Development of a Dynamic Identification Toolbox for Anthropometric Systems

Dinh Vinh Thanh Nguyen

ÉCOLE CENTRALE DE NANTES LAAS-CNRS

MASTER CORO-IMARO
"CONTROL AND ROBOTICS"

2020 / 2021

Master Thesis Report

Presented by

Dinh Vinh Thanh Nguyen

19/07/2021

Development of a Dynamic Identification Toolbox for Anthropometric Systems.

Jury

Evaluator(s):	Olivier Kermorgant	Associate Professor (ECN, LS2N)
	David Daney	Associate Professor (Inria Bordeaux)
Supervisor(s):	Vincent Bonnet	Associate Professor (LAAS-CNRS)
	Olivier Stasse	Dir. of Research (LAAS-CNRS)
	Maxime Gautier	Emeritus Professor (Univ. of Nantes, LS2N)

Laboratory: Laboratoire d'analyse et d'architecture des systèmes LAAS-CNRS

Abstract

The knowledge of dynamic parameters of robotic systems is essential to applications and research such as model-based controlling, simulating a robot, or planning motions. The dynamic identification process stands out as one of the most efficient and widely used method to estimate dynamic parameters. In this process, a dynamic model of a robot is created. Thanks to its linearity to dynamic parameters, an identification model is derived from the dynamic model. The identification model is applied to a sufficient data points obtained from specially defined trajectories in order to construct an over-determined linear system. Linear regression techniques such as least square estimation are utilized in order to identify dynamic parameters. Finally, the identified parameters must be validated by direct validation or cross validation. This thesis's work aims to develop a standard open-source toolbox that provides all functions and methods of dynamic identification for robotic systems, in general, and for anthropometric systems, in particular.

Acknowledgements

I am always grateful for everything that I have been given.

List of Figures

List of Tables

Contents

Introduction

In the robotics field, dynamic parameters such as mass, center of mass, inertia tensor are basic building blocks that construct a model for almost every application ranging from control, system simulation to motion planning. Consequently, they have been always of interest to be obtained with highest accuracy, specially in cases where the system is highly sensitive to uncertainties and errors. Theoretically, this intrinsic physical knowledge of a system is assumed to be either measurable by measuring devices done separately part by part, or known by CAD designers *before* manufacturing and assembling. However, by measuring separately, the accumulative errors apparently would be significant enough to have negative impacts on the performance of a system. More over, measuring part by part is time consuming with complex robots and might be inaccurate with inertia tensor values. For many applications, CAD data is normally considered as a reliable reference. Yet, not all the dynamic parameters given by CAD data are not reliable due to several factors. First, the designing process is done before actual manufacture and assemble procedures, so errors typically appearing during the manufacturing would cause a deviation in actual values compared to reference values. Second, CAD data cannot provide us dynamics effects such as friction which could only happen when robots are deployed in actual operation. In short, dynamic parameters are crucial to every robotic applications, and it is necessary to have a reliable, relatively simple method to estimate correctly.

Dynamic identification is considered to be a more practical method than theses above mentioned methods to recognise the actual parameters thanks to its simplicity of experimental settings, while keeping high level of precision for obtained values. The dynamic identification method can be shortly explained as: the estimation of dynamic parameters of a robot is done by minimizing the difference between measured values and predicted values from a mathematical model, through analyzing the input/output of the robot after tracking specifically defined trajectories, often referred as optimal exciting motions (optimal trajectories)[1].

Over decades, researcher have thoroughly investigated this method and implemented it in not only various industrial applications , academic research in robotics but also in bio-mechanics with human bodies. In recent years, this approach has been used in order to identify effective dynamic parameters in complex anthropometric robotic system and obtained significant results

with innovative solution deriving from the fundamental dynamic identification method.

0.1 Motivation

This thesis is motivated by the fact that the field has been missing a toolbox that allows to identify accurate dynamic parameters given any kind of structures despite the proven maturity of the dynamic identification methodology. In practice, the benefits of obtaining accurate dynamic parameters have been provenly essential in various tasks such as performing model-based control, closing the simulation-to-reality gap or estimating values which cannot be directly measured. However, in practice, mostly dynamic identification procedure is developed customizing for specific robots. It means that the reusability would be limited. Therefore, a general pipeline using the developed toolbox would benefit researchers and engineers. Secondly, these procedures did not take the advantages of excellent existing libraries such as dynamic modeling, trajectory planning and so on; or instead utilize independent libraries with difficulties of version control. Leveraging the existing tools within a common framework will not only reduce the amount of time to develop but also would be adding more values to the existing framework. With all the consideration above, making dynamic identification become a straight-forward and common practice to all general applications in robotics motivates me to carry out this thesis.

The work of this thesis is to fulfill the gap between the need of a practical and open-source identification toolbox to both industrial applications and academic research and the lack of a standard pipeline. It is worth mentioning that the development of this dynamic identification toolbox is closely supported by expertise of a strong research group in legged robotics, the Geppeto team at LAAS-CNRS, Toulouse, and a leading robotics company in Europe, PAL Robotics, Spain. The Geppeto team over years has continuously contributed to the robotics community with their notable research results including open-source software for robotics such as: Stack of Tasks, a framework for controlling redundant robots which includes popularly used libraries like Pinocchio, a rigid multi-body dynamics library. In addition to that, the work is provided opportunities to develop, test and implement on state-of-the-art hardwares manufactured by PAL Robotics. Such robots like human-sized humanoid robot TALOS [2], mobile manipulator Tiago by PAL Robotics have been widely received and deployed in research as part of collaborations with renowned research centers in many EU-funded projects.

0.2 Objectives

As mentioned above, the goal of the thesis is to develop a standard open-source toolbox of dynamic identification using in robotic systems in general and more particularly focused on

anthropometric systems. In order to achieve this goal, several specific objectives to be accomplished are proposed as following

- To develop a package providing robot model and dynamic algorithm: This package is built on top Pinocchio library to inherit its robot modeling and dynamic algorithms; converting robot MDH format modeling to a more modern and universal URDF format.
- To develop a package generating optimal exciting motions: This package's main functionalities includes: providing a wide range of trajectory planners; optimizing trajectories which would maximally excite dynamic parameters while respecting systems' geometrical and dynamical limits by solving non-linear optimization problem with IPOPT library.
- To develop a package providing standard identification methods: This package consists of modules whose functionalities: creating identification model; pre-processing input data; solving estimation problems; validating identified results.
- To unit-test, evaluate and validate each individual package's functionalities.
- To integrate, implement and document the use of the toolbox on actual robots in order to verify its performance.

0.3 Structure overview

This report is organized into 4 chapters:

- **Chapter 1** presents the state of the art approaches in dynamic identification.
- **Chapter 2** provides technical details of packages in the identification toolbox.
- **Chapter 3** describes actual implementations on available robots and discusses the results obtained from these implementations.

Chapter 1

State of the art

This chapter will present the state of the art approaches in dynamic identification. Dynamic identification is a procedure which contains cascading steps corresponding respectively to what model is chosen to represent the relationship between independent variables and dependent variables, how experimental data is generated based upon predefined criterion, how the generated experimental data is preprocessed and analyzed in order to estimate dynamic parameters. Theoretical background is the backbone of every step above through out the identification procedure. Therefore, they will be introduced here in great details. Following the theory, most common techniques that researchers have used to advance towards in dynamic identification will be shown and compared. Even though there are many innovations in research of dynamic identification in recent years, only most appropriate approaches to our toolbox that are taken into consideration of our work are presented in the chapter.

1.1 Overview of dynamic identification

Generally, parametric model expressing the relationship between input and output is of interest to most of applications. In the case of robotic applications, dynamic model that are constructed by dynamic parameters are crucial. Dynamic identification is a procedure to identify these parameters to build an accurate dynamic model. Mathematically, dynamic identification is a regression analysis problem. Unlike problems with unverified models, dynamic identification usually comes with a well-established model which could be dynamic model or, in some cases, power model. From the dynamic model or power model, the **identification model**, intuitively the regression model, is extracted. After the identification model is decided, data generation/collection is proceeded. Due to the characteristics of a physical system, data needs to be observed from specially designed experiments which would stimulate the dynamics of all parameters. This process is often done by solving an **trajectory optimization** problem. And then, the **identification algorithm**, mostly inherited from regression techniques, will be applied to estimate the parameters of the identification model. The last step of this procedure

is to analyze the errors and to validate the obtained model in direct/cross **validation**. The procedure could be repeated until the validating criterion are satisfied. With an intuition as such, an over scheme showing how a complete procedure of dynamic identification is presented in the following figure.

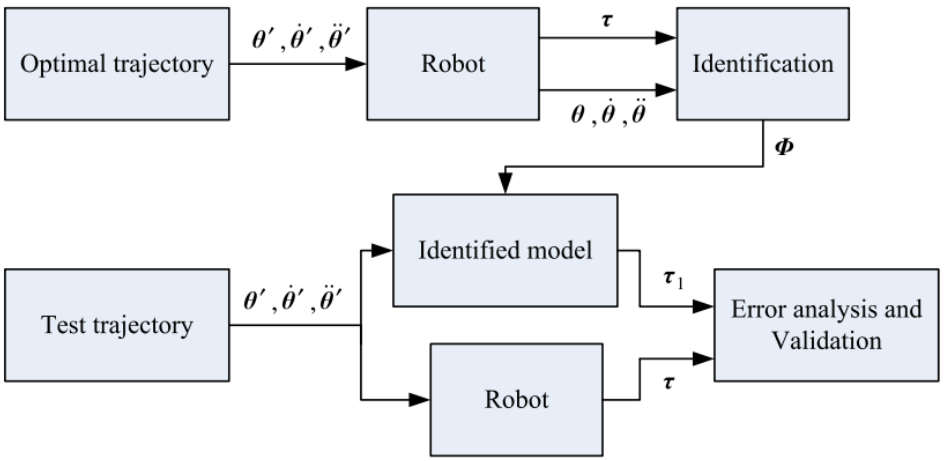

Figure 1.1: A generalized scheme of identification procedure

1.2 Identification model

Dynamic identification in robotics takes the advantage of linearity in inverse dynamic model of a robot with respect to its dynamic parameters. A common approach of building an identification model is to derive the inverse dynamic model. Then, by applying this model to a sufficient number of sample data points from specifically designed trajectories, we can obtain an over-determined linear system of equations. In general, a specific link of a robot only moves in a limited number of axes (degree of freedom). As a result, not every standard dynamic parameters would have effects on dynamics model. Hence, those ones are redundant, not identifiable, and can be excluded from the model without changing its dynamics. Secondly, as links are connected to each other by joints, when a link moves, it will have a dynamic impact on its precedent links. Thus, a number of dynamic parameters are bound together as a linear combination to represent the correlation between dynamics of links. This is our **identification model**.

1.2.1 Dynamic model

a) Dynamic model is the relationship between kinematic variables of the motion and the cause of the motion. In the literature, different methods can be deployed to derive the dynamic model such as: Newton-Euler method, Lagrange formulation, and virtual work principle. The

dynamic model can expressed as:

$$\mathbf{M}(q)\ddot{q} + \mathbf{H}(q, \dot{q}) + \mathbf{G}(q) = \tau \tag{1.1}$$

where:

- q, $\dot{q}$, $\ddot{q}$ are joint positions, velocities, accelerations;
- τ are joint forces/torques;
- $\mathbf{M}(q)$ is the symmetric positive definite inertial matrix;
- $\mathbf{H}(q, \dot{q})$ is the matrix representing Coriolis and centrifugal forces;
- $\mathbf{G}(q)$ represents gravitational force.

This is the most common form of dynamic model that can be found in literature. In the dynamic model, elements of matrices $\mathbf{M}$, $\mathbf{H}$, $\mathbf{G}$ are standard dynamic parameters: mass, first moment of inertia, inertia tensor. The data of joint positions is collected and sampled from measures of encoders and the data of joint forces/torques is estimated from actuator current in most cases, or obtained from torque sensors, which is less frequently available in most of robots.

b) Dynamics of base link

In the previous dynamic model above, the dynamics of base link is excluded. It is common that many applications use industrial robots with fixed base. Therefore, it is trivial to consider the dynamics of the base link. However, in more general cases such as free-floating base robots like legged robots, mobile robots or mobile base manipulators which contact with environment, the dynamics of the contacts between base link and/or links with their environment needs to be taken into account.

The dynamic equations that describes a system contacting with its environment can be rewritten in a form as below:

$$\begin{bmatrix} \mathbf{M}_{\omega\omega} & \mathbf{M}_{\omega c} \\ \mathbf{M}_{c\omega} & \mathbf{M}_{cc} \end{bmatrix} \begin{bmatrix} \ddot{q}_\omega \\ \ddot{q} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_\omega(q_\omega, \dot{q}_\omega) \\ \mathbf{B}_c(q, \dot{q}) \end{bmatrix} = \begin{bmatrix} 0 \\ \tau \end{bmatrix} + \sum_{k=1}^{N_c} \begin{bmatrix} \mathbf{J}_{\omega_k}^T \\ \mathbf{J}_{c_k}^T \end{bmatrix} F_k \tag{1.2}$$

where:

- subscript ω and c indicates parameters corresponding to the contacting links and the non-contacting ones, respectively;
- $\mathbf{M}$: inertial matrices;

- $\ddot{q}$: joint accelerations;
- **B**: representing centrifugal, Coriolis and gravity forces;
- F_k: contact forces system with environment at k-th contact point amongst N_c contact points;
- **J**: Jacobian matrices that maps these contact force into joint space.

If we assume that the robot does not have any other contact rather the ground contact at the base link, the dynamic equations of base link could actually represent the dynamics of the whole robot. This is true because as mentioned above, dynamics from end-effector link transferred down to the precedent link until the base link. Researchers [3] have utilized this property to apply in applications where a fast, simple dynamic identification required for a complex robot. Another application is that when measurement of force/torque at joint are unfeasible such as in human mechanic study, or measuring sensors are unreliable due to noise, a simple 6-axis force plate at the base link can be a good solution for dynamic identification problem.

c) Inclusion of joint drive chain friction and inertia

Despite the inevitable existence in almost every moving physical system, friction is commonly ignored in the dynamics equations. One may argue that the effect of friction having on overall dynamics of the system is negligible. However, poor performance in model-based control due to this ignorance proved otherwise as friction and inertia in joint drive chain may have a significant impact on the dynamics. Modeling of the effects of joint drive chain friction and inertia can be described as below.

$$\tau_g = \tau + \dot{q}f_v + sign(\dot{q})f_c + Ia(\ddot{q}) \tag{1.3}$$

where:

- τ_g: general expression of joint force/torque
- τ: joint force/torque without friction and actuator inertia effects
- f_v: viscosity coefficient
- f_c: Coulomb coefficient
- Ia: actuator inertia

Authors at [4] demonstrated how a full set dynamic parameters including friction, actuators, joint torques offset should be identified. In a further step of developing this approach, drive

gain which is used to estimate joint torques can be also identified accurately. The work of this technique is presented in the following publication [5].

d) Joint coupling

In manipulators, joints near the end-effector being mechanically coupled are common. This coupling could be done a couple of face gears or a set of gears. It is important to include the coupling in the dynamic model as it changes the values of actual joint velocities and joint torques compared to the values out of motors.

1.2.2 Identification model

A set of standard inertial parameters of a link i contains 10 values including: mass m_i, first moment of inertia [m_{x_i} m_{y_i} m_{z_i}] and 6 values that form 3x3 inertial tensor [I_{XXi} I_{YYi} I_{ZZi} I_{XYi} I_{YZi} I_{ZXi}]. If we rewrite the equation 1.2 as regrouping these parameters as variables, we may have:

$$\mathbf{W}\Phi = \begin{bmatrix} 0 \\ \tau \end{bmatrix} + \sum_{k=1}^{N_c} \begin{bmatrix} \mathbf{J}_{\omega_k}^T \\ \mathbf{J}_{c_k}^T \end{bmatrix} F_k \tag{1.4}$$

where:

- $\mathbf{W}$ is the observation matrix;
- Φ is a vector of standard inertial parameters;
- τ, J, F_k are defined as in above equations.

The model above gives us two possibilities with or without the availability of joint torque values. When the joint torques are not available, the upper part of the equation can be utilized in the identification process since only measurements of contact wrench at the base link are needed. If the joint torque information can be accessible, the identification can proceed based on the lower part of the equation.

a) Identifiability of the dynamic parameters

The authors [1] pointed out that dynamic parameters can be divided into three groups: fully identifiable, identifiable in linear combination, and completely unidentifiable. As a result, the observation matrix can be rank deficient, in other words, not a full rank column matrix. Therefore, in that case, to obtain a unique solution for standard parameters is not possible. It is obvious that one should consider to exclude the trivial, non-effectual parameters, and obtain a minimal set of identifiable parameters by linearly regrouping the identifiable standard parameters. These linear combinations are called base parameters (BP).

b) Regrouping standard parameters to form base parameters

There are two main ways to obtain the symbolic expression of base parameters. Authors at [6] suggested that symbolic forms of the base parameters can be manually found by applying a certain set of formulas based on dynamics rules. He demonstrated in obtaining base parameters of a serial PUMA robot. However, this method may become tedious and error-prone once the studied system is complex.

A second way to obtain the expressions of base parameters is to use numerical methods [7]. One of the most common techniques in literature is pivoted QR decomposition. QR decomposition is a process for decomposing a matrix into two matrices: the first one, matrix Q, is an orthogonal column basis of the matrix; the second one, matrix R, represents the corresponding magnitudes of each column of the matrix projected onto vectors of the basis matrix Q. Thus, in dynamic identification, using QR decomposition to decompose an observation matrix is actually evaluating the impact of each column in the whole dynamics of the system, in which each column of the observation matrix stands for the dynamic impact of one standard parameter. Following the decomposing process, the columns of the observation matrix can be rearranged by the descending order of their corresponding values on the diagonal of matrix R which indicate their magnitudes of impact on the over dynamics. This process is called pivoting. The columns with zero values on the diagonal of matrix R are so-called dependent parameters. In contrast, the ones with positive values are independent parameters. Acquiring this information would help to simplify the dynamic model by regrouping the dependent parameters into linear combinations with independent parameters. Eventually, we would obtain a minimal set of parameters, or base parameters, which can fully describe the dynamics of the system.

c) Identification model for base parameters

After obtaining the expression of base parameters, we can construct an over-determined linear system of equations:

$$\mathbf{Y}(\tau, f_{ext}) = \mathbf{W_B}(q, \dot{q}, \ddot{q})\Phi_B + e \tag{1.5}$$

where:

- $\mathbf{Y}(\tau, f_{ext})$ is a vector containing joint force/torques and/or contacting wrench at base link;
- $\mathbf{W_B}(q, \dot{q}, \ddot{q})$ is the regressor matrix that has a size of mxn, m is number of data points, n is number of base parameters, $m >> n$;
- e is the vector of residual error between measured values and estimated values.

1.3 Identification algorithm

Identification algorithm involves in estimating dynamic parameters using numerical methods, particularly regression analysis. This section introduce two mainly used algorithms in dynamic identification. More over, a technique to estimate standard dynamic parameters from identified base parameters is presented here. Finally, an approach that attempts to simplify the identification problem by categorizing dynamic parameters as "static" and "dynamic" referring to the ones not involving in movements and the ones that are, respectively.

1.3.1 Estimation methods for base parameters

a) Ordinary Least Square Estimation

In the identification model above, the vector e represents measurement noise and modelling errors are perturbed, or so called residues. The error e is assumed to have zero mean. The goal is to find a solution that will minimize the sum of residue norms.

The principle of ordinary least square estimation is to find a set of coefficients of a model by minimizing the sum of norms of the residues which represents the difference between measured entities and its corresponding estimated ones from the model. In linear algrebra, when we have a problem of ordinary least square estimation where one side is a product of 2 vectors/-matrices and the other side is a vector, and with the matrix confirmed full rank, an ordinary least square solution is simply calculating pseudo-inverse which can be done be Singular Value Decomposition (SVD) or QR decomposition.

If the regressor matrix $\mathbf{W_B}$ having column full-rank, the vector of base parameters Φ_B can be directly estimated by the ordinary least square approach in a linear algebra manner.

$$\hat{\Phi}_B = \text{argmin}(||e||_2) \tag{1.6}$$

In another word,

$$\hat{\Phi}_B = (\mathbf{W_B}^T\mathbf{W_B})^{-1}\mathbf{W_B}^T Y = \mathbf{W_B}^+ Y \tag{1.7}$$

where $\mathbf{W_B}^+$ is the pseudo-inverse matrix of $\mathbf{W_B}$.

b) Weighted Least Square Estimation

In the ordinary least square estimation, one assumption was made, that is the measurements was uncorrelated and have different uncertainties. However, in fact, the measurements in dynamic identification might be highly correlated in some cases, thus, the estimated results of ordinary least square may not present an appropriate estimation solutions. In this case, weighted least square (WLS) can be seen an alternative. In order for taking the account of correlation, a weighted sum of residues should be computed where the weights are the

reciprocal of the variances of measurements. The authors at [5] chose the inverse of a diagonal covariance matrix $\mathbf{\Omega}$ calculated from ordinary least square estimation as the weighting matrix. Consequently, the new solution obtained from WLS is:

$$\hat{\Phi}_{BWLS} = (\mathbf{W_B^T \Omega^{-1} W_B})^{-1} \mathbf{W_B^T \Omega^{-1}} Y \tag{1.8}$$

1.3.2 Estimation methods for standard parameters

Now supposing that we obtained the BP above, but the main interest is to obtain the values of standard dynamic parameters. However, it can be problematic if some of them falls into null space of the regressor, hence being unidentifiable. In recent studies, the main approach to solve for standard parameters is to solve a constrained quadratic programming optimization fitting the estimated ground reaction force and moments to measured ones, at the same time minimizing the difference between the estimated values and the reference values such CAD data [8], [5] or AT values for human body, and it is described as:

Find $\hat{\Phi}$ such that

$$min \quad \|Y - W\Phi\|^2 + \|\Phi_{CAD} - \Phi\|^2 \tag{1.9}$$

However, the function above does not guarantee an appropriate result since no physically meaningful constraint has been introduced. In [9], it demonstrated this effect where the final results of some positive parameters turned out negative which are impossible. We consider the following inequality constraints in order to obtain consistent standard parameters:

- all masses must be positive;
- CoM has to be bounded in a defined range;
- Inertial matrix must be positive definitive.

Now, the complete constraint QP optimization problem is defined as below:

$$\hat{\Phi} = \text{argmin}(\|(F - W\Phi)\|^2 + \|(\Phi_{CAD} - \Phi)\|^2) \tag{1.10}$$

subject to:

- $m_i > 0$;
- $CoM^- < CoM < CoM^+$;
- for any non-zero vector v satisfied $v^T I v > \epsilon$ with $\epsilon = 1e-3$.

1.3.3 Decoupling parameters influenced by static postures and dynamic motions

Up to this point, it is very interesting to see that the problem can be actually decoupled into two separate cases: static and dynamic cases. In [10], the base parameters involved in the static cases such as mass and center of mass of each link were estimated assuming that static postures are achieved, meaning no velocities and acceleration. It can be as simple as:

$$\mathbf{P}_{CoM} = R_b(\mathbf{q})\Phi_b \tag{1.11}$$

where: $\mathbf{P}_{CoM}$ is x, y components of CoM of the whole body estimated by COP, R_b is the regressor, Φ_b is the BP.

On the other hand, [5], [11] estimated both static parameter (mass and CoM) and dynamic ones (inertia) in a decoupling fashion, in which such parameters: mass and CoM are identified by exciting static postures while other inertial parameters are identified by exciting motions joining the previous static postures.

1.4 Optimal trajectory

The least square estimation or, more generally linear regression problem of identification above requires a large amount of data, a rule of thumb is that it should be at least 10 folds of numbers of estimated parameters. However, in addition to a large amount of data, in order to improve the convergence rate in the linear regression problem, the motions that generated the data need to excite the dynamics of the system. Such a trajectory generating data for identification is called optimal exciting trajectory. In short, finding optimal exciting trajectories is solving a constrained non-linear optimization problem.

1.4.1 Trajectory Optimization Problem

a) Objective function

One criterion to justify the excitation of postures/motions that has been extensively used in the literature [12] is the condition number of the regressor matrix. A large condition number shows a sign of a large sensitivity to data errors or noise [13]. In short, the aim of this motion generating process is to find a set of motions that minimizes the condition number of the observation matrix by solving an optimization problem.

In general, the use of condition number as an optimizing criteria can provide robust results in least square estimation, only it satisfies one important condition: well-equilibrated, meaning its rows (columns) share the same length in some norm. However, it is common that a robot

often have very large links with large inertia near the base and much lighter links at the end-effector. The scenario creates an bad-equilibrated regressor matrix for dynamic identification problem. As result, besides using condition number as the main criteria, one must consider some weighting factors in order to eliminate the illed-equilibration of the regressor matrix.

b) Constraints

Mechanical constraints

It is essential to take into account several constraints, which are joint limits, maximum velocities, and maximum effort tolerance. Each joint has its own range of motion, which is physically impossible to violate. Also, in any either static posture or dynamic motions, the balance of robot needs to be satisfied. Other constraints can be defined upon the specific configuration of the system or conditions of subject that are experimented on.

Self-collision constraints

Even though, most of robots these has a default mode of control to prevent self-collision by halting the operation, we do not want self-collision to be ignored while planing our trajectories. Therefore, an addition constraint on self-collision must be added to the optimization problem. The self-collision normally involves with high cost of computation if the structure of robot's links is complex. One must pay attention to use a simplified collision model to reduce the computation cost for the trajectory optimization problem.

Self-balance constraints

Furthermore, unlike healthy human subjects who can keep balance by themselves, humanoid robots are prone to bad self-balance. In another word, it actually creates an extra constraint to the non-linear optimization problem. The Zero-Moment-Point (ZMP) is the point where the tangent components of ground reaction moment is zero. The humanoid robot is guaranteed to be stable when this ZMP stays inside the support polygon. In [5], [11], a balance controller was proposed along with identification process, where the choice of links to used in balancing task and identification task is thoroughly considered and made in order to simplify the optimization problem. Therefore decoupling has not only been done by type of parameters as in the previous section, but also by link when different configuration giving different sub-regressors. And now,

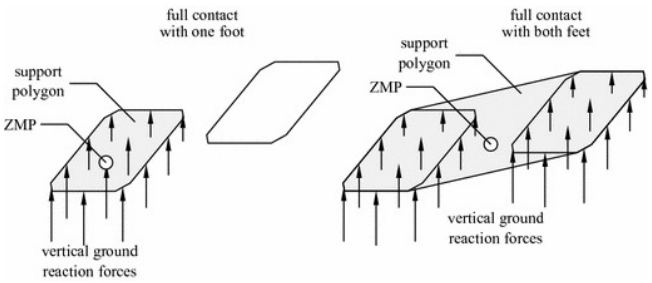

Figure 1.2: The ZMP and the support polygon during single support and double support

it is clearly seen that the generation of optimal exciting postures/motions is to solve another

constrained QP optimization problem.

c) Optimization Problem Formulation

With all objective functions, constraints and decision variables chosen, generating the optimal trajectories is equivalent to solve a constrained non-linear optimization. The criteria that this problem tries to satisfy is to find trajectories that create an well-conditioned observation matrix. Along with constraints being discussed, the problem can be mathematically presented by:

$$\hat{q} = argmin(cond(\mathbf{W_B})) \tag{1.12}$$

subject to:

- joint limits: ${q_i}^- < q_i < {q_i}^+$;
- velocity limits: $|\dot{q}_i| < \dot{q}_{imax}$;
- effort limits: $|\tau| < \tau_{max}$;
- self collision avoidance: $d(q) > 0$;
- self balance: $ZMP^- < ZMP < ZMP^+$;

In the case of considering weighting the regressor matrix, the criterion is described as:

$$C = cond(\mathbf{W_B} diag(\mathbf{Z})) \tag{1.13}$$

$\mathbf{Z}$ is the matrix of a priori knowledge of dynamic parameters which represents relative effect of each link's dynamic parameters. For instance, it could the weight of links which can be either already know from CAD model.

1.4.2 Other approaches

a) Static postures

Prior to the knowledge of the author, only 2 studies [10], [8] has attempted to estimate inertial parameters using static postures. In the earlier study, measurements were the x, y component of center of pressure (CoP) and joint angles measured by incremental encoders. In all the robot static postures used for measurements, necessary constraints were taken into account. Those are: joint's mechanical limits, the balance condition of robot, feet are not intersected, fixed feet configuration for several postures. Then, exciting postures were generated by setting randomly and manually certain variables such as position and orientation of the support foot, the height of CoM as well as the angle of waist level. This pool of postures was then applied

to the optimization problem that minimizes the condition number of the observation. The experiment failed to obtained enough numbers of exciting postures despite such an amount of effort being put in.

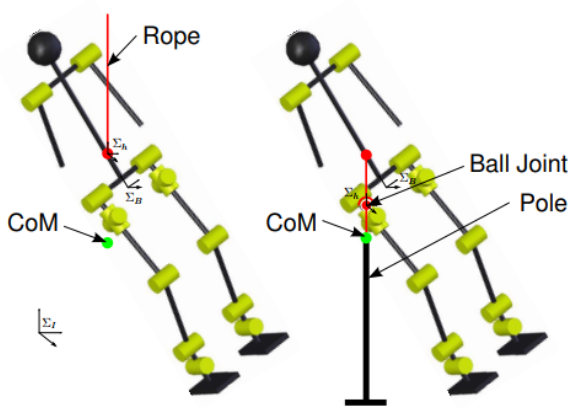

Figure 1.3: Torque-free mounting applied on the humanoid robot

In the second study, as mentioned above, an interesting idea of using torque-free mounting, illustrated in the figure 1.3 allowed to estimate important parameters such mass and CoM using only static postures. Data were collected from Inertial Measurement Unit (IMU) attached on upper body, and joint angle from encoders. Unlike the previous study, the author utilized the generic algorithm (GA) to search for optimal exciting static postures. First, within the working space of each joint, the angle was quantized into discrete positions equally distributed all over the working space. This resulted in a large set of different combinations for joint angles. Then, this is where the GA algorithm came in and searched for optimal exciting postures with criterion that is the ratio of largest singular value over the smallest singular value of the regressor matrix thanks to its linear independence characteristic. The steps of GA are generally described as: random selection from initial population, cross over, mutation, exit when condition is fulfilled. This method successfully generate a pool of exciting postures required for identification process.

b) Optimal trajectories generation using visual feedback

In order to estimate inertia of a system, dynamic motions are mandatory. Earlier, [9] proposed a method to estimate standard parameters of human body segments using real-time visual feedback. The approach intended to derive the standard parameters from the BP by solving QP problem, but without taking into account several important constraints. Instead of generating off-line a set of optimal exciting motions, this study proposed to utilize a visual feedback system, where an actor is visualized on screen along with modeled segments and colors that indicates how much that segment has been excited by the motion. When all the segments are considered excited enough for identification, the measurements of geometric parameters and external ground reaction force were performed by motion capture and force plate. Additionally, an extra weigh was added to different location of body with a purpose of generate new set of

base parameters. This approach did definitely have an advantage of real-time online motion generation, but it depended largely on the accuracy of prior standard parameters.

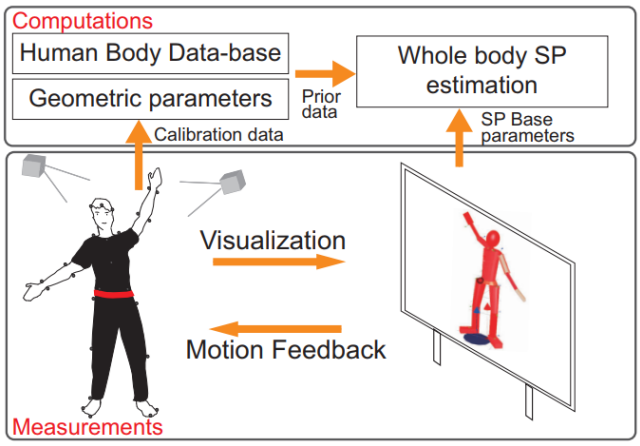

Figure 1.4: Diagram of the identification process using visual feedback

c) Improvement in solving the optimization problem of generating optimal trajectories

More recently, several research works have developed a significant improvement on performance of estimating standard parameters using optimal exciting motions. In the first study [14], the authors particularly aimed for a fast and reliable method to estimate some fundamental human body segment inertial parameters. Therefore, the choice of a simplified model with 3 planar segments to model the human body was determined. Taking into account of human body biomechanical constraints, a set of optimal exciting motions was obtained by using visual biofeedback system where 4 markers being tracked by Kinect camera and a feature-tracking software. The joint angles were also accurately obtained from this system despite being poorly estimated without markers. External GRF and resultant moments were measured by balance board. This approach did provide a good set of exciting motions for estimating, but it has a significant drawbacks such as motions are strictly limited to planar, simplified model would not be helpful for other purposes.

In [5], a more generalized approach was proposed, taken into account most of aspects in standard parameters estimation. In this approach, it had to deal with a complex model with large number of parameters. Instead of solving a tedious, giant nonlinear QP optimization problem, the author proposed to decouple the generation of optimal exciting motions by dividing into 2 processes: generating a set of static postures that excites static parameter such as mass and CoM, and then between two consecutive postures a optimal trajectory using B-spline interpolation technique to excite dynamic parameters i.e. inertia. These two processes correspond to two constrained optimization problems, taken into account all mechanical limits and dynamic balance. The criterion for excitation is still the condition number of each regressor

matrix but now it is modified by weighting to scale of each link's mass so that even small mass link can still be excited enough for the identification. [1]

1.5 Validation and application

Validating the identified parameters is a very important step which determines the success of the identification. There are many ways to evaluate the success of a model. It could direct validation or cross validation.

1.5.1 Direct validation

To justify the accuracy of the estimated BP, the relative standard deviation is calculated and assessed. Assuming that W_B is deterministic and e is zero mean with standard deviation σ. The variance-covariance matrix of the estimation error can be given as:

$$C_{\hat{\Phi}_B} = E[(\Phi_B - \hat{\Phi}_B)(\Phi_B - \hat{\Phi}_B)^T] = \sigma(W_B^T W_B)^{-1} \tag{1.14}$$

The absolute standard deviation of a i^{th} parameter can be found out as:

$$\sigma_i = \sqrt{C_{\hat{\Phi}_B}(i,i)} \tag{1.15}$$

From the expression above, the relative standard deviation of the parameter to justify the accuracy of the identification is obtained by:

$$\sigma_i\% = 100\% \frac{\sqrt{C_{\hat{\Phi}_B}(i,i)}}{\hat{\Phi}_{B_i}} \tag{1.16}$$

A threshold of minimum percentage of relative standard deviation is often to be defined to justify the accuracy of the estimated parameters. Most of the cases, it should be 10%, other wise the set data is considered to be not usable for identification. Then the process needs to be revised and repeated until it satisfy the threshold.

1.5.2 Cross validation

A more intuitive to validate a model is to perform experiments on the model obtained from identified parameters and compare with other measured results. The following methods have been seen in the literature: (1) carrying out experiments on different trajectories rather than optimal trajectories trajectories, then compared the values obtained from the identified model

with measured joint torques/measured ground reaction forces/measured joint positions; (2) re-identifying the parameters with the robot with known addition load added.

Chapter 2

Main Features of Dynamic Identification Toolbox - Figaro

This chapters will outline the main features of the dynamic identification toolbox, named Figaro, along with its main functions. The toolbox is arranged into three packages: modeling package, optimal trajectory package, identification algorithm package. Figaro is written in Python, using the common scientific library, Numpy; Rigid Body Dynamics Analysis software, Pinocchio; Flexible Collision Library, FCL; Interior Point Optimizer library, Ipopt; library for creating smooth cubic splines, NDCurves. The project is still actively in the developing phase; therefore, some of coding has not been applied with best practices in software development.

2.1 Modeling package

Inheriting methods and data structure from Pinocchio library, modeling package will provide users methods in a more adequate way adapting to identification problem.

2.1.1 Creating a robot object

The Pinocchio library allows to create the model of a robot by import files under urdf format or python code. One can create a robot object by calling:

Robot(path to urdf)

A fundamental point of a robot object is that it is of two main classes: Model, which includes physical description of the robot (kinematic tree, reference inertial parameters) and should not be modified; and Data, which contains all the values to be used for computation (positions, velocities, acceleration,...) and varying at any time of computation. With this basis, any algorithm in Pinocchio will start in a form of:

algorithm(model, data, arguments)

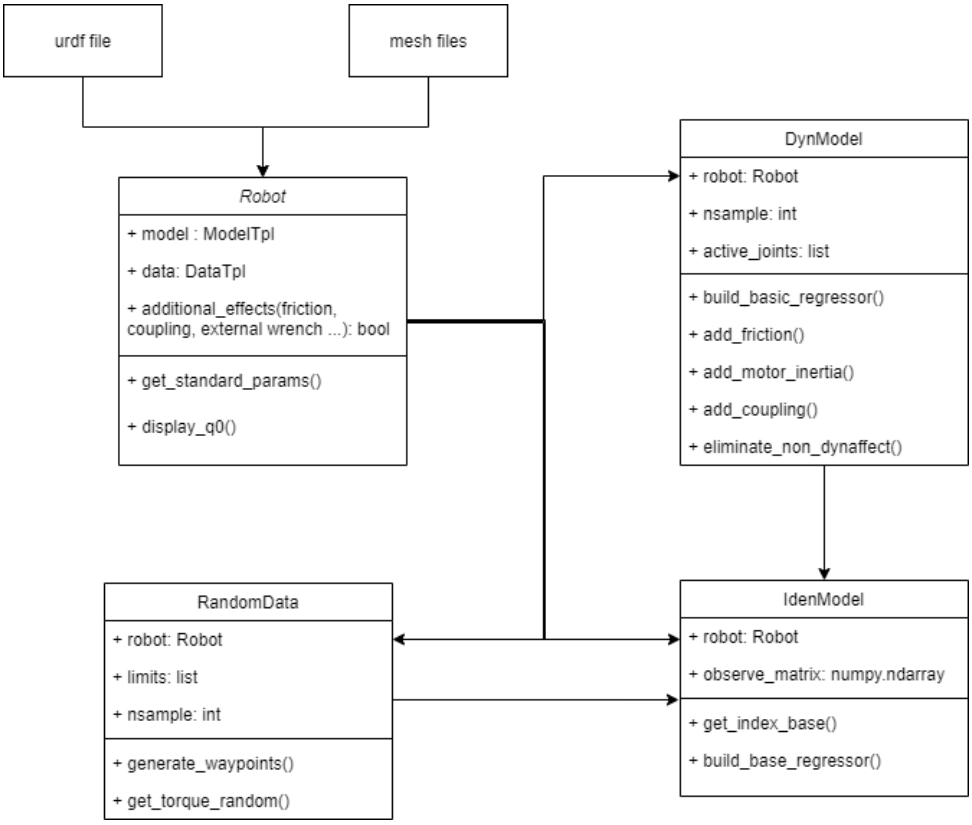

Figure 2.1: Class diagram of modeling package

2.1.2 Building dynamic model

a) Getting basic observation matrix

Thanks to the functionalities from Pinocchio, building a basic dynamic model can be done simply by calling:

build_basic_regressor(N,model, data, q, v, a)

- Arguments: data points of given joint configuration.
- Output: an observation matrix of dynamic model corresponding standard parameters parsing from urdf file.

b) Adding friction

To add friction to the dynamic model of the robot, one can call the function:
add_friction(robot, N)

- Arguments: robot object, N number of data points
- Output: updated observation matrix

c) Adding actuator inertia

To add actuator inertia to the dynamic model of the robot, one can call the function:
add_motor_inertia(robot, N)

- Arguments: robot object, N number of data points
- Output: updated observation matrix

d) Adding joint coupling

It is not always a case in robots, but in some robots, there is a mechanical joint coupling between links. Therefore, accounting this effect in dynamic model is necessary. One can call a function which is now only available for robot Staubli TX40, or create a function based on the template of this function for his/her specific robots.

add_coupling(robot, N, cpl_joints)

- Arguments: robot object, number of data points, coupling joints
- Output: updated observation matrix

e) Eliminating non-affecting parameters To eliminating the columns and standard parameters that have no effects the dynamics of the robot, one can call function:

eliminate_non_dynaffect(robot, W)

- Arguments: robot object, observation matrix
- Output: reduced observation matrix

2.1.3 Building identification model

a) Getting base parameters symbolic expression

After obtaining observation matrix with desired options, one can get the symbolic expression of base parameters by calling function:

get_index_base(W_e, params_r)

- Arguments: reduced observation matrix, list of reduced standard parameters
- Output: list of index of base parameters

b) Build base regressor

The base regressor is the main matrix that we will work with throughout the identification procedure. One can obtain the regressor matrix by calling:

build_base_regressor(W_e, index_base)

- Arguments: reduced observation matrix, index of base parameters
- Output: base regressor matrix

2.2 Optimal trajectory generation package

The package contains two main parts: trajectory planner and optimization solver. Firstly, we introduces trajectory planning modules that allow users to choose appropriate planners upon their application's needs, these including trapezoidal, double S curves, polynomial and cubic splines. The second part of the package presents the optimization solvers adopted from Ipopt library.

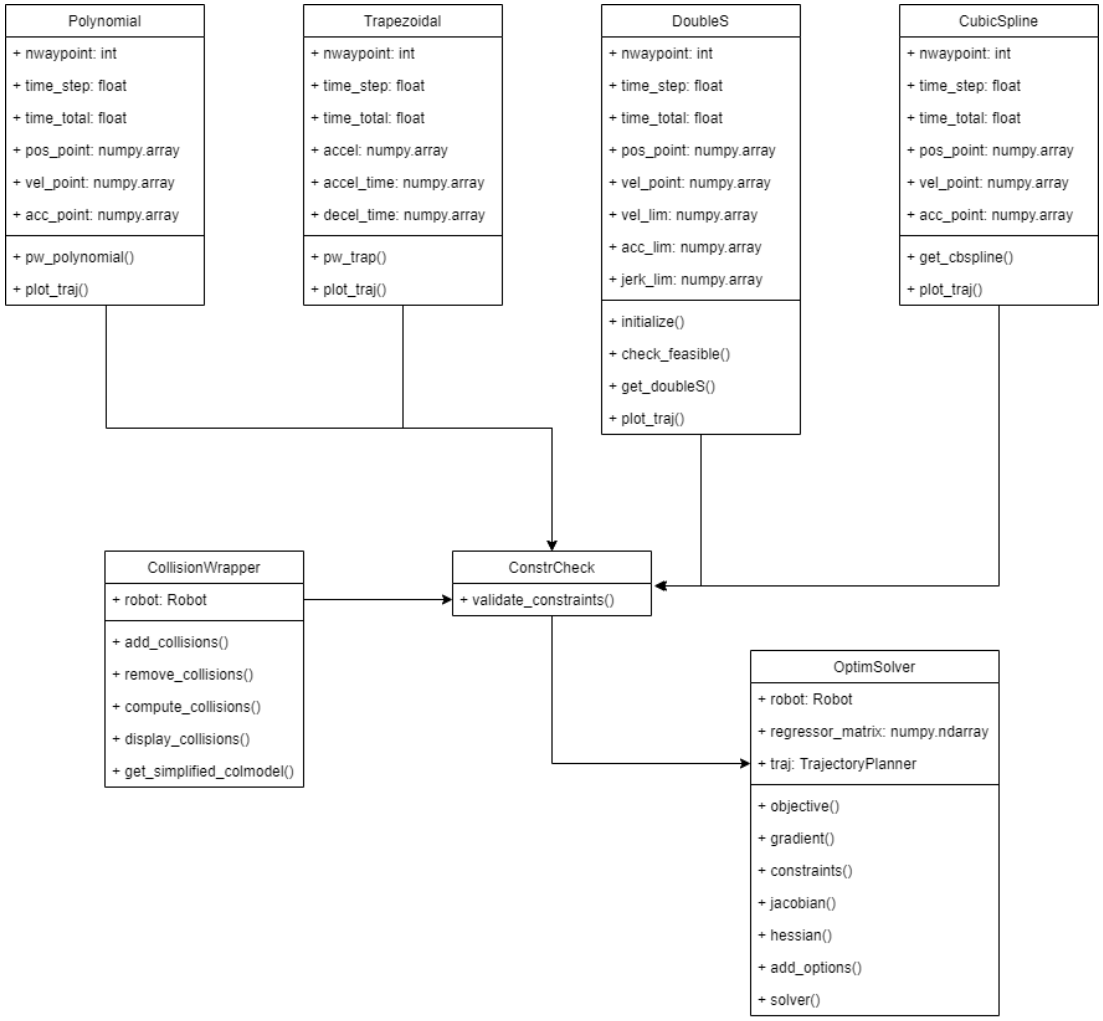

Figure 2.2: Class diagram of optimal trajectory generation package

2.2.1 Trajectory planner

In robotics, a trajectory refers to a sequence of points in either task space or joint space matching with a desired time history . Generating trajectories is to compute the reference for the control system as a function of time so that a robot can track the desired path. The technique to find

this function is called interpolation. Different types of interpolation between 2 points or via several intermediate points are presented in the following.

The NDCurves library allows user to create polynomial and cubic splines with various choices. In many applications of dynamic identification, trapezoidal and double S curves are also desired, as they have a segment of constant velocity in their velocity profile. This property specially becomes useful in investigating friction effects and how dynamics differs from each level of speed. Presented below are the functions covering all cases in planning a trapezoidal or double S curve.

One can create a desired trajectory by calling:
trajectory(robot, N, f, trajectory boundaries)

- Arguments: robot object, number of data points, frequency, boundaries
- Output: trajectory

a) Trapezoidal

One of the desire features of any trajectories is to have the continuity in position and velocity. Trapezoidal is an effective interpolating method to satisfy this criterion. In this method, the velocity profile consists 3 parts: accelerating, constant velocity, and decelerating. Accordingly, the profile of acceleration contains a non-zero constant acceleration duration, then a zero acceleration duration and finally a constant acceleration with an opposite sign of the first duration's one. As a result, the function of position is quadratic type blended with linear type.

Let's say a robot has:

- a number of joints defined as *njoints*
- a number of waypoints defined as *nwaypoints*
- an starting configuration q_0, dq_0 as initial position, velocity
- a desired total runtime t_f for the whole trajectory

Now, in order to generate trapezoidal trajectories for all joints, the following parameters need to be defined:

- acceleration corresponding to 3 phases in a trapezoidal profile: a_1,a_2, a_3
- time duration of 3 phases: δ_1, δ_2, δ_3

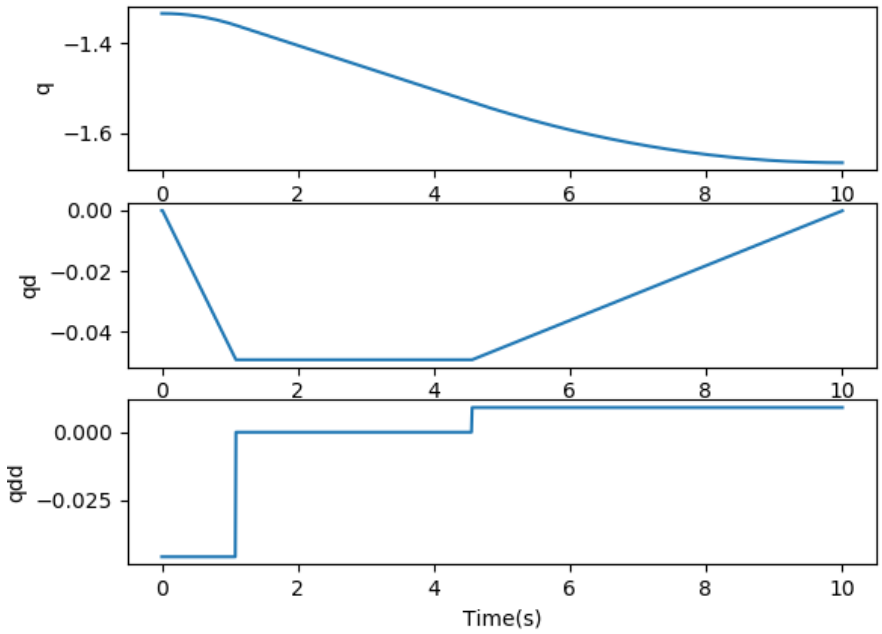

Figure 2.3: A trapezoidal trajectory between 2 points

By tuning these parameters, we can find desired trajectories upon the required criterion. For instance, we are to create a trapezoidal trajectory between 2 points for joint i within a runtime δ_f and the starting configuration q_{0i} and qd_{0i}. First of all, in the second phase, velocity maintains to be constant, so:

$$a_2 = 0 \tag{2.1}$$

Secondly, the change of velocity during the first phase and the third phase should be equal in term of magnitude. Consequently, we obtain a condition:

$$a_1\delta_1 = -a_3\delta_3 \tag{2.2}$$

A third condition is the overall time amount:

$$\delta_1 + \delta_2 + \delta_3 = \delta_f \tag{2.3}$$

Started with 6 parameters to be tuned, we only need to vary 3 parameters which are: either a_1 or a_3 and 2 values from δ_1, δ_2, δ_3. Then, the trajectory between two points for joint i is

presented as below:

$$\begin{cases} q_i(t) = q_{0i} + qd_{0i} * (t - t_i) + 1/2 * a_1 * (t - t_i)^2 \\ q_i(t) = q_{0i} + qd_{0i} * \delta_1 + 1/2 * a_1 * {\delta_1}^2 + (q_{d0} + a_1 * \delta_1) * (t - t_i - \delta_1) \\ q_i(t) = q_{0i} + qd_{0i} * (\delta_1 + \delta_2) + 1/2 * a_1 * ({\delta_1}^2 + 2 * \delta_1 * \delta_2) \\ \qquad + (q_{d0} + a_1 * \delta_1) * (t - t_i - \delta_1 - \delta_2) + 1/2 * a_3 * (t - t_i - \delta_1 - \delta_2)^2 \end{cases} \tag{2.4}$$

for 3 following phases, respectively: $\begin{cases} \text{for } t_i \leq t \leq t_i + \delta_1 \\ \text{for } t_i + \delta_1 \leq t \leq t_i + \delta_2 \\ \text{for } t_i + \delta_1 + \delta - 2 \leq t \leq t_i + \delta_f \end{cases}$

Joint limits, velocity, acceleration/torque constraints

While choosing the values of accelerations and time duration, we need to respect the limits of joint position, velocity and acceleration. For a joint i, the maximum absolute value of acceleration is a_{Mi}, then $|a_1|$ and $|a_3|$ should be smaller than this value. Similarly, the absolute value of velocity should be smaller than the maximum value v_{Mi}, and position should stay inside the range of joint limits $[{q_M}^-, {q_M}^+]$. These limits form a set of constraints as following:

$$\begin{cases} |a_{1i}|, |a_{3i}| \leq a_{Mi} \\ |qd_{0_i} + a_{1i} * \delta_{1i}| \leq v_{Mi} \\ {q_M}^- \leq q_i(t) \leq {q_M}^+ \end{cases} \tag{2.5}$$

Implementation and testing

Details of the code cen be found in the appendix section. The figure 2.4 shows an example of 3 trapezoidal trajectories of which each visits 4 waypoints.

b) Double S curves

A trapezoidal planner allows users to generate trajectories with continuity of velocity, but not with acceleration. A discontinue acceleration motion, which results in impulsive jerks, may cause mechanical stress and harms to the system, which normally results in undesired vibration. As a result, adopting a continuous acceleration profile into trajectories is demanded. A natural, evolving way from trapezoidal planners is to blend at the ends of accelerating/decelerating phase of velocity profile with parabolic blends. The resultant of this modification is illustrated in the figure 2.5. The shape of the velocity profile gave it the name of "double S"[15].

We detail a double S profile between two waypoints into 3 parts similar to the trapezoidal profile. Without losing the generality of the problem, we assume the final position is larger than the initial position. Indeed, the vice-versa case will be discussed at the end of the subsection.

Boundary conditions as well as some assumption for the simplicity are presented below:

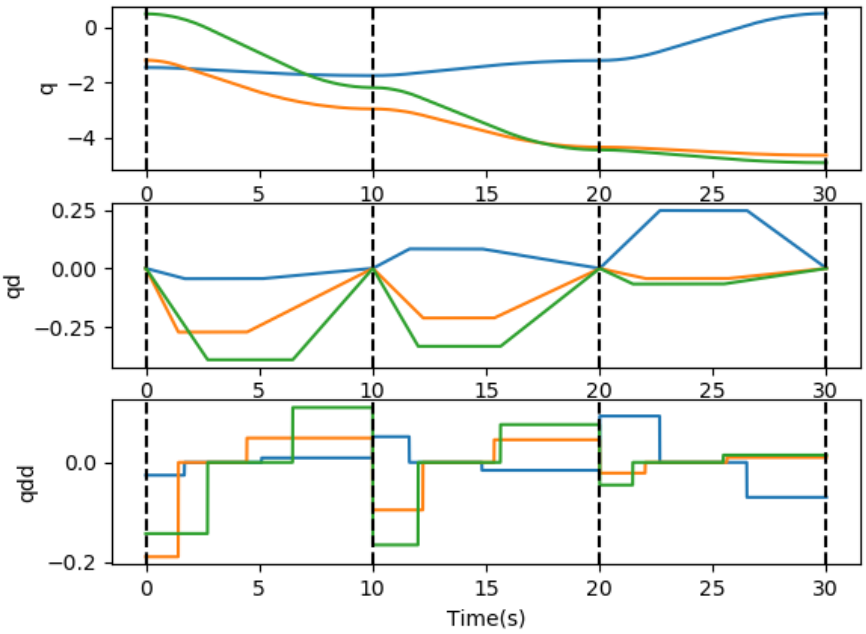

Figure 2.4: A trajectory for 3 joints with 4 waypoints

- Initial and final positions: $q_0 < q_1$
- Initial and final velocities: v_0, v_1
- Initial and final accelerations a_0, a_1 are set to zero.
- Maximum values of velocity, acceleration and jerk are $v_{max}, a_{max}, j_{max}$.
- Assume that the initial time is $t_0 = 0$.
- Assume that minimum values of velocity, acceleration, and jerk are $v_{min} = -v_{max}, a_{min} = -a_{max}, j_{min} = -j_{max}$.

Depending on values of boundary conditions, a double S trajectory may have all 3 phases or only 1 or 2 phases below:

- Accelerating phase: $0 < t < T_a$, acceleration increases linearly from zero, reaches its limits and maintains, then decreases linearly to zero within a duration of T_{j1} same as its increasing part.
- Constant velocity phase: $T_a < t < T_a + T_v$, acceleration is zero.
- Decelerating phase: $T - T_d < t < T = T_a + T_v + T_d$, opposite with the profile of accelerating phase with T_{j2} as the duration that acceleration changes linearly.

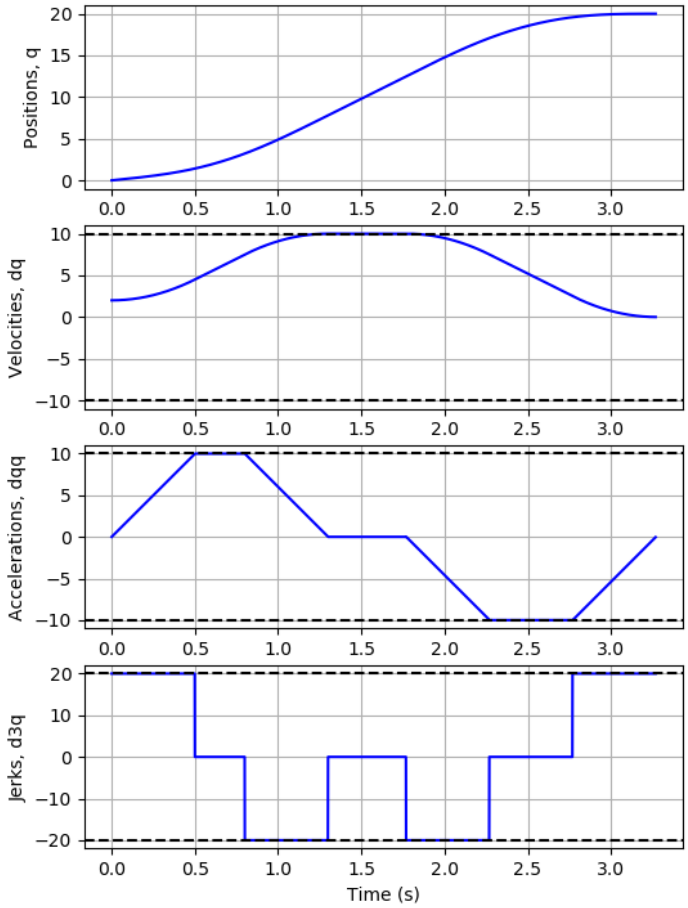

Figure 2.5: A double S curve

Checking feasibility of a double S trajectory: First of all, given those boundary conditions and constraints on kinematic limits, before looking for a solution of double S profile, we need to check if such an profile ever exists. For instance, a given displacement $q_1 - q_0$ is too small to perform a change of velocity from v_0 to v_1, as the area below the velocity profile with respect to time should be equal the magnitude of displacement.

Therefore, taking a minimal case of a double S profile where there is only one acceleration/deceleration phase as our limit case, as in figure 2.6, the given displacement should be larger or equal the area under this velocity profile. Otherwise, we can conclude a double S trajectory is not feasible to perform between two given waypoints with given conditions.

$$q_1 - q_0 > \begin{cases} \sqrt{\frac{|v_1 - v_0|}{j_{max}}}(v_1 + v_0) & , if \sqrt{\frac{|v_1 - v_0|}{j_{max}}} \leq \frac{a_{max}}{j_{max}} \\ \frac{1}{2}(\frac{a_{max}}{j_{max}} + \frac{|v_1 - v_0|}{a_{max}})(v_1 + v_0) & , if \sqrt{\frac{|v_1 - v_0|}{j_{max}}} > \frac{a_{max}}{j_{max}} \end{cases} \tag{2.6}$$

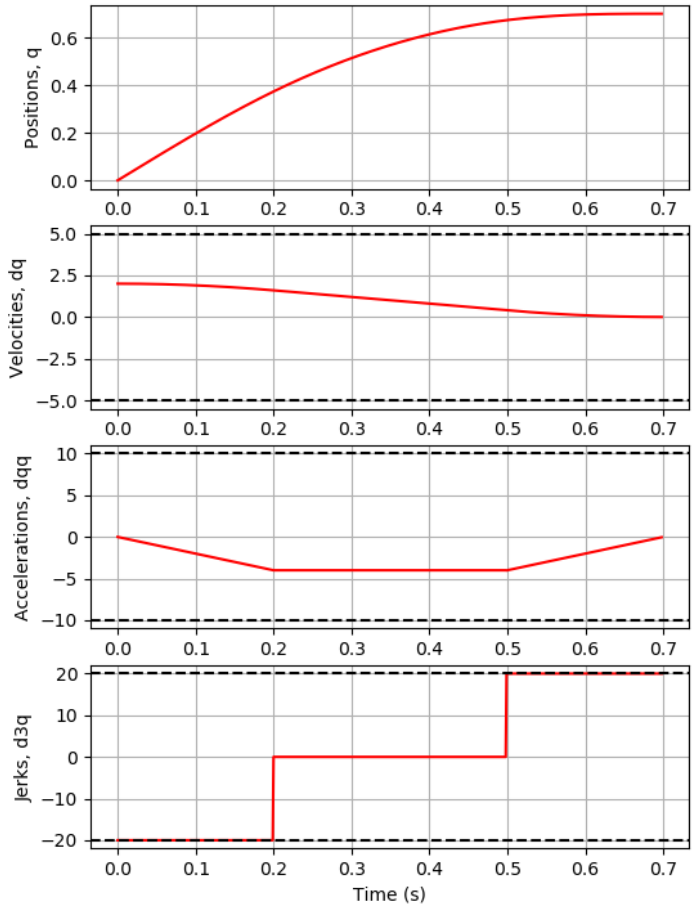

Figure 2.6: A minimum feasible case of double S curve

Checking if maximum velocity is reachable

An approach to check whether maximum velocity is reached or not is to calculate the duration of constant velocity, T_v, with the assumption of v_{max} reached. Thus, if T_v is smaller than zero, meaning maximum velocity is never reached, and vice-versa.

Checking if maximum(minimum) acceleration is reachable

For acceleration, we need to check for 2 phases. One is whether maximum acceleration is reached in accelerating phase and the other is whether minimum acceleration is reached in decelerating phase. The inequality conditions are:

$$\begin{cases} a_{max} \text{ is reached, if } \frac{(v_{max}-v_0)}{j_max} \geq {a_{max}}^2 \\ a_{min} \text{ is reached, if } \frac{(v_{max}-v_1)}{|j_min|} \geq {a_{min}}^2 \end{cases} \tag{2.7}$$

Cases of double S profiles

These expressions below are general expression for all cases considered above, where:

v_{lima}, v_{limd} :highest/lowest values of velocities no matter it reached maximum/minimum values or not

a_{lima}, a_{limd} :highest/lowest values of accelerations no matter it reached maximum/minimum values or not

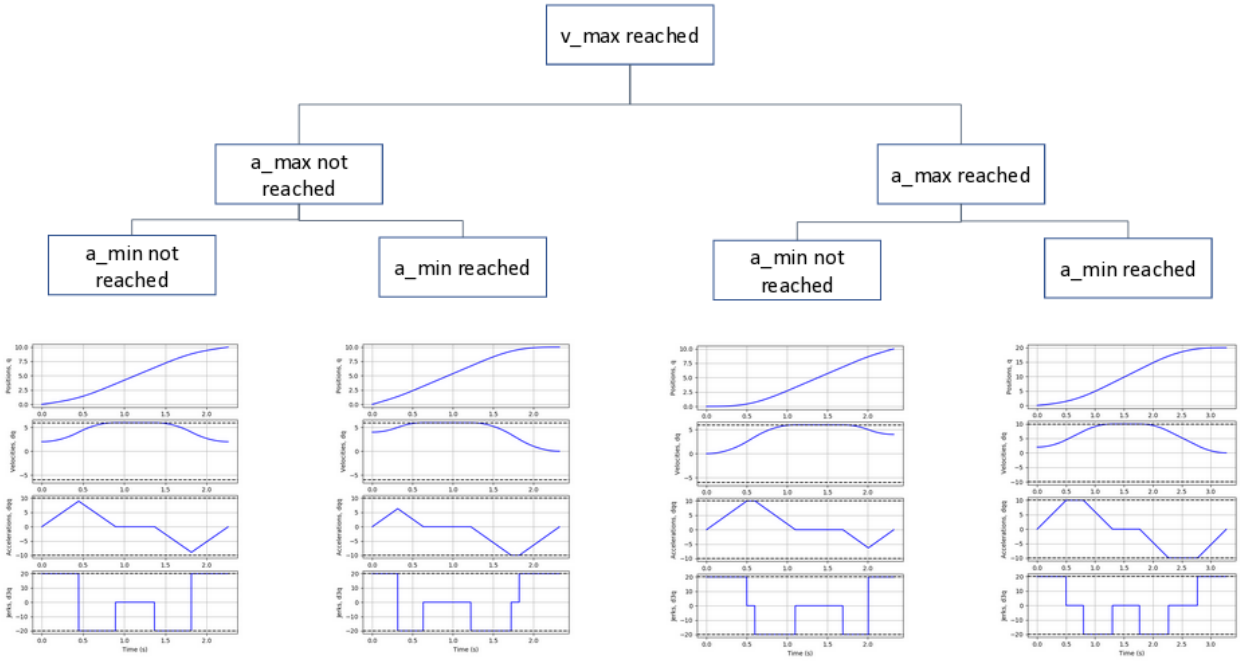

Figure 2.7: 4 cases with v_{max} reached

Case 1: In figure 2.7, v_{max} is reached + a_{max} reached/not reached + a_{min} reached/not reached, all 3 phases

If a_{max} is reached, duration of accelerating phase is calculated as:

$$T_{j1} = \frac{a_{max}}{j_{max}} \text{ and } T_a = T_{j1} + \frac{(v_{max} - v_0)}{a_{max}} \tag{2.8}$$

otherwise as,

$$T_{j1} = \sqrt{\frac{|v_{max} - v_0|}{j_{max}}} \text{ and } T_a = 2T_{j1} \tag{2.9}$$

If a_{min} is reached, duration of decelerating phase is calculated as:

$$T_{j1} = \frac{a_{max}}{j_{max}} \text{ and } T_a = T_{j1} + \frac{(v_{max} - v_1)}{a_{max}} \tag{2.10}$$

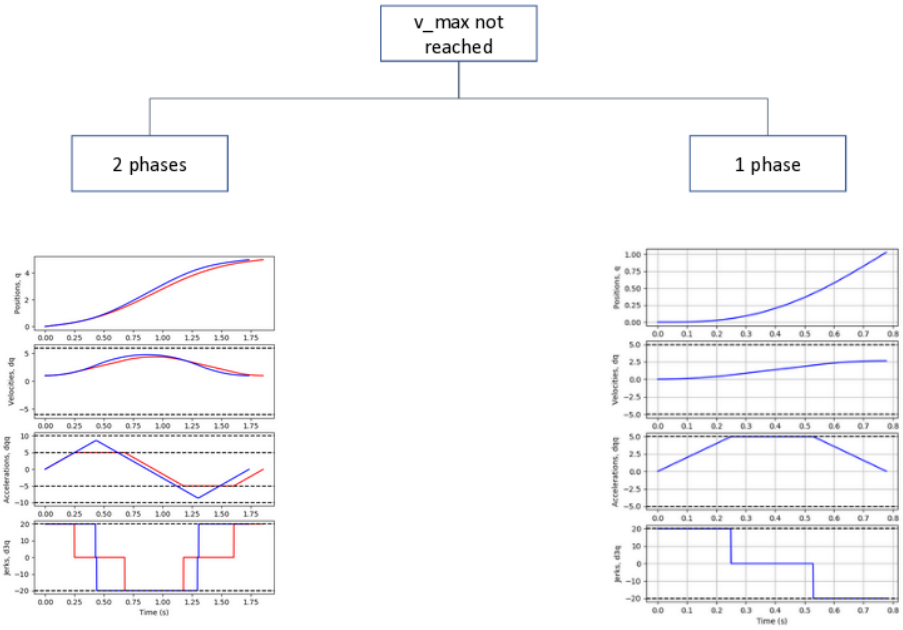

Figure 2.8: 2 cases with v_{max} not reached

otherwise as,

$$T_{j1} = \sqrt{\frac{|v_{max} - v_1|}{j_{max}}} \text{ and } T_a = 2T_{j1} \tag{2.11}$$

Duration of constant velocity phase then would be computed as:

$$T_v = \frac{q_1 - q_0}{v_{max}} - \frac{T_a}{2}(1 + \frac{v_0}{v_{max}}) - \frac{T_d}{2}(1 + \frac{v_1}{v_{max}}) \tag{2.12}$$

Case 2: In figure 2.8, v_{max} is not reached, only 2 phases, if a_{max}/a_{min} not reached (blue plot) , reduce value of a_{max} iteratively and repeat until both reached (red plot)

Since v_{max} is not reached, $T_v = 0$. Duration of accelerating phase and decelerating phase can be computed as below,

$$\begin{cases} T_{j1} & = \frac{a_{max}}{j_{max}} \\ T_a & = \frac{1}{2a_{max}} \left(\frac{{a_{max}}^2}{j_{max}} - 2v_0 \right. \\ & \quad \left. + \sqrt{\frac{{a_{max}}^4}{{j_{max}}^2} + 2({v_0}^2 + {v_1}^2) + a_{max}(4(q_1 - q_0) - 2\frac{a_{max}}{j_{max}}(v_0 + v_1))} \right) \end{cases} \tag{2.13}$$

$$\begin{cases} T_{j2} & = \frac{a_{min}}{j_{min}} \\ T_d & = \frac{1}{2a_{max}} \left(\frac{{a_{max}}^2}{j_{max}} - 2v_1 \right. \\ & \left. + \sqrt{\frac{{a_{max}}^4}{{j_{max}}^2} + 2({v_0}^2 + {v_1}^2) + a_{max}(4(q_1 - q_0) - 2\frac{a_{max}}{j_{max}}(v_0 + v_1))} \right) \end{cases} \tag{2.14}$$

Case 3: In figure 2.8, v_{max} is not reached, only 1 phases, deciding whether it is acceleration or deceleration.

If only accelerating phase is performed, then $T_v = 0$ and $T_d = 0$, T_a is calculated as:

$$\begin{cases} T_{j1} & = \frac{j_{max}(q_1 - q_0) - \sqrt{j_{max}(j_{max}(q_1 - q_0)^2 - (v_1 + v_0)^2(v_1 - v_0))}}{j_{max}(v_1 + v_0)} \\ T_a & = \frac{2(q_1 - q_0)}{v_1 + v_0} \end{cases} \tag{2.15}$$

If only accelerating phase is performed, then $T_v = 0$ and $T_a = 0$, T_d is calculated as:

$$\begin{cases} T_{j1} & = \frac{j_{max}(q_1 - q_0) - \sqrt{j_{max}(j_{max}(q_1 - q_0)^2 - (v_1 + v_0)^2(v_1 - v_0))}}{j_{max}(v_1 + v_0)} \\ T_a & = \frac{2(q_1 - q_0)}{v_1 + v_0} \end{cases} \tag{2.16}$$

Add a flowchart

Position, velocity, acceleration and jerk as functions of time in 3 phases:

Once a duration profile of double S curve has been defined,$[T_{j1}, T_{j2}, T_a, T_v, T_d]$, we can define the largest/lowest values of velocity and acceleration of the profile.

$$\begin{cases} v_{lima} & = v_0 + a_{lima}(T_a - T_{j1}) \\ v_{limd} & = v_1 - a_{limd}(T_d - T_{j2}) \\ a_{lima} & = j_{max} T_{j1} \\ a_{limd} & = j_{min} T_{j2} \end{cases} \tag{2.17}$$

Then, position, velocity, acceleration and jerk can be express as functions of time through 3 phase as following,

- Accelerating phase

  $t \in [0, T_{j1}]$

$$\begin{cases} q(t) & = q_0 + v_0 t + j_{max}\frac{t^3}{6} \\ \dot{q}(t) & = v_0 + j_{max}\frac{t^2}{2} \\ \ddot{q}(t) & = j_{max} t \\ q^{(3)}(t) & = j_{max} \end{cases} \tag{2.18}$$

$t \in [T_{j1}, T_a - T_{j1}]$

$$\begin{cases} q(t) & = q_0 + v_0 t + \frac{a_{lima}}{2}(3t^2 - 3T_{j1}t + {T_{j1}}^2) \\ \dot{q}(t) & = v_0 + a_{lima}(t - \frac{T_{j1}}{2}) \\ \ddot{q}(t) & = a_{lima} \\ q^{(3)}(t) & = 0 \end{cases} \tag{2.19}$$

$t \in [T_a - T_{j1}, T_a]$

$$\begin{cases} q(t) & = q_0 + (v_{lima} + v_0)\frac{T_a}{2} - v_{lima}(T_a - t) - j_{min}\frac{(T_a - t)^3}{6} \\ \dot{q}(t) & = v_{lim} + j_{min}\frac{(T_a = t)^2}{2} \\ \ddot{q}(t) & = -j_{min}(T - a - t) \\ q^{(3)}(t) & = j_{min} \end{cases} \tag{2.20}$$

- Constant-velocity phase

$t \in [T_a, T_a + T_v]$

$$\begin{cases} q(t) & = q_0 + (v_{lima} + v_0)\frac{T_a}{2} + v_{lima}(t - T_a) \\ \dot{q}(t) & = v_{lim} \\ \ddot{q}(t) & = 0 \\ q^{(3)}(t) & = 0 \end{cases} \tag{2.21}$$

- Decelerating phase

$t \in [T - T_d, T - T_d + T_{j2}]$

$$\begin{cases} q(t) & = q_1 - (v_{lim} + v_1)\frac{T_d}{2} + v_{lim}(t - T + T_d) - j_{max}\frac{(t - T + T_d)^3}{6} \\ \dot{q}(t) & = v_{limd} - j_{max}\frac{(t - T + T_d)^2}{2} \\ \ddot{q}(t) & = j_{max}(t - T + T_d) \\ q^{(3)}(t) & = j_{min} \end{cases} \tag{2.22}$$

$t \in [T - T_d + T_{j2}, T - T_{j2}]$

$$\begin{cases} q(t) & = q_1 - (v_{limd} + v_1)\frac{T_d}{2} + v_{limd}(t - T + T_d) + \\ & \frac{a_{limd}}{6}(3(t - T + T_d)^2 - 3T_{j2}(t - T + T_d) + {T_{j2}}^2) \\ \dot{q}(t) & = v_{limd} + a_{limd}(t - T + T_d - \frac{T_{j2}}{2}) \\ \ddot{q}(t) & = a_{limd} \\ q^{(3)}(t) & = 0 \end{cases} \tag{2.23}$$

$t \in [T - T_{j2}, T]$

$$\begin{cases} q(t) & = q_1 - v_1(T - t) - j_{max}\frac{(T-t)^3}{6} \\ \dot{q}(t) & = v_1 + j_{max}\frac{(T-t)^3}{2} \\ \ddot{q}(t) & = -j_{max}(T - t) \\ q^{(3)}(t) & = j_{max} \end{cases} \tag{2.24}$$

Implementation

The flowchart in the figure 2.9 illustrate the algorithm of generating a double S curve. Details of the code can be found in appendix section.

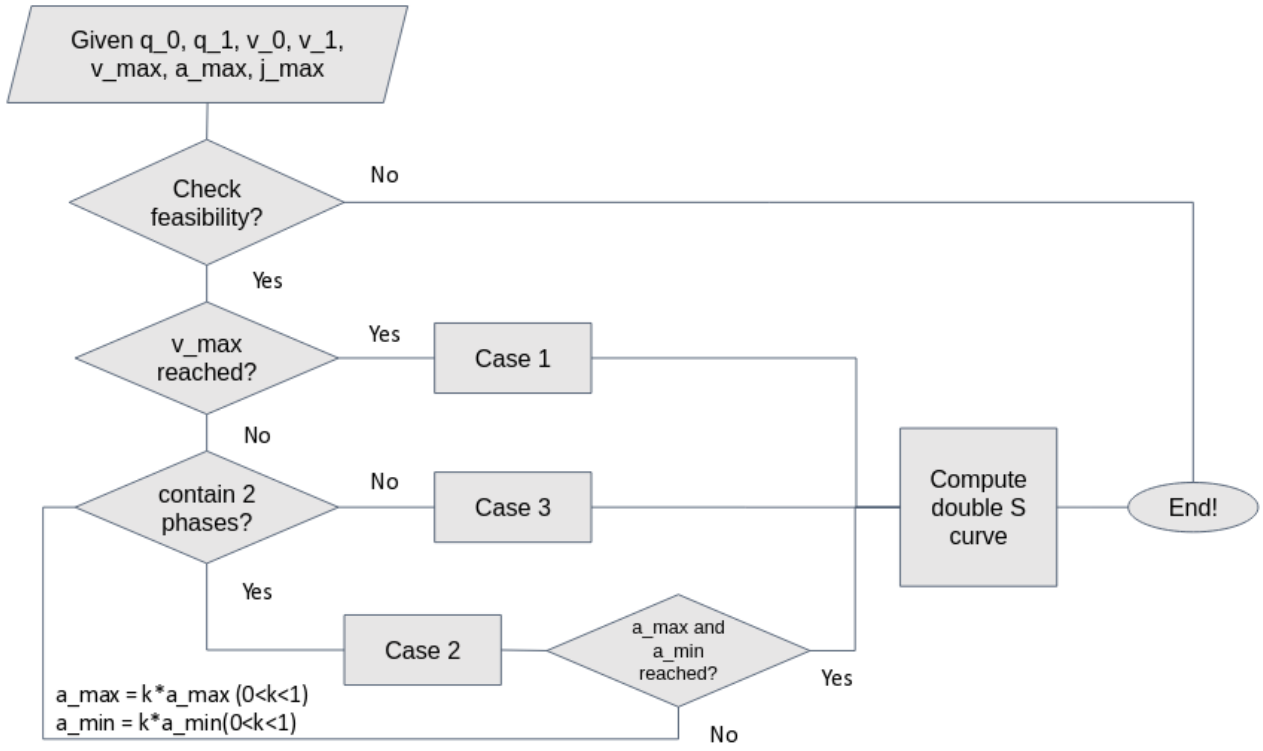

Figure 2.9: The flowchart of generating a double S curve

2.2.2 Optimization Problem

Ipopt is an Interior Point Optimizer which is best to be used in non linear problem with large size. This is also the reason why the author chose this library for finding optimal trajectory. One should expect a large number of data points, thus, a large size of a problem that an optimization solver needs to deal with. Therefore, the practice of minimizing unnecessary variables/constraints needs to be kept very well in mind. Also, a sound understanding of how Ipopt works is suggested when dealing non linear problem, so that the user can properly determine the objective function and initial guess. A sample of Ipopt problem is followed.

```
1        class Problem_cond_Wb:
2            def __init__(self, Ns, vel_wps, acc_wps,
3                         wp_init, vel_wp_init, acc_wp_init, W_stack,
   stop_flag):
4                self.W_stack = W_stack
5                self.wp_init = wp_init
6                self.vel_wp_init = vel_wp_init
7                self.acc_wp_init = acc_wp_init
8                self.vel_wps = vel_wps
9                self.acc_wps = acc_wps
10               self.stop_flag = stop_flag
11
12           def objective(self, X):
13               return objective_func(
14                   Ns, X, self.vel_wps, self.acc_wps, self.wp_init,
   W_stack_=self.W_stack
15               )
16
17           def gradient(self, X):
18               def obj_f(x): return self.objective(x)
19               grad_obj = nd.Gradient(obj_f)(X)
20               return grad_obj
21
22           def constraints(self, X):
23               constr_vec = get_constraints_all_samples(Ns,  X, self.
   vel_wps, self.acc_wps,
24                                                        self.wp_init, self.
   vel_wp_init, self.acc_wp_init)
25               return constr_vec
26
27           def jacobian(self, X):
28               def f(x): return self.constraints(x)
29               jac = nd.Jacobian(f)(X)
30               return jac
```

```
31
32              def hessian(self, X, lagrange, obj_factor):
33                  return False
34
35              def intermediate(
36                  self,
37                  alg_mod,
38                  iter_count,
39                  obj_value,
40                  inf_pr,
41                  inf_du,
42                  mu,
43                  d_norm,
44                  regularization_size,
45                  alpha_du,
46                  alpha_pr,
47                  ls_trials,
48              ):
49
50                  iter_num.append(iter_count)
51                  list_obj_value.append(obj_value)
52                  if self.stop_flag:
53                      return False
54
55
```

Listing 2.1: Ipopt problem

2.3 Identification algorithm package

2.3.1 Data preprocessing

Raw input data are usually very noisy, so a filtering step is needed especially in order to estimate velocity and acceleration by numerical difference. Numpy library and Scipy library provides various filtering methods to filtering out noise and outliers. The details of types and the set up of filters can be customized for specific robot. Therefore, one must explore by himself on this step. Even though, a sample filtering process is demonstrated on actual experimental with the robot Staubli TX40.

2.3.2 QR decomposition

Users have two options of implementing QR decomposition.

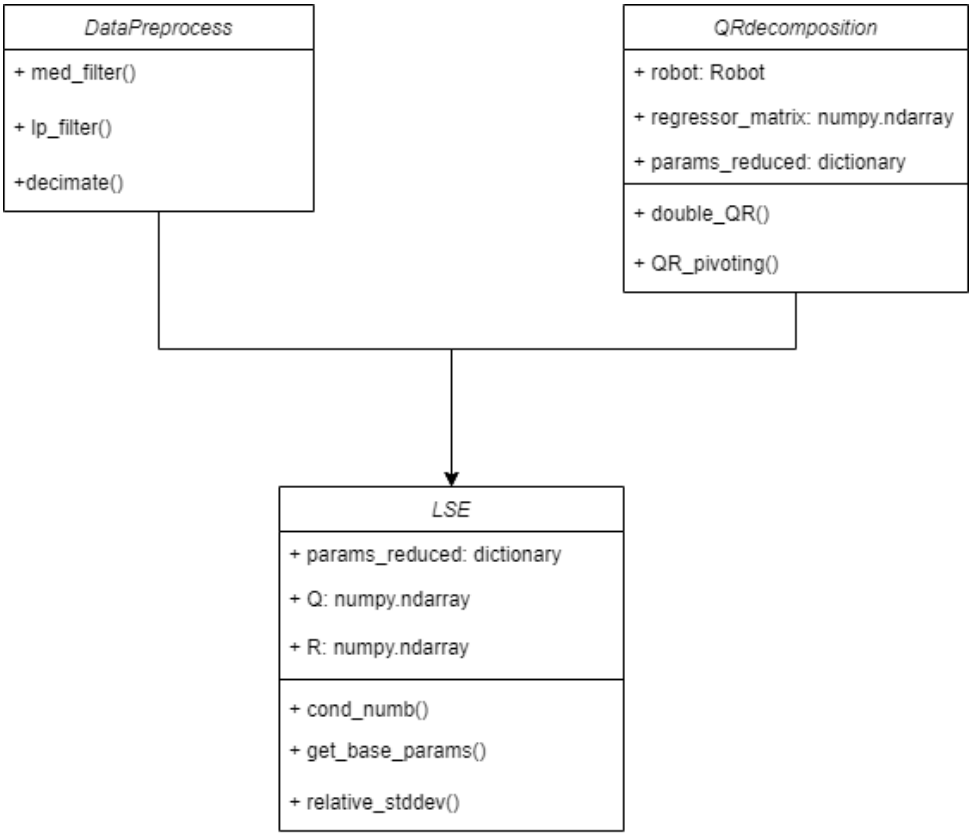

Figure 2.10: Class diagram of identification algorithm package

a) QR pivoting

The QR pivoting method will implement QR decomposition and sorting the columns of the matrix as order of the values on diagonal of the decomposed matrix.

QR_pivoting(W_e, tau, params_r)

- Arguments: reduced observation matrix, vector of joint efforts, list of base parameters
- Output: a dictionary of base parameters and its identified values

Note: This method would not respect the numbered order of joints, hence, the regrouping expression of base parameters can be different from symbolic expression obtained the following QR technique.

b) double QR

The double QR methods actually apply QR decomposition twice. First, QR decomposition is implemented in order to create symbolic expression of base parameters. After obtaining the base parameter expression and its corresponding base regressor, QR decomposition is applied second time to obtain the ordinary least square solution.

double_QR(W_e, tau, params_r)

- Arguments: reduced observation matrix, vector of joint efforts, list of base parameters
- Output: a dictionary of base parameters and its identified values

2.3.3 Least Square Estimation

Numpy library provides different method to do least square estimate such as QR decomposition, SVD decomposition or pseudo-inverse solution.

get_base_params(W_b, Q, R, option)

- Arguments: base regressor matrix, matrix Q, matrix R , option of OLS or WLS
- Output: base parameter values and their standard deviation

Chapter 3

Implementations and Experiments

In this chapter, the functionalities of the dynamic identification toolbox will be tested and verified. The first section will show an identification procedure in simulation with a simple 2 DOF model. After that, a more complete procedure of dynamic identification will be presented using provided experimental data with verified results for performance comparison on an industrial robot. Lastly, a standard pipeline including all steps in dynamic identification will be demonstrated with experimentation and analysis on a mobile base manipulator.

3.1 A simple simulation - 2 DOF manipulator

In this section, an identification method using QR decomposition with pivoting on a 2 DOF robot is presented. The theory of dynamic model as well as identification algorithms on how to handle rank deficiency using QR decomposition are detailed. Sample kinematic data on which the the identification process performs are randomly selected from an uniform distribution within a specific range. Following the theory, the implementation code and results of testing are also included.

3.1.1 Experiment designs

The Pinocchio library allows to import and specify the properties of a robot as well as define environment parameters regarding the interaction of the robot with environment. The geometric parameters including kinematic structure, number of links, joint types, location of joint frames, dynamic parameters such as mass, COM, inertia tensor, damping, and friction can be defined in URDF files.

a) Frames and transformation

Robot is a multi-body system which is described by multiple coordinate frames.

- There is a *global frame* which is fixed permanently.

- Attached to it is the base link B_o , which can have a fixed connection, free-floating w.r.t the global frame, with its reference frame called *local reference frame*.

- Similarly, any following link B_i is connected to its parent, the antecedent link B_{i-1} by joint J_i with its local reference frame.

- Inertial, visual, and collision frames are defined upon the local reference frame, serving to the calculation of dynamics, visualization and collision analysis.

Transformation from a base to a link or from a link to a link can be done by multiplying recursively transformation matrices between two consecutive frames.

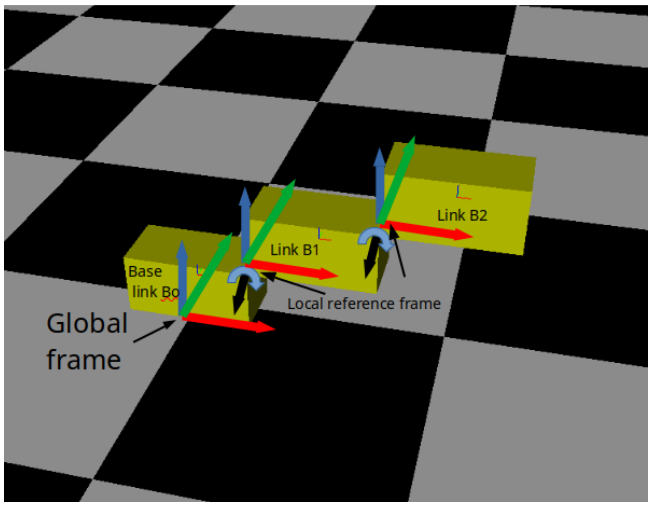

Figure 3.1: Frame structure of 2 DOF robot

b) Links and joints

The 2 DOF robot consist 3 identical box-shaped rigid links sizing of $0.25 \times 0.25 \times 0.5$ $(h \times w \times l)$. The link B_1 is mounted on the edge of base link B_0 by a revolute joint, following that is the link B_2 connects with link B_1 at the edge by another revolute joint. The two revolute joints are in the same direction and parallel with the y axis of the global frame as well as local reference frames, as illustrated in the figure 1.1.

If friction and inertia of actuators are not taken in account, dynamic properties of a single link can be described by its mass, its first moments of inertia and its inertia tensor. These values construct a set of standard inertial parameters for each link i: $\Phi = \{m_i, mX_i, mY_i, mZ_i, I_{XX_i}, I_{XY_i}, I_{YY_i}, I_{XZ_i}, I_{YZ_i}, I_{ZZ_i}\}$.

c) Miscellaneous parameters and notation

Vector gravity: $g = \begin{bmatrix} 0 \\ 0 \\ -9.81 \end{bmatrix} (m/s^2)$.

Velocity and acceleration of link i: $\begin{bmatrix} v_i \\ \omega_i \end{bmatrix}$ and $\begin{bmatrix} \dot{v}_i \\ \dot{\omega}_i \end{bmatrix}$.

Unit vector of frame i : $\{\mathbf{x_i}, \mathbf{y_i}, \mathbf{z_i}\}$

Wrench of a joint is defined as: $\begin{bmatrix} f_i \\ \tau_i \end{bmatrix}$

An robot object that contains information mentioned above can be called by:

robot = Robot(double_pendulum.urdf)

Model and data structure of the robot can be called by: *model = robot.model* and *data = robot.data*.

Names and reference values of standard parameters can be accessed by: *params_std = model(params_std)*

3.1.2 Methodology

In the inverse dynamic model of a system, the equations of motions clearly show that the output force/torque are actually linear with the inertial parameters such as mass, center of mass, inertia tensor. Thanks to the linearity of dynamic model in a rigid body system, one can take the advantage to create the identification model to identify inertial parameters. This model then would be used to form a linear least square problem. Solving the linear least square problem provides us the estimation of inertial parameters.

a) Dynamic model

Calculating velocities and accelerations in terms of generalized coordinates

For each link, velocities and accelerations are defined in its local reference frame.

For link B_1:

$$\begin{cases} v_1 = \mathbf{0} \\ \omega_1 = \dot{q}_1 \mathbf{y_1} \end{cases} \tag{3.1}$$

$$\begin{cases} \dot{v}_1 = \mathbf{0} \\ \dot{\omega}_1 = \ddot{q}_1 \mathbf{y_1} \end{cases} \tag{3.2}$$

For link B_2:

$$\begin{cases} v_2 = \frac{dO_0O_2}{dt} = v_1 + l_{O_1O_2}\dot{q}_1 \mathbf{z_1} \\ \omega_2 = (\dot{q}_1 + \dot{q}_1)\mathbf{y_2} \end{cases} \tag{3.3}$$

$$\begin{cases} \dot{v}_2 & = l_{O_1O_2}(\ddot{q}_1\mathbf{z_1} - \dot{q}_1^2\mathbf{y_1}) \\ & = l_{O_1O_2}((\ddot{q}_1\cos q_2 + \dot{q}_1^2\sin q_2)\mathbf{z_2} + (\ddot{q}_1\sin q_2 - {\dot{q}_1}^2\cos q_2)\mathbf{x_2}) \\ \dot{\omega}_2 & = (\ddot{q}_1 + \ddot{q}_1)\mathbf{y_2} \end{cases} \tag{3.4}$$

Recursive Newton-Euler algorithms

The dynamic model can be derived from either Lagrange formalism or Newton-Euler principles. For i^{th} body, the equation of motion w.r.t to frame i^{th} can be described as below using Newton-Euler principle. In a tree-structure or serial robot like this 2 DOF robot, Newton-Euler equations can be started from the free end link where no external forces are exerted on, then recursively following down the kinematic tree to the base link.

For link B_2, the equations are described as:

$$\begin{cases} f_2 + m_2 g = m_2\dot{v}_2 + mS_2 \times \dot{\omega}_2 + \omega_2 \times (\omega_2 \times mS_2) \\ \tau_2 + mS_2 \times g = \mathbf{I}_2\dot{\omega}_2 + mS_2 \times \dot{v}_2 + \omega_2 \times (\mathbf{I}_2\omega_2) \end{cases} \tag{3.5}$$

For link B_1, the dynamic equations are as below:

$$\begin{cases} f_1 - f_2 + m_1 g = m_1\dot{v}_1 + mS_1 \times \dot{\omega}_1 + \omega_1 \times (\omega_1 \times mS_1) \\ \tau_1 - \tau_2 - \overrightarrow{O_1O_2} \times f_2 + mS_1 \times g = \mathbf{I}_1\dot{\omega}_1 + mS_1 \times \dot{v}_1 + \omega_1 \times (\mathbf{I}_1\omega_1) \end{cases} \tag{3.6}$$

A basic dynamic model of the 2 DOF robot can be easily built by:

W = build_basic_regressor(N,model, data, q, v, a)

b) Identification model

Substituting expressions of velocities and acceleration from (1.1) − (1.4), the equations in generalized coordinates can be obtained. Since both revolute joints have the direction along the y-axis of global frame, only y component of joint torque vectors remains of interest. The expressions of them in generalized coordinates and standard inertial parameters,

$$\mathbf{W}\Phi = \boldsymbol{\tau_y} \tag{3.7}$$

,are presented as below:

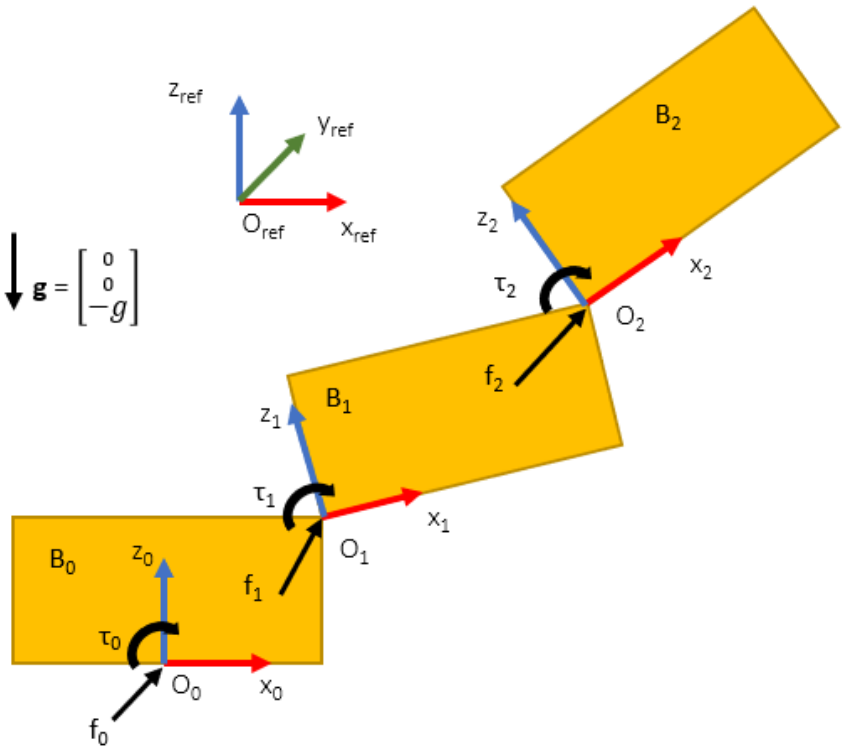

Figure 3.2: Diagram of 2 DOF robot

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ 0 & 0 & 0 & 0 & b_1 & b_2 & b_3 \end{bmatrix} \begin{bmatrix} mX_1 \\ mZ_1 \\ I_{YY_1} \\ m_2 \\ mX_2 \\ mZ_2 \\ I_{YY_2} \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{3.8}$$

where:

$$
\begin{aligned}
a_1 &= g\cos q_1 \\
a_2 &= -g\sin q_1 \\
a_3 &= \ddot{q}_1 \\
a_4 &= {l_{O_1O_2}}^2\ddot{q}_1 \\
a_5 &= l_{O_1O_2}\cos q_2(2\ddot{q}_1+\ddot{q}_2) - l_{O_1O_2}\sin q_2(\dot{q}_2^2+2\dot{q}_1\dot{q}_2) + g\cos(q_1+q_2) \\
a_6 &= -l_{O_1O_2}\sin q_2(2\ddot{q}_1+\ddot{q}_2) - l_{O_1O_2}\cos q_2(\dot{q}_2^2+2\dot{q}_1\dot{q}_2) - g\sin(q_1+q_2) \\
a_7 &= \ddot{q}_1+\ddot{q}_2 \\
b_1 &= l_{O_1O_2}(\ddot{q}_1\cos q_2+\dot{q}_1^2\sin q_2) + g\cos(q_1+q_2) \\
b_2 &= -l_{O_1O_2}(\ddot{q}_1\sin q_2-\dot{q}_1^2\cos q_2) - g\sin(q_1+q_2) \\
b_3 &= \ddot{q}_1+\ddot{q}_2
\end{aligned}
$$

Now, if we generate a trajectories that produces N samples of $(q,\dot{q},\ddot{q})$. We can obtain an identification model $\bar{\mathbf{W}}\Phi=\bar{\boldsymbol{\tau}}$ which has a form as:

$$
\begin{bmatrix}
W_{11} & W_{21} & . & . & . & W_{(m-1)1} & W_{m1} \\
. & . & . & . & . & . & . \\
. & . & . & . & . & . & . \\
. & . & . & . & . & . & . \\
W_{1n} & W_{2n} & . & . & . & W_{(m-1)n} & W_{mn}
\end{bmatrix}
\begin{bmatrix}
\varphi_1 \\ \varphi_2 \\ . \\ . \\ . \\ \varphi_{m-1} \\ \varphi_m
\end{bmatrix}
=
\begin{bmatrix}
\tau_1 \\ \tau_2 \\ . \\ . \\ . \\ . \\ \tau_n
\end{bmatrix}
\tag{3.9}
$$

$$
\begin{bmatrix}
W_{b_{11}} & W_{b_{21}} & . & . & . & W_{b_{(l-1)1}} & W_{b_{m1}} \\
. & . & . & . & . & . & . \\
. & . & . & . & . & . & . \\
. & . & . & . & . & . & . \\
W_{b_{1n}} & W_{b_{2n}} & . & . & . & W_{b_{(l-1)n}} & W_{b_{ln}}
\end{bmatrix}
\begin{bmatrix}
\varphi_{b_1} \\ \varphi_{b_2} \\ . \\ . \\ . \\ \varphi_{b_{(l-1)}} \\ \varphi_{b_l}
\end{bmatrix}
=
\begin{bmatrix}
\tau_1 \\ \tau_2 \\ . \\ . \\ . \\ . \\ \tau_n
\end{bmatrix}
\tag{3.10}
$$

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} & a_{51} & a_{61} & a_{71} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ a_{1N} & a_{2N} & a_{3N} & a_{4N} & a_{5N} & a_{6N} & a_{7N} \\ 0 & 0 & 0 & 0 & b_{11} & b_{21} & b_{31} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & b_{1N} & b_{2N} & b_{3N} \end{bmatrix} \begin{bmatrix} mX_1 \\ mZ_1 \\ I_{YY_1} \\ m_2 \\ mX_2 \\ mZ_2 \\ I_{YY_2} \end{bmatrix} = \begin{bmatrix} \tau_{11} \\ . \\ . \\ . \\ \tau_{1N} \\ \tau_{21} \\ . \\ . \\ . \\ \tau_{2N} \end{bmatrix} \tag{3.11}$$

Identification model with the inclusion of joint frictions

If we consider the friction at revolute joints can be modeled as:

$$\tau_F = \dot{q} f_v + sign(\dot{q}) f_c \tag{3.12}$$

where: f_v : viscosity coefficient and f_c : Coulomn coefficient.

then, the dynamic model in equation (1.8) becomes:

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & 0 & 0 \\ 0 & 0 & 0 & 0 & b_1 & b_2 & b_3 & 0 & 0 & b_4 & b_5 \end{bmatrix} \begin{bmatrix} mX_1 \\ mZ_1 \\ I_{YY_1} \\ m_2 \\ mX_2 \\ mZ_2 \\ I_{YY_2} \\ f_{v1} \\ f_{c1} \\ f_{v2} \\ f_{c2} \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{3.13}$$

where:

$$\begin{aligned} a_8 &= \dot{q}_1 \\ a_9 &= sign(\dot{q}_1) \\ b_4 &= \dot{q}_2 \\ b_5 &= sign(\dot{q}_2) \end{aligned}$$

To add friction to the existing dynamic model, one can simply call:

add_friction(robot, N)

Identification model w.r.t external wrench at the base link

In multi-body systems where the base link is not permanently fixed to the ground, such as humanoids, 6 equations of motion of the base link is sufficient to describe the dynamics of the whole system. This characteristic came from the fact that thanks to the recursivity of tree-structure system's dynamics where the children link can pass its dynamics to its parent's dynamic equations, consequently the base link as the final in that structure would include all of its children links' dynamics.

Therefore, the base link's dynamics can be used to identify base dynamic parameters of the system. Instead of using measurements of joint torques, it only requires 6 components of external wrench where the base link makes contact with environment.

Assuming in the case of 2 DOF robot, we can obtain the external wrench acting on the base link at the origin of base link's local reference frame. And if we assume that the base link is free-floating and stationary, the equations of motion for the base link B_0 are:

$$\begin{cases} f_0 - f_1 + m_0 g = 0 \\ \tau_0 - \tau_1 - \overrightarrow{O_0 O_1} \times f_1 + mS_0 \times g = 0 \end{cases} \tag{3.14}$$

In generalized coordinates, the equations can be described as a linear system of equations below.

$$\mathbf{W_o}_{(6\times 30)} \Phi_{(30\times 1)} = \begin{bmatrix} \mathbf{f_0}_{(3\times 1)} \\ \boldsymbol{\tau}_{0(3\times 1)} \end{bmatrix} \tag{3.15}$$

Since the equations of motion of the base link involves in all axes, we expect that there are more dynamic parameters presented in the identified results.

c) Identification algorithms

N samples of kinematic data are randomly chosen from a uniform distribution within a specific range, forming a sample trajectory. After that inputting the trajectory, Pinocchio library's dynamic algorithms allows users to calculate joint torques which correspond to the right side of equation (1.8) and generate a regressor matrix in agree with the matrix $\bar{\mathbf{W}}$ on the left side, accordingly.

Followed by previous steps, to get the identification model with regressor matrix and expression of base parameters. One can use the functions.

- To eliminate non-effect parameters: *W_e, params_r = eliminate_non_dynaffect(robot,W)*
- To get the base parameters expression: *index_base = get_index_base(W, params_r)*

- The regressor matrix of base parameters is: *W_b = build_base_regressor(W_e, index_base)*
- Finally, the identified results can be obtained by: *params_b = QR_pivoting(W_b, τ, params_r)*

Eliminating columns that has no effects on dynamics

By computing the L_2 norm of column vectors of the regressor, one can determine if a column has no effects on dynamics by comparing with a tolerance, σ, which is close to zero.

The diagonal of the product $\bar{\mathbf{W}}^T\bar{\mathbf{W}}$ would give the L_2 norm of columns vector. All columns that have L_2 norm smaller than tolerance, $diag[i] < \sigma$, will be eliminated. The initial regressor with 20 columns linking to 20 standard inertial parameters is expected to be reduced to 7 columns as $\bar{\mathbf{W}}$ in the symbolic equation (1.8).

QR decomposition with pivoting

The regressor in the identification problem does not guarantee column-linear independence, meaning there are columns as linear combination of other columns. By regrouping parameters according to this linear combination of corresponding columns, we create a minimum set of parameters, called base parameters. QR decomposition with pivoting is in favor to solve rank deficiency because it can provide an exact numerical rank of the rank-deficient regressor. Thus, by QR with pivoting, we can determine which columns to be deleted and how parameters can be regrouped.

The regressor $\bar{\mathbf{W}}$ can decomposed into 2 matrices $\mathbf{Q}$ and $\mathbf{R}$. A binary, unitary matrix $\mathbf{P}$ multiplying with $\bar{\mathbf{W}}$ so that the values on diagonal of matrix $\mathbf{R}$ decrease.

$$\bar{\mathbf{W}}\mathbf{P} = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}_{(2N-7)\times 7}\end{bmatrix} \tag{3.16}$$

$\bar{\mathbf{W}}$: the regressor, $(2N \times 7)$

$\mathbf{P}$: a permutation matrix, (7×7)

$\mathbf{Q}$: an orthogonal matrix, $(2N \times 2N)$

$\mathbf{R}$: an upper triangular matrix, (7×7)

By setting a threshold σ_r, one can determine the numerical rank by:

$$rank(\bar{\mathbf{W}}) = k : R_{kk} = min(R_{ii}) < \sigma_r \tag{3.17}$$

The threshold σ_r is determined as $\sigma_r = 2N * max(R_{ii}) * eps$ eps: machine epsilon.

Regrouping standard parameters into base parameters

The numerical rank k is also the number of base parameters. According, the matrices can be divided into two parts corresponding to linearly independent columns and linearly dependent columns as following:

$$\bar{\mathbf{W}}\mathbf{P} = \begin{bmatrix}\mathbf{W_1} & \mathbf{W_2}\end{bmatrix} = \begin{bmatrix}\mathbf{Q_1} & \mathbf{Q_2}\end{bmatrix} \begin{bmatrix}\mathbf{R_1} & \mathbf{R_2} \\ \mathbf{0}_{(2N-7)\times k} & \mathbf{0}_{(2N-7)\times 7-k}\end{bmatrix} = \begin{bmatrix}\mathbf{Q_1}\mathbf{R_1} & \mathbf{Q_1}\mathbf{R_2}\end{bmatrix} \tag{3.18}$$

Let $\mathbf{W_2} = \mathbf{W_1}\boldsymbol{\beta}$, then equation (1.7) becomes:

$$\begin{bmatrix}\mathbf{W_1} & \mathbf{W_2}\end{bmatrix} \begin{bmatrix}\mathbf{\Phi_1} \\ \mathbf{\Phi_2}\end{bmatrix} = \begin{bmatrix}\mathbf{W_1}\mathbf{\Phi_1} & \mathbf{W_1}\boldsymbol{\beta}\mathbf{\Phi_2}\end{bmatrix} = \mathbf{W_1}(\mathbf{\Phi_1} + \boldsymbol{\beta}\mathbf{\Phi_2}) = \bar{\boldsymbol{\tau}} \tag{3.19}$$

From the above equation, we can obtain the base regressor $\mathbf{W_b} = \mathbf{W_1}$, and base parameters as linear combination $\mathbf{\Phi_B} = \mathbf{\Phi_1} + \boldsymbol{\beta}\mathbf{\Phi_2}$. The coefficient $\boldsymbol{\beta}$ can be defined by:

$$\mathbf{W_2} = \mathbf{W_1}\boldsymbol{\beta} \Leftrightarrow \mathbf{Q_1}\mathbf{R_2} = \mathbf{Q_1}\mathbf{R_1}\boldsymbol{\beta} \Leftrightarrow \boldsymbol{\beta} = {\mathbf{R_1}}^{-1}\mathbf{R_2} \tag{3.20}$$

Finally, the identified values of base parameters are the solutions of linear least square on $\mathbf{W_B}$:

$$\mathbf{\Phi_B} = {\mathbf{R_1}}^{-1}{\mathbf{Q_1}}^T\bar{\boldsymbol{\tau}} \tag{3.21}$$

3.1.3 Result and discussion

A test performed on a trajectory with 1000 samples is shown below. As expected, the base parameters are correctly estimated compared with the standard values from 3D model. In addition, details of the regrouping are presented which indicates the advantage of this identification method.

a) Comments on results identification using joint torques measurement

- The identified base parameters did not include the mass of link 1. The reason is that the gravity effect which mass of link 1 contributes to the dynamics is already included in the first moment of inertia.
- The mass of link 2 is regrouped with all identified parameters of link. This indicates the dynamic effect of link 2 on link 1. The value $0.25 * m_2$ in $mz_1 + 0.25 * m_2$ and $0.5 * m_2$ in $mx_1 + 0.5 * m_2$ are the product of link 2's mass and distance from origin of frame F_2 to origin of frame F_1 along z and x axis. The value $0.3125 * m_2$ in $Iyy_1 + 0.3125 * m_2$ is equal to $(0.5^2 + 0.25^2) * m_2$.

b) Comments on results identification using external wrench

- The identified base parameters using external wrench showed that it is not possible to identify masses of links separately. This is because the gravity force of 3 links are accumulated in the base dynamic equations.

	Standard Parameters	Value		Standard Parameters	Value
1	m0	1.000000	18	Ixz1	-0.031250
2	mx0	0.000000	19	Iyz1	0.000000
3	my0	0.000000	20	Izz1	0.088542
4	mz0	0.125000	21	m2	1.000000
5	Ixx0	0.026042	22	mx2	0.250000
6	Ixy0	0.000000	23	my2	0.000000
7	Iyy0	0.041667	24	mz2	0.125000
8	Ixz0	0.000000	25	Ixx2	0.026042
9	Iyz0	0.000000	26	Ixy2	0.000000
10	Izz0	0.026042	27	Iyy2	0.104167
11	m1	1.000000	28	Ixz2	-0.031250
12	mx1	0.250000	29	Iyz2	0.000000
13	my1	0.000000	30	Izz2	0.088542
14	mz1	0.125000	31	fv1	0.050000
15	Ixx1	0.026042	32	fc1	0.010000
16	Ixy1	0.000000	33	fv2	0.050000
17	Iyy1	0.104167	34	fc2	0.010000

Table 3.1: Standard inertial parameters of links B_0, B_1, B_2, and friction coefficients

	Base Parameters	Value
1	mx2	0.250000
2	mz2	0.125000
3	mx1 + 0.5*m2	0.750000
4	mz1 + 0.25*m2	0.375000
5	Iyy2	0.104167
6	Iyy1 + 0.3125*m2	0.416667

(a) Without the inclusion of friction model

	Base Parameters	Value
1	mx2	0.250000
2	mz2	0.125000
3	mx1 + 0.5*m2	0.750000
4	mz1 + 0.25*m2	0.375000
5	Iyy2	0.104167
6	fc2	0.010000
7	fc1	0.010000
8	Iyy1 + 0.3125*m2	0.416667
9	fv2	0.050000
10	fv1	0.050000

(b) With the inclusion of friction model

Table 3.2: Identified base parameters

- Unlike the identified base parameters using joint torques, there are parameters in all axes involved. The reason is that we took into account the dynamics on all 6 axes at the base link.

- Identified base parameters did not include the friction coefficients. This confirmed that joint frictions are reaction force/torque between two consecutive links, and they are canceled in the dynamics of the base link.

	Base Parameters	Value
0	m2 + 1.0*m0 + 1.0*m1	3.000000
1	my0 + 1.0*my1 + 1.0*my2	-0.000000
2	mx0 - 0.25*m0	-0.250000
3	mz2	0.125000
4	mx2	0.250000
5	mz1 - 0.25*m0 - 0.25*m1	-0.375000
6	mx1 - 0.5*m0 - 0.5*m1	-0.750000
7	Ixy2	-0.000000
8	Iyz2	0.000000
9	Iyy2	0.104167
10	Ixy1 - 0.5*my2	0.000000
11	Iyz1 - 0.25*my2	0.000000
12	Iyy1 - 0.3125*m0 - 0.3125*m1	-0.520833

(a) Without friction inclusion

	Base Parameters	Value
0	m2 + 1.0*m1 + 1.0*m0	3.000000
1	my0 + 1.0*my1 + 1.0*my2	0.000000
2	mx0 - 0.25*m0	-0.250000
3	mz2	0.125000
4	mx2	0.250000
5	mz1 - 0.25*m1 - 0.25*m0	-0.375000
6	Ixy2	0.000000
7	Iyz2	-0.000000
8	mx1 - 0.5*m1 - 0.5*m0	-0.750000
9	Iyy2	0.104167
10	Ixy1 - 0.5*my2	-0.000000
11	Iyz1 - 0.25*my2	0.000000
12	Iyy1 - 0.3125*m1 - 0.3125*m0	-0.520833

(b) With friction inclusion

Table 3.3: Identified base parameters using external wrench

3.2 Implementation on given data - Staubli TX40

Staubli TX40 is a classical 6 DOF industrial manipulator robot, which has been a popular research platform for dynamic identification. In this section, the author will introduce a complete identification procedure using the developed toolbox with given experimental data. The given experimental data was generated and analyzed by other researchers. Identified parameters were obtained with validation and published in the literature. The aim of this section is to reproduce the identification procedure and compared with identified results of other researchers in literature.

3.2.1 Experiment designs

a) Robot description

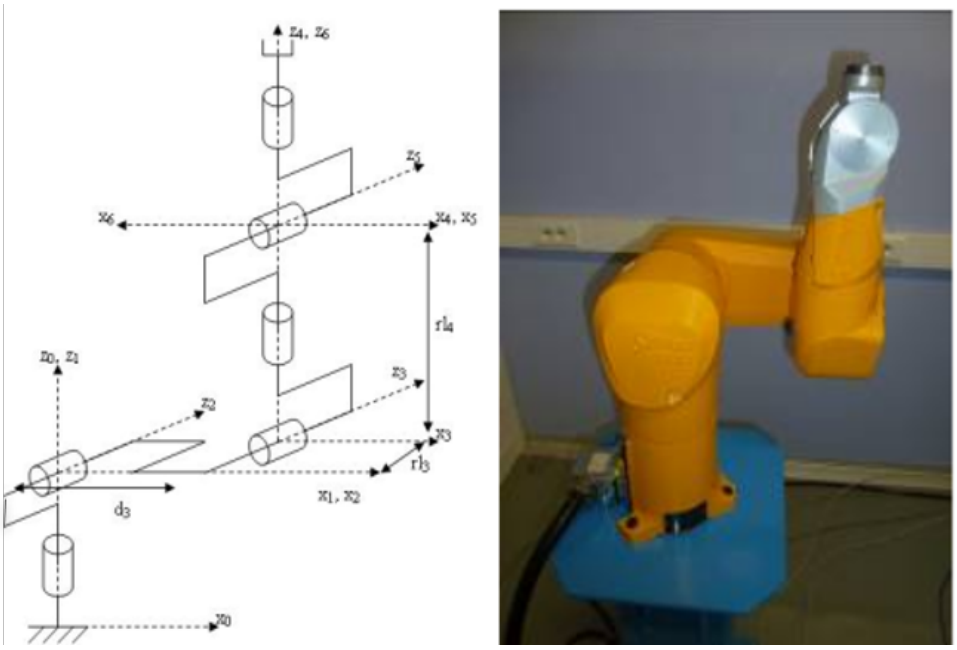

Figure 3.3: Staubli TX40 and its kinematic structure

The robot TX40 consists of 6 revolute joints, in which joint 5 and joint 6 are mechanically coupled. At each joint, encoder can measure and sample at high frequency (5kHZ) which gives an advantage of collecting data. The robot is not equipped with torque sensors, so joint torques values are estimated from actuator current references. Details of working space, joints limits,gear reduction ratio can be found in the catalogue of the manufacture Staubli.

b) Experimental data collection

The experiment was carried at Ecole Centrale de Nantes, by M. Gautier [4]. In the experiment, the robot followed an optimal S curves trajectory within a duration of 8 seconds. As illustrated below, the collected raw motor angle data is presented. Beside it are the estimated values of motor torques from current references of the current amplifiers which supply brushless motors.

3.2.2 Methodology

In the subsection, a step-by-step procedure of dynamic identification for Staubli TX40 will be introduced.

a) Converting MDH modeling format to urdf format

Since the modeling of TX40 was done with modified Denavit-Hartenberg convention, it is necessary to convert to urdf format in order to be utilizable in Pinocchio library. The table below

presents the MDH table for the robot TX40. Followed by the specific rules of MDH convention, the zero position of MDH modeling is different from the robot's default zero. Meanwhile, urdf format provides exactly modeling matching actual structure of the robot including reference parameters such as joint limits, velocity limits, effort limits, and so on.

With the toolbox, users can input the MDH table and additional information of the robot, then the toolbox would generate the appropriate urdf format of robot modeling.

j	σ_j	α_j	d_j	θ_j	r_j
1	0	0	0	q_1	0
2	0	-π/2	0	$q_2 - \pi/2$	0
3	0	0	$d3 = 0.225$m	$q_3 + \pi/2$	$rl3 = 0.035$m
4	0	π/2	0	q_4	$rl4 = 0.225$m
5	0	-π/2	0	q_5	0
6	0	π/2	0	$q_6 + \pi$	0
7	2	0	0	0	0

Figure 3.4: Modified DH table for robot TX40

joint	x (m)	y (m)	z (m)	roll (rad)	pitch (rad)	yaw (rad)
1	0	0	0.320	0	0	0
2	0	0	0	-π/2	0	0
3	0.225	0	0.035	0	0	0
4	0	-0.225	0	π/2	0	0
5	0	0	0	π/2	0	0
6	0	0	0	π/2	π	0

Figure 3.5: Transformation table for robot TX40

After converting mDH format to urdf format, a robot object of TX40 in Pinocchio can be instantiated by:

TX40 = Robot(TX_40.urdf)

model = TX40.model and *data = TX40.data*

A dictionary contains names and reference values of 87 standard parameters can be found through the function:

params_std = model.params_std

b) Dynamic model

In term of dynamic model, in addition to the basic dynamic model, with TX40 we considered additional factors which might contribute to the dynamics of the robot such as friction, actuator inertia, offset of joint effort and coupling effect at the wrist between joint 5 and 6.

With that given consideration, the full dynamic model of TX40 as followed:

$$\mathbf{M}(q)\ddot{q} + \mathbf{H}(q,\dot{q}) + \mathbf{G}(q) + \mathbf{F_v}\dot{q} + \mathbf{F_s}\text{sign}(\dot{q}) + \mathbf{F_{off}} + \mathbf{I_a}\ddot{q} = \tau \tag{3.22}$$

where:

- q, τ are joint positions and joint forces/torques, respectively,
- $\mathbf{M}(q)$ is the symmetric positive definite inertial matrix,
- $\mathbf{H}(q,\dot{q})$ is the matrix representing Coriolis and centrifugal forces,
- $\mathbf{G}(q)$ represents gravitational force.
- $\mathbf{F_v}$ is viscous friction constants vector.
- $\mathbf{F_s}$ is Coulomb friction constant vector.
- $\mathbf{F_{off}}$ is off joint effort vector.
- $\mathbf{I_a}$ is actuator inertia vector.

The coupling effect was included in the formula calculating joint torques and joint velocity as following [4].

Coupling effect between joint 5 and 6 :

$$\begin{bmatrix} \dot{q}r_5 \\ \dot{q}r_6 \end{bmatrix} = \begin{bmatrix} N_5 = 45 & 0 \\ N_6 = 32 & N_6 = 32 \end{bmatrix} \begin{bmatrix} \dot{q}_5 \\ \dot{q}_6 \end{bmatrix},$$
$$\begin{bmatrix} \tau_{c_5} \\ \tau_{c_6} \end{bmatrix} = \begin{bmatrix} N_5 & N_6 \\ 0 & N_6 \end{bmatrix} \begin{bmatrix} \tau_{r_5} \\ \tau_{r_6} \end{bmatrix}$$

Where:

$\dot{q}_{ri}$: *rotor velocity*
$\dot{q}_i$: *joint velocity*
τ_{ci}: *motor torque*
τ_{ri}: *electro − magnetic torque*
N_i: *transmission ratio*

Figure 3.6: Coupling effect of robot TX40

The friction induced by the coupling effect is presented as:

In conclusion, in addition to 10 standard inertia parameters $\{m,\ m_X,\ m_Y,\ m_Z,\ I_{XX},\ I_{XY},\ I_{XZ},\ I_{YY},\ I_{YZ},\ I_{ZZ}\}$ representing for mass, first of moments of inertia, inertia tensor, we

Coupling effect:

$$\tau_{c_5} = \tau_5 + Ia_{m_6}\ddot{q}_6 + Fv_{m6}\dot{q}_6 + Fs_{m6}\,\mathrm{sign}(\dot{q}_5 + \dot{q}_6),$$
$$\tau_{c_6} = \tau_6 + Ia_{m6}\ddot{q}_5 + Fv_{m6}\dot{q}_5 + Fs_{m6}\,\mathrm{sign}(\dot{q}_5 + \dot{q}_6)$$

Figure 3.7: Dynamics of coupling joints

included 4 more parameters for each joints $\{f_v,\ f_s,\ f_{off},\ I_a\}$, 3 parameters of coupling effects $\{I_{am6},\ f_{vm6},\ f_{sm6}\}$. Overall, the robot TX40 has 87 standard parameters dynamic parameters.

In the toolbox a full dynamic model of TX40 can be obtained by:

W = build_full_regressor(N, model, data, q, v, a)

Or, it can be built by calling a basic dynamic model, then calling individual effects that are necessary.

c) Identification model

To obtain a identification model, one need obtain two things which are a set of base parameters and its corresponding regressor matrix. These are all derived from dynamic model presented above.

As stated in the state of the art chapter, there are two ways to obtain linear combination of dynamic parameters, which is referred as base parameters. One is to manually derive from symbolic formulas, one is to use numerical method. For the sake of the generality, the toolbox only utilizes the latter method.

The details of applying QR decomposition to obtain base parameters are introduced above. Similarly, with a step by step approach, random data is generated. Then, the data is used to construct the observation matrix by substituting to the dynamic model. After that, columns with zero magnitude will be eliminated. These columns corresponded to the non-effect parameters to the dynamics of the robot which explained the zero magnitude. In general, this is set with threshold of 10^{-6}. After this step, we obtained reduced observation matrix. Next, QR decomposition is applied, separating the reduced observation matrix into two matrices. Mathematically detailed above, we obtained the regressor matrix and the expressions of base parameters.

We obtained a set of 61 base parameters. This is a minimal set of dynamic parameters that can fully describe the dynamics of robot TX40.

d) Data preprocessing

Data preprocessing is a process to remove the unwanted data so that only should more valuable and meaningful information be extracted from the collected data. As dynamic identification relies on statistical estimating techniques, noisy data oftentimes could be one of the biggest obstacles. Therefore, data preprocesing is a crucial step of dynamic identification. In literature, some practical guide lines for tuning filters and for avoiding a biased estimation

1	Izz1 + 1.0*Ia1 + 1.0*Iyy2 + 1.0*Iyy3 + 0.07*mz3 + 0.05185*m3 + 0.05185*m4 + 0.05185*m5 + 0.05185*m6
2	fv1
3	fs1
4	off1
5	Ixx2 - 1.0*Iyy2 - 0.050625*m3 - 0.050625*m4 - 0.050625*m5 - 0.050625*m6
6	Ixy2
7	Ixz2 - 0.225*mz3 - 0.007875*m3 - 0.007875*m4 - 0.007875*m5 - 0.007875*m6
8	Iyz2
9	Izz2 + 1.0*Ia2 + 0.050625*m3 + 0.050625*m4 + 0.050625*m5 + 0.050625*m6
10	mx2 + 0.225*m3 + 0.225*m4 + 0.225*m5 + 0.225*m6
11	my2
12	fv2
13	fs2
14	off2
15	Ixx3 - 1.0*Iyy3 + 1.0*Iyy4 + 0.45*mz4 + 0.050625*m4 + 0.050625*m5 + 0.050625*m6
16	Ixy3
17	Ixz3
18	Iyz3
19	Izz3 + 1.0*Iyy4 + 0.45*mz4 + 0.050625*m4 + 0.050625*m5 + 0.050625*m6
20	mx3
21	my3 - 1.0*mz4 - 0.225*m4 - 0.225*m5 - 0.225*m6
22	Ia3
23	fv3
24	fs3
25	off3
26	Ixx4 - 1.0*Iyy4 + 1.0*Iyy5
27	Ixy4
28	Ixz4
29	Iyz4
30	Izz4 + 1.0*Iyy5
31	mx4

Figure 3.8: Expressions of base parameters of TX40 (1)

of velocities and accelerations, taking advantages of non-causal off-line pass band filtering are presented in [16].

Median filter

Data from encoders and actuator current are collected and sampled at high frequency. As it can be seen in the figure below, there are noise and uncertainties in the signal. Since the data from encoders will be used to estimate joint velocities and joint accelerations by numerical difference, it is essential to eliminate out-liers and smoothen the noisy data. In order to eliminate out-liers, a median filter was applied on the position data. A median filter functions by a principle of replacing the current value by the median value of a batch of consecutive values including that current value.

Median filter can be called from scipy.signal library by: *scipy.signal.med_filt()*

Low-pass filter

Secondly, the sampling frequency was 5 kHz for encoders, apparently noise will be pick up with a high frequency like that. Furthermore, a dynamical system does not actual operate at that high frequency. Hence, a low-pass filter that only allows to pass certain signal under a specific frequency. For data from TX40,a zero-phase Butterworth low-pass filter was applied.

A zero-phase low-pass filter can be found in scipy.signal library by *scipy.signal.filtfilt()*

32	my4 + 1.0*mz5
33	Ia4
34	fv4
35	fs4
36	off4
37	Ixx5 - 1.0*Iyy5 + 1.0*Iyy6
38	Ixy5
39	Ixz5
40	Iyz5
41	Izz5 + 1.0*Iyy6
42	mx5
43	my5 - 1.0*mz6
44	Ia5
45	fv5
46	fs5
47	off5
48	Ixx6 - 1.0*Iyy6
49	Ixy6
50	Ixz6
51	Iyz6
52	Izz6
53	mx6
54	my6
55	Ia6
56	fv6
57	fs6
58	off6
59	Iam6
60	fvm6
61	fsm6

Figure 3.9: Expressions of base parameters of TX40 (2)

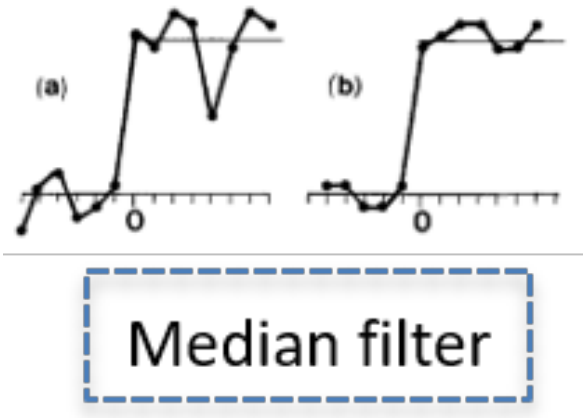

Figure 3.10: A median filter

Truncating trivial data

One needs to pay attention to trivial data. Normally, when data is recorded, there is trivial data at the beginning and ending part. This data should not be counted as data input.Furthermore, the data at the ends of sequences should be removed after filtering as they have been distorted by filtering.

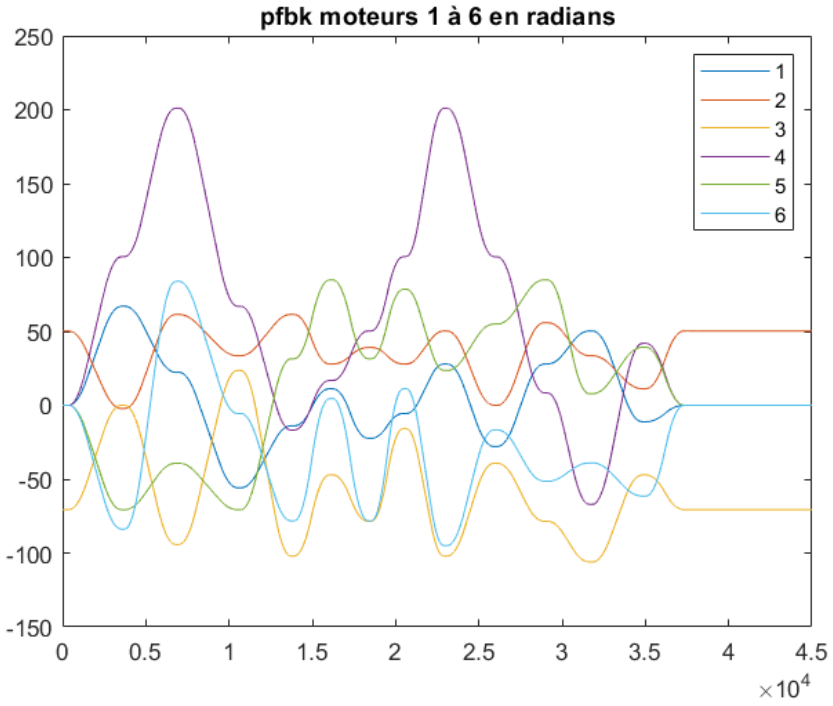

Figure 3.11: Motor angles measured at 5 kHz

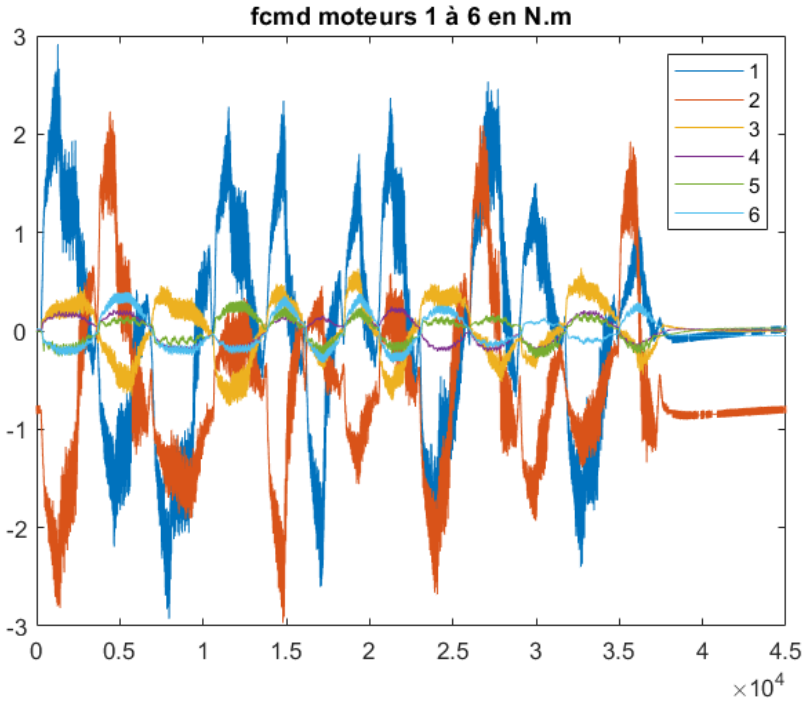

Figure 3.12: Motor torques estimated from current references

Zero-crossing velocity elimination

Near-zero velocity of the motion, the friction is contributed by a non linear effect called Stribeck effect as seen in the figure 3.8. Modeling this non linear effect could be difficult and it

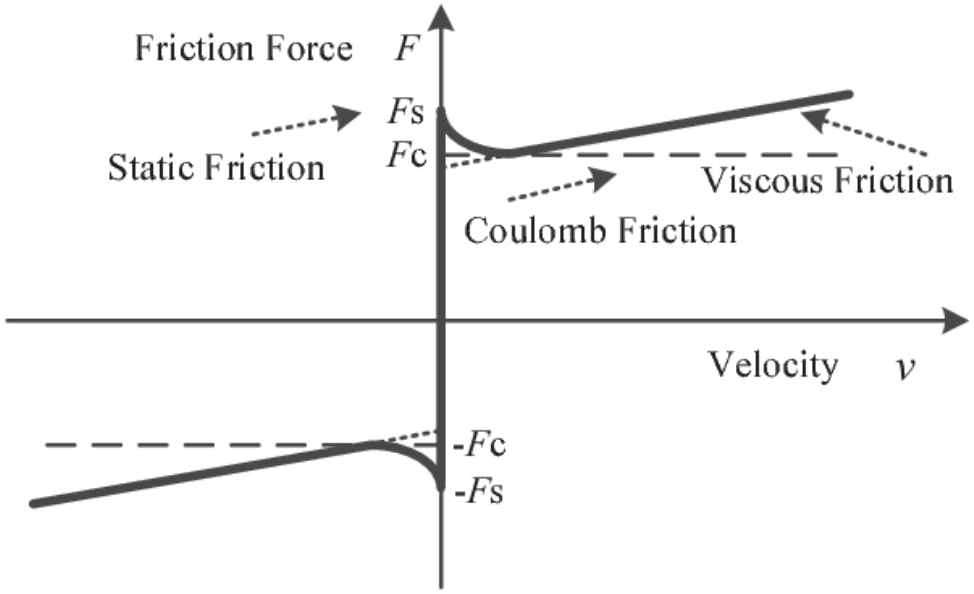

Figure 3.13: Stribeck friction effect

usually is ignored by only considering Coulomb friction. As a result, to keep the identification from this non linear effect, better should we keep this term out of the equation. Hence, data associating near-zero velocities should be eliminated. This is a small amount of data compared to the total collected data.

Parallel decimation

Similar to the data collected by encoders, the data from actuator currents also contained noises. We need to apply a low-pass filter to the joint torques data estimated from actuator currents. Morever, due to high frequency recording, we can pick up meaningful data points without taking the cost of computation by downsampling. Combining low-pass filter and downsampling, a parallel decimation filter can be applied. It should be noted that the filter should be applied on not only the joint torques vector but also the columns of observation matrix. Since they are in a linear relation, thus, our estimation would not be affected by the filter.

Decimation data can be done by function: *scipy.signal.decimate()*.

e) Identification algorithm

After obtaining necessary preprocessed data and identification model, we can proceed the next step which is estimating the dynamic parameters values. Similar to working steps with 2DOF robot, the steps to obtain base parameters are following:

- To eliminate non-effect parameters: *W_e, params_r = eliminate_non_dynaffect(robot,W)*
- To get the base parameters expression: *index_base = get_index_base(W, params_r)*
- The regressor matrix of base parameters is: *W_b = build_base_regressor(W_e, index_base)*

- Finally, the identified results can be obtained by: *params_b = double_QR(W_b, τ, params_r)*

Weighted Least Square Estimation

The results are above is of ordinary least square estimate. In order to obtain results with weighted least square estimate, one must call for ordinary least square, then get the results and variance of estimated values.

Calling an additional function, we can get the results from WLS estimate: *params_wls = get_wls(params_b, stddev)* with *stddev = get_stddev(τ, W_b, params_b).*

3.2.3 Result and discussion

The figures below has shown the identified results, the ones marked with "thanh" are identified by the toolbox, the ones marked with "prof_gautier" are provided from literature [4].

a) Comparison with identified results

While the provided results has been verified with cross validation by being used to track a test trajectory and matched closely with the measurement of torques, the toolbox has given identified results which are close in a group of base parameters, not in another group. One can clearly see that the closely matched parameters have large inertia, but the diverge results are all with large deviation,corresponding to not identifiable parameters. This lead to the introduction of another set of parameters, namely essential parameters which are presented in the following paragraph.

The details of comparison can be found the appendix with full description.

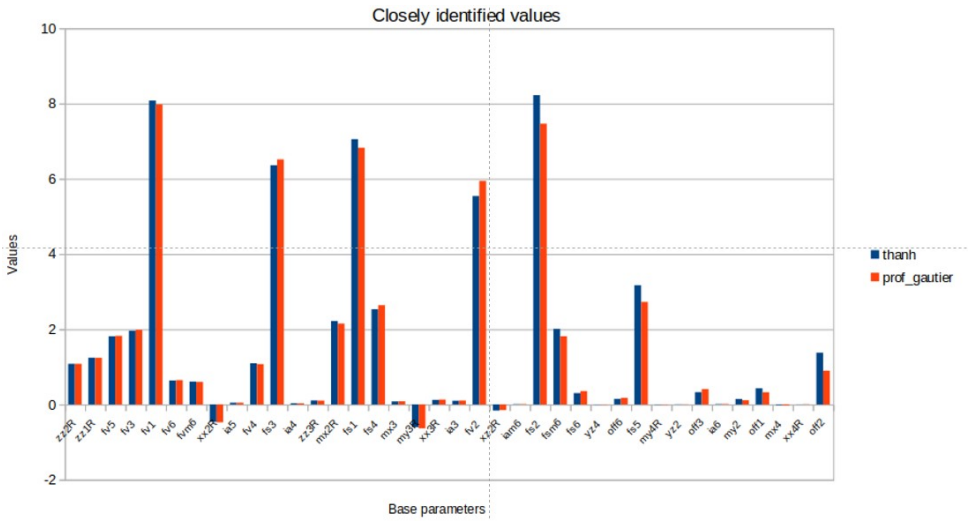

Figure 3.14: Comparison of identified parameters of TX40 with results from literature - close values

b) Essential parameters

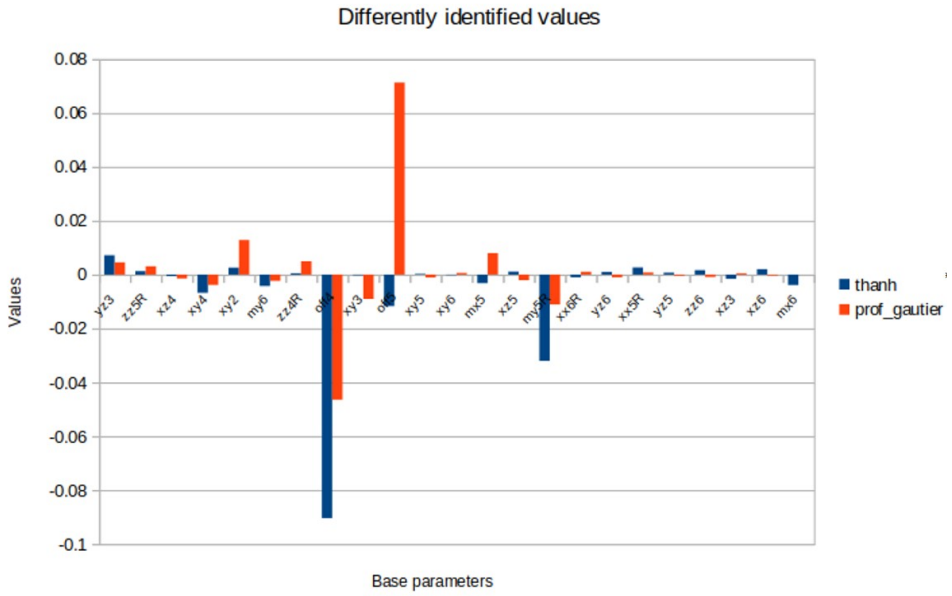

Figure 3.15: Comparison of identified parameters of TX40 with results from literature - diverge values

It has not been the focus in the dynamic identification of robotics, but essential parameters can be seen a more practical results that one should look for. Essential parameters are a group of base parameters which are dominant the dynamics of the system, why keeping the model of the system more simpler. In the case of TX40, the number of essential parameters is 28, compared to 61 base parameters, while the sum of residual error between estimated torques from two models by essential parameters and base parameters remain less than 2%.

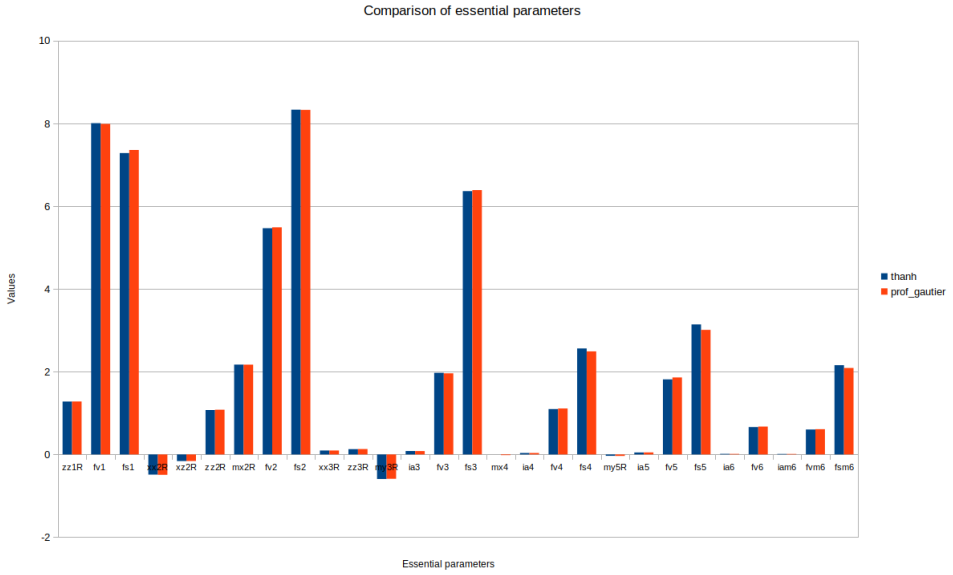

Figure 3.16: Comparison of identified essential parameters of TX40 with results from literature

The algorithm of obtaining essential parameters can be described as followed.

- After performing weighted least square estimation, check if the ratio between the largest deviation and smallest deviation is large than a threshold.
- If yes, remove the parameter with the largest deviation, then repeat step 1 until the ratio becomes equal or smaller than the threshold.

Reasoning for this algorithm is pretty straight forward. The parameter with large deviation, meaning its estimation has been spread over the space. In other word, the parameter has not actually contribute much in dynamics of the robot so that it could form a clear linear relation that should be realized by linear regression technique such as least square estimation. In the table of comparison of identified results, one may easily recognise that those parameters with large deviation have small magnitude which explains their slight impact on dynamics.

In the toolbox, the step of finding essential parameters is repeating WLS estimation. Therefore, it is straight forward to implement with a while loop.

3.3 A complete procedure - TIAGo

In the first section of the chapter, the dynamic identification on a simulated manipulator has shown how the basic methods should be called and what output would be expected from each method. However, it had excluded such a significant effect of uncertainty from obtaining data from actual system. Therefore, the second section had naturally transitioned to the next step of dealing with these uncertainties by implementing toolbox on a set of raw experimental with well documented data. The final step to complete the whole procedure of dynamic identification is to generate exciting trajectories that would produce experimental data for analysis. This is the main purpose we aimed to achieve in the experimentation with TIAGo, a mobile base manipulator.

3.3.1 Experiment designs

a) Robot descriptions

TIAGo is a mobile base manipulator composing of a mobile base, a torso, an arm with a wrist, an end-effector and head. This experimentation will focus on identifying dynamic parameters of the torso and the arm and and the wrist's parameters. The torso is mounted on the mobile base and allows to attach the arm and the head on it. It contains of 1 lift joint with a stroke of 35 cm which permits the translational up and down motion. The arm and wrist has 7 revolute joints with a reaching range of 87 cm. The last two joints at the wrist are coupled and driven by a differential face gear set. The following figure and table gives the

detail specification of the robot TIAGo. About software, TIAGo can be controlled utilizing ROS system and data recording is also done by reading sensor data from bag files.

Figure 3.17: TIAGo robot main components

Figure 3.18: TIAGo's software

b) Experiment setup

The experimentation on TIAGo is to generate necessary data for the analysis step of dynamic identification later. In this experiment setup, the torso's lift joint and 7 joints of 7 DOF arm

will be controlled and tracking a pre-designed trajectory. Other joints such as the mobile base and the head are not of interest to this experimentation, hence there are kept fixed during the experiments.

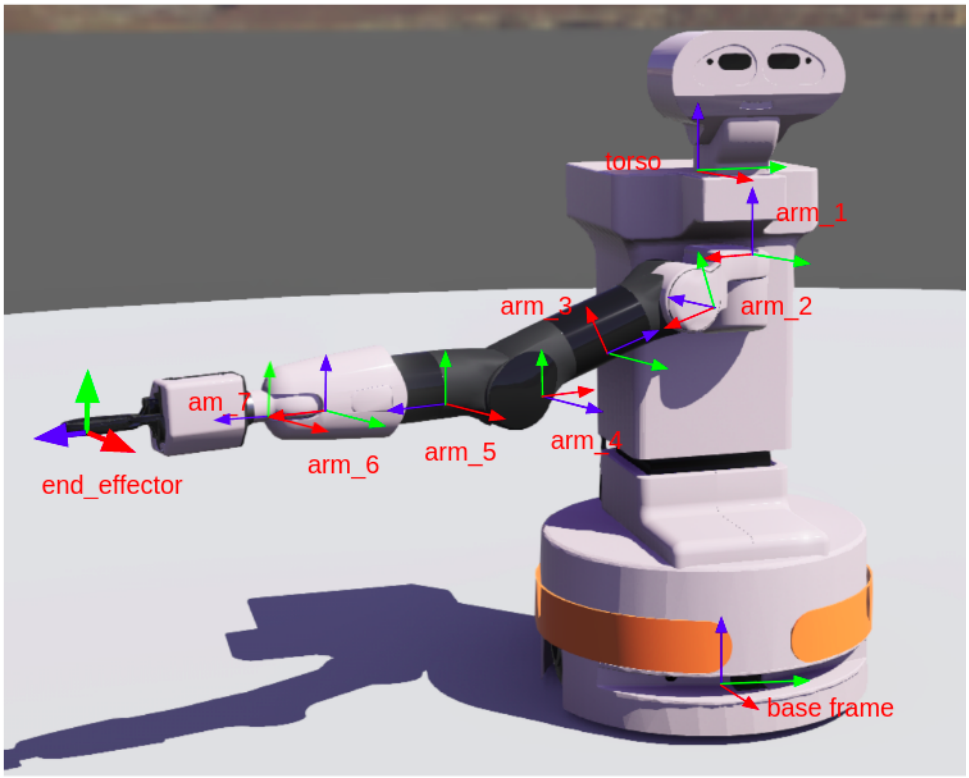

Figure 3.19: Kinematics tree structure of TIAGo Steel robot

The detailed steps of the experiments are presented as following:

- Robot startup: robot is started up with the initial position of the links near the starting position of the trajectory. The low-level controller will make an additional motion to move the links to the exact starting position later.
- Reference trajectory load-up: The reference points of the trajectory is saved in text file with a timeline at a frequency of 50 HZ which is in range of the controller's frequency.
- Trajectory tracking: The low-level controller will track the loaded trajectory.
- Data recording: The joint data such as positions, velocities and efforts are record in the bag files, a data structure of ROS system which allows to replay the motion and inspect the data in details.
- Validating obtained data: Finally, the obtained data will be plotted to spot abnormal outliers or failures of tracking or saturation. If none of these issues happens, the obtained data is validated to proceed to the analysis step.

3.3.2 Methodology

In this methodology sub-section, similar to steps done with the robot TX40 above, first dynamic model of the robot will be derived, then an identification model of interested links will be determined. After that, the methods of creating trajectories that optimizes certain criterion will be shown. Collected data from experimets tracking these trajectories will preprocessed. Finally, the pre-processed data will be analyzed by identification algorithms. Afterwards, the estimated parameters for the analysis process will be validated.

a) Dynamic model

First of all, a robot object will be instantized by calling:

tiago_robot = Robot(tiago.urdf)

model = tiago_robot.model and *data = tiago_robot.data*

Similar to implementation on TX40, the dynamic model can be built by calling the function:

W = build_full_regressor(N, model, data, q, v, a)

The matrix W is the observation matrix containing full dynamic modeling of the TIAGo robot with inclusion of friction, offset, and joint couplings.

b) Identification model and identification algorithm

Thanks to the generality of the function, an identification model and the values of base parameters can be obtained for the robot TIAGo by calling the same functions as following:

- To eliminate non-effect parameters: *W_e, params_r = eliminate_non_dynaffect(robot,W)*
- To get the base parameters expression: *index_base = get_index_base(W, params_r)*
- The regressor matrix of base parameters is: *W_b = build_base_regressor(W_e, index_base)*
- Finally, the identified results can be obtained by: *params_b = double_QR(W_b, τ, params_r)*

c) Data preprocessing

Although, data preprocessing is normally customized for a specific series of robots. However, the basic filtering and preprocessing steps used with TX40 in the above section could be considered as a general pipeline. Applying the same processes to TIAGo, we would obtain "clean" and reliable data ready for analysis. The figure 3.20 is showing an example of acceleration estimates before and after filtering and preprocessing.

d) Optimal trajectories generation

The main difference, also the advance of the dynamic identification procedure for TIAGo comparing to the one for TX40 is to generate optimal exciting trajectories. In this implementation, the trajectory profile chosen is the cubic spline profile. The optimizing criteria is the condition number of the base regressor. The decision variables are the control points of

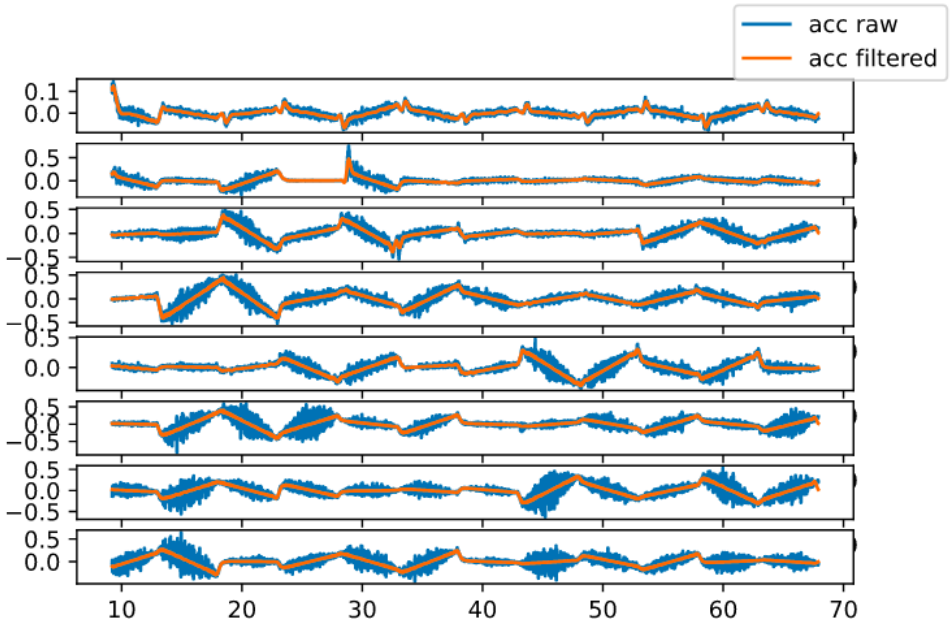

Figure 3.20: Accelerations of the lift joint and 7 arm joints before and after filtering

cubic spline curve. Constraints of joint position limits, velocity limits and joint effort limits are imposed. More over, the self collision is also taken into account to the consideration of constraints. With the information above, we could formulate an optimization problem for generating optimal exciting trajectory.

Stacking technique

An optimizing technique is adopted from literature by V. Bonnet [11]. Considering the fact that a cubic spline CB is a series of cubic polynomials $\{cb_1, cb_2, ...cb_k, ...cb_n\}$ which share the end-points, we could create an optimal exciting cubic spline trajectory by consecutively impose n optimization problems on n segments of the cubic spline.

- 1^{st} repetition: we would have obtained a sub-regressor W_{b1} which has minimum condition number corresponding to first segment cb_1. At this point, our main base regressor W_B is equivalent to the first sub-regressor W_{b1}.

- 2^{nd} repetition: we would generate the second cubic polynomial segment cb_2 which would produce a sub-regressor W_{b2}. We would stack the sub-regressor W_{b2} to our main base regressor W_B, then solves the optimization problem for the minimum condition number of the current W_B with only searching variables of the second segment.

- k^{th} repetition: we would repeat the procedure above. At the step, we would obtain a new main base repressor W_B stacked by a sub-regressor W_{bk}. We repeat this process until the n^{th} time which signals the deviation of optimized condition number is relatively small compared to its previous repetition.

This technique would help to reduce significant amount computing time for a contrained nonlinear optimization problem.

Algorithm 1 "Stacking" regressor optimization

Input : robot model
$\epsilon \leftarrow 0.05$
$CB \leftarrow [\,]$
$W_B \leftarrow [\,]$
while $\Delta \geq \epsilon$ **do**
  $var \leftarrow cubic\_spline()$ ▷ var is control points of an individual cubic polynomial segnent.
  $cb \leftarrow NLP.solve(var)$ ▷ NLP is an optimization solver object.
  if *cb is not constraint-violated* **then**
    $CB \leftarrow stack(CB, cb)$
    $W_B \leftarrow stack(W_B, W_b)$
    $\Delta \leftarrow \frac{|cond(W_B) - cond\_prev|}{cond(W_B)}$
    $cond\_prev \leftarrow cond(W_B)$
  else
    $CB \leftarrow CB$
    $W_B \leftarrow W_B$
  end
end
Output: A optimal exciting cubic spline CB

3.3.3 Result and discussion

a) Optimal trajectories

The figure 3.21 has shown the evolution of the condition number of main base regressor. We can observe that it reduces over the optimizing iteration counts and stacking repeats.

Generating optimal exciting trajectories has been remaining one of the most challenging obstacles in dynamic identification. The problem with optimization approach lies with its large size, non-linearity and highly constrained boundaries. Therefore, it has not only made the problem difficult to solve but also unsolvable in some cases.

Stacking technique has made one significant progress in solving this problem. It has divided the optimization into sub-problem and solving one by one. The quickly descending change of condition number over repetition can be interpreted that in the first segments, the solver has tried to optimizing on parameters with large magnitudes. As it proceeded forward, the solver shifted its focus on the smaller magnitude parameters.

b) Comparison of reconstructed measured values of joint torques

Based on the results we obtained by reconstructing joint efforts from identified base parameters and standard reference parameters, we could make some observations as follow:

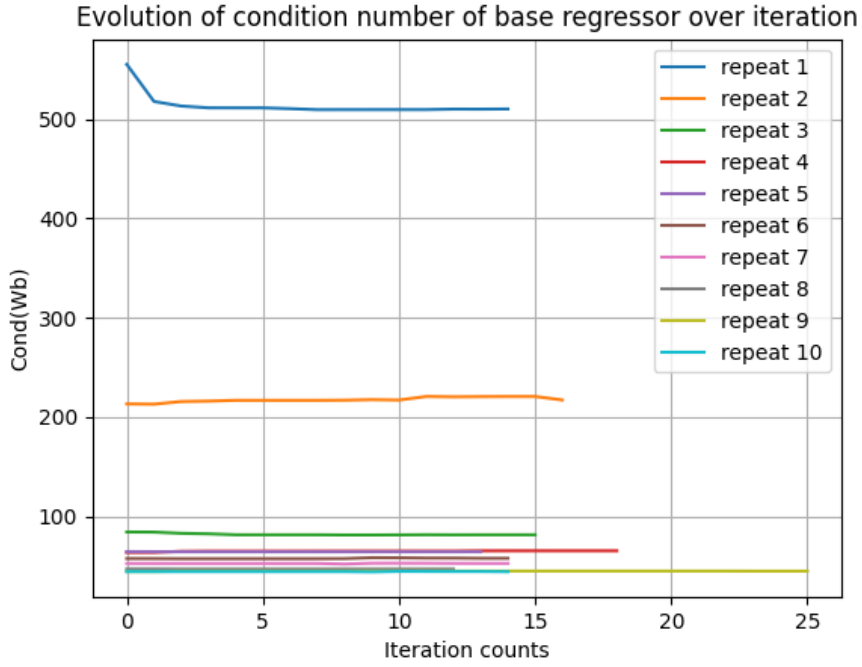

Figure 3.21: Optimizing condition number using stacking technique

- For the lift joint, both joint efforts constructed by identified parameters and reference parameters provided similar "shape" to measured values, but clearly the reference one has missed out an significant offset which identified model has been able to addressed. The interpretable reason could be that the lift joint has not been well-calibrated.

- For arm joint 1, 2, 3, 4, the joint efforts by standard reference parameters are very questionable as they deviated largely from measured values. Meanwhile, the constructed joint efforts by identified parameters have delivered much more closely estimated values.

- For arm joints 5, 6, 7 which correspond to the wrist, both reconstructed values did not succeed resembling the measured values. Identifying these three joints have remained a challenging problem. Firstly, it needs to address the hardware difficulty with joint couplings between joint 6 and joint 7. Joint coupling affected only the joint effort data but also the joint position and velocities. These effects changed the estimated final results. Secondly, end-effector's light weight and its position at the end of a long arm made it prone to jerky vibration during the experiment which brought in unexpected noise and disturbance to the data. Finally, inaccurate drive gain could be a factor that magnifies the deviation in the final estimated parameters.

Due to the time limit of this internship, the fore mentioned problems remain to be resolved in a further study.

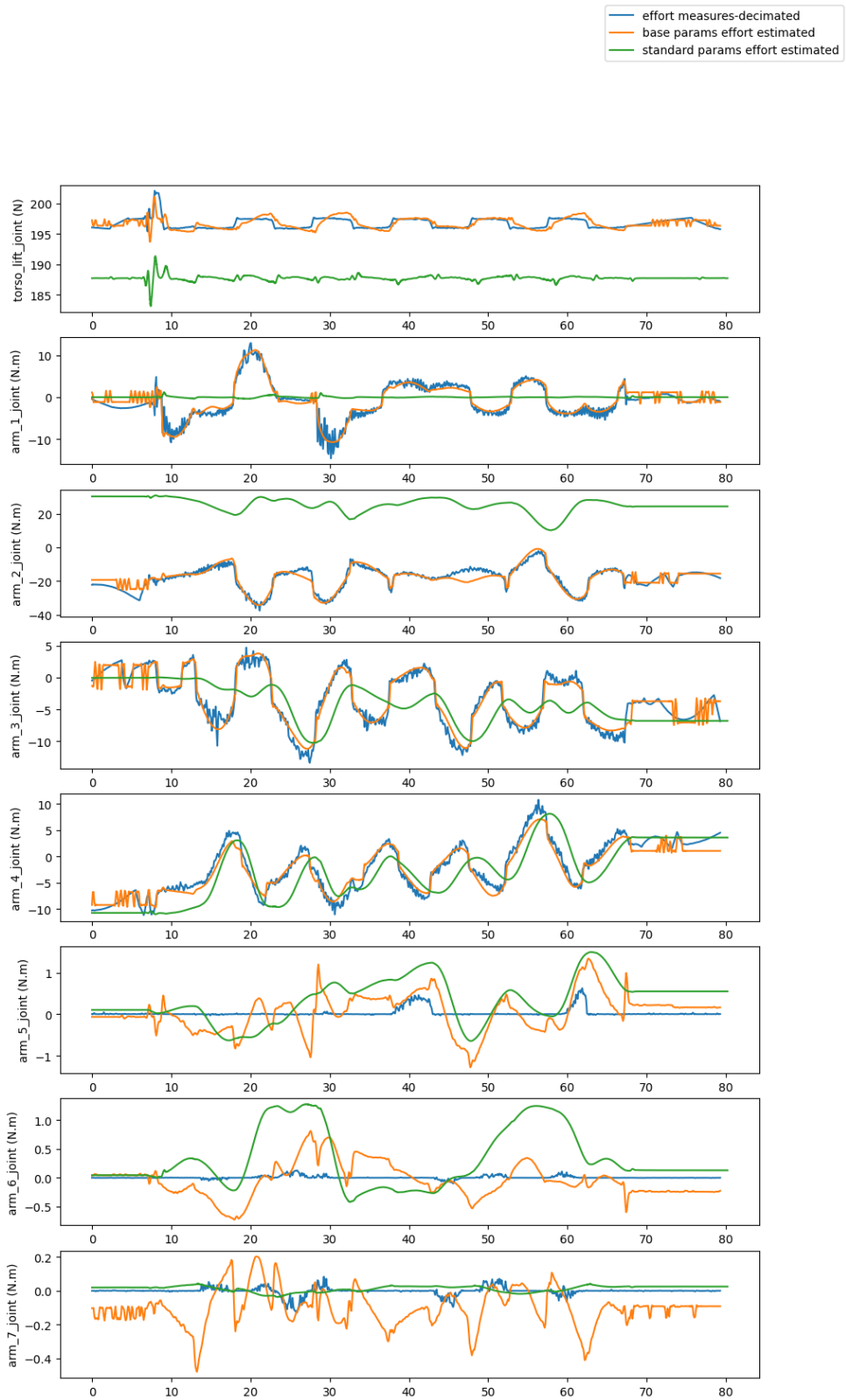

Figure 3.22: Comparison of re-constructed joint efforts by reference parameters and identified parameters with measured values

Conclusion

In this thesis report, the development of a standard open-source dynamic identification was presented. Firstly, an overview on the state of the art in dynamic identification was introduced in the first chapter. The theory background was given in details along with major approaches by researchers in recent years in dynamic identification. This chapter lays the foundation of the toolbox. Main features and functions of the toolbox were explained chapter 3. The toolbox was built on top of the Rigid Body Dynamic library, Pinocchio and utilized popular computing libraries. In order to verify the functionalities of the toolbox, a full description of identification procedures on simulation and on actual robots was presented in chapter 4. Even building a toolbox based on theory mostly means translating conceptual ideas to practical tools, a pipeline of how step-by-step these tools should be used is also very important. Therefore, the chapter of implementation plays a critical role in this report. The development of the toolbox was able to deliver main functionalities for dynamic identification in robotics. However, future works remains open to expand the toolbox to geometric calibration, real-time identification as well as to optimize the performance of the toolbox with best practices.

Appendix A

Additional tables

	Base parameters	Identified results by toolbox				Provided results	
		OLS estimate		WLS estimate		WLS estimate	
		Values	Std. Deviation(%)	Values	Std. Deviation(%)	Values	Std. Deviation(%)
1	zz1R = zz1 + ia1 + yy2 + yy3 + 0.07*mz3 + 0.05185*m3 + 0.05185*m4 + 0.05185*m5 + 0.05185*m6	1.244255	0.95	1.242096	1.07	1.24	1.1115
2	fv1	8.101884	0.49	8.078526	0.66	8.05	0.6769
3	fs1	6.996061	1.65	7.053945	2.21	7.14	2.2356
4	off1	0.437283	14.19	0.428999	19.91	0.392	21.8748
5	xx2R = xx2 - yy2 - 0.050625*m3 - 0.050625*m4 - 0.050625*m5 - 0.050625*m6	-0.463077	2.12	-0.464318	2.56	-0.466	2.5875
6	xy2	0.003468	133.97	0.002746	221.92	0.00358	171.7591
7	xz2R = xz2 - 0.225*mz3 - 0.007875*m3 - 0.007875*m4 - 0.007875*m5 - 0.007875*m6	-0.1586	3.23	-0.160258	4.09	-0.16	4.1141
8	yz2	0.004893	78.19	0.004758	103.13	0.00331	149.3633
9	zz2R = zz2 + ia2 + 0.050625*m3 + 0.050625*m4 + 0.050625*m5 + 0.050625*m6	1.082211	0.82	1.079656	1.01	1.08	1.0119
10	mx2R = mx2 + 0.225*m3 + 0.225*m4 + 0.225*m5 + 0.225*m6	2.21705	1.87	2.219835	2.43	2.22	2.4256
11	my2	0.137491	26.23	0.146245	31.53	0.136	32.9494
12	fv2	5.550123	0.81	5.539886	1.05	5.53	1.0416
13	fs2	8.204463	1.3	8.225992	1.71	8.26	1.6916
14	off2	1.346143	36.64	1.405915	43.57	1.37	44.5933
15	xx3R = xx3 - yy3 + yy4 + 0.45*mz4 + 0.050625*m4 + 0.050625*m5 + 0.050625*m6	0.128705	7.57	0.122164	8.99	0.127	8.9567
16	xy3	0.004355	122.74	-0.000168	3645.34	0.000639	990.9884
17	xz3	-0.001705	357.29	-0.001314	504.63	-0.00372	186.9976
18	yz3	0.008569	42.6	0.006997	56.19	0.00599	68.6018
19	zz3R = zz3 + yy4 + 0.45*mz4 + 0.050625*m4 + 0.050625*m5 + 0.050625*m6	0.107067	7.13	0.106102	7.97	0.107	8.1704
20	mx3	0.073643	30.83	0.080582	30	0.0786	31.9877
21	my3R = my3 - mz4 - 0.225*m4 - 0.225*m5 - 0.225*m6	-0.593092	1.98	-0.602621	2.02	-0.601	2.1079
22	ia3	0.096196	6.69	0.09881	6.91	0.0988	7.2114
23	fv3	1.954614	1.64	1.960738	1.54	1.97	1.6612
24	fs3	6.387475	1.66	6.35754	1.57	6.34	1.7063
25	off3	0.335282	56.68	0.343723	57.95	0.326	63.2178
26	xx4R = xx4 - yy4 + yy5	0.005178	89.74	0.004109	112.65	0.00464	103.6509
27	xy4	-0.002743	76.91	-0.006808	28.97	-0.00687	30.0017
28	xz4	-0.004828	52.61	-0.000568	397.24	-0.00178	132.0915
29	yz4	-0.004899	57.3	-0.005611	40.44	-0.00599	39.5343
30	zz4R = zz4 + yy5	-0.004236	75.94	0.000518	466.98	0.00116	216.3227

Figure A.1: Identified base parameters of TX40 comparison table (1)

31	mx4	-0.026733	21.81	-0.018964	26.12	-0.0219	23.3969
32	my4R = my4 + mz5	0.00031	2373.65	-0.008383	64.16	-0.00726	77.0029
33	ia4	0.033301	10.64	0.031799	6.72	0.0313	7.1472
34	fv4	1.093212	2.92	1.097282	1.36	1.11	1.4061
35	fs4	2.517084	4.66	2.529803	2.15	2.48	2.2869
36	off4	-0.09156	66.1	-0.090049	30.58	-0.102	28.1788
37	xx5R = xx5 - yy5 + yy6	0.007201	45.32	0.002517	86.02	0.00293	76.8254
38	xy5	-0.00099	152.21	0.000383	284.6	0.00055	203.3751
39	xz5	0.001995	80.3	0.00112	101.54	0.000998	117.127
40	yz5	0.003666	40.31	0.000683	167.37	0.000716	163.805
41	zz5R = zz5 + yy6	0.005609	44.7	0.001286	181.25	0.00109	217.0015
42	mx5	0.006812	73.11	-0.001966	236.22	-0.00153	307.6661
43	my5R = my5 - mz6	-0.031986	13.9	-0.032621	11.67	-0.0309	12.5921
44	ia5	0.042922	9.84	0.04731	8.02	0.0468	7.8959
45	fv5	1.848582	2.2	1.846107	1.92	1.86	1.8268
46	fs5	3.044712	3.51	3.054238	3.07	3.03	2.9482
47	off5	0.028416	238.46	-0.010262	570.65	-0.0288	194.4682
48	xx6R = xx6 - yy6	-0.003386	38.45	-0.001105	84.13	-0.00136	71.0549
49	xy6	0.000122	558.53	-0.000109	449.19	-2.00E-06	31015.493
50	xz6	0.001364	69.84	0.002122	30.33	0.00226	29.4666
51	yz6	0.002103	40.76	0.000974	61.29	0.000896	68.8983
52	zz6	0.003256	37.93	0.001607	49.83	0.00185	45.0712
53	mx6	-0.008965	33.44	-0.003624	69.67	-0.00316	83.3765
54	my6	-0.007074	41.52	-0.004121	59.23	-0.00498	51.0892
55	ia6	0.008887	25.09	0.010827	10.96	0.0105	11.8534
56	fv6	0.637031	3.21	0.642761	1.52	0.65	1.5859
57	fs6	0.234177	56.35	0.287204	27.98	0.282	28.7752
58	off6	0.164858	37.97	0.146695	20.39	0.127	24.669
59	iam6	0.010213	17.41	0.009553	12.99	0.00964	12.9987
60	fvm6	0.59	2.68	0.612876	1.57	0.616	1.6093
61	fsm6	2.07401	4.47	1.993034	3.48	1.95	3.4705

Figure A.2: Identified base parameters of TX40 comparison table (2)

		Essential parameters	Thanh (WLS)	Relative deviation (%)	Gautier	Relative deviation (%)
1	zz1R	Izz1 + 1.0*Ia1 + 1.0*Iyy2 + 1.0*Iyy3 + 0.07*mz3 + 0.05185*m3 + 0.05185*m4 + 0.05185*m5 + 0.05185*m6	**1.2778**	0.47	**1.2800**	0.49
2	fv1	fv1	**8.0088**	0.64	**7.9900**	0.67
3	fs1	fs1	**7.2855**	2.04	**7.3600**	2.12
4	xx2R	Ixx2 - 1.0*Iyy2 - 0.050625*m3 - 0.050625*m4 - 0.050625*m5 - 0.050625*m6	**-0.4890**	1.85	**-0.4940**	1.90
5	xz2R	Ixz2 - 0.225*mz3 - 0.007875*m3 - 0.007875*m4 - 0.007875*m5 - 0.007875*m6	**-0.1612**	3.53	**-0.1600**	3.64
6	zz2R	Izz2 + 1.0*Ia2 + 0.050625*m3 + 0.050625*m4 + 0.050625*m5 + 0.050625*m6	**1.0732**	0.54	**1.0800**	0.55
7	mx2R	mx2 + 0.225*m3 + 0.225*m4 + 0.225*m5 + 0.225*m6	**2.1709**	0.50	**2.1700**	0.50
8	fv2	fv2	**5.4690**	1.04	**5.4900**	1.05
9	fs2	fs2	**8.3347**	1.66	**8.3300**	1.69
10	xx3R	Ixx3 - 1.0*Iyy3 + 1.0*Iyy4 + 0.45*mz4 + 0.050625*m4 + 0.050625*m5 + 0.050625*m6	**0.0941**	9.43	**0.0947**	9.99
11	zz3R	Izz3 + 1.0*Iyy4 + 0.45*mz4 + 0.050625*m4 + 0.050625*m5 + 0.050625*m6	**0.1272**	3.86	**0.1300**	4.06
12	my3R	my3 - 1.0*mz4 - 0.225*m4 - 0.225*m5 - 0.225*m6	**-0.5948**	1.46	**-0.5890**	1.57
13	ia3	Ia3	**0.0815**	4.25	**0.0801**	4.82
14	fv3	fv3	**1.9715**	1.49	**1.9600**	1.69
15	fs3	fs3	**6.3660**	1.54	**6.3900**	1.72
16	mx4	mx4			**-0.0171**	17.95
17	ia4	Ia4	**0.0333**	3.68	**0.0342**	3.89
18	fv4	fv4	**1.0964**	1.28	**1.1100**	1.36
19	fs4	fs4	**2.5614**	2.04	**2.4900**	2.26
20	my5R	my5 - 1.0*mz6	**-0.0345**	8.65	**-0.0376**	8.51
21	ia5	Ia5	**0.0476**	6.30	**0.0481**	6.24
22	fv5	fv5	**1.8131**	1.93	**1.8600**	1.81
23	fs5	fs5	**3.1425**	3.01	**3.0100**	2.94
24	ia6	Ia6	**0.0125**	7.11	**0.0123**	7.85
25	fv6	fv6	**0.6618**	1.32	**0.6730**	1.38
26	iam6	Iam6	**0.0107**	10.86	**0.0110**	11.02
27	fvm6	fvm6	**0.6027**	1.46	**0.6090**	1.52
28	fsm6	fsm6	**2.1566**	2.00	**2.0900**	2.12

Figure A.3: Identified essential parameters of TX40 comparison table

Bibliography

[1] W Khalil and E Dombre. "Chapter 12 - Identification of the dynamic parameters". In: *Modeling, Identification and Control of Robots*. Ed. by W Khalil and E Dombre. Oxford: Butterworth-Heinemann, 2002, pp. 291–311. ISBN: 978-1-903996-66-9. DOI: https://doi.org/10.1016/B978-190399666-9/50012-9. URL: http://www.sciencedirect.com/science/article/pii/B9781903996669500129.

[2] O. Stasse et al. "TALOS: A new humanoid research platform targeted for industrial applications". In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. 2017, pp. 689–695. DOI: 10.1109/HUMANOIDS.2017.8246947.

[3] Ko Ayusawa, Gentiane Venture, and Yoshihiko Nakamura. "Identification of humanoid robots dynamics using floating-base motion dynamics". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 2854–2859. DOI: 10.1109/IROS.2008.4650614.

[4] Maxime Gautier and Sébastien Briot. "Global identification of drive gains parameters of robots using a known payload". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 2812–2817. DOI: 10.1109/ICRA.2012.6225099.

[5] V. Bonnet et al. "Optimal Exciting Dance for Identifying Inertial Parameters of an Anthropomorphic Structure". In: *IEEE Transactions on Robotics* 32.4 (2016), pp. 823–836. DOI: 10.1109/TRO.2016.2583062.

[6] M. Gautier and W. Khalil. "Direct calculation of minimum set of inertial parameters of serial robots". In: *IEEE Transactions on Robotics and Automation* 6.3 (1990), pp. 368–373. DOI: 10.1109/70.56655.

[7] M. Gautier. "Numerical calculation of the base inertial parameters of robots". In: *Proceedings., IEEE International Conference on Robotics and Automation*. 1990, 1020–1025 vol.2. DOI: 10.1109/ROBOT.1990.126126.

[8] J. Mayr and H. Gattringer. "Static inertial parameter identification for humanoid robots using a torque-free support". In: *2014 IEEE-RAS International Conference on Humanoid Robots*. 2014, pp. 99–104. DOI: 10.1109/HUMANOIDS.2014.7041344.

[9] K. Ayusawa, Y. Nakamura, and G. Venture. "Optimal estimation of human body segments dynamics using realtime visual feedback". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 1627–1632. DOI: 10.1109/IROS.2009.5354711.

[10] J. Baelemans, P. van Zutven, and H. Nijmeijer. "Model parameter estimation of humanoid robots using static contact force measurements". In: *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2013, pp. 1–6. DOI: 10.1109/SSRR.2013.6719328.

[11] Vincent V. Bonnet et al. "Self-Generation of Optimal Exciting Motions for Identification of a Humanoid Robot". In: *International Journal of Humanoid Robotics* 15.6 (Dec. 2018), p. 1850024. DOI: 10.1142/S021984361850024X. URL: https://hal.archives-ouvertes.fr/hal-02048085.

[12] Gene H Golub and Charles F Van Loan. *Matrix computations. Johns Hopkins studies in the mathematical sciences*.

[13] C. Presse and M. Gautier. "New criteria of exciting trajectories for robot identification". In: *[1993] Proceedings IEEE International Conference on Robotics and Automation*. 1993, 907–912 vol.3. DOI: 10.1109/ROBOT.1993.292259.

[14] V. Bonnet and G. Venture. "Fast Determination of the Planar Body Segment Inertial Parameters Using Affordable Sensors". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 23.4 (2015), pp. 628–635. DOI: 10.1109/TNSRE.2015.2405087.

[15] Luigi Biagiotti and Claudio Melchiorri. "Multipoint Trajectories". In: *Trajectory Planning for Automatic Machines and Robots*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 151–219. ISBN: 978-3-540-85629-0. DOI: 10.1007/978-3-540-85629-0_4. URL: https://doi.org/10.1007/978-3-540-85629-0_4.

[16] M. Gautier. "Dynamic identification of robots with power model". In: *Proceedings of International Conference on Robotics and Automation*. Vol. 3. 1997, 1922–1927 vol.3. DOI: 10.1109/ROBOT.1997.619069.