilon}_D^{\mathcal{D}} \end{bmatrix} + K_d \begin{bmatrix} \boldsymbol{v}_d \\ \boldsymbol{\omega}_d^{\mathcal{D}} \end{bmatrix} + K_p \mathbf{e}_{\mathrm{SE}(3)}(T_{\mathcal{W}}^{\mathcal{E}}, T_{\mathcal{W}}^{\mathcal{D}}) = \mathbf{0} \end{aligned} \tag{3.33}$$

Hence, the A and $\mathbf{b}$ are following

$$\begin{aligned} A &= \begin{bmatrix} -J_{E,v} \\ -R_D^T J_{E,\omega} \end{bmatrix} \\ \mathbf{b} &= \begin{bmatrix} \dot{J}_{E,v} \\ R_D^T [\dot{J}_{E,\omega} - \hat{\boldsymbol{\omega}}_D J_{E,\omega}] \end{bmatrix} \mathbf{v} - \begin{bmatrix} \mathbf{a}_D^{\mathcal{D}} \\ \boldsymbol{\epsilon}_D^{\mathcal{D}} \end{bmatrix} - K_d \begin{bmatrix} \boldsymbol{v}_d \\ \boldsymbol{\omega}_d^{\mathcal{D}} \end{bmatrix} - K_p \mathbf{e}_{\mathrm{SE}(3)}(T_{\mathcal{W}}^{\mathcal{E}}, T_{\mathcal{W}}^{\mathcal{D}}) \end{aligned} \tag{3.34}$$

Thus, the common algorithm is

Algorithm 3.4.1 Computing A and $\mathbf{b}$ for control of fixed point

1: **function** GETFIXEDCONTROLCONSTRAINT(combined system $\mathbf{S}_c$, $\mathbf{q}$, $\mathbf{v}$, E, $T_{\mathcal{W}}^{\mathcal{D}}$, $\begin{bmatrix} \boldsymbol{v}_D^T & [\boldsymbol{\omega}_D^{\mathcal{D}}]^T \end{bmatrix}^T$, $\begin{bmatrix} \mathbf{a}_D^T & [\boldsymbol{\epsilon}_D^{\mathcal{D}}]^T \end{bmatrix}^T$) :

2: Compute attitudes $T_{\mathcal{W}}^{\mathcal{E}}$ for $\mathbf{S}_c$ at state $\mathbf{q}$

3: Compute velocities $\boldsymbol{v}_E$ and $\boldsymbol{\omega}_E$ for $\mathbf{S}_c$ at state $\mathbf{q}$, $\mathbf{v}$

4: Compute Jacobians $J_{E,v}$ and $J_{E,\omega}$ for $\mathbf{S}_c$ at state $\mathbf{q}$

5: Compute Jacobians time derivatives $\dot{J}_{E,v}$ and $\dot{J}_{E,\omega}$ for $\mathbf{S}_c$ at state $\mathbf{q}$, $\mathbf{v}$

6: Compute A and $\mathbf{b}$ by equation (3.34)

7: **return** A, $\mathbf{b}$

8: **end function**

Now let's consider the case of controlling point fixed on rigid body that couples systems. The schematic representation of it is demonstrated on Fig. 3.3.

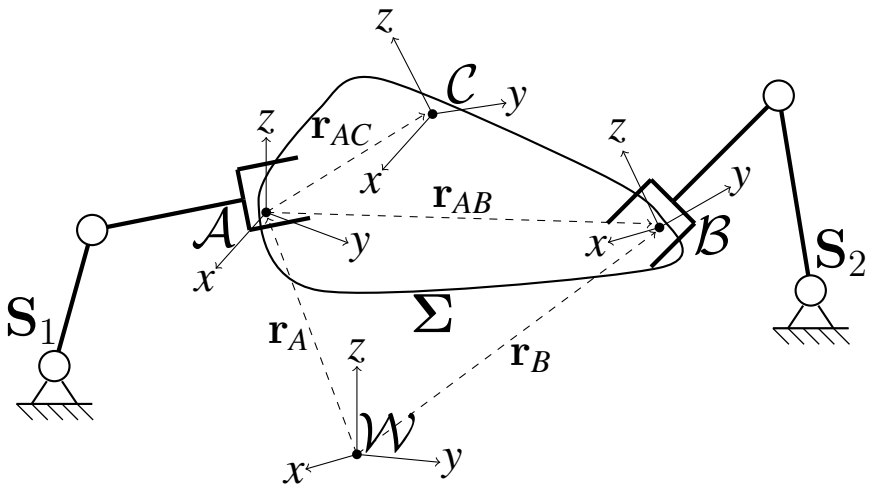

Fig. 3.3. Rigid body (Σ) coupling with fixed frame on it

It is easy to see that control law in this case is similar to equation (3.18). The key difference is that target frame has no Jacobians to represent frame velocities. Also it is necessary to add acceleration part to $\mathbf{b}$. It leads to

$$
\begin{aligned}
A &= \begin{bmatrix} \tilde{J}_v \\ \tilde{J}_\omega \end{bmatrix} \\
\mathbf{b} &= -\begin{bmatrix} \dot{\tilde{J}}_v \\ \dot{\tilde{J}}_\omega \end{bmatrix} \mathbf{v} - \begin{bmatrix} \mathbf{a}_D \\ \boldsymbol{\epsilon}_D^{\mathcal{D}} \end{bmatrix} - K_d \begin{bmatrix} \boldsymbol{\upsilon}_d \\ \boldsymbol{\omega}_d^{\mathcal{D}} \end{bmatrix} - K_p \mathbf{e}_{\mathrm{SE}(3)}(T_{\mathcal{W}}^{\mathcal{E}} \cdot T_{\mathcal{A}}^{\mathcal{C}}, T_{\mathcal{W}}^{\mathcal{D}})
\end{aligned}
\tag{3.35}
$$

where $\boldsymbol{\upsilon}_d$, $\boldsymbol{\omega}_d^{\mathcal{D}}$, $\tilde{J}_v$, $\tilde{J}_\omega$, $\dot{\tilde{J}}_v$, and $\dot{\tilde{J}}_\omega$ are

$$
\begin{aligned}
\boldsymbol{\upsilon}_d &= \boldsymbol{\upsilon}_D - \boldsymbol{\upsilon}_E - \boldsymbol{\omega}_E \times \mathbf{r}_{AC} \\
\boldsymbol{\omega}_d^{\mathcal{D}} &= \boldsymbol{\omega}_D^{\mathcal{D}} - \boldsymbol{\omega}_E^{\mathcal{D}} \\
\tilde{J}_v &= -J_{E,v} + [R_E \mathbf{r}_{AC}^{\mathcal{A}}]^{\wedge} J_{E,\omega} \\
\tilde{J}_\omega &= -R_E^T J_{E,\omega} \\
\dot{\tilde{J}}_v &= -\dot{J}_{E,v} + \hat{\boldsymbol{\omega}}_E \hat{\mathbf{r}}_{AC} J_{E,\omega} + \hat{\mathbf{r}}_{AC} \dot{J}_{E,\omega} \\
\dot{\tilde{J}}_\omega &= -R_E^T [\dot{J}_{E,\omega} - \hat{\boldsymbol{\omega}}_E J_{E,\omega}]
\end{aligned}
\tag{3.36}
$$

The algorithm is following

Algorithm 3.4.2 Computing A and $\mathbf{b}$ for control of the connecting rigid body

1: **function** GETRIGIDBODYCONTROLCONSTRAINT(combined system $\mathbf{S}_c$, $\mathbf{q}$, $\mathbf{v}$, E, $T_{\mathcal{A}}^{\mathcal{C}}$, $T_{\mathcal{W}}^{\mathcal{D}}$, $\left[\boldsymbol{\upsilon}_D^T \quad [\boldsymbol{\omega}_D^{\mathcal{D}}]^T\right]^T$, $\left[\mathbf{a}_D^T \quad [\boldsymbol{\epsilon}_D^{\mathcal{D}}]^T\right]^T$) :
2: Compute attitudes $T_{\mathcal{W}}^{\mathcal{E}}$ and velocities $\boldsymbol{\upsilon}_E$, $\boldsymbol{\omega}_E$ for $\mathbf{S}_c$ at state $\mathbf{q}$, $\mathbf{v}$
3: Obtain Jacobians $J_{E,v}$, $J_{E,\omega}$ and Jacobians time derivatives $\dot{J}_{E,v}$, $\dot{J}_{E,\omega}$ for $\mathbf{S}_c$ at state $\mathbf{q}$, $\mathbf{v}$
4: Get $\boldsymbol{\upsilon}_d$, $\boldsymbol{\omega}_d^{\mathcal{D}}$, $\tilde{J}_v$, $\tilde{J}_\omega$, $\dot{\tilde{J}}_v$, and $\dot{\tilde{J}}_\omega$ by equation (3.36)
5: Calculate A and $\mathbf{b}$ by equation (3.35)
6: **return** A, $\mathbf{b}$
7: **end function**

3.5 Task prioritization

After defining algorithm for obtaining A and $\mathbf{b}$ it is necessary to consider the prioritization problem. Let's consider the following case. The combined system $\mathbf{S}_c$ has n_v degree of freedom. There are n_c rigid body constraints and n_u controls laws imposed on it. Thus, the minimal requirement for problem to be solvable is

$$
n_v \geq 6 \cdot (n_c + n_u) \tag{3.37}
$$

Although this equation is useful for brief estimation, but it is still necessary condition. The control issues can arise due to the complex topological structure of joint space manifold. Hence, there are two options prove that desired trajectory is out off null space, or introduce some prioritization technique.

In the first case, it is enough to use combined matrices: $A = \begin{bmatrix} A_c^T & A_u^T \end{bmatrix}^T$, and $\mathbf{b} = \begin{bmatrix} \mathbf{b}_c^T & \mathbf{b}_u^T \end{bmatrix}^T$ with the Udwadia-Kalaba approach, where A_c, A_u, $\mathbf{b}_c$, and $\mathbf{b}_u$ are

$$\begin{aligned} A_c &= \begin{bmatrix} A_{c_1}^T & A_{c_2}^T & \dots & A_{c_{n_c}}^T \end{bmatrix}^T \\ A_u &= \begin{bmatrix} A_{u_1}^T & A_{u_2}^T & \dots & A_{u_{n_u}}^T \end{bmatrix}^T \\ \mathbf{b}_c &= \begin{bmatrix} \mathbf{b}_{c_1}^T & \mathbf{b}_{c_2}^T & \dots & \mathbf{b}_{c_{n_c}}^T \end{bmatrix}^T \\ \mathbf{b}_u &= \begin{bmatrix} \mathbf{b}_{u_1}^T & \mathbf{b}_{u_2}^T & \dots & \mathbf{b}_{u_{n_u}}^T \end{bmatrix}^T \end{aligned} \tag{3.38}$$

However, the discussed approach is not general. It is rare to proof controllability in real life cases.

The second way to figure out the problem is prioritization. The most rigorous technique to achieve is utilizing null space. Generally, it is preferable to strongly preserve the rigid body constraints and try to find the least square solution for control task. Such method can be expressed as

$$\begin{aligned} \min_{\dot{\mathbf{v}}} \quad & [A_u\dot{\mathbf{v}} - \mathbf{b}_u]^T[A_u\dot{\mathbf{v}} - \mathbf{b}_u] \\ \text{s.t.} \quad & \dot{\mathbf{v}} = A_c^+\mathbf{b}_c + \dot{\mathbf{v}}_u \\ & \dot{\mathbf{v}}_u \in \mathcal{N}(A_c) \end{aligned} \tag{3.39}$$

The above optimization prioritization is analytically solvable but solution is not convenient to use in terms of equation (3.7). Therefore, it is easy to reformulate the Gauss least constraint principle and include prioritization:

$$\begin{aligned} \min_{\dot{\mathbf{v}}} \quad & [\dot{\mathbf{v}} - \mathbf{a}]^T M [\dot{\mathbf{v}} - \mathbf{a}] + \omega [A_u \dot{\mathbf{v}} - \mathbf{b}_u]^T [A_u \dot{\mathbf{v}} - \mathbf{b}_u] \\ \text{s.t.} \quad & A_c \dot{\mathbf{v}} = \mathbf{b}_c \\ & \mathbf{a} = M^{-1} \mathbf{Q} \end{aligned} \tag{3.40}$$

where $\omega > 0$ tunable parameter. Due to $M \succcurlyeq 0$ and $A_u^T A_u \succcurlyeq 0$ the optimization problem (3.40) is quadratic task. Hence, it is always has a solution and can be solved fast by modern frameworks.

Chapter 4

Implementation and evaluation

In this chapter the implementation of proposed method and results of numerical experiments are shown. Firstly, Section 4.1 contains information about chosen system and its properties. Then, in Section 4.2 the discussion of used frameworks and utils is presented. Consequently, these tools are used in Section 4.3 to show how the proposed algorithms can be implemented. Finally, Section 4.4 contains results of numerical experiments with their evaluation.

4.1 Dual-arm YuMi

To prove the workability of the suggested methodology the dual-arm YuMi manipulator is chosen, see Fig. 4.1. It has 18 DoF, namely: 7 rotation and 2 prismatic (fingers) joints per each arm.

In simulation the total DoF was reduced to 14 by fixing end-effectors' fingers. Also for demonstration purposes the joint limits and body collisions were removed.

Otherwise, their presence could influence on final results.

Fig. 4.1. Dual-arm YuMi IRB 14000

The discussed manipulator was selected because there is URDF (Unified Robot Description Format) for it. However, there was no necessary representation for the chosen simulator. Thus, it was generated. Details is conducted later in Section 4.3.

4.2 Frameworks

For implementation the discussed algorithms C++ is used. Beyond of chosen language it is also necessary to select a simulator, a tool for computing dynamics, and library for solving the Gauss least constraint principle (3.3). In this section all of them is reviewed.

Firstly, let's discuss chosen simulator, MuJoCo [2]. It is general physical engine. In this study this framework is used to calculate system behavior under given Control. Also, it provides way to visualize robot. From control point of view MuJoCo is only needed to give $\mathbf{q}$ and $\mathbf{v}$ at each step.

The system state is transferred to dynamic computing tool. This paper is utilizing Pinocchio [3] library. It is modern and efficient instrument for calculating kinematic and dynamic for given system. Notably, this frameworks supports the URDF, which is very convenient in the context of the selected system. Moreover, Pinocchio provides fast computational utils. Authors of tool give the following quantities, see Fig. 4.2. For implementation this library is necessary to calculate Jacobians, attitudes of frames, and velocities.

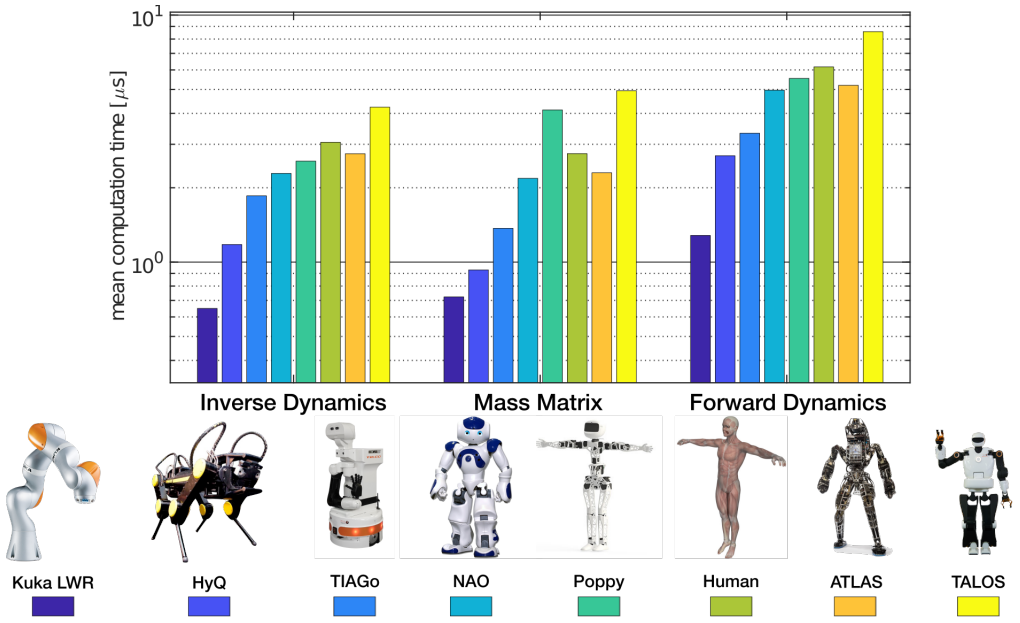

Fig. 4.2. Pinocchio performance demonstration

Finally, the last needed element is optimization problem solver. In this paper ProxSuite [19] is used for such purpose. The suggested methodology lies on quadratic problem. Authors of ProxSuite shows that their tool is fast of solving it. The mean computation time is 10 microseconds, see Fig. 1 in [19]. Although, the discussed instrument is efficient but the problem (3.40) can be directly solved. Thus, it is needed to be reformulated. Further details is provided later in Section 4.3

4.3 Implementation details

This section contains details of implementation of the suggested methodology. It can be split into two parts, namely: computing A and $\mathbf{b}$ with Pinocchio, and solving the problem (3.40) by ProxSuite. Remarkably, in this section there is no details about simulation and visualization because they described in MuJoCo documentation. However, it is necessary to mention that there is no YuMi representation for this simulator. Let's first discuss this moment.

The MuJoCo physical engine supports only specific XML description. It is usually called MJCF. For numerical experiment the total DoF was reduced to 14. Only rotation joints were remained. For demonstration purposes it is enough.

Now let's consider implementation of proposed algorithms 3.3.1, 3.4.1, and 3.4.2. For all of them it is obligatory to compute Jacobians, attitude of frames, and velocities. Let's starts from system model and data definition:

```
namespace pin = pinocchio;
...
pin::Model m;
pin::urdf::buildModel("path/to/urdf", m);
pin::Data d(m);
```

Listing 4.1: Model and data

Next it is necessary to compute forward kinematics for given model. As input this step requires generalized coordinates and velocities ($\mathbf{q}$ and $\mathbf{v}$). See below snippet:

```
pin::forwardKinematics(m, d, q, v);
pin::computeJointJacobians(m, d, q);
pin::computeJointJacobiansTimeVariation(m, d, q, v);
pin::updateFramePlacements(m, d);
```

Listing 4.2: Forward kinematics

Now Jacobians, attitudes of frames, and velocities can be calculated. If E is origin of some frame attached to system, then it has a special id in Pinocchio description. Let's call it `eIdx`. Thus, the code for Jacobians is

```
using MatXd = Eigen::MatrixXd;
...
MatXd jac = MatXd::Zero(6, m.nv);
MatXd dJac = MatXd::Zero(6, m.nv);
pin::getFrameJacobian(m, d, eIdx, pin::LOCAL_WORLD_ALIGNED,
   jac);
pin::getFrameJacobianTimeVariation(m, d, eIdx,
   pin::LOCAL_WORLD_ALIGNED, dJac);
```

Listing 4.3: Jacobians

Here `jac` and `dJac` are Jacobian and its derivative over time respectively. It is important to mention that these quantities from $\mathbb{R}^{6\times n_v}$. The first three rows are corresponding to classical linear velocity, $J_{E,v}$ or $\dot{J}_{E,v}$. The bottom three rows are angular part, $J_{E,\omega}$ or $\dot{J}_{E,\omega}$. All of them are expressed in the world coordinates.

For attitudes and velocities it is enough to execute

```
pin::Frame eFrame = m.frames[idx];
pin::FrameIndex pIdx = frame.parent;
pin::SE3 localToWorldY = pin::SE3::Identity();
localToWorldY.rotation(d.oMf[eIdx].rotation());
pin::SE3 eAtt = d.oMf[eIdx];
pin::Motion eVel =
   localToWorldY.act(frame.placement.actInv(d.v[pIdx]));
```

Listing 4.4: Attitudes of frames

In snippet (4.4) `eAtt` and `eVel` are frame attitude and velocity respectively. `eVel` contains linear and angular part.

The last parts worth mentioning are the skew-symmetric operator ($[.]^\wedge$) and matrix logarithm ($\log(.)$). In Pinocchio they are `pin::skew` and `pin::log3` function respectively. Other mathematical operations are not worth to be explained

due to their triviality.

All snippets above are used for experiment. They can be found in the repository REF TO REP. However, for optimization and readability they are realized not in the same order. In the provided code algorithm 3.3.1 is demonstrated in `getConstraintsAffineDesc`, algorithm 3.4.1 is shown in `getControlAffineDesc`, the last one 3.4.2 can be implemented by function `getAffineDesc`.

Now let's discuss the transformation of equation (3.40) for ProxSuite framework. This library can handle quadratic problem of the following form

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{g}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b} \\ & \mathbf{y}_l \leq C\mathbf{x} \leq \mathbf{y}_u \end{aligned} \tag{4.1}$$

where $[.] \leq [.]$ is element-wise vector operator, $H \succeq 0$, and other quantities are arbitrary.

In case of the Gauss least constraint principle the last line of quadratic problem (4.1) is unnecessary. Hence, it is enough to pen brackets of equation (3.40):

$$\begin{aligned} \min_{\dot{\mathbf{v}}} \quad & \frac{1}{2}\dot{\mathbf{v}}^T[M + \omega A_u^T A_u]\dot{\mathbf{v}} + [-\mathbf{Q} - \omega A_u^T \mathbf{b}_u]^T\dot{\mathbf{v}} \\ \text{s.t.} \quad & A_c\dot{\mathbf{v}} = \mathbf{b}_c \end{aligned} \tag{4.2}$$

The removing of unconstrained acceleration $\mathbf{a}$ is possible by equation (3.2). Thus, $H = M + \omega A_u^T A_u$ and $\mathbf{g} = -\mathbf{Q} - \omega A_u^T \mathbf{b}_u$. The problem (4.2) is solvable by the following code

```
namespace prox = proxsuite::proxqp;
...
```

```
prox::dense::QP<double> qp(m.nv, 6, 0);
qp.settings.eps_abs = eps_abs;
qp.settings.initial_guess =
   prox::InitialGuessStatus::NO_INITIAL_GUESS;
qp.settings.verbose = false;
qp.init(
    M + omega * Au.transpose() * Au,
    -Q - omega * Au.transpose() * bu,
    Ac, bc,
    proxsuite::nullopt, proxsuite::nullopt, proxsuite::nullopt
);
qp.solve();
```

Listing 4.5: Quadratic problem solution

After first computation it is recommended to use `prox::InitialGuessStatus::WARM_START` to speed up calculation. This flag enables initial guess taken from previous solution.

4.4 Simulation results

Chapter 5

Evaluation and Discussion

Chapter 6

Conclusion

Appendix A

Brief Lie theory

The Lie group, a mathematical concept dating back to the 19th century, was first proposed by Sophus Lie, laying the foundation for continuous transformation groups. Although it was initially abstract, over time its influence has spread to various scientific and technological fields. Before proceeding further, it is necessary to consider the basics of Lie theory. The mathematical tools discussed in this section are crucial for the research discussed here.

Let $\mathcal{G}$ is smooth manifold that satisfies the group axioms. For any $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ from $\mathcal{G}$ the following statements are true:

$$\begin{aligned} &\mathcal{X} \circ \mathcal{Y} \in \mathcal{G} && \text{(I)} \\ &\exists\, \mathcal{E} :\ \mathcal{E} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{E} = \mathcal{X} && \text{(II)} \\ &\exists\, \mathcal{X}^{-1} :\ \mathcal{X}^{-1} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{X}^{-1} = \mathcal{E} && \text{(III)} \\ &(\mathcal{X} \circ \mathcal{Y}) \circ \mathcal{Z} = \mathcal{X} \circ (\mathcal{Y} \circ \mathcal{Z}) && \text{(IV)} \end{aligned} \tag{A.1}$$

For such group $T_{\mathcal{E}}\mathcal{G}$ is the Lia Algebra defined at $\mathcal{E}$ element. The geometric interpretation of algebra is tangent plane that touchs the smooth manifold (group). $\mathfrak{m} \equiv T_{\mathcal{E}}\mathcal{G}$ is always a vector space.

The next crucial defenition is a group action

$$f : \mathcal{G} \times \mathcal{V} \to \mathcal{V} \tag{A.2}$$

where $\mathcal{V}$ is some set. The operation defined above should satisfy the axioms ($v \in \mathcal{V}$; $\mathcal{X}, \mathcal{Y} \in \mathcal{G}$),

$$\begin{aligned} &\mathcal{E} \cdot v = v && \text{(I)} \\ &(\mathcal{X} \circ \mathcal{Y}) \cdot v = \mathcal{X} \cdot (\mathcal{Y} \cdot v) && \text{(II)} \end{aligned} \tag{A.3}$$

For instance, if the SO(n) rotations is considered as Lie group, than the transformation $R \cdot x \equiv Rx$ ($x \in \mathbb{R}^n$) is a group action.

The exponential map $\exp : \ \mathfrak{m} \to \mathcal{G}$ converts elements of Lie algebra to corresponding group. The log map do inverse operation. However, it is neccesary to remember that this map applicable only for "identity" algebra. However, there is a linear transformation between $T_{\mathcal{X}}\mathcal{G}$ and $T_{\mathcal{E}}\mathcal{G}$. It is called adjoin.

It is known that $\mathfrak{m}$ is isomorphic to the vector space $\mathbb{R}^m$. One can write it as $\mathfrak{m} \cong \mathbb{R}^m$. Due to convinience this isomorphism would be highly utilized. Moreover, in the bellow sections only algebras constructed on $\mathbb{R}^m$ are considered. Thus, a maping between sets can be defined in the following manner (hat-vee notation)

$$\begin{aligned} &[*]^{\wedge} : \ \mathbb{R}^m \to \mathfrak{m} && \boldsymbol{\tau}^{\wedge} = \sum_{i=1}^{m} \tau_i E_i \\ &[*]^{\vee} : \mathfrak{m} \to \mathbb{R}^m && \boldsymbol{\tau} = \sum_{i=1}^{m} \tau_i \mathbf{e}_i \end{aligned} \tag{A.4}$$

where $\mathbf{e}_i$ are a base of $\mathbb{R}^m$ and E_i are base vectors of $\mathfrak{m}$. Obviously, $\mathbf{e}_i^{\wedge} = E_i$. Using this mapping, the exponential / log map can be modified

$$\begin{aligned} \exp: &\quad \mathcal{X} = \exp(\boldsymbol{\tau}^{\wedge}) \\ \log: &\quad \boldsymbol{\tau} = \log(\mathcal{X})^{\vee} \end{aligned} \tag{A.5}$$

Or, in the most convinient form

$$\begin{aligned} \text{Exp}: &\quad \mathcal{X} = \exp(\boldsymbol{\tau}^{\wedge}) = \text{Exp}(\boldsymbol{\tau}) \\ \text{Log}: &\quad \boldsymbol{\tau} = \log(\mathcal{X})^{\vee} = \text{Log}(\mathcal{X}) \end{aligned} \tag{A.6}$$

Through this definitions it easy to introduce plus and minus operations

$$\begin{aligned} &\text{right-}\oplus : \mathcal{X} \oplus {}^{\mathcal{X}}\boldsymbol{\tau} \equiv \mathcal{X} \circ \text{Exp}({}^{\mathcal{X}}\boldsymbol{\tau}) \in \mathcal{G} \\ &\text{right-}\ominus : \mathcal{Y} \ominus \mathcal{X} \equiv \text{Log}(\mathcal{X}^{-1} \circ \mathcal{Y}) \in T_{\mathcal{X}}\mathcal{G} \end{aligned} \tag{A.7}$$

TODO: continue section

Bibliography cited

[1] F. Udwadia and R. Kalaba, "A new perspective on constrained motion," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 439, no. 1906, pp. 407–410, Nov. 1992. DOI: 10.1098/rspa.1992.0158.

[2] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, Oct. 2012. DOI: 10.1109/IROS.2012.6386109.

[3] J. Carpentier, G. Saurel, G. Buondonno, *et al.*, "The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," *International Symposium on System Integration SII*, Jan. 2019. DOI: 10.1109/SII.2019.8700380.

[4] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[5] H. S. et al., "Application of the udwadia-kalaba approach to tracking control of mobile robots," *Nonlinear Dynamics*, vol. 83, no. 1–2, pp. 389–400, Aug. 2015. DOI: 10.1007/s11071-015-2335-3.

[6] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal, "A unifying framework for robot control with redundant dofs," *Autonomous Robots*, vol. 24, no. 1, pp. 1–12, Oct. 2007. DOI: 10.1007/s10514-007-9051-x.

[7] F. E. Udwadia, "Constrained motion of hamiltonian systems," *The Computer Journal*, vol. 84, no. 3, pp. 1135–1145, Dec. 2015. DOI: 10.1007/s11071-015-2558-3.

[8] M. Chignoli, N. Adrian, and P. M. Wensing, "Recursive rigid-body dynamics algorithms for systems with kinematic loops," *arXiv.org*, Nov. 2023. DOI: 10.48550/arXiv.2311.13732.

[9] S. Kuindersma, R. Deits, M. Fallon, and et al., "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, pp. 429–455, Jul. 2015. DOI: 10.1007/s10514-015-9479-3.

[10] S. Sovukluk, J. Englsberger, and C. Ott, "Whole body control formulation for humanoid robots with closed/parallel kinematic chains: Kangaroo case study," *International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2023. DOI: 10.1109/IROS55552.2023.10341391.

[11] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 1, no. 1, pp. 1–16, Jun. 1972. DOI: 10.1016/0045-7825(72)90018-7.

[12] B. T. Lopez and J.-J. E. Slotine, "Sliding on Manifolds: Geometric Attitude Control with Quaternions," *IEEE International Conference on Robotics*

and Automation (ICRA), May 2021. DOI: 10.1109/ICRA48506.2021.9561867.

[13] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," *IEEE Conference on Decision and Control (CDC)*, Dec. 2010. DOI: 10.1109/CDC.2010.5717652.

[14] N. A. Chaturvedi, A. K. Sanyal, and N. H. McClamroch, "Rigid-Body Attitude Control," *IEEE Control Systems Magazine*, vol. 31, pp. 30–51, May 2011. DOI: 10.1109/MCS.2011.940459.

[15] H. Mishra, M. D. Stefano, A. M. Giordano, and C. Ott, "Output feedback stabilization of an orbital robot," *IEEE Conference on Decision and Control (CDC)*, Dec. 2020. DOI: 10.1109/CDC42340.2020.9304044.

[16] H. Mishra, M. D. Stefano, A. M. Giordano, and C. Ott, "A Nonlinear Observer for Free-Floating Target Motion using only Pose Measurements," *American Control Conference (ACC)*, Jul. 2019. DOI: 10.23919/ACC.2019.8814815.

[17] F. Udwadia and A. Schutte, "Equations of motion for general constrained systems in lagrangian mechanics," *Acta Mechanica*, vol. 213, pp. 111–129, Feb. 2010. DOI: 10.1007/s00707-009-0272-2.

[18] F. Udwadia and R. Kalaba, *Analytical Dynamics: A New Approach*. Cambridge University Press, 1996, ISBN: 9780521482172.

[19] A. Bambade, S. El-Kazdadi, A. Taylor, and J. Carpentier, "PROX-QP: Yet another Quadratic Programming Solver for Robotics and beyond," in *RSS 2022 - Robotics: Science and Systems*, New York, United States, Jun. 2022. [Online]. Available: https://hal.inria.fr/hal-03683733.