Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

Singularity detection in the Vereshchagin hybrid dynamics solver

Supriya Vadiraj

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Paul G. Plöger
M.Sc. Sven Schneider

January 2019

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Supriya Vadiraj

Abstract

In the field of robot manipulation, singularities can be defined as state in which robot is incapable to moving its end effector regardless of its movements in joints. The workspace of the robot is bounded by singularities. The analysis, thus forms a very critical step in manipulator design. There can be two situations derived out of occurrence of singularities, one such situation is to seek singularity and the other is to avoid them. This is based on their level of evaluations conducted on forces or motions.

This project extends the existing Popov Vereshchagin hybrid dynamics solver to detect singularities occurring at runtime or to detect if it is close to singularity. The solver only relies on considering the dynamics primitives available in the existing solver to detect singularities. The solver after extension should be able to provide a feedback as a separation of concern between detection mechanism and policy of exploiting the decision.

In this work, a detailed hypothesis has been performed over situations where singularities could possibly occur within the solver. The evaluation has been performed over a KUKA-LWR (7R) manipulator model in simulation to validate the desired results in the case of singular configurations, where they satisfy the required task constraints and their equivalence with the traditional methodology. future work

Acknowledgements

I would like to express my sincere gratitude to Prof. Dr. Paul Plöger for his beneficial feedback and support.

I would like to specially thank M.Sc. Sven Schneider for his continuous support throughout the project. With his patience and immense knowledge in the field has helped me learn a lot in this research. As an advisor, his help is more than I could acknowledge for here. I would also like to thank Djorje Vuckevic for his helpful comments throughout the project.

Finally, I would like to thank my family and friends for their love, guidance and endless support in whatever I pursue.

Contents

List of Figures

List of Tables

List of symbols

M	Inertia matrix that maps between joint space domain and force domain
$\ddot{X}$	Cartesian level acceleration
H	Rigid body inertia
$q, \dot{q}, \ddot{q}$	Joint position, velocity,acceleration vectors
ν	Lagranage multiple(vector of magnitude of constraint forces)
A_N	Matrix of unit constraint forces
b_N	Matrix denotes the vector of acceleration energy for segment N
$C(q, \dot{q})$	Coriolis and centrifugal force vector
d	Joint rotor inertia
P	Control Plant iTASC control scheme
C	Control block iTASC control scheme
$M + E$	Block representing Model update and Estimation in iTASC control scheme
X_u	Geometric disturbances as input to iTASC control scheme
y	System output iTASC control scheme
z	Sensors measurements in iTaSC control scheme
K	Feedback term iTASC
τ_a	Joint space input forces

S	Motion subspace matrix
F^{ext}	External force applied on rigid body
F_{bias}	Bias force acting on rigid body which takes into account Coroilis, centrifugal effects and external forces
d	Joint rotor
P^A	Projection operator, projects inertia and forces for articulated body
X_f	Feature co-ordinates in iTASC control scheme
r	Penalty function which is optimized - iTASC control scheme
$\times, \times^*$	Operators representing spatial cross products
$(.)^T$	Operator representing matrix transpose
$(.)^{-1}$	Operators representing matrix inverse
E	Rotation matrix - spatial coordinates
ω	Angular velocity vector- three components
v	Linear velocity vector- three components
$I(q)$	Joint space inertia matrix
Λ	Operational space matrix
F_c	Constraint force
$E_{present}$	Current acceleration energy generated by bias forces, external forces or joint torques
$E_{required}$	Acceleration energy required to satisfy constraints in solver
$\mathcal{L}$	Constraint coupling matrix

Acronymns

Abbreviation		Full forms
ABA	=	Articulated Body Algorithm
ADAMS	=	Automated Dynamic Analysis of Mechanical Systems
DART	=	Dynamic Animation and Robotics Toolkit
ODE	=	Open Dynamics Engine
KDL	=	Kinematics and Dynamics Library
CRBA	=	Composite Rigid Body Algorithm
RNEA	=	Recursive Newton Euler Algorithm
MujoCo	=	Multi-Joint dynamics with Contact
WBC	=	Whole Body Control
WBOSC	=	Whole Body Operational Space Control
OCP	=	Optimization control problem
DE	=	Differential Equation
OSF	=	Operational space formulation
SVD	=	Singular Value Decomposition
DLS	=	Damped Least Squares
RRQR	=	Rank Revealing QR decomposition
LDL	=	Cholesky Decomposition
HAL	=	Hardware Abstraction Layer
GIK	=	Generalized Inverted Kinematics
SOT	=	Stack of Tasks
RBDL	=	Rigid Body Dynamics Library

1

Introduction

Robot manipulation is an important field of research in robot technology. In the past years, there has been a large and varied growth in the field of robotic systems, concerning its tasks. Within industries around the world, robots perform a variety of manipulation tasks on a daily basis. They lift heavy objects, move with high speed, and repeat complex performances precisely. The field of robot manipulation can be further sub-classified to kinematics and dynamics, motion planning and control and also higher mathematics.

Kinematics studies the possible motions of the robot without taking into consideration the forces that are necessary to produce that motion. There has been substantial research effort in the field of kinematics in-order to develop a general solution to the inverse kinematic problems. The major drawback in dealing with these problems is the situation of arising singular configurations in robots. Intuitively for a standard six joint industrial manipulator, these configurations can be visualized as incapability of achieving motion in a particular direction irrespective of its movements in its joints. The workspace of robots is thus limited by singularities. From the mathematical point of view, kinematic singularity can be made precise in terms of rank loss of Jacobian matrix [16]. This can be interpreted as the change in instantaneous number of degrees of freedom of the robot [16].

The main objective of the research aims at extending the already available Popov Vereschchagin hybrid dynamics solver to detect the singularities occurring at the runtime or to detect if it is close to singularity. Popov Vereschchagin is

a domain specific solver which computes control commands based on constraints imposed on them. The downside of this solver is its inability to determine runtime singularity. The solver only considers the robot's dynamics and therefore is unaware of the concept of kinematic Jacobians. Hence, the "traditional" means of singularity detection do not apply. This instigates that there is necessity to detect singularity that rely on the dynamics primitives available in the existing solver. The aforementioned reasons indicate the prerequisite to extend the solver to solve and detect for dynamic singularities.

The report is structured as follows. Firstly, chapter 2 provides required background and technical soundness in understanding the field of rigid body systems and their dynamics. This is followed by explaining the necessary state of the art. Main focuses are on kinematic singularities, task specifications such as Whole Body Control, Stack of tasks etc. Chapter 3 address the problem statement of the research. Chapter 4 gives a comprehensive overview of the Popov Vereschchagin solver. Followed by chapter 5 and 6 outlines the methodology and the experiments conducted to evalaute the solutions to the problem of detecting singularities. Finally, the conclusion and the future reference to this project is stated in chapter 7.

1.1 Motivation

Depending on the tasks performed by the robot, two situations could be of concern. The first situation is avoiding and the second is to exploit (utilize) singular configurations in the robot.

Considering an example, referring to the case in the human body, where human's posture while in standing (stretched legs) position. Evaluation on the level of forces, this assists by allowing all the forces to be applied on the joint constraints of the legs while reducing the forces applied to the muscles. An important motive behind the problem is to utilize the opportunity of exploiting singularities in the context of tasks and which allows the robot to minimize energy consumption.

There are various reasons for the occurrence of singularities. The importance of singularities from an engineering perspective are mentioned below [4]:

In many cases, robot manipulators are required to perform tasks under contact with objects or environment. The endpoint of the manipulator is in contact with

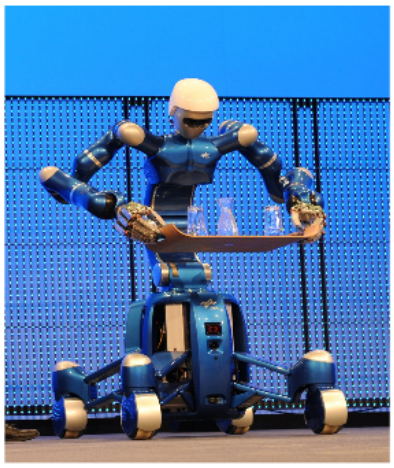

Figure 1.1: Situation depicting robot carrying heavy tray [60]
Case1: Necessity to avoid singular configuration

Figure 1.2: Situation depicting human carrying heavy box [3]
Case2: Necessity to utilize singular configuration

the environment. We have several examples describing the above situation, one of them can be by considering a robot which is required to push heavy loads and is also in contact with the object. A mechanical advantage can be realized for eg: load-bearing capacity [4]. Hence for above scenario, it is best to seek singularity.

A robot's task is described usually in its task space. In order to provide significant application efficiency and flexibility to a robotic operation a direct implementation of a task level controller is performed. However, the existence of singularities is the crucial problem in the application of task level controllers [12]. While approaching singular configuration, the task level controllers may generate high joint torques. The outcome is large errors and in instability in the task space [62]. In general a 6 degrees of freedom (DOF) task, such as welding will require the robot to avoid singular configuration in-order to perform the task. Also near the singular configurations, a large movement of joint variables may result in small motions of the endeffector [17]. In these cases, it is necessary to avoid singularities.

These instances provide an insight into the problem, where we distinguish into mechanism and policy where the mechanism is to detect singularities at the runtime and also the policy is to know how to exploit this knowledge. This provides the best situation in order to avoid or utilize singularity.

The above given examples are extremely dependent on the type of robot task. The question would be how to make use of them in our applications. Here, the focus is on how to ensure that the singularity does not influence the task. Based on the type of the singularity, for eg. internal singularity can be avoided by control. But workspace singularities require the help of mobile base. Therefore with this knowledge, we will be able to derive valuable information about how to handle them [10].

2

State of the Art

This chapter assesses the technical knowledge required in understanding classical mechanics and their related fields such as rigid body dynamics. It also aims at providing state of art relating in task specifications in rigid bodies. Further examination state of art for kinematic singularities and some of the decomposition techniques in linear algebra.

2.1 Rigid body dynamics

Rigid body dynamics can be defined as studying motions of systems in chained bodies to which the external forces are applied [44]. A rigid body system is made of links (rigid bodies) and joints connecting the links together. The joints acts as motion constraints or as a constraint force to the rigid body, allowing motion in particular directions [19].

The field of dynamics studies or focuses its study on accelerations and forces of rigid bodies. The dynamic algorithms analyze the equations of motion of rigid bodies and allow mathematical calculation for variables in concern. As mentioned above, the equation of motion for general rigid body system gives information about dynamics of that body which can be represented as in [19]:

$$I(q)\ddot{q} + C(q, \dot{q}) = \tau \tag{2.1}$$

Where, I represents the inertia matrix in joint space and q is the vector representing position, $\dot{q}$ and $\ddot{q}$ are the vectors representing velocity and acceleration variables

respectively, C represents bias force effects of Coriolis and Centrifugal forces along with gravitational and other forces acting on the system in joint space and finally τ represents the vector of joint forces [19].

In the field of robotics, there are two main dynamics algorithms which are of importance they are forward and inverse dynamics [19].

- *Forward Dynamics* : can be defined as an algorithm which calculates the joint acceleration response when the force/torque is being applied on rigid body [19].

$$\ddot{q} = FD(model, q, \dot{q}, \tau) \tag{2.2}$$

- *Inverse Dynamics* which can be defined as an algorithm which performs the opposite of forward dynamics, i.e. calculates the force/torque which needs to be applied onto the rigid body for the joint acceleration provided [19].

$$\tau = ID(model, q, \dot{q}, \ddot{q}) \tag{2.3}$$

where FD and ID denotes computation of forward dynamics and inverse dynamics respectively. On comparing the above equations to 2.1, it can be seen that FD and ID can be rewritten as $I^{-1}(q)(\tau - C(q, \dot{q}))$ and $I(q)\ddot{q} + C(q, \dot{q})$ respectively [19]. Hybrid dynamics can be defined as combination of forward and inverse dynamics operations in which the interest is in calculating the unknown forces and accelerations when there is information about forces/torques at certain joints and the information about joint accelerations at other joints [19].

The dynamics algorithms can also be used to identify the inertial elements of a rigid body and thus attends the problem of hybrid dynamics as well [19]. There are different types of dynamic algorithms are used for finding solutions for problems related to inverse, forward and hybrid dynamics problems. Some of them are:

1. *Composite Rigid Body Algorithm* algorithm which computes forward dynamics and joint space inertial matrix of a rigid body [20]

2. *Recursive Newton Euler* algorithm and *Lagrangian* formulated algorithm solves for inverse dynamics.

3. *Articulated-Body Hybrid Dynamics* Algorithm (ABA) [19], *Popov-Vereshchagin algorithm* [55] solves problem of hybrid dynamics.

Constraints can be expressed as an algebraic equation depicting the number of allowed motion directions. They can be classified into equality and inequality constraints [19]. They can be further classified into physical and virtual constraints. The physical constraints are the indicated through for example contact with the environment or contact between bodies whereas virtual constraints can be desired motion specified during task specification.[19]. The equation of motion of an non-constrained rigid-body system is represented as in [19]:

$$I\ddot{q} + C = \tau \tag{2.4}$$

The rigid body system is subjected to motion constraints, then the equation of motion can be represented as in [19]

$$I\ddot{q} + C = \tau + \tau_c \tag{2.5}$$

where τ_c indicates the constraint force.

In real world, currently calculation of dynamics has been adopted different fields such as computer gaming, virtual-reality softwares, in certain simulators also used in animation tools and in different engineering tools [19]. Along with the applications there are different softwares which provide the implementations for algorithms of kinematics and dynamics application. Some of them are ODE (Open Dynamics Engine) [57], Project Chrono which is an Open-Source Physics Engine [1] , ADAMS (Automated Dynamic Analysis of Mechanical Systems) is a software for multi body dynamics simulation [2], DART (Dynamic Animation and Robotics Toolkit) [25], MuJoCo physics engine [64] etc. Some of the different softwares made use of are:

- KDL (Kinematics and Dynamics Library): This library is used for modeling and computing kinematic chains of robots. It includes implementations of different class libraries for geometrical objects, and also allow specifying motion [48]. KDL also includes source code available for inverse and hybrid dynamics algorithms such as Popov Vereschagin hybrid dynamics algorithm

and Recursive Newton Euler algorithm.

- RBDL (Rigid Body Dynamics Library): This library contains source code for both forward and inverse dynamics algorithms. They contain Recursive Newton Euler Algorithm (RNEA), Composite Rigid Body Algorithm (CRBA) and Articulated Body Algorithm (ABA) [20].

2.2 Task Specification

The task specification for redundant robots are not merely specifying pose of a single effector. Instead they might involve specifying task for different end effectors and also involve specifying motion associated with the arms, legs, torso, head etc. There have been different task controlled approaches performed on redundant robots [32].

2.2.1 Whole Body Operational

Prior to the whole body operational space control literature (WBOSC) [32] was introduced, the field of robotics focused on operational space formulation (OSF) [31]. The OSF framework applied onto manipulators of rigid body systems control and analyze them by writing control equations at the end effector which account only for forces [31]. The WBOSC is a framework which enables real time control of multiple task primitives related to motion and force of redundant humanoid robots [23].

In-order to assure the safety of the robot and the environment in which it is deployed, a control hierarchy has been described in which they prioritize the most important constraints over non safety constraints [53][32]. They are categorized into three different types of control primitives as in [54]:

- *Constraint handling task*: These primitives (constraints) deal with movements and physical limitations that could harm the robot or the physical environment of the robot. Examples: contacts, avoiding objects/bodies nearby.

- *Operational tasks*: These low level primitives deals with manipulation and locomotion, by controlling different parts of robot's body such as feet, hands.

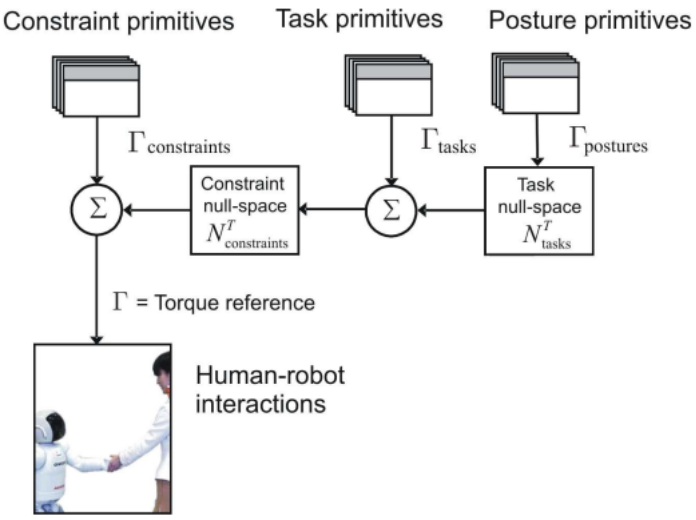

Figure 2.1: Control scheme representing null space projections [54]

- *Posture tasks*: These constraints deal with residual redundancy.

The control hierarchy indicates that the constraints are taken into consideration as primary tasks and they are never violated, while operational tasks (task primitives and posture primitives) are projected onto the constraint motion null-space. In-order to make use of the additional redundancy i.e with the remaining degrees of freedom, posture primitives are projected into both the null-spaces of constraints and the task. Thus indicating that the posture space of the task consists of all the possible motions which do not interfere the performance of the task [54] [32].

This hierarchy is used to prevent the secondary constraints from interfering with the primary constraints in the robots body. The prioritization can be visualized as projecting Jacobian into null space of tasks which are given higher priortization. If the Jacobian gets singular, it indicates the that task which is active is ill conditioned, the movement of the robot is not feasible [54]. As mentioned in [54] the higher priority tasks are maintained while removing the control from the task trajectory.

The complete joint torque is given by the individual torques are control vectors (operational task, posture task and constraint handling). This is mentioned in the control hierarchy as in [54]:

$$\tau = \tau_{constraints} + N^T_{constraints}(\tau_{tasks} + N^T_{tasks}\tau_{posture}) \tag{2.6}$$

In the above equation N^T_{tasks}, $N^T_{constarints}$ are the matrices representing null space.

Inorder to provide control for the control primitives in different levels of hierarchy, OSF is described as the decomposition on the level of torques of the desired task (operational task). The torque can be calculated by force transformation [53]

$$\tau_t = J_t^T F_t \tag{2.7}$$

Similarly, the joint torques for tasks, constraints and postures described in the equation 2.6 can be represented as in [53]:

$$\tau_{tasks} = J^T_{tasks} F_{task} \tag{2.8}$$

$$\tau_{constarints} = J^T_{constraints} F_{constarints} \tag{2.9}$$

$$\tau_{postures} = J^T_{postures} F_{postures} \tag{2.10}$$

$J_{constraints}$, J_{tasks} represent the projections of task space Jacobian onto constraint spaces. $J_{postures}$ represents posture Jacobian which operates in additional redundancy. $F_{constarints}$, F_{task}, $F_{postures}$ represents constraint force vectors which control the respective control primitives [53].

2.2.2 Stack of Tasks

The Stack of Tasks is a control framework which is developed for redundant manipulator control and complex robots like humanoids. The framework supports control based on both kinematic and dynamic descriptions of the robot. Methods based on a hierarchy of tasks (stack of tasks) have motion specification supported for equality and inequality constraints [39].

Authors in [39] indicate that, contrarily to WBOSC framework, the tasks use features which can be position of end effector, center of gravity or mass of the robot [38]. The tasks then can be defined as the difference between desired feature (s^*) and the feature based current state of the robot (s) as in [39]:

$$e = s - s^* \tag{2.11}$$

The above error function depicting the task is regulated to 0. The components of task can be mapped to both equality or inequality constraints and compute the motion by minimizing the error. Each component representing a task can be *bilateral constraints* or set of represented by inequality constraints. Examples of inequality constraints are singularity and collision avoidances, joint limits etc., [41].

The authors Mansard et al., develop their methods which address for task sequencing using Generalized Inverted Kinematics (GIK) [42] and and then they expand it to task description which uses operational space formulation [18, 49, 50, 40, 51]. The difficulty lies in calculating the desired motion when imposed by the unilateral constraints represented as $e < 0$. This is addressed in [41] by various control laws which propose a common activation matrix. The activation matrix is then smoothened at every time step by activating task for each of the inequality constraint values. However, the above approach did not consider the constraints by rigid contacts.

The authors in [50] indicate an extension to incorporate modeling of unilateral rigid contacts which can be processed in a generic hierarchal approach using sequence of quadratic programs (QP) which also resolves the equality and inequality constraints. QP is able to take into account two tasks where the first task is developed for equality and inequality constraints. In each of the hierarchical levels the slack constraints are introduced for every task.

The software framework for controlling redundant robots (SOT) is an open source library with Software Development Kit (SDK) [42]. This framework is efficiently integrable with CORBA (for communication of systems which have been deployed on different platforms).

The framework has been developed to compute the value of the feature at the current time instance. The signals indicate the data flow within the system and can be called as input and output signals. The output signal supplies information onto the component called Entity which computes the desired information. The tasks and features are two different classes of entities. There are specific control parameters such as factories of entities, which loads libraries to create, run and destroy entities and a pool is a collection of entities. Some of the other classes of entities are tasks, entities specific for robot dynamic model, entities for walking etc. SOT entities use feature entity and compute the task error function (control law)

by regulating it 2.11.

2.2.3 iTaSC

The original concept of iTASC has been discussed in [58]. The abbreviated iTASC is called as (Instantaneous Task Specification and Control) [58]. It is a systematic approach which allows to specify constraints on tasks given as relative motions or by allowing dynamic interaction between robot and environment. The constraints can be geometric or dynamic. In general, it can be called as an organized approach to specify constraints for complicated tasks in handling multiple sensor based systems [58].

These complicated tasks could be combination of subtasks, feedback from sensors, substantial variety of robot systems, uncertain environments [63]. This framework can be applied to many other robotic systems, some of them are mobile robots also to numerous robot systems etc. [58]. iTASC has certain important advantages over the traditional motion specification methodologies. Firstly, there is no necessity of constraining the complete 6D system and hence constraints can be partially specified. The constraints can be prioritized and these constraints are in-turn instinctively optimize the robot's motion by solving the constrained optimization problem [47]

In general there are two modeling procedures for the task, one is application independent control and estimation scheme and the other one is systematic application dependent task modeling procedure [63]. The *Control and Estimation scheme* is as indicated in the figure 2.2. The plant P consists of the main robot system and the environment. The controller is denoted as C, the model update and estimation is denoted as M+E. The control input u to is given to the plant, in robots it is given as joint velocities in velocity scheme ($\dot{q}$). The system output is denoted by y, which is given by controller variables. The reference values to output is given as y_d. The measurements z is used to observe the plant and the geometric disturbances is given as X_u [63][58]. The equations for control system in iTASC is mentioned in [14]:

The robot system can be given by the how the changing of joint co-ordinates

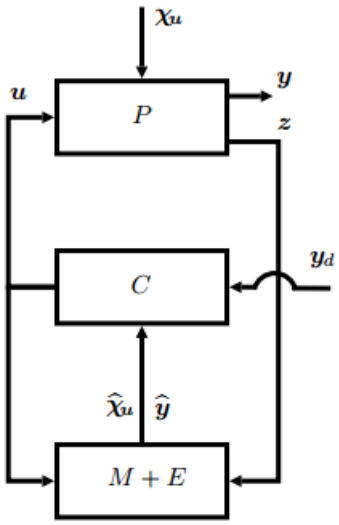

Figure 2.2: iTASC control scheme [58]

i.e the state of the robot system is being affected over control input u.

$$\frac{du}{dt}(q, \dot{q})^T = s(q, \dot{q}, u) \tag{2.12}$$

The output equation can be represented as a function of joint velocities and feature co-ordinates.

$$y = f(q, X_f) \tag{2.13}$$

By differentiating equation 2.13 w.r.t. time the resulting output equation is obtained in velocity level. The system output can be imposed with constraints y_d on y.

$$y = y_d \tag{2.14}$$

Further differentiating equation 2.14 we obtain,

$$\dot{y_d} = \dot{y} + K(y_d - y) \tag{2.15}$$

where the equation consists of feed forward term and K is the feedback term which compensates for various errors.

The loop closure equation gives the relation between joint velocities, feature

co-ordinates and uncertainty co-ordinates.

$$l(q, X_f, X_u) = 0 \tag{2.16}$$

By differentiating the above equation w.r.t. time the resulting loop closure equation is obtained in velocity level.

For task modeling iTASC makes use of task co-ordinates. There are two types of task co-ordinates feature co-ordinates (X_f) and uncertainty co-ordinates (X_u). The task co-ordinates are usually defined in user defined frames. The end result of using the procedure adopted by this framework allows to chose the frames and task co-ordinates such that the task specification becomes very innate [63]. The procedure follows by initially identifying the objects and features and also assigning the reference frames. Later on choosing feature co-ordinates which can be a edges, surface, reference frame etc. And every feature is linked to objects which specify relative motions/force between objects. This can be performed by imposing constraints on feature of one object and then similarly on feature of the other object [58][63].

The authors have extended the framework to support for inequality constraints as well [14][15]. They initially validate the approach on non instantaneous tasks. Thus this approach ensures versatility in specifying robot behavior and eases the task adaptations [15].

Equations for generalizing the objective function of the framework's control equation is given as in [14].

$$u = r(\mu, \gamma) \tag{2.17}$$

where r is denoted as the penalty function for any convex problem defined by user and μ and γ are two objective functions. Even though the convex functions can be solved very robustly, based on the real time requirements the penalty function can be restricted along with the number of constraints [14]. Thus equation 2.17 can be applied for "instantaneous task specification for velocity based control", if there are no geometric uncertainties considered, then in general the equation of optimized problem is given as in [15][14]:

$$u = argmin(r(\mu, \gamma_1, \gamma_2)) \tag{2.18}$$

which is conditioned to,

$$\frac{du}{dt}(q,\dot{q})^T = s(q,\dot{q},u)$$

$$y = f(q, X_f)$$

$$l(q, X_f) = 0$$

$$y = y_d + \mu$$

$$\frac{d^i}{dt^i}y \leq y_{i,max} + \gamma_1$$

$$\frac{d^i}{dt^i}y \geq y_{i,min} + \gamma_2$$

2.2.4 ControlIt! framework

The main objective of *ControlIt!* focuses on general usage of the whole body controller algorithms. It is an open source software framework which addresses the whole body controller algorithms, such that they enable their incorporation onto a substantial system. This framework, can provide its extension to implement on different robot platforms framework, for that reason it requires model description of that robot system URDF, plugin-based modular architecture and also performs addition of new Whole Body Control (WBC) constraints [23]. The authors suggest the need for integrating of the software with external process as there are robot systems like humanoid robots which are highly redundant and contain multi layered software architecture binding the whole body controller. Hence the distributed component software architecture was proposed as they enable the process to run independently as threads [23].

The software architecture of *ControlIt!* acts as the core component in the framework. It is categorized into three groups: configuration software abstractions, whole body control, and hardware abstraction layer. As indicated in the figure 2.3 the compound task (tasks prioritized), robot model (computation of kinematic and dynamic properties) and set of constraints are the components of configuration. The WBC forms the binding co-ordinator/manager which brings in other components jointly and acts as servo-loop. While the Hardware Abstraction Layer (HAL) consists of servoclock and robot interface [23] .

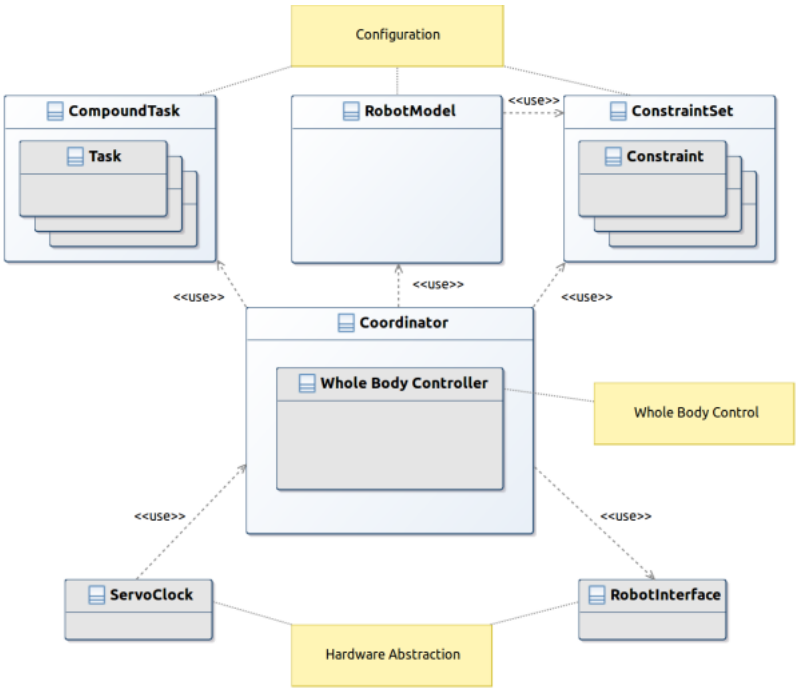

Figure 2.3: Classification of Primary Software Architecture of ControlIt! into three categories [23]

The main principles considered in the development of the software architecture were such that it allowed separation of concerns as mentioned in figure 2.3, platform independent systems (plug-in architecture), reusing the existing software and code, being conscious of considerations during real time and also being able to support software developers who will use this framework and also researchers who can alter this framework [23]. Prior to the implementation of ControlIt, the researchers developed Stanford-WBC [45], which implemented WBOSC algorithms however did not extend implementation for tree structured robots. Later, UTA-WBC was developed to overcome the disadvantages of Stanford-WBC and was tried on humanoid robot. However, this was developed to certain robot and to those robots behavior specific, and could not assist for common usage of applications [23]. The authors make a comparison between UTA-WBC and ControlIt!, the model library which the ControlIt makes use of is RBDL (Rigid Body Dynamics Library). The current integration of ControlIt is with ROS supports Hydro and Indigo versions also can also simulate on Gazebo platform [67].

The authors from [23] suggest some of the drawbacks that lie in the design of the above framework. The current framework supports only WBOSC based plugins which can be solved analytically. However, there is no support afforded to quadratic programming [18] which optimizes inequality constraints. They suggest a future research in identifying if there is possiblity of providing quadratic programming as a plugin to this framework [23].

2.3 Kinematic Singularities

There have been analysis of different methods which analyze and solve the problem of inverse kinematics. Inverse kinematics finds its application in controlling the motions of the rigid body. Inorder to represent inverse kinematics problem , consider a rigid body with links and joints of varied types. Then the joint angles can be represented by a column vector $\theta = (\theta_1, ..., \theta_k)^T$ and the end effector pose can be given by $s = s(\theta)$ indicating they are functions of joint rates. The desired pose for the end effector should reach are indicated by $d = (d_1, ...d_n)^T$ where d_i is the position of i_{th} end effector. Thus the IK will find the joint angles for all i such that

$$d_i = s_i(\theta) \tag{2.19}$$

But unfortunately there does not exist any unique solution to this problem. Therefore the Jacobian matrix was introduced as an iterative solution [10]. This matrix J maps the set of joint angles(θ) of the robot to the end effector pose(s).

$$J(\theta) = \frac{\delta s}{\delta \theta} \tag{2.20}$$

The robot reaches kinematic singularity when there is a failure in attaining some of the configurations (i.e. joint rates) of the kinematic chain. The mapping between joint and Cartesian spaces are not smooth, i.e. a minute change in one of the spaces (giving end effector a samll amount of displacement) will not result in smooth behavioral change in the other space. Mathematically it is indicated by Jacobian matrix being rank deficient as there is a change in the number of motion degrees of freedom [8]

There are mainly two different types of singularity that can occur in the robot they are workspace singularities and internal singularities. The can occur when the

arm of the robots are completely stretched or drawn back and the later one could be due to lining up of joint axes [16]. The behavior of kinematic chain is different when it reaches singularity. There could be practical effects of singularities in planning a trajectory, the velocity of joints reaching infinity, accuracy in achieving the Cartesian motions etc. [8].

The problem of inverse kinematics can be solved by determining the joint parameters of the robot when the desired position and orientation of the end effector is given [10]. Few of the most commonly used methods used to solve inverse kinematics include simple analytical methods, iterative/recursive optimizations techniques such as Jacobian transpose methods [6], along with heuristic procedure which finds the solution faster and performs cheaper computations such as Cyclic Coordinate Descent [36]. The manipulators when in singular configuration can be solved as differential equations, presenting solutions to inverse kinematics [56]. Some of the other approaches which can be used to detect singularities and to solve the problem of inverse kinematics are numerical methods of SVD, damped least square methods [61].

The authors in [10] mention their contribution to the methods which will allow dynamic tracking of end effector to its desired pose which are not reachable. Also the necessity to deal with the unreachable positions arise because the methods such as Jacobian inverse method, pseudo inverse methods will fluctuate greatly. The authors try to elucidate on the mathematical formulations on other methods such as Singular value decompostion(SVD) [13], selectively damped least squares (SDLS) [9] [10]. Another recursive computation method for detecting kinematic singularities in floating space manipulators where the problem is formulated as optimal control theory which is the similar method adopted in the Popov Vereshchagin solver [35].

Several measures have been used in order to design the manipulators and determine their pose in working environment, one of them is *manipulability measure* denoted as $J^T J$. This quantitative measure determines the ease of moving the manipulator in the Cartesian space but does not have any physical meaning [8] [68]. By bringing dynamics into consideration, authors in [68] introduce *dynamic manipulability measure* which introduces an *weighting matrix*. This allows to match the physical units. Dynamic manipulability measure can be defined as the inertia is felt when it moves in Cartesian space [8].

2.4 Different methods for linear algebra

There are different mathematical decomposition techniques which allow the efficient decomposition/factorization of matrix into simpler components which in-turn ease the complicated matrix operations. Matrix decompositions can be applicable to solve systems of linear equations, or could be based on eigenvalues and concepts related to them and some other decompositions in general. There are certain decomposition techniques and their applications which were looked into and contributed to the mathematical part of the research.

2.4.1 Decomposition and rank one updates

- **LU Decomposition** : The main applications of this factorization is in determining the inverse of a matrix and also the determinant of the matrix []and helps in solving the linear equation or finding solution to the simultaneous linear equation. Also the calculation of determinant of a matrix is easier using this decomposition.

- **Cholesky Decomposition** : This is applicable to positive definite matrices and can be represented as $A = LDL^T$ where L is a lower triangular matrix and D represents diagonal elements where there can be non positive entries [29]. This decomposition is also used to solve linear equations $Ax = b$, where A denotes a symmetric and a positive semi definite matrix. The other advantage is that it requires lesser memory storage [29].

- **SVD** : The main use of this decomposition is that to find a good approximation of low rank matrix which is not trivial. It can also be used for obtaining numerical stability i.e. by controlling the condition number which indicates the sensitiveness of the matrix to the variations of input and also rounding errors in the results [59]. The other advantages are solving problems relating to linear least square, it is also used in K means clustering for selecting k values, noise filtration, outliers detection in multivariate data etc. [33]

- **QR Decomposition**: This decomposition is mainly used in finding the eigenvalues of matrix in computer programming languages. This decomposition as

all the above solves linear systems, and also to find approximations for least square problems [33]. A variant of the QR decomposition is RRQR (Rank Revealing QR decomposition) which can be alternative to SVD as it involves detection of rank of a matrix. It is also useful in matrix approximations. Generally RRQR is called as QR decomposition along with column pivoting [30]

- **UTV Decomposition**: This factorization gives a performance with higher efficiency than SVD. They can be easily updated or downdated. They are also rank revealing like SVD, RRQR. They can also be called as two sided orthogonal decomposition. The implementation for rank one update of UTV decomposition is available in matlab [22].

- **VSV Decomposition**: These decomposition techniques are also rank revealing like SVD. There have been algorithms which have worked on semi definte matrices and also indefinite matrices which has been discussed in [37][5]. When compared to UTV decomposition this decomposition makes use of symmetry in the matrix and thus saves the processing time in computers [21].

2.5 Limitations of previous work

write here

3

Problem Formulation

The existing methods for detecting singularity are based on kinematic level. Here, there is no consideration of arm dynamics. Though it can be applied to build theoretical design of manipulator and be able to perform singularity avoidance. It may not be applicable for complicated arm designs and also not applicable for designing arm in detail or for higher speed and also high-speed motion control [68].

All the calculations were previously performed on the Jacobian matrix, however, on another side, different types of hybrid dynamic solvers, such as Popov-Vereshchagin algorithm, are derived from optimization problem, which results in the absence of Jacobian matrix and requires other means of detection. However, detection of singularities at runtime on an algorithmic level can be performed. This can be done by inspecting the matrices related to the robot's dynamics. Since each singularity is associated to rank loss of the matrix involved, simply by calculating the rank of the matrix will help in the detection of singularities. Another possibility to measure the distance from singular configuration is calculating the condition number of the matrix. However, these methods have the following drawbacks:

- **Inefficiency**: To compute the determinant on an $n \times 6$ matrix with a complexity of $\mathrm{O}(n^3)$.

- **Insufficient Reasoning**: To detect at which joint is the singularity first noticed. Also to ensure if the singularity interferes with the task.

- **Constrained task conflict**: It can also be recognized that the assigned tasks cause the occurrence of singularities. The desired motion of end effector is affected by constrained task conflicts [11].
- **Separation of concerns**: There should be a separation of concerns in mechanism and policy, which is not present in the current state of the art. When there are means to detect singular configuration the decision of exploiting this knowledge is made by others.

The procedure involved will also assist in actually measuring the expensiveness to perform tasks in the joint space, i.e cost in terms of energy usage. In non singular configurations the cost is low, as the usage on energy is comparatively less than in singular configurations, because a small motion in the end effector will require high velocities in joint space.

4

Popov-Vereshchagin Hybrid Dynamics Solver

The previous chapter 4.1 gives an overview on modeling and controlling of robot through task based control i.e. by computing the required control commands which resolves the constraints imposed on the robot by task specifications. This takes into consideration the complete dynarimitmic properties of the robot.

In-order to establish task level control for tasks such as robots being in contact with environment, pushing heavy objects etc. the Popov Vereshchagin solver [65, 46, 66] was developed in the early 70's by E.P Popov and A.F.Vereshchagin to compute an instantaneous solution to hybrid dynamics problem (control commands) by resolving the constraints imposed on the end effector. The imposed constraints are Cartesian acceleration constraints, external force constraints from the environment, feedforward joint constraints.

4.1 Task Specification

Based on the motivation mentioned earlier this section gives a comprehensive overview on the task specifications given to the solver. There are different types of task specifications which can be given as input to the solver. The tasks can be specified as *external forces*, *Cartesian accelerations* and as *joint feed-forward torques*. The task specifications and their associated interfaces can be given as:

1. **External force** tasks specifications defined by real or virtual forces F_{ext}.

2. **Joint** task specification which can be given by feed forward torques τ.

3. The **Cartesian acceleration** task specification on the end effector $A_N\ddot{X}_N = b_N$.

The solver has the capability to take task specifications as their input and compute the control commands. The different task specifications are their uses can be noted as:

- **External force** F_{ext} : The task specification through external forces are helpful to achieve tasks with situations where the manipulator in contact with the environment or while performing reconfiguration of robot. They are also useful in tasks involving impedance control.

- **Feed-forward joint torques** τ: The task specification through joint torques is useful for handling situations with joint limit avoidance where these constraints act as virtual spring by pushing away from singularities when the limits are close to singularity, null space motion.

- **The Cartesian acceleration** $A_N\ddot{X}_N = b_N$: The task specification through Cartesian acceleration constraints can also be useful to tasks involving manipulators being in contact with the environments or to perform tasks such as manipulation by specifying accelerations on end effector. These can be specified in the form of physical or virtual constraints. The solver requires the user to specify the directional information of the constrained acceleration and the amount of acceleration energy applied in that particular direction.

  Thus the constraints specified on the end effector segment can be given as: [55]

$$A_N^T\ddot{X}_N = b_N \tag{4.1}$$

  The columns of A_N represent that the directions in which the segment is constrained. The vector b_N signifies the acceleration energy set points (amount of acceleration) in each direction. Depending on the number of constraints (m), the dimensions of matrix A_N and vector b_N will be $6 \times m$ and $m \times 1$ respectively.

The constraints specified by the user, can be complete or partially stated in such a way that the they are less than m number of constraints. For eg: constraining the motion of the end effector/segment in linear z and angular x directions. This means that the robot cannot accelerate in linear z and angular x directions, rest of the directions remain unconstrained. The unit constraint matrix for this case can be defined as [55]:

$$A_N = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad b_N = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In the above case the initial three rows of the unit constrained force matrix (A_N) allow to specify the linear constraints in x, y and z directions and similarly the lower three rows specify angular constraints in the respective directions. The acceleration energy (b_N) has been set to zero which indicates the robot does not produce any acceleration energy in these particular directions.

Considering an additional case indicating complete specification of acceleration constraints. By constraining all the *six* directions of the end effector, an identity matrix of size 6×6 is obtained as in: [55]

$$A_N = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad b_N = \begin{bmatrix} \ddot{X}_N \end{bmatrix}$$

Since A_N is an identity matrix, the magnitudes of two vectors representing Cartesian accelerations ($\ddot{X}_N$) and acceleration energy set point (b_N) both of

dimensions 6×1 are being assigned to each other, in spite of their difference in their physical units. This means that in this scenario, the $A_N \ddot{X}_N = b_N$ where A_N represents unit forces, the magnitudes of acceleration energy and acceleration match up. This signifies that the acceleration energy is produced by constraint forces which is dealt for each of the directional constraints. In the above scenario, every column of unit constrained force matrix indicate a value of 1 which indicates the direction of the constrained force. Similarly the value of Cartesian acceleration is equal to the acceleration energy set point in the respective directions of the constrained forces [67].

By the definition of hybrid dynamics we calculate the unknown forces and accelerations with given forces and accelerations at joints. The solution to the Popov Vereshchagin solver is given by $\ddot{q}$ (joint accelerations), τ_{ctrl} in joints and $\ddot{X}$ (Cartesian accelerations) on links. Since hybrid dynamics solver is a combination of both forward and inverse dynamics solvers. The results $\ddot{q}$ and $\ddot{X}$ are provided as solution to forward dynamics solver and the τ_{ctrl} for inverse dynamics solver.

4.2 Functionality of the solver

In general the term *solver* can be defined as the one which is able to solve a differential equation (equations of Constrained motion), where optimal control problem(OCP) is one of the special cases of differential equations [**?**]. There are special cases of differential equations and OCP's which can be resolved analytically instead of using numerical techniques. Hence, Popov Vereshchagin solver can be categorized one among them. The solvers can be categorized into domain specific solvers and domain independent solvers. The domain specific solvers use optimization techniques and specific to a particular domain. Examples of such solvers are ABA, ACHD, Newton-Euler [55]. The domain independent solvers solve OCP using numerical methods [55]. The Popov Vereshchagin solver is also a domain specific solver. The solver performs recursion over kinematic tree, the search space is reduced by the structure of the kinematic tree.

The main functionality of the solver is that it computes the system's constrained motion. As mentioned in [66] this solver is based on *Gauss's principle of least*

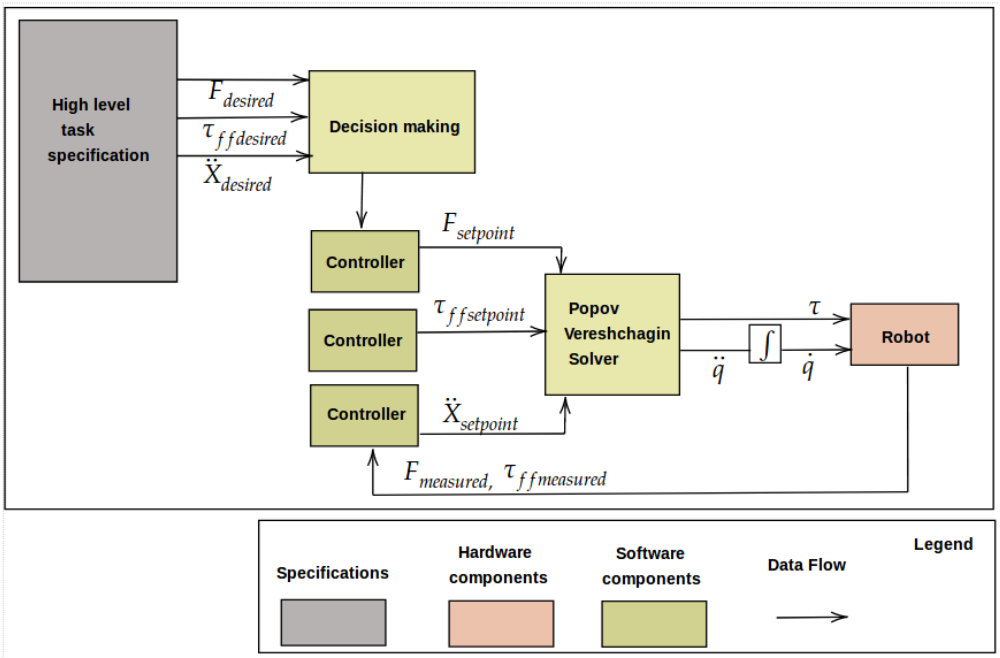

Figure 4.1: Schematic representation of task level specification in Popov Vereshchagin solver

constraint which belongs to the variational principles of classical mechanics. It states that the real motion (constrained motion) of the system is determined by minimum of the convex function which are subjected to linear constraints [66][24][67]. It indicates that in a constrained system inorder to find the true accelerations of masses, the accelerations are initially computed by treating them as unconstrained and then projecting them onto the nearest accelerations which satisfy the constraints [43].

The Gauss function which is minimized is thus given as [66]:

$$\min_{\ddot{q}} Z(\ddot{q}) = \sum_{i=0}^{N} \frac{1}{2} \ddot{X}_i^T H_i \ddot{X}_i + F_{bias,i}^T \ddot{X}_i + \sum_{j=1}^{N} \frac{1}{2} d_j \ddot{q}_j^2 - \tau_j \ddot{q}_j \tag{4.2}$$

where Z is defined as the Zwang (Degree of constraint), H_i is the rigid body inertia which is a symmetric matrix of size 6×6, $\ddot{X}$ represents a 6×1 vector for Cartesian accelerations of particular robot links. $F_{bias,i}$ represents the effects of external forces, Coriolis and centrifugal forces, which is collectively called as bias

force which is a vector of size 6×1 , τ_j the feed-forward joint torques, d represents the inertia of the motor actuating the joint rotor [66].

The Gauss function is subjected to [67]:

$$\ddot{X}_{i+1} = {}^{i+1}X_i\ddot{X}_i + S_i\ddot{q}_{i+1} + \ddot{X}_{bias,i+1}$$

$$A_N^T\ddot{X}_N = b_N$$

In the above equation A_N represents matrix of unit constraint forces. As mentioned earlier they contain the direction of the acceleration constraints. b_N denotes the vector of acceleration energy set point for segment N.

Reformulating the equation 4.2 in-order to account for acceleration constraints on segments as:

$$\min_{\ddot{q},\nu} Z(\ddot{q},\nu) = \sum_{i=0}^{N}\frac{1}{2}\ddot{X}_i^T H_i\ddot{X}_i + \sum_{j=1}^{N}\frac{1}{2}d_j\ddot{q}_j^2 - \tau_j\ddot{q}_j + \nu^T A_N^T\ddot{X}_i \tag{4.3}$$

The above equation uses the method based on Lagrange equations [55][34]. ν represents the Lagrange multiplier (magnitude of constraint forces). Using Bellman's principle of optimality the iterative formulation of equation 4.3 has been converted to recursive formulation [7].

The solution to the Popov Vereshchagin solver is *joint accelerations* ($\ddot{q}$). The solver also finds magnitude of *constraint forces* (ν) and *Cartesian accelerations* ($\ddot{X}_i$).

The Popov Vereshchagin solver has been derived from *Gauss Principle*. The underlying working principle for the Popov Vereshchagin solver can be represented as:

$$F_c = A_N\nu \tag{4.4}$$

$$\nu = \frac{F_c}{A_N} = \frac{E_{present}}{E_{required}}$$

where F_c is a force constraint and A_N is the directional constraint and the ν is Lagrange multiplier which is equivalent to applying $E_{present}$ which is the current acceleration energy which is generated by bias forces, external forces or joint torques and and $E_{required}$ is the acceleration energy which is required to satisfy the

constraints in Popov Vereshchagin solver. Since both the energies can be calculated and ν can be derived as a result of this. Constantly scaling constraint directions will allow computation of F_c.

The input requirements to the solver are the kinematic chain model parameters, joint angles and joint velocities at the current time instant, feedforward joint torques, the external forces applied to individual segments and A_N is the desired constraint (unit force) applied and b_N is the desired acceleration energy set point on the end effector respectively.

The Popov Vereshchagin algorithm is a three pass/sweep algorithm. The algorithm performs two outward and one inward computational sweep on the kinematic chain [55]. The first is the outward sweep that controls the calculation of pose, velocity and bias terms from the root of the robot (base) to the leaves (end effector). Then, the inward sweep is involved in calculation of articulated-body inertias and joint forces. The last sweep is another outward sweep again, which calculates the end result of joint torques, joint accelerations and Cartesian accelerations for each segment [55]. In addition to the above calculations, the solver after the second recursion computes the magnitude of constraint forces (Lagrange multiplier), which is performed when the solver visits the base segment. This hybrid dynamics solver has the run time complexity of $O(n)$ where n is the number of segments, as the solver visits every segment only one time.

Algorithm 1: Popov Vereschchagin Hybrid Dynamics Solver [66, 46, 55, 67]

Input : **Kin Chain model**, $\mathbf{q}$, $\dot{\mathbf{q}}$, τ, $\ddot{\mathbf{X}}_\mathbf{0}$, $\mathbf{F_{ext}}$, $\mathbf{A_N}$, $\mathbf{b_N}$

Output : $\tau_\mathbf{ctrl}$, $\ddot{\mathbf{q}}$, $\ddot{\mathbf{X}}$

begin

  for i = 0 to N − 1 do

    ${}^{i+1}_{i}X = ({}^{d_i}_{i}X \;\; {}^{i+1}_{d_i}X(q_i))$;

    $\dot{X}_{i+1} = {}^{i+1}X_i\dot{X}_i + S_{i+1}\dot{q}_{i+1}$;

    $\ddot{X}_{bias,i+1} = \dot{X}_{i+1} \times S_{i+1}\dot{q}_{i+1}$;

    $F^A_{bias,i+1} = \dot{X}_{i+1} \times^* H_{i+1}\dot{X}_{i+1} - {}^{i+1}X^*_0 F^{ext}_{0,i+1}$;

    $H^A_{i+1} = H_{i+1}$;

  end

  $L_N = 0, U_N = 0$

  for i = N − 1 to 0 do

    $D_{i+1} = d_{i+1} + S^T_{i+1}H^A_{i+1}S_{i+1}$;

    $P_{i+1} = 1 - H^A_{i+1}S_{i+1}D^{-1}_{i+1}S^T_{i+1}$;

    $H^a_{i+1} = P_{i+1}H^A_{i+1}$;

    $H^A_i = H^A_i + {}^iX^*_{i+1}H^a_{i+1}{}^iX_{i+1}$;

    $F^a_{bias,i+1} = P^A_{i+1}F^A_{bias,i+1} + H^A_{i+1}S_{i+1}D^{-1}_{i+1}\tau_{i+1} + H^a_{i+1}\ddot{X}_{bias,i+1}$;

    $F^A_{bias,i} = F^A_{bias,i} + {}^iX^*_{i+1}F^a_{bias,i+1}$;

    $A_i = {}^iX^*_{i+1}P^A_{i+1}A_{i+1}$;

    $U_i = U_{i+1} + A^T_{i+1}\{\ddot{X}_{bias,i+1} + S_iD^{-1}_i(\tau_{i+1} - S^T_i(F^A_{bias,i+1} + H^A_{i+1}\ddot{X}_{bias,i+1}))\}$;

    $\mathcal{L}_i = \mathcal{L}_{i+1} - A^T_{i+1}S_{i+1}D^{-1}_{i+1}S^T_{i+1}A_{i+1}$;

  end

  $\nu = \mathcal{L}^{-1}_0(b_N - A^T_0\ddot{X}_0 - U_0)$;

  for i = 0 to N − 1 do

    $\ddot{q}_{i+1} = D^{-1}_{i+1}\{\tau_{i+1} - S^T_{i+1}(F^A_{bias,i+1} + H^A_{i+1}({}^{i+1}X_i\ddot{X}_i + \ddot{X}_{bias,i+1}) + A_{i+1}\nu)\}$;

    $\ddot{X}_{i+1} = {}^{i+1}X_i\ddot{X}_i + \ddot{q}_{i+1}S_{i+1} + \ddot{X}_{bias,i+1}$;

  end

end

4.3 Comprehensive explanation of the solver

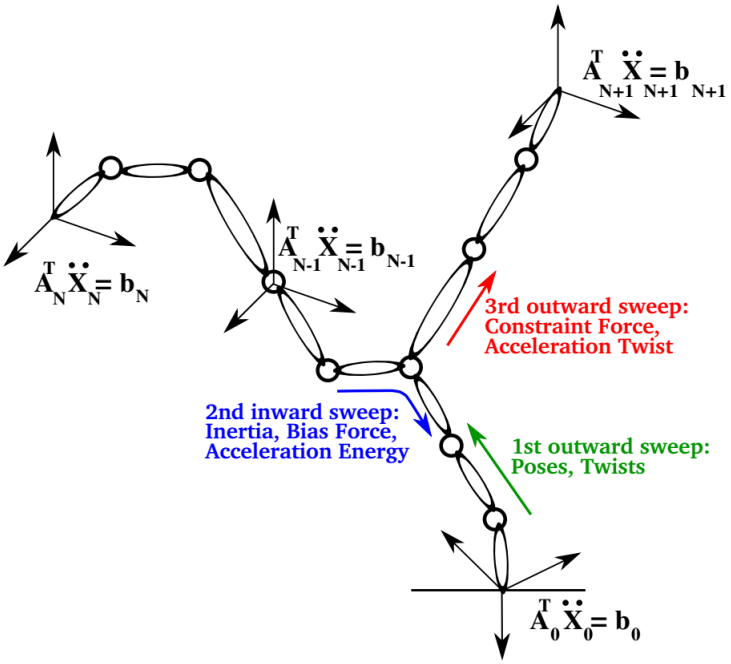

Figure 4.2: Sweeps present in Popov Vereschchagin solver [55]

- **Outward sweep** : The first outward sweep performs recursion of position, velocity and bias acceleration.

  The line 3 of the algorithm performs the calculation of pose relation between the links {i+1} and {i} i.e. calculation of pose on proximal frames of segment {i+1} w.r.t segment {i}. This can be seen in figure 4.3. The pose calculation as mentioned above is given by composing transformations of ${}^{D_i}_{P_i}X$, between proximal and distal frames of segment i and transformations of ${}^{P_{i+1}}_{D_i}X(q_i)$ between proximal frame of link i+1 and distal frames of link i and also shows its dependence of joint position variable q_i [55].

  In order to find the Cartesian velocity as in line 4 i.e. $\dot{X}$ of segment i+1, it requires contribution from the joint velocity twist $\dot{X}_i$ along with joint velocity $\dot{q}_{i+1}$. The expression uses S which is the motion subspace matrix which represents constraining every joints motion to motion subspace [19]. Also

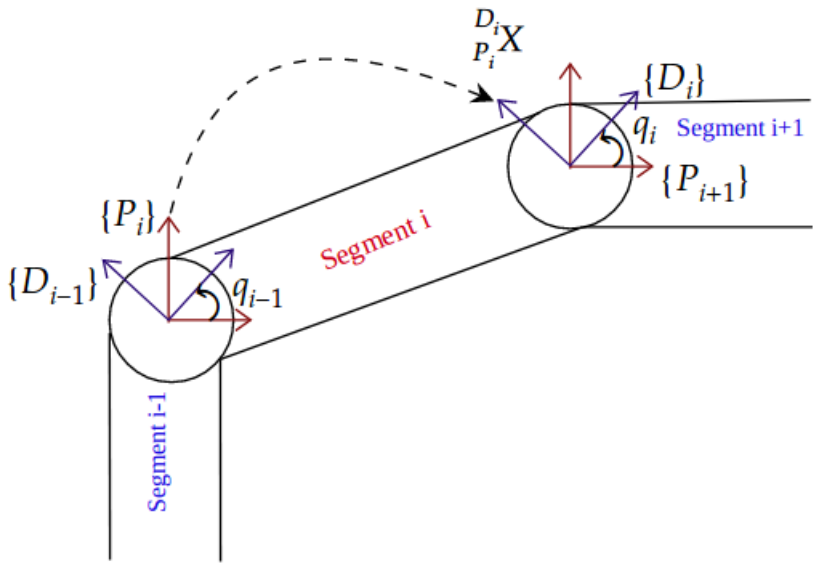

Figure 4.3: Frames and transformations assignment between segments in kinematic chain

the term ${}^{i+1}X_i$ indicates a co-ordinate transformation of motion vector (refer A.2) relating co-ordinates between {i} and {i+1} [55].

Given the velocity twist and joint rate of link {i+1}, the line 5 denotes the calculation the linear and angular components of bias acceleration $\ddot{X}_{bias,i+1}$. The detailed expression uses the property of spatial cross product operator (refer A.3).

Later, in the consecutive step the solver computes bias force vector $F_{bias,i+1}$, which is calculated given spatial velocity twist $\dot{X}_{i+1}$and rigid body inertia H_{i+1}. The external forces are taken into account and the effect of applying these external forces are subtracted from the bias force vector of segment {i+1}. The expression ${}^{i+1}_{0}X^*$ is matrix representation of change of coordinates of force vector between 0 to $i+1$ refer A.2 [67] which means $F^{ext}_{0,i+1}$ are measured in frame {i+1} but is expressed in base frame 0 [55].

The lines 7 and 6 indicate initial initializations of respective rigid body values to *articulated rigid body inertia* matrix and *bias force* vector.

- **Inward Sweep**: During the inward sweep the algorithm deals with calculation of articulated body quantities.

  The line 11 is given as the joint space inertia of segment {i+1} added along with the joint rotor of segment {i+1}. The line 12 defines the *projection matrix* of segment {i+1}, for inertias and forces of the articulated body over the joint axis. The projections are over the joint axis. The next steps (see lines 13, 14) define *articulated body inertia* and *apparent body inertia* which are summed up from all children segments in the kinematic chain. Similarly, summing up bias forces transmitted by children segments as F^a_{bias} (see line 16) contribute to the *articulated body bias forces* [67].

  Due to the specification of end effector constraints the unit forces are felt or experienced on every segment. The line 17 in the algorithm calculates the unit constraint forces which have been propagated and experienced on the segment {i} because of the specification of the end effector constraints. The term A represents a unit constraint force matrix of dimension $6 \times m$ where m is the number of constraints and indicates the direction of the acceleration constraint. However, the magnitude of constraint forces are still unknown in this step.

  The next subsequent line 18 defines the matrix U_i which accumulates the generated (present) acceleration energy which has been generated by external forces, joint torques and contributions from inertial forces over recursive computation from end effector segment to the segment {i}.

  The aforeknown contribution of generated acceleration energy need not additionally contribute to the constraint forces which is to be calculated in the balance equation. The matrix U_N is initialized to 0. The line 19 represents $\mathcal{L}_i$ called the constraint coupling matrix initialized to 0. This keeps monitoring the desired acceleration energy which is required by the unit constraint forces. If the specified constraints on the end effector are not achievable by the kinematic chain i.e if the robot has lesser DOF's than it is required for the task. In such cases the constraint coupling matrix loses its rank. The run time singularity cannot be detected in the current state of the art of the Popov Vereshchagin solver. The basic solution presented by [55] on the solver as a

solution to singularity is weighted least square method. The early detection of singularity has been worked upon and evaluated as in 5.

The core of the algorithm represented by line 21 lies in the computation of the magnitude of constraint forces (Lagrange multiplier ν) balances the between the desired acceleration energy which has already been generated and acceleration energy which is required for the task.

- **Outward Sweep**: The final outward sweep provides the solution to equation 4.3 where joint accelerations and control torques are calculated. The resulting control torque is a combined effect of feed-forward torques, torques due to bias forces, and also torques generated due to constraint forces. This represents line 23 the real/true acceleration of constrained motion. This can be given as in [66]:

$$\ddot{q}_{i+1} = D_{i+1}^{-1}\{\overbrace{\underbrace{\tau_{i+1}}_{\tau_{ff}} - \underbrace{S_{i+1}^T A_{i+1}\nu}_{\tau_{constraint}} - \underbrace{S_{i+1}^T(F_{bias,i+1}^A + H_{i+1}^A({}^{i+1}X_i\ddot{X}_i + \ddot{X}_{bias,i+1})}_{\tau_{bias}}}^{\tau_{ctrl}}\} \tag{4.5}$$

From the above results, finally the Cartesian acceleration is calculated for every segment represented in line 24.

4.4 Summary

The current implementation Popov Vereshchagin algorithm is provided in OROCOS Kinematics and Dynamics library (KDL) [48]. This is an open source library and the computer language is C++ in which the complete library has been implemented.

This algorithm was implemented mainly by Ruben Smits, Herman Bruyninckx, and Azamat Shakhimardanov. For the tasks to be executed safely and to be monitored during the runtime the user should implement software mechanism externally. Inorder to or provide an extension to the solver in the case of singularity, where the constraint coupling matrix $\mathcal{L}$ becomes rank deficient which can be due to many reasons. The KDL library does not provide the user useful information for debugging.

The work by Shakhimardanov on Popov-Vereschagin also provides extension to be able to apply multiple constraints on different or same segments/end effector in tree structured kinematic chains [55]. Due to the above mentioned reasons, this algorithm has been of great interest and benefit to the robotics community.

5

Approach

5.1 Overview

By definition singularity in manipulators can be defined as the state in which the end effector is incapable of moving in particular direction regardless of its movements in joints. They can also be defined on the level of forces as the configuration of manipulator achieved when force applied on the end effector does not produce any torque in the joints. There could be two situations of concern as mentioned earlier (refer section 1.1) depending on the task performed by the robot. In few cases when the tasks are handled on the level of motions i.e., via velocities or accelerations, the necessity is that to avoid such singular configurations. Contrarily, when singularities are evaluated on the level of forces, then there are certain tasks for which we could exploit singularity.

The main advantage of Popov Vereshchagin solver is that the cartesian acceleration, external forces and feedforward torques can be given as input task specifications. There are two main reasons to infer that the kinematic chain reaches singular configuration. The first one is that the kinematic chain when not be able to generate desired task specifications. The other reason is that the task constraints mentioned at end effector are not independent of each other [55]. This means that different task specifications can be specified together and have can be conflicting to each other. The above reasons motivate the approach taken for the detection of singularities.

As mentioned earlier in section 3, the requirement is to find an early and efficient detection for singularities i.e. to be able to detect the singularities earlier in the initial sweeps of the solver and also to find a decomposition technique which is computationally less expensive than the existing one. The current implementation of the Popov Vereshchagin solver lacks to check for the ill conditions of the kinematic chain being in singular configuration. There are certain insights for the approach which have been derived from [55]. The overall approach can be outlined as backtracking the detection mechanism within the solver.

This chapter summarizes the procedure involved in detecting singularities early and efficiently. The solutions proposed are specific to the Popov-Vereshchagin Constrained hybrid dynamics solver. Initial discussion focuses on difficulties in identifying the singularities particular to the solver. The latter part of the chapter enunciates on the mathematical considerations necessary for producing an efficient solution . Then the chapter outlines on the implementation and hypothesis performed on the level of algorithm and finally, the limitations of the methodology.

5.2 Solver specific singularities

This section explains in detecting singularities which rely on the dynamic primitives available within the solver.

The Popov Vereshchagin is a hybrid dynamics solver which is derived from an optimization problem, and does not possess Jacobian matrix as a result, hence the traditional methods do not apply. Solution to the traditional approach can be obtained by building up the manipulator Jacobian for all the joints. Further analyzing on rank of the Jacobian following Singular value decomposition technique. However, this analysis does not contribute any insight on the early or efficient detection process within the solver. As the solver relies on matrices related to robot dynamics. This further initiates the need to explore other approach mechanisms.

Within the solver, there are specific matrices related to robot dynamics which indicate mathematical singularity. This can be seen in the inward sweep of the algorithm (see line 12) and in while determining the magnitude of constraint forces (see line 21) where the matrices are being inverted. These matrices are analyzed to provide further inputs to the detection mechanism. Referring to line 12 where term D is always a positive non zero scalar value indicating the sum

of the joint inertia and joint rotor and thus indicates that the matrix is non singular and always invertible. The next step involves numerical examination of the constraint coupling matrix $\mathcal{L}$ (see line 21) in order to check if the matrix is always invertible. The constraint coupling matrix is a symmetric matrix as are all its prior contributors (refer inward sweep in section 4.3) and is inverted during the calculation of magnitude of constraint forces, which forms the core part to be analyzed.

The mathematical indication of the kinematic chain not being able to achieve the desired task specification can be indicated through $\mathcal{L}$ being singular.

5.3 Singularity analysis at the base

As mentioned in the previous section, the issues are specific to the constraint coupling matrix and needs to be analyzed mathematically. There are naive approaches which can contemplate the detection process on an algorithmic level. The determination of rank of matrices in concern will already provide certain hints to the further problem solving. In the original algorithm, to solve the equations calculating ν at the base (see section 4.3), the SVD over $\mathcal{L}$ matrix is performed. As we are aware that the $\mathcal{L}$ will lose rank when the robot has lesser DOF's than required to achieve task and is not able to generate the desired unit constraint forces. Thus merely by determining if the number of nonzero singular values during the SVD of $\mathcal{L}$ will provide the information on singularity.

5.4 Singularity analysis during the inward sweep

The further steps necessitated in further early detection within the sweeps. By backtracking the inverted constraint coupling matrix to the generation of constraint coupling matrix forms the next step in the process (see line 19). One such procedure, is by performing rank one update over matrix decompositions. However, in this scenario, it is necessary to find a decomposition technique which can satisfy certain properties of matrix $\mathcal{L}$. The recursion of $\mathcal{L}$ matrix begins with a positive semi definite (zero matrix), and every recursive update is symmetric. Thus, the decomposition should be able to handle the symmetricity and positive semi definite matrices in-order for the rank one updates to be possible. The other properties are that the decompositions should be rank revealing. The decomposed matrix should in-turn

help in solving the linear equation at the base. The decomposition should also be computationally less expensive than the existing (SVD).

The table 5.1 provides a comparison of different matrix factorizations and the properties they can handle. The main comparisons are performed over LDL, SVD, QR and RRQR, UTV and VSV.

Table 5.1: Comparison of properties handled by different decompostions

Properties	***Sub-properties***	***Decompositions***						
		LDL	**LU**	**QR**	**SVD**	**RRQR**	**UTV**	**VSV**
Rank one update	**Handle positive semi definite matrix**	✓			✓	✓[30]		✓[28]
	Exploits symmetric shape	✓	✓			✓		✓
Rank revealing			✓	✓	✓	✓	✓	
Computational complexity		$O(n^2)$	$O(nm^2)$		$O(n^2)$		$O(mn^2)$ [27]	
Ability to solve linear equation		✓		✓	✓			✓
Unique decomposition		✓	not always	✓			unknown	unknown

Refer section 2.4 for decomposition techniques and their uses. These properties are essential inorder to provide an efficient detection process. As we can see the LDL decomposition is able to perform rank one update efficiently, satisfying all of the properties listed. The drawback of LU decomposition is that it does not handle positive semi definite matrices. Though RRQR satisfies all the properties, the computational complexity is greater than in the current implementation (SVD). UTV and VSV decompositions were looked into as they satisfied the rank one update conditions, but could not extract much information regarding the computational complexity and numerical stability.

From the above analysis on decomposition techniques, it is seen that the rank one updates through LDL decomposition satisfies all the prior conditions to perfrom an efficient rank one update. Within the inward sweep of the algorithm (see algorithm1 3) line 19 where it indicates that the recursion begins with a zero matrix and for every recursion it adds a rank-1 symmetric update which is rank revealing. Thus the authors in [55] suggest the possibility of rank revealing LDL

decomposition to provide an efficient decomposition which is presented in [52]. If solved using the above mentioned method then on finally reaching the base segment, the magnitude of constraint forces (ν) can easily be determined as the matrix of constraint coupling matrix is in its decomposed form, which remains with solving for the linear equation in the form $Ax = b$.

5.5 Rank one update of LDL decomposition

As mentioned in 5.3 when arriving at the base segment the numerical decomposition procedure should allow solving a linear set of equations [55]. This section gives detailed explanation of the numerical approach involved in solving constraint coupling matrix efficiently.

The LDL decomposition of the positive semi definite constraint coupling matrix gives:

$$\mathcal{L} = \mathcal{L}_L.\mathcal{L}_D.\overline{\mathcal{L}}_L \tag{5.1}$$

where $\mathcal{L}$ is a lower triangular matrix and $\mathcal{L}_D$ is the diagonal matrix. The constraint coupling matrix is updated by a symmetric matrix of rank one to:

$$\overline{\mathcal{L}} = \mathcal{L} + \alpha.zz^T \tag{5.2}$$

where α is a scalar value and z is a $n \times 1$ vector. This equation can be compared to the line 19 and rewritten as :

$$\overline{\mathcal{L}} = \mathcal{L}_{i+1} + (A_{i+1}^T S_{i+1} D_{i+1}^{-1} S_{i+1}^T A_{i+1}) \tag{5.3}$$

On comparing equations 5.2 and 5.3 we see that α is same as $1/D$ as it a scalar value, $A_{i+1}^T S_{i+1}$ can be compared to z and similarly the transpose. In the original algorithm (see line 19) the minus sign indicates the algorithm is computing the reaction force to the constraint forces [8]. The sign change to positive in equation 5.3 indicates the calculation of actual forces to constraint forces. To calculate the reaction forces again, it is possible by inverting the magnitude of constraint forces at the base (see algorithm 2 line 10).

With the given decomposition, exploiting the decomposed factors of $\mathcal{L}$ in-order to obtain $\overline{\mathcal{L}}_L$, $\overline{\mathcal{L}}_D$ and $\overline{\mathcal{L}}_L$ which are efficient rank one update factors. The algorithm

(refer appendix)proposed by [52] is provided as a slight modification to one of the methods in [26] in-order to account for $\alpha > 0$ when $\mathcal{L}$ is positive semi definite.

Algorithm 2: Extension to inward sweep in Popov Vereshchagin for efficient singularity detection

1 **begin**
2 $\quad \mathcal{L}_N = 0$
3 $\quad$ // Inward sweep
4 $\quad$ **for i = N − 1 to 0 do**
5 $\quad\quad$ refer lines 11 - 18 from original algorithm 3
6 $\quad\quad$ $\alpha = \frac{1}{D}$, $w = A_{i+1}^T S_{i+1}$;
7 $\quad\quad$ $\mathcal{L}_i = \mathcal{L}_{i+1} + A_{i+1}^T S_{i+1} D_{i+1}^{-1} S_{i+1}^T A_{i+1}$;
8 $\quad\quad$ $\mathcal{L}_L^{'}, \mathcal{L}_D^{'} = rank\_one\_update(\mathcal{L}_L, \mathcal{L}_D, \alpha, w)$;
9 $\quad$ **end**
10 $\quad$ $A = \mathcal{L}_0, x = -\nu, b = b_N - A_0^T \ddot{X}_0 - U_0$
11 $\quad$ $\mathcal{L} D \mathcal{L}^T x = b$
12 $\quad$ $D\mathcal{L}^T x = forwardsubstitution(\mathcal{L}, b) = y$
13 $\quad$ $\mathcal{L}^T x = D^{-1} y = z$
14 $\quad$ $x = backwardsubstitution(\mathcal{L}^T, z)$
15 **end**

5.6 Algorithmic level extension for singularity detection

This section explains in detail the evaluation on the level of algorithm for detecting singularities.

As mentioned in section 5.3 the initial investigation is performed on constraint coupling matrix $\mathcal{L}$. Also as discussed in sections 5 and 5.3, for a faster and numerically efficient matrix inversion, the algorithm performs rank one symmetric update in every recursion of sweep. This has been depicted on the level of algorithm as an extension to the inward sweep 2. In lines 6, 7 it is shown that the α which is a scalar value is given the value of $1/D$. Further performing LDL decomposition on $\mathcal{L}$ will result in decomposed factors which are updated using rank one update as in 8. Thus at the base the linear equation is easily solvable using forward and

backward substitution.

Based on the criteria of run time complexity we analyze two decomposition techniques LDL and SVD inorder to measure the expensiveness of their operations. Firstly we evaluate the run time behavior of LDL decomposition technique. The rank one update performed through this decomposition will yield with a complexity of $O(n.nc^2)$ i.e. $O(nc^2)$ where nc is the no of constraints and n is the number of segments. This decomposition results in complexity of $O(nc^2)$ at the base segment due to the forward and backward substitutions. Similarly the SVD decomposition which is currently implemented in the solver will perform the operation with a complexity of $O(n.nc^2)$ and $O(n.nc^3)$ at the base. The comparison can be made based on number of segments and number of constraints, if the number of segments is increases then SVD is expected to perform better, contrarily if number of constraints are more then the performance of LDL decomposition is better.

For the real time scenario, SVD decomposition gives better performance. Generally for a kinematic chain upto six number of constraints per segment can be provided. By introducing more and more constraints into the intermediate segments, the SVD operations become more expensive as they are quadratic, and expensive at the base. In terms of performance LDL is better than SVD. But in terms of numerical precision SVD takes the lead.

5.7 Analysis on Projections

On comparing the run time complexities and examining their performance, LDL rank one update does not seem to perform better than SVD in real case scenarios and also does not computationally lessen the operations with lesser number of constraints on segments. Hence, on the level of algorithm, analysis is performed on other dynamic primitives for find an efficient mechanism. The evaluation is performed on the level of composition of multiple projection and transformations (see line 14). The hypothesis is that if there is no joint canceling out the end effector force then it means that joint cannot produce movement in that direction. In other words, if any of the applied end effector forces is experienced at the base then it indicates that it means that the manipulator cannot move in those directions of the applied force. This involved in checking if the projections and transformations are accumulated over the sweeps. If there is accumulation of

projection and transformation values then it signifies that the manipulator cannot move in those directions. See line 14, it indicates the accumulation of projections and transformations. The projection over here is over the joints and the composition sums up the apparent inertias from child segments. Identity projection matrix without full rank. If they are redundant then they do not lose rank.

For efficient detection mechanism, the idea is based on the equation 23 where it indicates the projection of constraint forces over joint space. The input constraint forces in the solver can be specified upto six and they encode the directional information, i.e. it represents the linearly and angularly specified constraints. Lets consider an example, where specification of linear x constraint is provided as input. It is necessary to discover if there have been torques generated due to this force. That indicates that if none of the torques have been generated in joints due to the forces specified on the end effector, then we can conclude that the kinematic chain has achieved singular configuration.

5.8 Implementation

The extension to the current implementation of Popov Vereshchagin solver is given as software proof of concept implementation. This is provided in C++ language. The implementation depends on KDL library [48] mentioned earlier. Referring to figure 4.1 which proposes the original schematic representation of task level specification in Popov Solver. This does not take into account in checking ill conditions for singular configuration. But after providing extension for the above case refer figure 5.1, the singularity detection is added as a feedback from the solver to the decision making. In other words, when the solver reaches singular configuration, it is detected and sent as a feedback to the decision making mechanism, which will further exploit this knowledge. Thus achieving separation of concern.

5.9 Lessons learned

As discussed previously in sections 5.3 and 5.1, the evaluation on rank one updates is performed anticipating the performance at the base is less expensive computationally than the existing one, i.e., performing decomposition directly at the base.

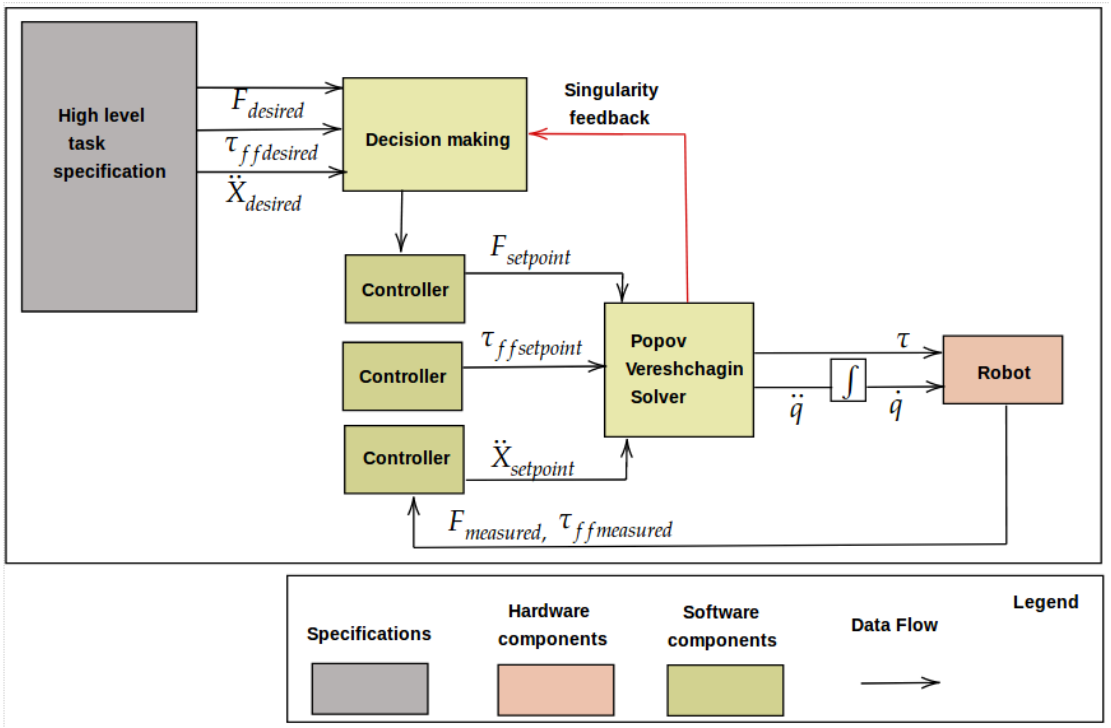

Figure 5.1: Schematic representation of task level specification extension in Popov Vereshchagin solver

The LDL rank one update performed within the inward sweep of solver, will help reduce the computations (see algorithm 2). It indicates that the constraint coupling matrix is obtained in its decomposed form at the base, on further solving for a linear system of equations, it is possible solve for magnitude of constraint forces. This was the initial hypothesis, which was based on also suggestions by [**?**]. Inorder to test this case, the rank one update LDL decomposition was implemented and tested.

The implementation however did not provide any computationally better procedure for calculating the magnitude of constraint forces. This further led to a comparison between the run time complexities to benchmark for future use.

6

Experimental Evaluation & Results

This chapter presents experiments performed to evaluate and analyze on the *proof of concept implementation* given to the extension of Popov Vereshchagin solver. The evaluation gives theoretical and practical explanation to the hypothesis which has been discussed in chapter 5. The chapter begins by discussing over evaluation on singularities and then providing a comparison between run time complexities of decomposition techniques by benchmarking them.

6.1 Experimental setup

All of the tests were conducted in simulation environment. The test setup created to show the equivalence between implemented approach of detecting the singularities within the solver and the traditional (ground truth) singularities. Different scenarios have been evaluated, firstly in-order to find the equivalence between the traditional singularities and the dynamic singularities detected. Later on to test by randomizing input task constraints on the end effector, to check if task constraints are satisfiable in the kinematic chain. The main reason for conducting the experiments is to give a validation for the proposed approach in chapter 5.

For the conducted experiments, the design of the robot model follows the specifications of KUKA LWR. Mass of each segment is 2 kg and the joint specifications and frame assignment follow the model specified in OROCOS KDL. The model is also maintained to not compensate for gravity effects. Also mainly for the experimentation, giving the end effector with different task constraints specifications and

joint specifications, we try to validate the results if they indicate the cases with singularity and their correctness of measurement.

6.2 Test scenarios

In the first scenario, the KUKA-LWR (7R manipulator) is explicitly maintained in different singular configurations . The first configuration of the manipulator is completely extended (see figure 6.1), with zero joint configuration ($q = 0$) for all the joints. The end effector is completely constrained for this case. This indicates that the arm is in singular configuration and cannot achieve all the task constraints as specified in the end effector as some of the joints are restricted. Similarly, the second singular configuration can be seen in figure 6.2 as the manipulator loses few DOF's.

In the second case, the input joint configurations are randomized along with end effector Cartesian acceleration constraints. These cases validated for random joint configurations as seen in figures 6.3 and 6.4. The main intention behind this test case, is to provide validation of the derived approach to check if it is able to detect singularities and able to perform separation of concerns between detection of singularity and providing decision of exploiting this knowledge to others (decision making refer 5.1). This setup is also mainly developed to check if all the task constraints applied are satisfiable and if they are detected within the sweeps.

6.3 Results and Discussions

The results in the first case indicated that certain task constraints are satisfied while some of them cannot. This specifies that the kinematic chain is in singular configuration and some of the joints are restricted. In case of figure 6.1 the algorithm validates that the task constraint can be satisfied linearly in x directions and angularly in y and z directions which complement the physical interpretations. However, to match the findings in Popov Vereshchagin solver to traditional singularity, the rank of the constraint coupling matrix is determined and compared to the rank of Jacobian. The results indicate that they are equivalent. Similarly the result of other configuration in singularity (see figure 6.2) also indicated that all the end effector constraints are not satisfiable and task specification in linear z direction is not achievable.

The results for the second scenario indicated that all the task constraints are achievable and the kinematic chain is not in singularity.

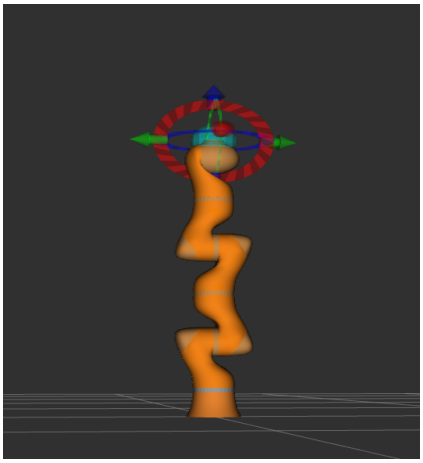

Figure 6.1: Arm in home configuration

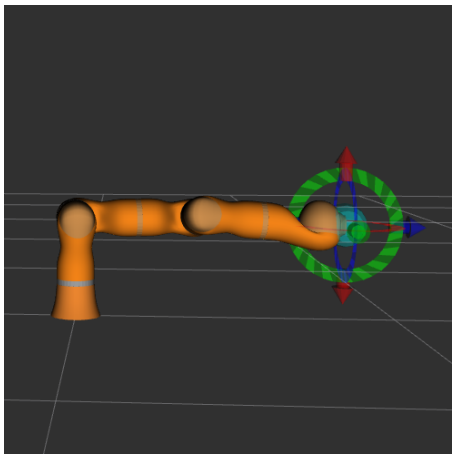

Figure 6.2: Arm in singular configuration

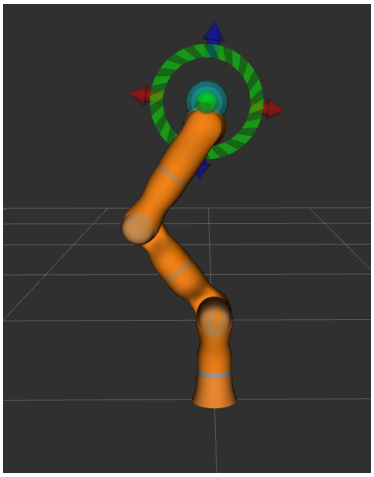

Figure 6.3: Arm in non singular configuration

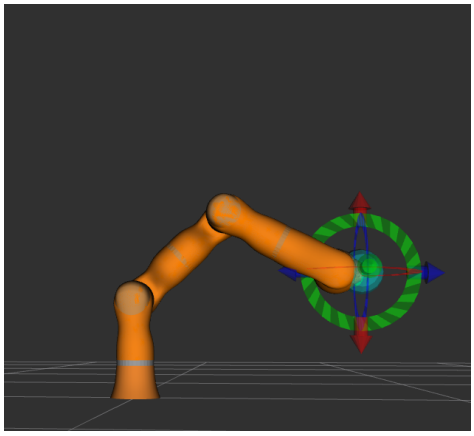

Figure 6.4: Arm in non singular configuration

6.4 Evaluation on exploiting singularities

The below table indicates the evaluation exploiting singular configurations. As discussed earlier, depending on the task performed by robot, there are two

situations which are of concern. In cases of singular configurations, the indication is that the application of arbitrary external force on the end effector, does not produce any torque in the joints for the directions of unsatisfiable task constraints i.e, in directions which the arm is singular.

6.4.1 Experimental Setup

The evaluation is conducted on two robot models KUKALWR4 and PUMA560. For KUKALWR4, the manipulator is maintained in home configuration with zero joint configuration (see fig 6.1). For PUMA560 is also maintained in home configuration, without including shoulder offset. The design of the kinematic chain follows the the joint specifications and frame assignments specified in OROCOS KDL. The models are also maintained to not compensate for gravity effects.

6.4.2 Results

See 6.1 and 6.2 where the input external forces have been applied for all directions in the end effector. The only torques generated due to application of external forces is considered as the output resultant torque. It can be seen that the there is no torque generated at all for certain directions of external forces. This indicates that, in that direction, application of arbitrary external forces results in torques in the joints which does not have any impact of external forces in them.

Table 6.1: Table indicating evaluation on exploiting singularities -PUMA560

External force	Resultant torques	Robot Model
linear x : [1,0,0,0,0,0]	[-0.0064, -0.05079, -0.4318,0,0,0]	PUMA560
linear y : [0,1,0,0,0,0]	[0.4318,0,0,0,0,0]	
linear z : [0,0,1,0,0,0]	[0.0060, 0.4318, 0,0,0,0]	
angular x : [0,0,0,1,0,0]	[0,0,0,0,0,0]	
angular y : [0,0,0,0,1,0]	[0.028, 0.227, 1.935,0,-1,0]	
angular z: [0,0,0,0,0,1]	[0,0,0,0,0,1]	

Table 6.2: Table indicating evaluation on exploiting singularities -KUKALWR4

External force	Resultant torques	Robot Model
linear x : [1,0,0,0,0,0]	[7.9e-05,0.0887,0.00052,0.39,0,0,0]	KUKALWR4
linear y : [0,1,0,0,0,0]	[0,0,0,0,0,0,0]	
linear z : [0,0,1,0,0,0]	[0,0,0,0,0,0,0]	
angular x : [0,0,0,1,0,0]	[0,0,0,0,0,0,0]	
angular y : [0,0,0,0,1,0]	[-0.0003,-0.379,-0.0022,-1.668,0.0026,-1,0]	
angular z: [0,0,0,0,0,1]	[0,0,0,0,0,0,1]	

6.5 Benchmarking on decompositions

As discussed earlier in [55], the authors suggest the decomposition of rank one update as in [52] as a computationally less expensive decomposition than existing SVD.

Measuring and analyzing the algorithm complexity aims at comparing different decomposition techniques and their rank one updates and their run time behaviors. The evaluation on the level of decompositions are necessary to measure the expensiveness of their operations. The efficiency of the algorithm depends on the time it

takes to execute it completely which is based on the number of parameters used as input.

On analyzing the graphs 6.5, this graph performs a comparison of SVD decomposition and rank one update of LDL decomposition. The input parameters are varied for the number of constraints and number of segments. For the real time scenarios, it can be seen that the SVD performs better than the rank one update of LDL decomposition [52]. Similarly by comparing the run times for rank one update LDL decomposition, SVD and also the rank one update. It can be seen that SVD and rank one update of LDL decomposition has a linear relation for number of segments, and cubic relation on the number of constraints. But for rank one update on the LDL decomposition also shows a linear behaviour on number of segments and quadratic behavior on number of constraints.

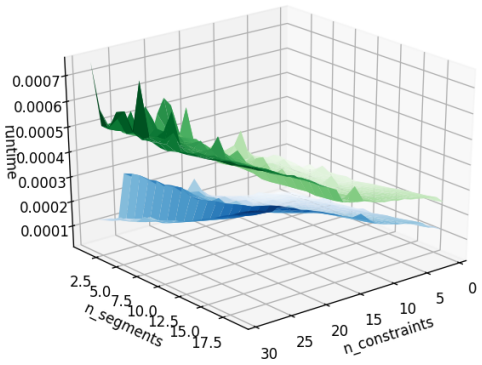

Figure 6.5: Comparison of run time complexity for SVD and rankone update LDL

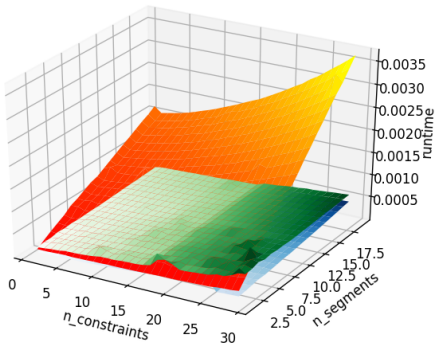

Figure 6.6: Situation depicting human carrying heavy box [3]
Case2: Necessity to utilize singular configuration

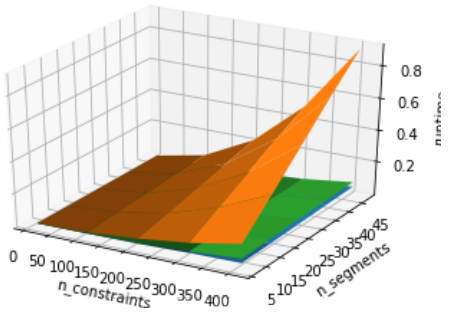

Figure 6.7: Situation depicting robot carrying heavy tray [60]
Case1: Necessity to avoid singular configuration

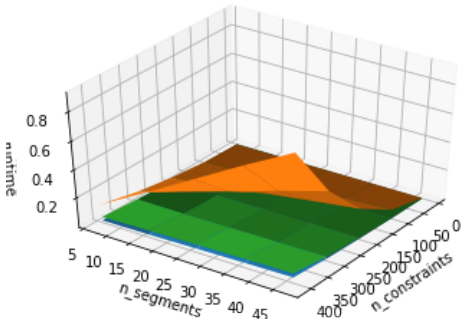

Figure 6.8: Situation depicting human carrying heavy box [3]
Case2: Necessity to utilize singular configuration

7

Conclusions & Future Scope

The existing Popov Vereshchagin hybrid dynamics solver has been extended to support the detection of singularities during the runtime. The approach involved in detecting singularities by evaluating and analysing the kinematic and dynamic state of the existing Popov Vereshchagin solver. Based on the evaluation presented in chapter 5 the solver is now able:

- To detect the singularity early during the runtime.
- To determine the satisfiable task constraints and their equivalence with traditional singularities.

The proposed methodology was tested in simulation with KUKA LWR model defined in OROCOS KDL with different joint configurations and different task constraints. The results have also shown that the knowledge of the detection mechanism can be exploited by others. Hence the separation of concerns has also been achieved as mentioned in chapter 3. Sufficient reasoning has been provided as to where singularity can be noticed early in the sweeps. In order to provide an efficient solution detection mechanism, benchmarking has been performed on the runtime complexities of different decomposition techniques.

The future work aims at extending the solver to analyze more on projections, to find the major contributor to singularity early in the sweeps and also to detect if it is close to singularity. Also, currently work has been performed only on detecting

singularities early and providing feedback to the exploit the decision. But the question still remains as to what is to be done with the decision, either avoid or exploit. The other future goals are to extend the work on different platforms such as mobile bases for detection singularities tree structured in kinematic chains.

A

Appendix A

A.1 Plücker co-ordinates representation for spatial vectors

These co-ordinates represent the 6D spatial vectors, which can be both velocities and forces. This representation involves combination of angular and linear elements [19].

In general cases the plücker co-ordinate representation of a motion vector (M^6) and force vector(F^6) can be given as in [19]:

$$M = \begin{bmatrix} m_{l,x} \\ m_{l,y} \\ m_{l,z} \\ m_{a,x} \\ m_{a,y} \\ m_{a,z} \end{bmatrix} \quad F = \begin{bmatrix} f_{l,x} \\ f_{l,y} \\ f_{l,z} \\ f_{a,x} \\ f_{a,y} \\ f_{a,z} \end{bmatrix}$$

Here the first three elements of both the vectors represent the linear components and then followed by the angular components.
In case of velocities the linear and angular velocities are represented by symbols v and ω which are represented as two 3D vectors. The spatial accelerations are derivatives of velocities which are represented by symbols $\dot{v}$ and $\dot{\omega}$ as linear and

angular accelerations

$$\dot{X} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} \ddot{X} = \begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \\ \dot{w_x} \\ \dot{w_y} \\ \dot{w_z} \end{bmatrix}$$

A.2 Spatial Co-ordinates

This section gives more information on co-ordination transformation of force and motion vectors and their representations.
The co-ordinate transformation of motion vector from A to B co-ordinates can be represented as ${}^{B}X_{A}$. This can be represented as in [19]:

$${}^{B}X_{A} = \begin{bmatrix} E & 0 \\ -Er\times & E \end{bmatrix}$$

The co-ordinate transformation of a force vector from A to B co-ordinates can be represented as ${}^{B}X_{A}^{*}$:

$${}^{B}X_{A}^{*} = \begin{bmatrix} E & -Er\times \\ 0 & E \end{bmatrix}$$

The above matrices represents product of translation by r of 3×1 position (translational) vector and rotation by E which is a matrix with size of 3×3. The operator "$\times$" maps from r to 3×3 matrix. This can be represented as:

$$r\times = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

The co-ordinate transformation matrix thus has the size of 6×6.

A.3 Spatial cross product

The two operators $\times$ and $\times^{*}$ are necessary to represent the cross products for both motion and force vectors separately. They can be represented through the

same relation as mentioned for the spatial co-ordinates. Both the operators act as dual to the other. The 6×6 matrix representation of $\dot{X}\times$ and $\dot{X}\times^*$ is given as in [19]:

$$\dot{X}\times = \begin{bmatrix} \omega \\ v \end{bmatrix} \times = \begin{bmatrix} \omega\times & 0 \\ v\times & \omega\times \end{bmatrix}$$

$$\dot{X}\times^* = \begin{bmatrix} \omega \\ v \end{bmatrix} \times^* = \begin{bmatrix} \omega\times & v\times \\ 0 & \omega\times \end{bmatrix}$$

B

Appendix B

B.1 LDL rank one update

The algorithm which has been used for rank one update of LDL is by Sentana [52]. They present an algorithm for to update symmetric factors of a positive semi definite matrix.

The implementation details of the algorithm is listed below:

Algorithm 3: Rank one update on LDL decomposition [52]

Input : $\alpha^{\mathbf{1}} = \alpha, \omega^{\mathbf{1}} = \mathbf{z},\ \mathbf{j = 0},\ \mathbf{N = 6}$

begin
  for i = 0 to N do
    if $\omega_j^j \neq 0$ **then**
      if $aD_j \neq 0$ **then**
        $p = \omega_j^j$;
        $aD_{new,j} = aD_j + \alpha p_j^2$;
        $\beta_j = p_j \alpha^j / aD_{new,j}$;
        $\alpha^{j+1} = aD_j \alpha^j / aD_{new,j}$;
        for r = j to N do
          $\omega_r^{j+1} = \omega_r^j - p_j aL_{r,j}$;
          $aL_{new,r,j} = aL_{r,j} + \beta_j \omega_r^{j+1}$;
        end
      end
      if $aD_j = 0$ **then**
        $p_j = \omega_j^j$;
        $aD_{new,j} = \alpha p_j^2$;
        $\beta_j = 1/p_j$;
        for r = j to N do
          $aL_{new,r,j} = \beta_j \omega_r^j$;
        end
        for i = j + 1 to N do
          $aD_{new,i} = aD_i$;
          for r = i to N do
            $aL_{new,r,i} = aL_{r,i}$;
          end
        end
        and stop
      end
    end
  end
end

Algorithm 4: continued. Rank one update on LDL decomposition [52]

begin
    else condition
    $aD_{new,j} = aD_j$;
    $\alpha_{j+1} = \alpha_j$;
    for r = j to N do
        $\omega_r^{j+1} = \omega_r^j$;
        $aL_{new,r,j} = aL_{r,j}$;
    end
end

References

[1] *Chrono: An Open Source Framework for the Physics-Based Simulation of Dynamic Systems @ONLINE*. Accessed Dec 29, 2018. URL `http://www.projectchrono.org`.

[2] *ADAMS: The Multibody Dynamics Simulation @ONLINE*. Accessed Dec 29, 2018. URL `http://www.mscsoftware.com/product/adams`.

[3] Man carrying a heavy box [online], Accessed Dec 31, 2018. URL `https://www.google.com/search?q=man+carrying+a+heavy+box&client=ubuntu&hs=4oO&channel=fs&source=lnms&tbm=isch&sa=X&ved=0ahUKEwih88bZwNHfAhUFGewKHY7QCUMQ_AUIDigB&biw=1533&bih=818&dpr=1.2#imgrc=ozFg-RpcWuqCfM`.

[4] Reza Yazdanpanah Abdolmalaki. Geometric jacobians derivation and kinematic singularity analysis for smokie robot manipulator & the barrett wam. *arXiv preprint arXiv:1707.04821*, 2017.

[5] Eugene Scott Baker and Ronald D DeGroat. A correlation-based subspace tracking algorithm. *IEEE transactions on signal processing*, 46(11):3112–3116, 1998.

[6] A Balestrino, G De Maria, and L Sciavicco. Robust control of robotic manipulators. *IFAC Proceedings Volumes*, 17(2):2435–2440, 1984.

[7] Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719, 1952.

[8] H. Bruyninckx. Robot kinematics and dynamics @ONLINE. 2010.

[9] Samuel R Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics tools*, 10(3):37–49, 2005.

[10] Samuel R. Sr Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *University of California, San Diego, Typeset manuscript ...*, 132(4):1–19, 2004. ISSN 0306-4522. doi: 10.1016/j.neuroscience.2005.01.020.

[11] Stefano Chiaverini. Singularity-Robust Task-Priority Redundancy Resolution for Real-Time Kinematic Control of Robot Manipulators. 13(3):398–410, 1997.

[12] Santa Clara, Daniel S Rice, Amith Yamasani, San Jose, Jason B Parks, Santa Clara, Evan Millar, Mountain View, and P C Richardson. (12) United States Patent. 2(12), 2012. ISSN 02277576. doi: 9168268.

[13] L De Lathauwer, B De Moor, J Vandewalle, and Blind Source Separation by Higher-Order. Singular value decomposition. In *Proc. EUSIPCO-94, Edinburgh, Scotland, UK*, volume 1, pages 175–178, 1994.

[14] Wilm Decré, Ruben Smits, Herman Bruyninckx, and Joris De Schutter. Extending itasc to support inequality constraints and non-instantaneous task specification. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 964–971. IEEE, 2009.

[15] Wilm Decré, Herman Bruyninckx, and Joris De Schutter. Extending the itasc constraint-based robot task specification framework to time-independent trajectories and user-configurable task horizons. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1941–1948. IEEE, 2013.

[16] Peter Donelan. Kinematic singularities of robot manipulators. In *Advances in Robot Manipulators*. InTech, 2010.

[17] PS Donelan. Singularities of robot manipulators. In *Singularity Theory: Dedicated to Jean-Paul Brasselet on His 60th Birthday*, pages 189–217. World Scientific, 2007.

[18] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7):1006–1028, 2014.

[19] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

[20] Martin L Felis. Rbdl: an efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, 41(2):495–511, 2017.

[21] Ricardo D Fierro and Per Christian Hansen. Utv expansion pack: Special-purpose rank-revealing algorithms. *Numerical Algorithms*, 40(1):47–66, 2005.

[22] Ricardo D Fierro, Per Christian Hansen, and Peter Søren Kirk Hansen. Utv tools: Matlab templates for rank-revealing utv decompositions. *Numerical Algorithms*, 20(2-3):165, 1999.

[23] Chien-Liang Fok, Gwendolyn Johnson, John D Yamokoski, Aloysius Mok, and Luis Sentis. Controlit!a software framework for whole-body operational space control. *International Journal of Humanoid Robotics*, 13(01):1550040, 2016.

[24] Carl Friedrich Gauß. Über ein neues allgemeines grundgesetz der mechanik. *Journal für die reine und angewandte Mathematik*, 4:232–235, 1829.

[25] Institute of Technology @ONLINE Georgia. Dart physics engine. Accessed Dec 29, 2018. URL `http://dartsim.github.io/`.

[26] PE Gill, GH Golub, W Murray, and MA Saunders. Methods for modifying matrix factorizations. *Milestones in matrix computation: the selected works of Gene H. Golub with commentaries, Oxford University Press, Walton Street, Oxford OX2 6DP, UK*, pages 309–344, 2007.

[27] Ming Gu and Stanley C Eisenstat. Efficient algorithms for computing a strong rank-revealing qr factorization. *SIAM Journal on Scientific Computing*, 17(4): 848–869, 1996.

[28] Per Christian Hansen and Plamen Y Yalamov. Computing symmetric rank-revealing decompositions via triangular factorization. *SIAM Journal on Matrix Analysis and Applications*, 23(2):443–458, 2001.

[29] Nicholas J Higham. Cholesky factorization. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(2):251–254, 2009.

[30] Yoo Pyo Hong and C-T Pan. Rank-revealing factorizations and the singular value decomposition. *Mathematics of Computation*, 58(197):213–232, 1992.

[31] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[32] Oussama Khatib, Luis Sentis, Jaeheung Park, and James Warren. Whole-body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics*, 1(01):29–43, 2004.

[33] Nishith Kumar. *Matrix Decompositions and its application in statistics*. Accessed Dec 30, 2018. URL `https://slideplayer.com/slide/4598334/`.

[34] Joseph Louis Lagrange. *Mécanique analytique*, volume 1. Mallet-Bachelier, 1853.

[35] G Le Vey. Kinematics of free-floating systems through optimal control theory. In *Advances in Robot Kinematics: Analysis and Design*, pages 177–184. Springer, 2008.

[36] David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

[37] Franklin T Luk and Sanzheng Qiao. A symmetric rank-revealing toeplitz matrix decomposition. *Journal of VLSI signal processing systems for signal, image and video technology*, 14(1):19–28, 1996.

[38] N. Mansard and F. Valenza. Pinocchio library [online]. Accessed Dec 29, 2018. URL `https://stack-of-tasks.github.io/documentation/a_introduction.html`.

[39] Nicolas Mansard. A Dedicated Solver for Fast Operational-Space Inverse Dynamics. In *2012 IEEE International Conference on Robotics and Automation*, pages 4943–4949, St Paul, United States, May 2012. IEEE. URL `https://hal.archives-ouvertes.fr/hal-00707200`.

[40] Nicolas Mansard. A dedicated solver for fast operational-space inverse dynamics. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4943–4949. IEEE, 2012.

[41] Nicolas Mansard, Oussama Khatib, and Abderrahmane Kheddar. A unified approach to integrate unilateral constraints in the stack of tasks. *IEEE Transactions on Robotics*, 25(3):670–685, 2009.

[42] Nicolas Mansard, Olivier Stasse, Paul Evrard, and Abderrahmane Kheddar. A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–6. IEEE, 2009.

[43] Gauss's Principle of Least Constraint. Gauss's principle of least constraint [online]. Accessed Dec 31, 2018. URL `http://preetum.nakkiran.org/misc/gauss/`.

[44] Burton Paul. *Kinematics and dynamics of planar machinery*. Prentice-Hall Englewood Cliffs, New Jersey, 1979.

[45] Roland Philippsen, Luis Sentis, and Oussama Khatib. An open source extensible software package to create whole-body compliant skills in personal mobile manipulators. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1036–1041. IEEE, 2011.

[46] Je P Popov, Anatolij Fedorovič Vereščagin, and Stanislav Leonidovič Zenkevič. *Manipuljacionnyje roboty: Dinamika i algoritmy*. Nauka, 1978.

[47] H. Bruyninckx R. Smits and J. D. Schutter. itasc (instantaneous task specification using constraints [online]. Accessed Dec 31, 2018. URL `http://www.orocos.org/wiki/orocos/itasc-wiki`.

[48] H. Bruyninckx R. Smits, E. Aertbelien and A. Shakhimardanov. Kinematics and dynamics library (kdl) @ONLINE. Accessed Dec 29, 2018. URL `http://www.orocos.org/kdl`.

[49] Layale Saab, Nicolas Mansard, François Keith, Jean-Yves Fourquet, and Philippe Souères. Generation of dynamic motion for anthropomorphic systems under prioritized equality and inequality constraints. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1091–1096. IEEE, 2011.

[50] Layale Saab, O Ramos, Nicolas Mansard, Philippe Souères, and Jean-Yves Fourquet. Generic dynamic motion generation with multiple unilateral constraints. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4127–4133. IEEE, 2011.

[51] Layale Saab, Oscar E Ramos, François Keith, Nicolas Mansard, Philippe Soueres, and Jean-Yves Fourquet. Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. *IEEE Transactions on Robotics*, 29(2):346–362, 2013.

[52] Enrique Sentana. Econometric applications of positive rank-one modifications of the symmetric factorization of a positive semi-definite matrix. *Spanish economic review*, 1(1):79–90, 1999.

[53] Luis Sentis and Oussama Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(04):505–518, 2005.

[54] Luis Sentis and Oussama Khatib. A whole-body control framework for humanoids operating in human environments. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2641–2648. IEEE, 2006.

[55] Azamat Shakhimardanov. Composable robot motion stack: Implementing constrained hybrid dynamics using semantic models of kinematic chains. 2015.

[56] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[57] R. Smith. *ODE physics engine @ONLINE*. Accessed Dec 29, 2018. URL `http://ode.org/`.

[58] Ruben Smits, Tinne De Laet, Kasper Claes, Herman Bruyninckx, and Joris De Schutter. itasc: a tool for multi-sensor integration in robot manipulation. In *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pages 426–433. IEEE, 2008.

[59] Harald Sonnberger. Regression diagnostics: Identifying influential data and sources of collinearity. *Journal of Applied Econometrics*, 4(1):97–99, 1989.

[60] Sabina Stodolak. Rollin justin robotsvoice [online].

[61] Tadeusz Szkodny. Avoiding of the kinematic singularities of contemporary industrial robots. In *International Conference on Intelligent Robotics and Applications*, pages 183–194. Springer, 2014.

[62] Jindong Tan, Ning Xi, and Yuechao Wang. A singularity-free motion control algorithm for robot manipulatorsa hybrid system approach. *Automatica*, 40 (7):1239–1245, 2004.

[63] Gianni Borghesan Erwin Aertbeli en Herman Bruyninckx Joris De Schutte Tinne De Laet, Dominick Vanthienen. itasc: a framework for constraint-based task specification from theory to software [online]. Accessed Dec 31, 2018. URL `https://robohow.eu/_media/meetings/delaet.pdf`.

[64] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[65] AF Vereshchagin. Computer simulation of the dynamics of complicated mechanisms of robot-manipulators. *Eng. Cybernet.*, 12:65–70, 1974.

[66] AF Vereshchagin. Modeling and control of motion of manipulational robots. *SOVIET JOURNAL OF COMPUTER AND SYSTEMS SCIENCES*, 27(5): 29–38, 1989.

[67] Djordje Vukcevic. Extending a constrained hybrid dynamics solver for energy-optimal robot motions in the presence of static friction. *Technical Report/Hochschule Bonn-Rhein-Sieg-University of Applied Sciences, Department of Computer Science*, 2018.

[68] Tsuneo Yoshikawa. Dynamic manipulability of robot manipulators. *Transactions of the Society of Instrument and Control Engineers*, 21(9):970–975, 1985.