Passive Compliance Control of Redundant Serial Manipulators

Jacob J. Rice
Marquette University

PASSIVE COMPLIANCE CONTROL OF REDUNDANT SERIAL MANIPULATORS

by

Jacob J. Rice, M.S.

A Thesis submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy

Milwaukee, Wisconsin

May 2019

ABSTRACT

PASSIVE COMPLIANCE CONTROL OF REDUNDANT SERIAL MANIPULATORS

Jacob J. Rice, M.S.

Marquette University, 2019

Current industrial robotic manipulators, and even state of the art robotic manipulators, are slower and less reliable than humans at executing *constrained manipulation tasks*, tasks where motion is constrained in some direction (e.g., opening a door, turning a crank, polishing a surface, or assembling parts). Many constrained manipulation tasks are still performed by people because robots do not have the manipulation ability to reliably interact with a stiff environment, for which even small commanded position error yields very high contact forces in the constrained directions. Contact forces can be regulated using *compliance control*, in which the multi-directional elastic behavior (force-displacement relationship) of the end-effector is controlled along with its position. Some state of the art manipulators can directly control the end-effector's elastic behavior using kinematic redundancy (when the robot has more than the necessary number of joints to realize a desired end-effector position) and using variable stiffness actuators (actuators that adjust the physical joint stiffness in real time). Although redundant manipulators with variable stiffness actuators are capable of tracking a time-varying elastic behavior and position of the end-effector, no prior work addresses how to control the robot actuators to do so. This work frames this passive compliance control problem as a redundant inverse kinematics path planning problem extended to include compliance. The problem is to find a joint manipulation path (a continuous sequence of joint positions and joint compliances) to realize a task manipulation path (a continuous sequence of end-effector positions and compliances). This work resolves the joint manipulation path at two levels of quality: 1) instantaneously optimal and 2) globally optimal. An instantaneously optimal path is generated by integrating the optimal joint velocity (according to an instantaneous cost function) that yields the desired task velocity. A globally optimal path is obtained by deforming an instantaneously generated path into one that minimizes a global cost function (integral of the instantaneous cost function). This work shows the existence of multiple local minima of the global cost function and provides an algorithm for finding the global minimum.

ACKNOWLEDGMENTS

Jacob J. Rice, M.S.

I wish to share a few words of gratitude for the support I received from so many. To all the faculty at Marquette University, I thank you for nurturing my knowledge and skill in various engineering topics. To my labmates, I thank you for the helpful discussions we shared and the feedback you offered. I can honestly say I enjoyed working alongside you all. To my advisor, Dr. Schimmels, I thank you for encouraging me to pursue my Doctorate and for having confidence in me. I thank you for your great assistance in my professional development - especially in my writing. But above all, I thank you for your genuine care for me as an individual; I could not ask for a better advisor. To my wife - words cannot express how grateful I am to have you at my side. You are much more precious to me than my career; I would have set aside my Ph.D. if you needed me to. I am thankful that you, instead, encouraged my Ph.D. all the way through to its completion. Your constant encouragement was the single most helpful thing to me in finishing this work. To my God and Savior, Jesus Christ, thank you for creating me and for saving me by giving your life for my sake on the cross. Whatever intelligence and skill I have is a gift from you. Thank you for guiding me every step in my life; thank you for being a *lamp to my feet and a light to my path* - Psalm 119:105.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Current industrial robotic manipulators, and even state of the art robotic manipulators, are slower and less reliable than humans at executing *constrained manipulation tasks*, tasks where motion is constrained in some direction (e.g., opening a door, turning a crank, polishing a surface, or assembling parts). Many constrained manipulation tasks are dull, difficult, dirty, or dangerous, but are still performed by people because robots do not have the manipulation ability to reliably interact with a stiff environment, for which even small commanded position error yields very high contact forces in the constrained directions. Some form of interaction control is required in order to simultaneously regulate the robot end-effector's motion and its interaction force for reliable and safe execution of the task.

An early method of interaction control is hybrid force/position control [1] in which the task is decoupled into orthogonal subspaces and each direction is either position controlled or force controlled. However, the hybrid force/position approach does not work in general, as tasks cannot be orthogonally decoupled [2]. Impedance control is an interaction control approach which is valid for any physical system; currently, it is the predominant interaction control method. Almost every state of the art interaction controller implements a form of impedance control. A description of impedance control and its implementation is provided below.

1.1 Impedance Control

Impedance control, formalized in [3], is an approach for controlling the interaction behavior between a manipulator and the environment. This control approach is firmly based on modeling principles of physical systems, where the interaction behavior of a physical system depends on its impedance. Impedance is the relationship between force and motion (displacement, velocity, and acceleration). It characterizes how a manipulator accepts motion and returns force to the environment[1].

There are three types of impedance components: 1) *stiffness* which determines the force returned by the manipulator due to displacement imparted by the environment, 2) *damping* which

[1]In *admittance* (inverse of impedance) control, the causality is reversed and the manipulator is seen to return a motion due to force imparted by the environment.

determines the force returned by the manipulator due to velocity imparted by the environment, and 3) *inertia* which determines the force returned by the manipulator due to acceleration imparted by the environment. For manipulation in a coordinate space greater than one dimension, each impedance component is given by a matrix where each element (i,j) describes the force/torque along/about coordinate i from the motion along/about coordinate j.

Impedance control adjusts the manipulator's impedance components (stiffness, damping, inertia) to produce a desirable force-motion relationship appropriate for the task. Impedance control is commonly implemented in one of two coordinate spaces:

1. *Joint Space*: The desired impedance of the manipulator is expressed in terms of the manipulator joints (i.e., the coordinates most convenient for commanding the manipulator motion). Joint impedance control is used for purposes that do not require knowledge of a desirable impedance in task coordinates, such as for collision safety [4] [5] or for mimicking the measured joint impedance of a human demonstrator [6].
2. *Task Space*: The desired impedance of the manipulator's end-effector is expressed in terms of the task coordinates (i.e., the coordinates most convenient for describing the task motion). The original description of impedance control implementation used task coordinates [7]. It is very common to describe a desired impedance with respect to a Cartesian frame of reference [8].

Because control of an impedance is ultimately implemented with joint actuators, which are naturally described using joint coordinates, implementation in joint space has fewer challenges than implementation in task space. Task space impedance control requires a mathematical relationship between the joint space motion of the manipulator and the task space motion of the end-effector. This relationship, in general, is not one-to-one. *Redundant manipulators*, for which the number of joint coordinates is greater than the number of task coordinates, have an infinite number of joint motions that yield the same task motion. The dimension of the null space (space of joint motions that do not yield task motion) is equal to the *degree of redundancy*, calculated by subtracting the number of task coordinates from the number of joint coordinates. In addition to the task impedance, redundant manipulators require a null space impedance to prevent erratic null space motion.

Although there are more challenges in implementing impedance control in task space compared to joint space, it is more appropriate when the desired impedance is described in task coordinates. For constrained manipulation tasks, constraint directions are not easily described using joint coordinates, but are easily described relative to a Cartesian frame of reference.

Currently, the most common method of implementing impedance control is using *active control* with sensor feedback to impose any virtual joint impedance matrix [7]. The end-effector motion is measured and the actuator torques are controlled, based on those measurements, to emulate the response of a system with the desired impedance. A large body of work addresses active impedance control for: rigid manipulators [7] [9], flexible joint manipulators [10], redundant manipulators requiring null-space impedance [11] [12], and stable time-varying stiffness control [13].

A less established method of implementing impedance control is using kinematic and actuator redundancy to directly control the physical task impedance matrix [7]. This method is often called passive impedance control to distinguish it from active impedance control and because the physical impedance matrix is passive (forces oppose direction of motion). Passive impedance control is less common than active impedance control because it requires manipulators with more complex hardware. However, many state of the art manipulators [14] [15] [16] [17] [18] are capable of directly controlling impedance components, especially stiffness. Details on passive stiffness control including its implementation and discussion on its advantages and limitations relative to active control are provided below.

1.2 Passive Stiffness Control and Passive Compliance Control

In constrained manipulation tasks, stiffness is the most critical impedance component to control. Because of uncertainty of the environment and limited actuator precision, there is always discrepancy between the commanded end-effector position and the actual end-effector position. Instead of penetrating the contact constraint surface, the commanded end-effector position is displaced to the actual configuration on the constraint surface. The resulting contact force depends on the net stiffness (force-displacement relationship) of the manipulator and environment. When both are very stiff, which is typical in many manufacturing tasks, the contact forces are very high.

Passive stiffness control directly controls the end-effector's physical stiffness to be beneficial for the task. A lower stiffness in contact-constrained directions reduces the interaction force due to position misalignments. A higher stiffness in unconstrained directions reduces end-effector displacement due to force uncertainties in those directions (e.g., uncharacterized friction or mechanical work). For manipulation tasks in which the constraint direction changes, a

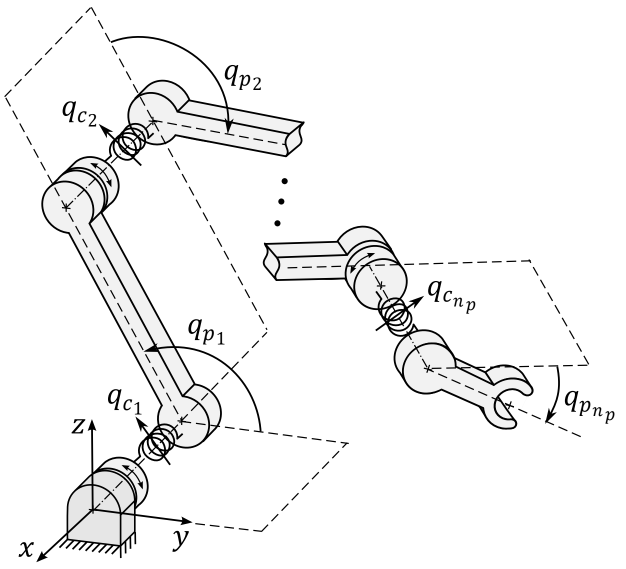


Figure 1.1: A redundant serial manipulator with elastic joints. If joint i is equipped with an SEA, the kinematic joint position q_{p_i} is controlled by the actuator and q_{c_i} is a fixed joint compliance (inverse of stiffness). If joint i is equipped with a VSA, each joint has two actuators such that both q_{p_i} and q_{c_i} are independently adjustable.

time-varying task stiffness is beneficial.

1.2.1 Implementation

Two methods for directly controlling the task stiffness matrix [7] involve using: 1) *kinematic redundancy* to adjust the kinematic joint configuration (i.e., spatial locations of the joints) which adjusts the task impedance matrix without adjusting the joint impedance matrix, or 2) *actuation redundancy* to adjust the physical joint stiffness matrix (a property of the human musculoskeletal system in co-contraction of antagonistic muscles).

A time varying task stiffness can be implemented passively (directly) using manipulators with structures like the serial manipulator in Fig. 1.1 with elastic elements between consecutive links at each joint. A manipulator of this type, has a diagonal joint stiffness matrix. Kinematically redundant manipulators with fixed joint stiffness from series elastic actuators (SEAs) [14] [15] [16] can control the task stiffness using Method 1. Non-redundant manipulations with real-time adjustable joint stiffnesses, from variable stiffness actuators (VSAs), can control the task stiffness using Method 2. Redundant manipulators with VSAs [17] [18] can use both methods.

A simple example of using these methods to adjust a particle planar elastic behavior realized by a manipulator with three elastic joints is provided below. Figure 1.2a shows the elastic

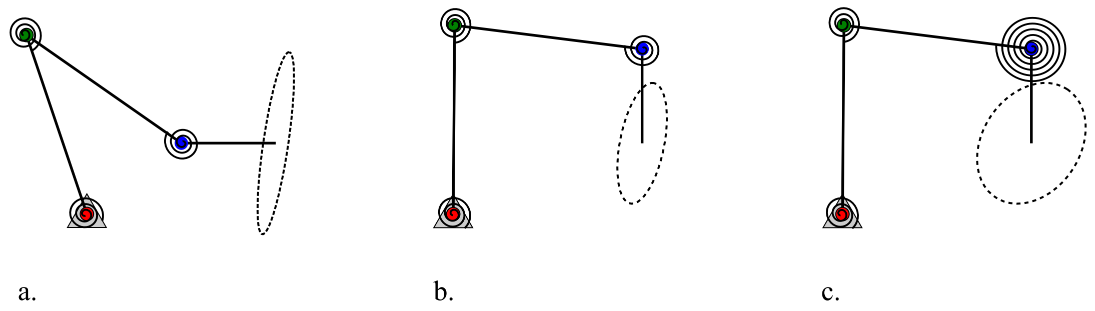


Figure 1.2: Physically implemented task compliances of the end-effector: a) with the original kinematic configuration and joint compliance values, b) then with an adjusted kinematic joint configuration, c) then with an adjusted compliance value (third joint compliance increases by a factor of 5). The dashed lines are compliance ellipses.

behavior as a task compliance ellipse (see Appendix A) showing the displacement characteristics of the end-effector from a unit force. The elastic behavior can be adjusted using Method 1 by changing the kinematic joint configuration without changing the joint elastic properties (illustrated by Fig. 1.2a to 1.2b), using Method 2 by changing the joint elasitc properties (using variable stiffness actuators), without changing the kinematic joint configuration (illustrated by Fig. 1.2b to 1.2c), or using Method 1 and Method 2 by changing the kinematic joint configuration *and* the joint elastic properties (illustrated by Fig. 1.2a to 1.2c). In all illustrated cases, the end-effector position is the same.

With passive stiffness control, each joint's equilibrium position and joint stiffness are *independently* controlled. This contrasts with active stiffness control, where the primary actuators are controlled based on sensor feedback to emulate the desired stiffness at a commanded virtual equilibrium point. As such, passive implementation of the desired task stiffness requires identifying a set of joint positions and joint stiffnesses that realize that task stiffness at the desired end-effector position.

The elastic behavior (force-displacement relationship) of the end-effector, described in terms of stiffness or compliance (inverse of stiffness), is determined by the kinematic joint configuration of the manipulator and each joint's elastic property. For serial manipulators, the mathematical expression (mapping) for the end-effector elastic behavior is easy to derive in terms of compliance, where the compliance of the end-effector is the superposition of the task-compliance resulting from each joint compliance, individually. The relationship between the elastic behavior of the joints of a serial manipulator and the task elastic behavior of its end-effector is difficult to derive in terms of stiffness. Whereas, the end-effector compliance is readily determined by the joint compliance through compliance addition. For this reason, this work uses *passive compliance control* where the joint positions and joint *compliances* are controlled to realize a desired task *compliance* at the desired end-effector position. Note, passive stiffness control and passive compliance control are two different approaches to achieving the same overall behavior. They both directly control the physical elastic properties of the manipulator, but they use different descriptions of the elastic behavior.

1.2.2 Advantages and Disadvantages Relative to Active Control

The advantage of passive control over active control is that the manipulator responds more robustly to force or displacement disturbances compared to active control. Active stiffness control has a bandwidth limitation [19] where the manipulator control loop may not respond to the environmental interaction fast enough to emulate the desired elastic behavior. Passive control does not have this bandwidth limitation because it operates *directly* on the physical elastic property without a control loop. A comparison [20] between passive and active stiffness control of a 3R manipulator inserting a planar-rectangular block into a slot with a lead-in chamfer showed that passive stiffness control typically performed the assembly task 8 times faster and experienced smaller contact forces. A constant joint stiffness was used for the task.

A limitation of passive compliance control is the difficulty of its realization. As noted in [21], it is easy to implement an arbitrary task compliance with a bandwidth-limited active control, but difficult, if not impossible, to implement the desired compliance passively. As shown in [22] [23] [24] [25] [26] [27], the manipulator's ability to produce a desired end-effector passive compliance strongly depends on its kinematic configuration. The set of realizable end-effector compliances is extremely limited for a non-redundant manipulator that cannot adjust its kinematic joint configuration. The set of implementable end-effector compliances is much larger for a redundant manipulator, for which the manipulator can adjust its kinematic joint configuration without affecting the end-effector's position.

Another challenge associated with passive compliance control is that robots with elastic joints experience deflection due to gravity. The commanded kinematic joint configuration of a manipulator with non-compliant joints is different than the static equilibrium position of the same manipulator with compliant joints. Gravity compensation is addressed [10] [28] using quasi-static analysis to identify the actuator positions that yield the desired joint configuration at static equilibrium.

Although there are challenges in passive compliance control that are not present in active control, the performance advantage is potentially very great. The more difficult challenge of realizing the task position and compliance passively is surmounted using kinematic redundancy with joints having controllable elastic behavior.

1.3 Problem Addressed

This work addresses passive compliance control of *redundant* serial manipulators with real-time adjustable joint compliances for executing constrained manipulation in task space without using active control. When damping and inertia are omitted from the dynamic model of the system, manipulation can be treated as a time-indexed sequence of configurations in static equilibrium. As such, a constrained manipulation task is described by a *task manipulation path*, a continuous sequence of desired end-effector positions and compliances in task space. This work addresses the problem of identifying a *joint manipulation path*, a continuous sequence of joint positions and joint compliances that execute the task manipulation path. The joint positions and compliances of the joint path must be *feasible* for which the manipulator hardware can attain.

Although kinematic redundancy increases the ability to attain the desired positions and compliances of the task manipulation path, it complicates the problem of selecting the joint manipulation path because the path is no longer unique. A feasible joint path is a path through the feasible solution space given by the set of feasible joint positions and feasible joint compliances that realize the task positions and task compliances of the task manipulation path. The feasible solution space typically is not simply connected, it typically has a complex connection structure. For some task manipulation paths, no feasible joint path exists. Because of redundancy, when a path does exist, there is an infinite set of joint paths that can complete the task. Prior work that address aspects of this problem are reviewed below.

1.3.1 Prior Work in Passive Stiffness/Compliance Control

Others have investigated the space of realizable end-effector stiffnesses by sampling the kinematic joint positions and joint stiffnesses and calculating the end-effector stiffness. The space of realizable stiffness is visualized using scatter plots [29] or "stiffablity maps" [30]. These sampling approaches, however, do not provide information about the connectivity of the joint positions and joint stiffnesses.

Necessary and sufficient conditions for a manipulator to achieve a specified passive compliance have been identified for a variety of manipulator types. The conditions and geometric interpretations for 2 and 3 joint manipulators realizing a particle planar compliance have been identified for revolute joints [22] and for a combination of revolute or prismatic joints [23]. The conditions and the geometric interpretations have been identified for 3R [24], 4R [25], and 5R [26] manipulators realizing a general planar compliance. The conditions and the geometric interpretations have also been identified for 6R [27] manipulators realizing a general spatial compliance. For each type of manipulator, the conditions on manipulator geometry are decoupled from the conditions on the joint compliance values. Results apply to finding a single joint configuration (position and compliance) at a single task instance. The connectivity of joint configurations at a single task instance is not addressed, nor is the generation of a time indexed joint path through the set of admissible joint configurations of all task instances.

The problem of selecting the passive joint compliances for redundant manipulators similar to the type illustrated in Fig. 1.1 is addressed in [31][30] using optimization. The optimization criterion minimizes the Frobenius norm of the difference between the desired task compliance and the passive end-effector compliance. The optimization in [31] selects the passive joint compliance without adjusting the kinematic joint configuration. The passive joint compliances attained by this optimization are not guaranteed to realize the desired task compliance. To compensate, the elastic behavior is supplemented with active stiffness control. In [30], a null space controller is used to adjust the kinematic joint configuration based on a local optimization for improving the task compliance. This approach adjusts the kinematic joint configuration without adjusting the passive joint compliance. These approaches [31][30] used for resolving the passive joint compliance and resolving the joint kinematics *separately* avoid using nonlinear constraints in the optimization, but sacrifice optimal joint control.

A method of simultaneous resolution of passive joint stiffness and joint kinematics is described in [32] without gravity compensation. The method tracks the *change* in task stiffness with respect to time using a stiffness Jacobian matrix. Little attention, however, was given to deriving the stiffness Jacobian. In [32], the stiffness Jacobian is approximated numerically using finite differences of a function that maps joint stiffness to task stiffness. The stiffness mapping function used in [32] is the inverse of the mapping commonly used in active stiffness control [33][34] for selecting the joint stiffness matrix from a desired task stiffness matrix. This choice of stiffness mapping, however, does not apply to redundant manipulators. For redundant

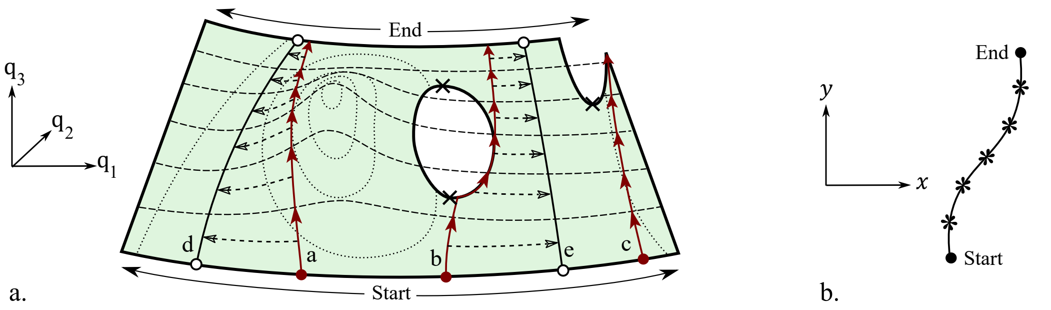


Figure 1.3: Redundant inverse kinematic (RIK) solution space with instantaneously and globally resolved joint paths. a) RIK solution space. b) Task path. Dashed lines are self-motion manifolds corresponding the task instances ($*$). Dotted lines are contour lines of the RIK solution space. Solid lines with arrow sequences are instantaneously resolved joint paths. Solid with open circle endpoints are globally resolved joint paths.

manipulators, the joint stiffness that implements the task stiffness is not unique and the stiffness matrix inverse does not exist. The active-stiffness-resolution-expression of [33][34] will select a joint stiffness matrix, likely one with off-diagonal terms, that cannot be implemented passively with a manipulator like that in Fig. 1.1 even if a passive solution exists. This approach can be used to generate a joint manipulation path for non-redundant manipulators. However, the set of task manipulation paths that can be realized by non-redundant manipulators is very limited.

1.4 Approach

Resolving redundancy in the set of joint positions and joint compliances that realize a desired task position and task compliance has not been addressed. However, resolving redundancy in manipulator kinematics (no compliance) has been widely addressed. In the problem of finding a path in the joint space to track a desired path in the task space, path planning occurs in the joint subspace that contains the set of feasible joint configurations that yield end-effector configurations on the task path. This work refers to this subspace as the feasible redundant inverse kinematic (RIK) solution space. The RIK solution space of a single task instance is a set of self-motion manifolds, each a continuously connected set of joint configurations that yield the same end-effector configuration. The RIK solution space of a task path is sequence of self-motion manifolds. Depending on the manipulator hardware, only a subset of the RIK solution space configurations are feasible configurations. The shaded surface in Fig. 1.3a represents a feasible RIK solution space of a task path in Fig. 1.3b. Each dashed line in Fig. 1.3a is a self-motion manifold of a single task instance corresponding to a $*$ in Fig. 1.3b.

Identifying joint configurations of the RIK solution space addresses the inverse kinematics (IK) problem. Selecting a unique joint configuration at each task instance addresses the redundancy resolution (RR) problem. The IK and RR problems are often resolved simultaneously *without* identifying the feasible RIK solution space beforehand. The redundant inverse kinematic (RIK) path planning problem can be resolved in real-time (i.e., the path is calculated as the manipulator executes it) using instantaneous resolution methods by identifying the optimal joint velocity for advancing the task. The RIK path planning problem can be optimally resolved off-line (i.e., the path is calculated before the manipulator executes it) using global resolution methods by identifying the joint path that minimizes a global cost function, a function of the entire joint path.

Differences between instantaneously and globally resolved paths are illustrated on the RIK solution space in Fig. 1.3. Instantaneous resolution requires a starting joint configuration and identifies the most direct joint motion to bring the joint path to the next self-motion manifold. The joint motion is integrated over the entire task to generate the path. This process is roughly illustrated by the sequence of arrows on paths a-c in Fig. 1.3. Although the instantaneously generated path is instantaneously optimal, the instantaneous optimization may guide the path into undesirable regions in the RIK solution space, e.g., path b must go around the hole and path c travels into a region that cannot complete the task. A shorter path can be obtained using a global resolution in which an initial joint path is deformed over the RIK solution space into a *locally* optimal path such that any infinitesimal deformation yields a longer joint path. There can be multiple locally optimal joint paths (e.g., paths d and e), especially when the RIK solution space is not simply connected. Finding the globally optimal (best) joint path is difficult when many locally optimal joint paths exist. No prior work addresses finding the globally optimal joint path in the *entire* RIK solution space.

This work provides an algorithm that identifies the globally optimal joint path for RIK problems with one degree of redundancy (or identifies if no solution exists). The algorithm does so by compactly characterizing the connectivity of the RIK solution space using feasible subsets of self-motion manifolds at bifurcation points. *Bifurcation points* are instances on the RIK solution space where feasible subsets of self-motion manifolds (bifurcation branches) split or converge (e.g., each $\times$ in Fig. 1.3). The algorithm uses a novel instantaneous path planner and a novel global resolution method with the bifurcation branches to efficiently and thoroughly search the entire RIK solution space for the globally optimal joint path. This algorithm is, therefore, called the bifurcation branch algorithm (bb-algorithm).

This work frames the challenge of passive compliance control as an extension of the RIK path planning problem to include compliance (RIK $\rightarrow$ RIKC). To extend the RIK problem, the joint space of the manipulator and the task space of the end-effector are augmented to include compliance coordinates. As such, the mapping from the joint space to the task space is augmented to include compliance; and the Jacobian matrix (tangent space mapping) that maps a joint velocity to a task velocity is also augmented to include compliance.

This work provides an instantaneous resolution approach for generating joint manipulation paths in the combined (kinematic and compliance) joint space for RIKC problems with any degree of redundancy. Redundancy is resolved by minimizing the norm of the actuator velocities (primary and VSAs). The actuator positions are not the same as the joint positions; they have a separate coordinate space. The actuator space is related to the joint space by a mapping function and actuator velocities are related to joint velocities by an actuator Jacobian (tangent space mapping). These mappings are used to compensate for deflection due to gravity and to resolve the manipulation planning in the joint space. The joint path is then converted to an actuator path, a sequence of actuator positions (both kinematic and compliance adjusting actuators). The actuator paths are executed with tracking control to implement passive compliance control. This approach is similar to the mathematical operations provided in [32]. However, the coordinate spaces are not clearly articulated in [32] and the choice of mapping functions is different. Unlike the stiffness mapping used in [32], the compliance mapping derived in this work does not require inverting the kinematic Jacobian matrix and is, therefore, valid for redundant manipulators and is computationally easier to evaluate. Additionally, the mappings in this work compensate for joint deflection due to gravity, which is not addressed in [32].

This work provides a global resolution approach for finding the best joint manipulation path (or identifies the non-existence of a joint path) for RIKC problems with one degree of redundancy. The bb-algorithm is used with the RIKC framework to identify the best joint manipulation path for passive compliance control. However, RIKC solution spaces are much more complex than RIK solution spaces, having multiple disconnected self-motion subspaces resulting from a greater number of self-motion manifolds and additional hardware limits from the VSAs. This work identifies the self-motion manifolds of RIKC problems and provides algorithms for identifying the bifurcation points and associated bifurcation branches.

1.5 Overview

This work addresses a challenge related to passive compliance control of redundant serial manipulators with adjustable joint compliances. The challenge is to determine a joint manipulation path (sequence of joint positions and compliances) that yields end-effector configurations of a desired task manipulation path (sequence of task positions and compliances). This challenge is addressed by extending the redundant inverse kinematic (RIK) path planning problem to include compliance (RIK $\rightarrow$ RIKC). This work provides an instantaneous resolution approach for RIKC problems with any degree of redundancy. This work also provides a global resolution approach to identify the best joint manipulation path for RIKC problems with one degree of redundancy.

Relevant background in the RIK path planning problem and its resolution using instantaneous methods is provided in Chapter 2. Chapter 2 then provides the details for extending the RIK problem to the RIKC problem, provides an instantaneous method of generating joint manipulation paths, and demonstrates the manipulation planner for 3R-VSA and 5R-VSA planar manipulators. Relevant background in the RIK path planning problem and its resolution using global methods is provided in Chapter 3 along with technical background relating to characterizing the RIK solution space. Chapter 3 also motivates, describes, and demonstrates the bifurcation branch algorithm (bb-algorithm) that is used for identifying the globally optimal joint path for RIK path planing problems with one degree of redundancy. Chapter 4 provides means of identifying the bifurcation points and the associated bifurcation branches in RIKC solution spaces. This information is used with the bb-algorithm and the RIKC framework to identify the globally optimal (best) joint path in the RIKC solution space. Chapter 5 summarizes the contributions of this work.

CHAPTER 2

OPTIMAL INSTANTANEOUS RESOLUTION OF PASSIVE COMPLIANCE CONTROL

Passive compliance control adjusts the robot end-effector's position and physical compliance in real-time to robustly execute constrained manipulation tasks. A redundant serial manipulator with adjustable joint compliance, like that illustrated in Fig. 1.1, can independently control its equilibrium joint positions and its joint compliances using its primary actuators and variable stiffness actuators (VSAs). The challenge in passive compliance control of redundant serial manipulators is identifying a *joint manipulation path*, a continuous sequence of joint positions and joint compliances executing actuator commands that yield the desired *task manipulation path*, a continuous sequence of desired task positions and task compliances. This challenge is addressed here by extending the redundant inverse kinematic (RIK) path planning problem to include compliance coordinates in joint and task configuration spaces.

First, this chapter reviews the RIK path planning problem and its resolution using the weighted pseudoinverse method to identify the instantaneously optimal joint velocity for advancing the task manipulation path. The weighted pseudoinverse method uses the task Jacobian matrix, which maps a joint velocity to a task velocity, to resolve inverse kinematics; and uses a weighting matrix to resolve redundancy according to a quadratic objective in the joint velocities.

Second, this chapter describes the augmentation of the joint and task coordinate spaces to include compliance in the total configuration spaces. The augmentation of the mapping functions used in resolving the inverse kinematics and compliance are also described. The compliance mapping function for any serial manipulator is provided and the compliance Jacobian matrix is derived.

Third, this chapter describes the weighing matrix used to select a joint velocity that minimizes the norm of the full set of actuator velocities (i.e., primary and VSA motor velocities). A distinction between the joint space and the actuator space (set of primary and VSA motor position coordinates) is made to account for link deflection due to gravity and to account for the nonlinear relationship between the VSA actuator position and the joint's compliance. The mapping function from the joint space to the actuator space is used to compensate for the link deflection due to gravity. An *actuator Jacobian* that relates joint velocities to actuator velocities is introduced. The actuator Jacobian defines the weighting matrix used in the weighted pseudoinverse to identify the joint velocities that minimize the norm of the actuator velocities.

Finally, this chapter explicitly shows how all these mappings are used in an instantaneous resolution method to generate a continuous sequence of actuator commands for implementing passive compliance control. The instantaneous RIKC resolution method is demonstrated for a 3R-VSA manipulator performing a particle planar compliance task and for a 5R-VSA manipulator performing a particle planar compliance task and a general planar compliance task.

2.1 Review of Redundant Inverse Kinematics and Instantaneous Resolution

Consider a serial manipulator with n joint configuration coordinates where the end-effector performs a motion task described by m coordinates. The number of task coordinates depends on the type of manipulation; $m = 2$ for particle planar manipulation, $m = 3$ for general planar manipulation, $m = 6$ for spatial manipulation. Figure 2.1 shows the spatial configuration of a serial manipulator and the spacial configuration of its end-effector. The manipulator configuration is determined by each joint position q_i. The end-effector configuration is given by the end-effector frame relative to the base frame.

The forward kinematic map $\boldsymbol{f}(\mathbf{q}) : Q \to X$ is a nonlinear function that defines the relationship between the manipulator's configuration in joint space $\mathbf{q} \in Q$ (relative angles between consecutive links) and the end-effector configuration in task space $\mathbf{x} \in X$ (position and orientation[1] of the end-effector).

The relationship between the joint velocity $\dot{\mathbf{q}}$ and task velocity $\dot{\mathbf{x}}$ is linear:

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}, \tag{2.1}$$

where $\mathbf{J}(\mathbf{q}) = \partial \boldsymbol{f}(\mathbf{q})/\partial \mathbf{q}$, is the $m \times n$ task Jacobian evaluated at the current joint configuration $\mathbf{q}$. This Jacobian is also called the analytical Jacobian, which is different from the geometric Jacobian[2] $\boldsymbol{J}(\mathbf{q})$ for spatial manipulation [35], but they are the same for planar manipulation.

Consider a task path $\boldsymbol{x}(t) \in X$, a continuous sequence of desired task configurations corresponding to the desired motion of the manipulator's end-effector. The task path is indexed by normalized time $t \in [0, 1]$. The inverse kinematic (IK) path planning problem is to find a joint

[1]For spatial orientation there is not an obvious choice of minimal coordinates; common choices include [35] Euler angles and roll, pitch, yaw angles.

[2]The geometric Jacobian defines the relationship: $[v, \omega]^T = \boldsymbol{J}(\mathbf{q})\dot{\mathbf{q}}$, where v is the linear velocity of frame (x', y', z') relative to (x, y, z) and ω is the angular velocity of frame $(x^{\mathrm{E}}, y^{\mathrm{E}}, z^{\mathrm{E}})$ relative to (x', y', z'). The analytical Jacobian and geometric Jacobian are related to each other by a transformation matrix that depends on the orientation representation parameters (e.g., Euler angles).

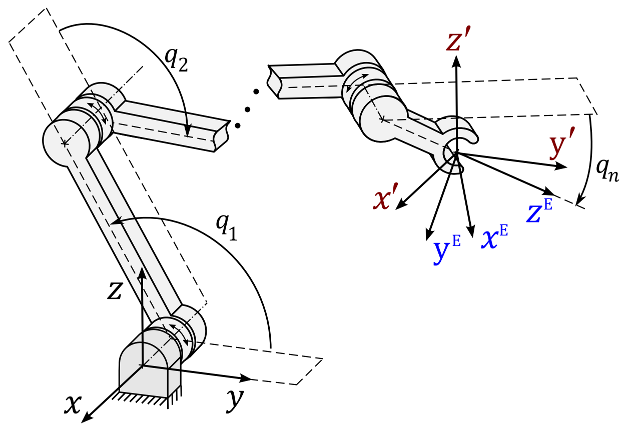


Figure 2.1: Joint configuration of a serial manipulator and the task configuration of its end-effector. Coordinate frames: 1) (x, y, z) attached to the base, 2) (x^E, y^E, z^E) attached to the end-effector, 3) (x', y', z') with its origin the same as the end-effector frame and its orientation the same as the base frame.

path $\boldsymbol{q}(t) \in Q$ such that $\boldsymbol{f}(\boldsymbol{q}(t)) = \boldsymbol{x}(t)$ for all time t, where $\boldsymbol{q}(t)$ is a continuous set of joint configurations. For redundant manipulators $(n > m)$, there is an infinite number of solutions to the IK problem. Choosing a unique joint path from the IK solution space is known as the redundancy resolution (RR) problem. The IK and RR problems are resolved together simultaneously using instantaneous optimization.

2.1.1 Instantaneous Resolution

A solution to the RIK path planning problem is generated using instantaneous resolution [36] by integrating the optimal joint velocity (instantaneous joint motion):

$$\boldsymbol{q}(t) = \mathbf{q}_0 + \int_{t=0}^{t=1} \dot{\boldsymbol{q}}^*(t)\, dt, \tag{2.2}$$

where $\mathbf{q}_0 \in Q \mid \boldsymbol{f}(\mathbf{q}_0) = \boldsymbol{x}(0)$ is an admissible initial joint configuration, and $\dot{\boldsymbol{q}}^*(t)$ is the optimal admissible joint velocity identified by solving the optimization problem:

$$\begin{aligned} \text{min.} \quad & G_{\text{inst}}(\mathbf{q}, \dot{\mathbf{q}}) \\ \text{s.t.} \quad & \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} = \dot{\mathbf{x}}, \end{aligned} \tag{2.3}$$

where $\mathbf{q}$ is the joint configuration at the current time-instance t and $\dot{\mathbf{q}}$ and $\dot{\mathbf{x}}$ are the joint and task velocities, respectively. Satisfying the constraint function (the bottom expression in (2.3)) addresses the IK problem by ensuring the selected joint velocity realizes the necessary end-effector velocity to advance the task. Minimizing the objective function (the top expression in (2.3)) addresses the RR problem by selecting the best joint velocity that advances the task. The objective function is commonly selected to be quadratic in joint velocity:

$$G_{\text{inst}} = \frac{1}{2}\dot{\mathbf{q}}^T \mathbf{W} \dot{\mathbf{q}}, \tag{2.4}$$

where $\mathbf{W}$ is an $n \times n$ positive definite matrix of weighting values and the domain of the optimization is the set of joint velocities that produce the required task velocity given by linear constraints (2.1). The weighting matrix may be selected to minimize the norm of the joint velocities, minimize the kinetic energy of the manipulator [36], minimize the distance from joint limits [37], or some other criterion.

The unique joint velocity $\dot{\mathbf{q}}^*$ that minimizes (2.4) while satisfying (2.1) is readily identified [36] using

$$\dot{\mathbf{q}}^* = \mathbf{J}^\dagger \dot{\mathbf{x}}, \tag{2.5}$$

where $\mathbf{J}^\dagger$ is the weighted pseudoinverse matrix of $\mathbf{J} \equiv \mathbf{J}(\mathbf{q})$, calculated by

$$\mathbf{J}^\dagger = \mathbf{W}^{-1}\mathbf{J}^T(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^T)^{-1}, \tag{2.6}$$

when $\mathbf{J}$ is full-rank[3] ($\text{rank}(\mathbf{J}) = m$). The weighted pseudoinverse simultaneously addresses the IK problem with $\mathbf{J}$ and addresses the RR problem with $\mathbf{W}$.

If the selected optimization criterion is non-quadratic (e.g., maximizing the manipulability index [40]), the optimal *non*-admissible joint velocity $\dot{\mathbf{q}}_N$ (that does not satisfy the task constraints of the Jacobian) is identified from the gradient of the instantaneous objective function. The closest *admissible* joint velocity (in the weighted least squares sense) solves

$$\begin{aligned} \text{min.} \quad & \tfrac{1}{2}(\dot{\mathbf{q}} - \dot{\mathbf{q}}_N)^T \mathbf{W} (\dot{\mathbf{q}} - \dot{\mathbf{q}}_N) \\ \text{s.t.} \quad & \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} = \dot{\mathbf{x}}, \end{aligned} \tag{2.7}$$

where $\dot{\mathbf{q}}_N$ is the “preferred” joint velocity. The optimization objective (2.4) is equivalent to (2.7) with $\dot{\mathbf{q}}_N = \mathbf{0}$.

The joint unique velocity $\dot{\mathbf{q}}^*$ that solves (2.7) is readily identified [41] using:

[3]If $\mathbf{J}$ is not full-rank, the weighted pseudoinverse is evaluated using singular value decomposition [38][39].

$$\dot{\mathbf{q}}^* = \mathbf{J}^\dagger \dot{\mathbf{x}} + (\mathbf{I} - \mathbf{J}^\dagger \mathbf{J})\dot{\mathbf{q}}_N, \tag{2.8}$$

where $\mathbf{I}$ is the identity matrix. Equation 2.5 is equivalent to 2.8 with $\dot{\mathbf{q}}_N = \mathbf{0}$.

2.1.2 Instantaneous Resolution Considering Hard Joint Limits

Most manipulators have fundamental limits where each joint variable is bounded by $q \in [q_{\min}, q_{\max}]$, elements of the joint configuration limits $[\mathbf{q}_{\min}, \mathbf{q}_{\max}]$. The instantaneously resolved joint path, generated by integrating (2.8) (solution to (2.7)), may yield joint configurations that violate the joint limits.

To prevent this, joint velocity is subject to the following inequality constraints:

$$\frac{\mathbf{q}_{\min} - \mathbf{q}}{\Delta t} \le \dot{\mathbf{q}} \le \frac{\mathbf{q}_{\max} - \mathbf{q}}{\Delta t}, \tag{2.9}$$

where Δt is the numerical integration step size and $\mathbf{q}$ is the current joint configuration of the generated path. The optimal admissible joint velocity solves:

$$\begin{aligned} \text{min.} \quad & \tfrac{1}{2}(\dot{\mathbf{q}} - \dot{\mathbf{q}}_N)^T \mathbf{W} (\dot{\mathbf{q}} - \dot{\mathbf{q}}_N) \\ \text{s.t.} \quad & \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} = \dot{\mathbf{x}} \\ & \tfrac{\mathbf{q}_{\min} - \mathbf{q}}{\Delta t} \le \dot{\mathbf{q}} \le \tfrac{\mathbf{q}_{\max} - \mathbf{q}}{\Delta t}. \end{aligned} \tag{2.10}$$

The joint velocity solving (2.10) with $\mathbf{W} = \mathbf{I}$ is identified using the Saturation in the Null Space (SNS) Algorithm [42]. There are three versions of the algorithm, a basic, optimal, and fast version. The basic version is briefly described here.

The SNS algorithm first calculates $\dot{\mathbf{q}}$ using (2.8), then checks if inequality constraints (2.9) are satisfied. If they are not, the most violated joint value, q_i is saturated at its limiting value by setting element i of $\dot{\mathbf{q}}_N$ to element i of the inequality constraint (2.9). Equation (2.8) is re-calculated, replacing $\mathbf{J}^\dagger$ with $\mathbf{J}^\dagger_{\text{SNS}} = (\mathbf{JS})^{\#}$, where $(\cdot)^{\#}$ is the Moore-Penrose[4] pseudoinverse and $\mathbf{S}$ is the saturation matrix: an $n \times n$ diagonal matrix with element $S_{ii} = 1$ for non-saturated joints or $S_{ii} = 0$ for saturated joints. This forces the non-saturated joints to compensate. The process is repeated until either a feasible joint velocity is identified or until $\text{rank}(\mathbf{JS}) < m$.

[4]The Moore-Penrose pseudoinverse is the weighted pseudoinverse with an identity weighting matrix.

2.1.3 Summary

A joint path $\boldsymbol{q}(t) \in Q$ that satisfies a desired task path $\boldsymbol{x}(t) \in X$ is generated by integrating the optimal admissible joint velocity identified using the weighted pseudoinverse. The SNS algorithm is used to accommodate joint limits.

By augmenting the joint configuration coordinates and task configuration coordinates to include compliance (inverse of stiffness), the weighted pseudoinverse based instantaneous RIK resolution approach is extended below to track both a desired equilibrium task position and a desired task compliance.

2.2 Compliance Extended Inverse Kinematics

To include compliance in the instantaneous inverse kinematics (IK) resolution approach, the joint space Q and task space X are augmented to include joint compliance coordinates and task compliance coordinates, respectively. The mapping function $\mathbf{x} = \boldsymbol{f}(\mathbf{q})$ from the joint space to the task space and the tangent space mapping $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$ are augmented with additional mappings for compliance. This section provides the details. To emphasize the distinction between the typical kinematic and the augmented compliance coordinates and mappings, subscript $(\cdot)_p$ indicates kinematic coordinates or mappings and subscript $(\cdot)_c$ indicates compliance coordinates or mappings.

2.2.1 Compliance Augmented Joint Space

The class of manipulators considered in this work are *serial* manipulators with n_p revolute joints where each joint has adjustable compliance (as depicted in Fig. 1.1). The *kinematic* joint configuration is

$$\mathbf{q}_p = [q_{p_1}, q_{p_2}, \dots, q_{p_{n_p}}]^T,$$

where the coordinate $q_{p_i} \in (-\infty, \infty)$ is the relative angle between link i and link $(i-1)$ (link 0 is the ground). The kinematic joint configuration $\mathbf{q}_p$ is the static equilibrium configuration of the manipulator with respect to all *known* external loads such as those due to gravity. The kinematic joint configuration is the actual position/orientation of the end-effector disturbances when no disturbance acts on the manipulator (i.e., no commanded position error due to constrained interaction).

The passive compliance of each joint is independently controlled, therefore, the *compliance* joint configuration is

$$\mathbf{q}_c = [q_{c_1}, q_{c_2}, \ldots, q_{c_{n_p}}]^T,$$

where the coordinate $q_{c_i} \in (-\infty, \infty)$ is the compliance of joint i. Note, the joint compliance must have a positive value to be realized; this issue is treated the same as joint limits (discussed in Section 2.4). The joint compliance matrix $\mathbf{C}_q$ is an $n_p \times n_p$ diagonal matrix with $\mathbf{q}_c$ as the diagonal elements.

Because the kinematic joint configuration and the compliance joint configuration are controlled independently, the total joint space configuration $\mathbf{q} \in Q$ of the manipulator is formed by concatenating the kinematic and compliance joint configurations:

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_p \\ \mathbf{q}_c \end{bmatrix}.$$

The total number of joint coordinates is $n = 2n_p$, when each joint compliance is adjustable.

2.2.2 Compliance Augmented Task Space

The task manipulation path for compliance control involves both kinematics and compliance. The robot end-effector has a specific position/orientation given by the $m_p \times 1$ vector $\mathbf{x}_p$, where m_p is the number of task kinematic degrees of freedom. The end-effector elastic behavior is described with respect to a coordinate frame attached to the end-effector with the orientation of the base frame (e.g., (x', y', z') in Fig. 2.1). The specific elastic behavior is given by $\mathbf{C}_x$, an $m_p \times m_p$ positive definite, symmetric task compliance matrix.

Because $\mathbf{C}_x$ is symmetric, the task compliance is fully and uniquely described by the upper triangular elements. Therefore, these elements are selected as the task compliance coordinates of the task compliance configuration:

$$\mathbf{x}_c = \mathrm{sort}^{\triangle}(\mathbf{C}_x)\,, \tag{2.11}$$

where $\mathrm{sort}^{\triangle}(\cdot)$ is a sorting operation for arranging the upper triangular elements of $\mathbf{C}_x$ into an $m_c \times 1$ vector based on element position in the matrix. The number of task compliance coordinates is $m_c = \frac{1}{2}m_p(m_p+1)$.

The augmented task configuration $\mathbf{x} \in X$ is formed by concatenating the task kinematic and task compliance configurations:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_p \\ \mathbf{x}_c \end{bmatrix}.$$

2.2.3 Compliance Augmented Forward Maps

The augmented joint configuration $\mathbf{q}$ and augmented task configuration $\mathbf{x}$ are each a concatenation of kinematic and compliance configurations. Because of this structure, the forward mapping function $\boldsymbol{f}(\mathbf{q})$ is a concatenation of the forward kinematics and forward compliance functions:

$$\boldsymbol{f}(\mathbf{q}) = \begin{bmatrix} \boldsymbol{f}_p(\mathbf{q}) \\ \boldsymbol{f}_c(\mathbf{q}) \end{bmatrix},$$

where the forward kinematic function $\boldsymbol{f}_p$ gives the task kinematic configuration $\mathbf{x}_p$ and the forward compliance function $\boldsymbol{f}_c$ gives the task compliance configuration, $\mathbf{x}_c$.

In the instantaneous compliance resolution approach, compliance and kinematics are resolved simultaneously at the instantaneous (differential) level. Rather than finding the appropriate joint compliance configuration for each task instance, the best infinitesimal change in the joint configuration (kinematic and compliance) is identified to produce the desired infinitesimal change in the task configuration (kinematic and compliance). The differential task requirements are a set of linear constraints given by the task Jacobian matrix $\mathbf{J} = \partial \boldsymbol{f}(\mathbf{q})/\partial \mathbf{q}$.

Because the joint space and task space coordinates are concatenations of kinematic and compliance coordinates, the partitioned form of the Jacobian matrix mapping a joint velocity to a task velocity is

$$\begin{bmatrix} \dot{\mathbf{x}}_p \\ \dot{\mathbf{x}}_c \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{f}_p(\mathbf{q})}{\partial \mathbf{q}_p} & \frac{\partial \boldsymbol{f}_p(\mathbf{q})}{\partial \mathbf{q}_c} \\ \frac{\partial \boldsymbol{f}_c(\mathbf{q})}{\partial \mathbf{q}_p} & \frac{\partial \boldsymbol{f}_c(\mathbf{q})}{\partial \mathbf{q}_c} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_p \\ \dot{\mathbf{q}}_c \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{pp} & \mathbf{J}_{pc} \\ \mathbf{J}_{cp} & \mathbf{J}_{cc} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_p \\ \dot{\mathbf{q}}_c \end{bmatrix}. \qquad (2.12)$$

where $\mathbf{J}_c = [\mathbf{J}_{cp}, \mathbf{J}_{cc}]$ is the compliance Jacobian. The partition $\mathbf{J}_{pp} = \partial \boldsymbol{f}_p(\mathbf{q})/\partial \mathbf{q}_p$ is the kinematic Jacobian (analytical Jacobian in IK literature). The partition $\mathbf{J}_{pc} = \partial \boldsymbol{f}_p(\mathbf{q})/\partial \mathbf{q}_c$ describes how the end-effector task position changes with changes in the joint compliance variables. Because robot kinematics do not depend on the joint compliance variables, $\mathbf{J}_{pc} = \mathbf{0}$. The partition $\mathbf{J}_{cp} = \partial \boldsymbol{f}_c(\mathbf{q})/\partial \mathbf{q}_p$ describes how the end-effector task compliance changes with changes in the joint kinematic variables. The partition $\mathbf{J}_{cc} = \partial \boldsymbol{f}_c(\mathbf{q})/\partial \mathbf{q}_c$ describes how the end-effector task compliance changes with changes in the joint compliance variables.

In summary, the compliance augmented Jacobian for serial manipulators with variable stiffness actuation (those like that illustrated in Fig. 1.1) is

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{pp} & \mathbf{0} \\ \mathbf{J}_{cp} & \mathbf{J}_{cc} \end{bmatrix}, \tag{2.13}$$

where each partition is described above.

2.2.4 Constructing the Compliance Jacobian

In this section, the compliance forward map and the compliance Jacobian are constructed using the geometric Jacobian and the joint compliance matrix.

Serial manipulators with independently controlled elastic joints have a simple mapping from the manipulator's joint compliance matrix to its end-effector task compliance matrix. The end-effector's task compliance is the superposition of the task compliance contribution from each joint and is given by [43]:

$$\mathbf{C}_x = \sum_{i=1}^{n_p} q_{c_i} \mathbf{t}_i \mathbf{t}_i^T, \tag{2.14}$$

where q_{c_i} is the compliance of joint i and $\mathbf{t}_i$ is the twist associated with joint-i. Each twist is the instantaneous kinematic motion of the end-effector due to the instantaneous kinematic motion of that joint, (i.e., a column in the geometric Jacobian matrix). The task compliance of the end-effector is compactly given by [43]:

$$\mathbf{C}_x(\mathbf{q}) = \boldsymbol{J}_{pp} \mathbf{C}_q \boldsymbol{J}_{pp}^T, \tag{2.15}$$

where $\boldsymbol{J}_{pp}$ is the $m_p \times n_p$ geometric Jacobian matrix and $\mathbf{C}_q$ is the diagonal joint compliance matrix with $\mathbf{q}_c$ as the diagonal elements. The end-effector's task compliance matrix $\mathbf{C}_x(\mathbf{q})$ is symmetric and positive definite. Note that (2.15) does not require calculating the inverse Jacobian matrix (a step needed in the mapping function described in [32]).

The forward compliance function as a *configuration* mapping is readily constructed by sorting the upper triangular elements of $\mathbf{C}_x$ into a vector:

$$\boldsymbol{f}_c(\mathbf{q}) = \mathrm{sort}^{\triangle}\Big(\boldsymbol{J}_{pp} \mathbf{C}_q \boldsymbol{J}_{pp}^T \Big). \tag{2.16}$$

The compliance Jacobian is $\mathbf{J}_c = \partial \boldsymbol{f}_c(\mathbf{q})/\partial \mathbf{q}$ and the variation in task compliance configuration is $\delta \mathbf{x}_c = \mathbf{J}_c \delta \mathbf{q}$, where $\delta \mathbf{q}$ is a variation in the augmented joint configuration. The compliance Jacobian, composed as a set of columns vectors is

$$\mathbf{J}_c = \left[\frac{\partial \boldsymbol{f}_c(\mathbf{q})}{\partial q_1}, \frac{\partial \boldsymbol{f}_c(\mathbf{q})}{\partial q_2}, \ldots, \frac{\partial \boldsymbol{f}_c(\mathbf{q})}{\partial q_n} \right] = \sum_{k=1}^{n} \frac{\partial \boldsymbol{f}_c(\mathbf{q})}{\partial q_k}, \tag{2.17}$$

where $\frac{\partial \boldsymbol{f}_c(\mathbf{q})}{\partial q_k}$ is the k^{th} column of $\mathbf{J}_c$ and q_k is the k^{th} joint space variable.

Consider the partial derivative of the task compliance function with respect to the k^{th} joint variable,

$$\frac{\partial \boldsymbol{f}_c(\mathbf{q})}{\partial q_k} = \frac{\partial}{\partial q_k} \text{sort}^{\triangle}\Big(\boldsymbol{J}_{pp} \mathbf{C}_q \boldsymbol{J}_{pp}^T \Big) . \tag{2.18}$$

Both the sorting operation and the partial derivative of a matrix with respect to a single variable are element-by-element operations. Therefore, sorting may occur after taking the partial derivative. Substituting (2.15) into (2.18) yields

$$\frac{\partial \boldsymbol{f}_c(\mathbf{q})}{\partial q_k} = \text{sort}^{\triangle}\left(\frac{\partial \mathbf{C}_x(\mathbf{q})}{\partial q_k} \right) \tag{2.19}$$

.

Using the product rule, the partial derivative of (2.15) expands to

$$\frac{\partial \mathbf{C}_x(\mathbf{q})}{\partial q_k} = \begin{cases} \left(\frac{\partial \boldsymbol{J}_{pp}}{\partial q_k}\right) \mathbf{C}_q \boldsymbol{J}_{pp}^T + \boldsymbol{J}_{pp} \mathbf{C}_q \left(\frac{\partial \boldsymbol{J}_{pp}}{\partial q_k}\right)^T & 1 \leq k \leq n_p \\ \boldsymbol{J}_{pp} \left(\frac{\partial \mathbf{C}_q}{\partial q_k}\right) \boldsymbol{J}_{pp}^T & n_p + 1 \leq k \leq n \end{cases} , \tag{2.20}$$

where q_k is a kinematic joint variable for $1 \leq k \leq n_p$ and a compliance joint variable for $n_p + 1 \leq k \leq n$. This decomposition is similar to that used in the conservative congruence transform [44] for finite deflection of a mechanism stiffness. This decomposition, however, is on compliance and it includes an additional term that accounts for variation in the joint compliance. Equation (2.20) may be separated into cases because $\mathbf{C}_q$ is independent of the kinematic variables $\mathbf{q}_p$, and $\boldsymbol{J}_{pp}$ is independent of the compliance variables $\mathbf{q}_c$.

The columns of the compliance Jacobian are constructed by substituting (2.20) into (2.19), where (2.20) is readily calculated as $\partial \boldsymbol{J}_{pp}/\partial q_k$ and $\partial \mathbf{C}_q/\partial q_k$ are evaluated element-by-element. Evaluating the partial derivative of the geometric Jacobian with respect to each kinematic joint variable is described in [45]. The partial derivatives of the joint compliance matrix components have a simple form given by

$$\left(\frac{\partial \mathbf{C}_q}{\partial q_k}\right)_{ij} = \begin{cases} 1 & i = j = k - n_p \\ 0 & \text{otherwise} \end{cases}. \tag{2.21}$$

With these results, the compliance augmented Jacobian (total Jacobian) **J** is readily calculated and used for instantaneously resolving the compliance extended inverse kinematics problem using the weighted pseudoinverse method.

2.3 Compliance Extended Redundancy Resolution

When a solution to the compliance extended instantaneous IK problem exists, the solution is not unique for redundant manipulators. There are many kinematic and compliance *joint motions* that produce the desired kinematic and compliance *task motion*. Here, the redundancy is resolved by minimizing the velocity norm of the actuators used to obtain the desired instantaneous change in the kinematic and compliance configurations.

For traditional industrial manipulators with rigid joints, the joint coordinates (relative angle between consecutive links) and the actuator coordinates (primary actuator positions) are the same. This is not that case for manipulators with elastic joints; actuator coordinates form a separate space Φ from the joint space Q. The actuator coordinates of serial manipulators with variable stiffness actuators (VSAs) are separate from its joint coordinates for two reasons: 1) externally applied loads displace the joint position from the no-load equilibrium position commanded by the primary actuators, and 2) the VSA actuator position and the joint compliance have different units and therefore separate configuration spaces. The differences between the actuator coordinates and joint coordinates for both kinematics and compliance are illustrated in Fig. 2.2.

In this section, a mapping function from joint space to actuator space for both kinematic and compliance variables is described. A linear mapping from the joint tangent space to the actuator tangent space (the actuator Jacobian) is also described. This actuator Jacobian is used in the pseudoinverse weighting matrix for minimizing the actuator motion.

2.3.1 Actuator Coordinates

As stated previously and illustrated in Fig. 1.1, manipulators having variable stiffness actuators allow kinematic joint position and joint compliance to be independently controlled. Although each joint coordinate value is independently controlled, it does not need to be controlled by a single actuator. For instance, antagonistic tendons with non-linear stiffness elements [46] control the joint position and joint compliance using two actuators. Moving both actuators in the same direction changes the joint position; moving both actuators in opposite directions changes the joint compliance. Many VSAs [47][48][49] have a dedicated joint compliance adjusting actuator (VSA) that is fully responsible for the joint compliance (i.e., the joint compliance depends only on the compliance actuator position), and a dedicated kinematic adjusting actuator (primary actuator) for controlling the kinematic joint position. An example of a joint with separate kinematic and compliance actuators is illustrated in Fig. 2.2. Each link i is connected to link $(i-1)$ by a revolute joint. The kinematic actuator, attached to link $(i-1)$, drives an intermediate body that is elastically coupled to link i. In Fig. 2.2, the spring illustrated is a simple elastic coupling between two points. The compliance actuator illustrated in Fig. 2.2 adjusts the effective joint compliance by controlling the orientation of the elastic element using a motor and ball-screw. This section describes the actuator coordinates and actuator mappings for which each joint has kinematic and compliance adjusting actuators (e.g., [47][48][49]).

The kinematic actuator configuration is given by

$$\boldsymbol{\phi}_p = [\phi_{p_1}, \phi_{p_2}, \ldots, \phi_{p_{n_p}}]^T,$$

where the coordinate ϕ_{p_i} is the kinematic actuator position of joint i. The compliance actuator configuration is given by

$$\boldsymbol{\phi}_c = [\phi_{c_1}, \phi_{c_2}, \ldots, \phi_{c_{n_p}}]^T,$$

where the coordinate ϕ_{c_i} is the compliance actuator position[5] of joint i. The total actuator configuration $\boldsymbol{\phi} \in \Phi$ is

$$\boldsymbol{\phi} = \begin{bmatrix} \boldsymbol{\phi}_p \\ \boldsymbol{\phi}_c \end{bmatrix}.$$

[5]The compliance actuator position ϕ_{c_i} in Fig. 2.2 is set to be on the rotary motor (as opposed to the translation of the spring connection point) so that all actuator positions have consistent units.

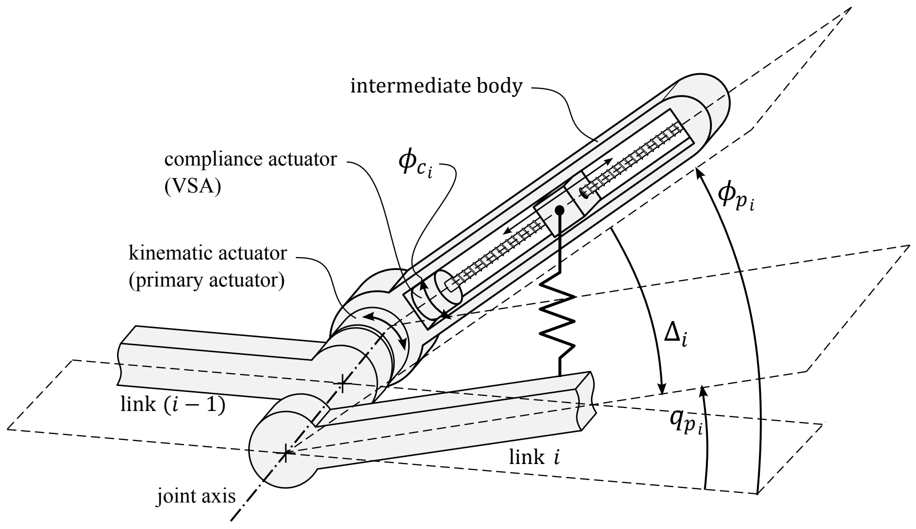


Figure 2.2: Actuator coordinates of joint i. The kinematic actuator position ϕ_{p_i} is the no-load joint position. External loads cause a joint displacement Δ_i resulting in the static equilibrium joint position q_{p_i}. The amount of displacement depends on the joint compliance q_{c_i} that is determined by the compliance actuator position ϕ_{c_i}.

There is a nonlinear function $\boldsymbol{\phi} = \boldsymbol{h}(\mathbf{q})$ that uniquely maps a *joint* configuration to an *actuator* configuration. Because the compliance actuator coordinates are separate from the kinematic coordinates, the actuator mapping function is partitioned as

$$\boldsymbol{h}(\mathbf{q}) = \begin{bmatrix} \boldsymbol{h}_p(\mathbf{q}) \\ \boldsymbol{h}_c(\mathbf{q}) \end{bmatrix}.$$

The kinematic actuator mapping function defines the relationship between the joint configuration $\mathbf{q}$ and the kinematic actuator configuration $\boldsymbol{\phi}_p$. The compliance actuator mapping $\boldsymbol{h}_c$ defines the relationship between the joint configuration $\mathbf{q}$ and the compliance actuator position $\boldsymbol{\phi}_c$. These mappings are described in detail below.

2.3.2 Kinematic Actuator Mapping

In the absence of external loads, kinematic joint positions and kinematic actuator positions are the same. However, gravitational loads are normally present and may cause significant joint deflection from the commanded position, i.e., $\boldsymbol{\phi}_p - \mathbf{q}_p \neq \mathbf{0}$. Commanded kinematic joint positions may be compensated for known loads (e.g., gravity) such that the deflected joint position is the desired position. At static equilibrium

$$\mathbf{K}_q[\boldsymbol{\phi}_p - \mathbf{q}_p] = \boldsymbol{g}(\mathbf{q}_p), \tag{2.22}$$

where $\mathbf{K}_q$ is the joint stiffness matrix and $\boldsymbol{g}(\mathbf{q}_p)$ is a vector of applied torques at the manipulator joints. For gravitational loads, the torque at each joint from the load is a function of the joint configuration determined by the link mass and moment arm.

When joint stiffness is independent of joint deflection, the kinematic actuator command for gravity compensation is [10]

$$\boldsymbol{\phi}_p = \mathbf{q}_p + \mathbf{K}_q^{-1}\boldsymbol{g}(\mathbf{q}_p). \tag{2.23}$$

Because $\mathbf{K}_q^{-1} = \mathbf{C}_q = \text{diag}(\mathbf{q}_c)$, both $\mathbf{q}_c$ and $\mathbf{q}_p$ are used in the kinematic actuator mapping function given by

$$\boldsymbol{h}_p(\mathbf{q}) = \mathbf{q}_p + \mathbf{C}_q\boldsymbol{g}(\mathbf{q}_p), \tag{2.24}$$

Unlike the gravity compensation method used in [10], because both motion and compliance planning are resolved here in joint space, the problem of finding the inverse function $\boldsymbol{h}_p^{-1}$ is unnecessary.

2.3.3 Compliance Actuator Mapping

The torque experienced at each VSA may be described by $\tau_{\text{VSA}_i}(\phi_{c_i}, \Delta_i)$, a nonlinear function of the VSA actuator coordinate ϕ_{c_i}, and the link deflection $\Delta_i = \phi_{p_i} - q_{p_i}$. The shape of the function is determined by the VSA design.

If the torque-deflection relationship is modeled as linear such that the stiffness does not change significantly with link deflection Δ_i, then the joint stiffness can be described by the compliance actuator position alone: $k_i(\phi_{c_i})$.

For VSA designs in which the compliance property is independently controlled, when the VSA has a strictly increasing (or decreasing) compliance profile: $q_{c_i} = 1/k_i(\phi_{c_i})$, there exists an inverse function h_{c_i} such that

$$\phi_{c_i} = h_{c_i}(q_{c_i}). \tag{2.25}$$

An exponential [49] compliance versus actuation profile allows a VSA to attain a very large range of elastic behaviors with only a small change in actuator position. The joint compliance is given by

$$q_{c_i} = c_0 \exp(\xi \phi_{c_i}), \tag{2.26}$$

where ξ and c_0 are constants. The compliance actuator position, then, is given by

$$h_{c_i}(q_{c_i}) = \frac{1}{\xi} \ln\left(\frac{q_{c_i}}{c_0}\right). \tag{2.27}$$

Because each VSA is independently controlled, the compliance actuator mapping function is

$$\boldsymbol{h}_c(\mathbf{q}) = [h_{c_1}(q_{c_1}), h_{c_2}(q_{c_2}), \ldots, h_{c_{n_p}}(q_{c_{n_p}})]^T. \tag{2.28}$$

2.3.4 Actuator Jacobian

Recall, that in our approach, joint motion is resolved by finding the minimum norm actuator motion that realizes the desired task space motion and compliance adjustment. Below, the pseudoinverse weighting matrix used in this redundancy resolution (RR) criterion is constructed from the actuator Jacobian.

Assuming the mapping function $\boldsymbol{h}(\mathbf{q})$ is continuous and differentiable, the linear mapping from the joint tangent space to the actuator tangent space is given by

$$\dot{\boldsymbol{\phi}} = \mathbf{J}_\phi \dot{\mathbf{q}}, \tag{2.29}$$

where $\mathbf{J}_\phi = \partial \boldsymbol{h}(\mathbf{q})/\partial \mathbf{q}$ is the actuator Jacobian matrix. The actuator Jacobian can be evaluated column-by-column, taking the partial derivative on the actuator mapping function with respect to each joint variable. If the actuator kinematic and compliance variables are independent, the actuator Jacobian is partitioned as

$$\mathbf{J}_\phi = \begin{bmatrix} \frac{\partial \boldsymbol{h}_p(\mathbf{q})}{\partial \mathbf{q}_p} & \frac{\partial \boldsymbol{h}_p(\mathbf{q})}{\partial \mathbf{q}_c} \\ \frac{\partial \boldsymbol{h}_c(\mathbf{q})}{\partial \mathbf{q}_p} & \frac{\partial \boldsymbol{h}_c(\mathbf{q})}{\partial \mathbf{q}_c} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{\phi_{pp}} & \mathbf{J}_{\phi_{pc}} \\ \mathbf{J}_{\phi_{cp}} & \mathbf{J}_{\phi_{cc}} \end{bmatrix}. \tag{2.30}$$

Note that, for the VSAs considered, $\boldsymbol{h}_c$ only uses *compliance* joint variables as inputs. As such, the partition $\mathbf{J}_{\phi_{cp}} = \mathbf{0}$. This contrasts with the task Jacobian where the diagonally opposite component $\mathbf{J}_{pc} = \mathbf{0}$.

The redundancy resolution criterion used to minimize the norm of the actuator velocities is equivalent to minimizing the quadratic objective function:

$$G_{\text{inst}} = \frac{1}{2}\dot{\boldsymbol{\phi}}^T\dot{\boldsymbol{\phi}}. \tag{2.31}$$

Substituting (2.29) into (2.31) yields

$$G_{\text{local}} = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{J}_\phi^T\mathbf{J}_\phi\dot{\mathbf{q}}. \tag{2.32}$$

Note that (2.32) and (2.4) have the same form. As such, the pseudoinverse weighting matrix is selected to be

$$\mathbf{W}(\mathbf{q}) = \mathbf{J}_\phi^T\mathbf{J}_\phi. \tag{2.33}$$

This selection of the weighting matrix minimizes the norm of the *actuator* velocity, while resolving the inverse kinematics in *joint* tangent space.

The compliance extended weighted pseudoinverse is calculated by substituting (2.33) and (2.13) into (2.6). This weighted pseudoinverse is used to solve the *instantaneous* compliance extended RIK problem (RIKC problem). The procedure for integrating the instantaneous RIKC solution for the whole ($t = 0 \rightarrow 1$) task manipulation path is provided below.

2.4 Instantaneous Manipulation Planning Approach

Passive compliance control of redundant serial manipulators is implemented by executing an actuator path containing the kinematic and compliance actuator positions for producing the desired task manipulation path. In this section, a basic numerical procedure for generating the actuation path is provided.

A summary of the weighted pseudoinverse RIK resolution approach is depicted in Fig. 2.3. Three configuration spaces are relevant: 1) task space X, 2) joint space Q, and 3) actuator space Φ. The desired task manipulation path $\boldsymbol{x}(t) \in X$, including both position and compliance, is provided as input to the procedure. The actuator path $\boldsymbol{\phi}(t) \in \Phi$, needed for implementing passive compliance control, is the output of the procedure.

If the manipulator were non-redundant, the actuator path could be directly calculated as $\boldsymbol{\phi}(t) = \boldsymbol{h}(\boldsymbol{f}^{-1}(\boldsymbol{x}(t)))$. For redundant manipulators considered here, a direct mapping $\boldsymbol{f}^{-1} : X \rightarrow Q$ from the task space to joint space does not exist; $\mathbf{J}^{-1}$ does not exist either. However, an admissible joint motion that produces the desired task motion is identified with the weighted pseudoinverse $\mathbf{J}^\dagger$ which resolves redundancy by minimizing the actuator velocity norm. Joint space

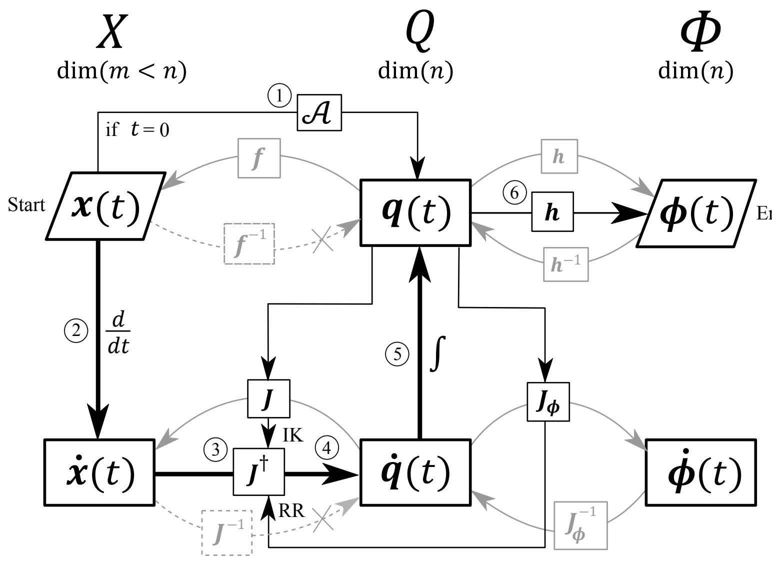


Figure 2.3: Instantaneous manipulation planning approach using the weighted pseudoinverse $\mathbf{J}^{\dagger}$ at a single instant in time. User provided input is $\boldsymbol{x}(t)$ and procedure output is $\boldsymbol{\phi}(t)$. Thick straight arrows show the integration cycle to generate the joint path $\boldsymbol{q}(t)$. Solid straight arrows show necessary calculations. Solid curved arrows represent mappings between spaces, whereas dashed curved arrows with an $\times$ indicate non-existent mappings between spaces.

Q serves as an intermediary between task space and actuator space in which the inverse kinematics (IK) and redundancy resolution (RR) problems are resolved at the differential level using the weighted pseudoinverse.

The instantaneous manipulation planning approach uses the following steps:

1. Determine an admissible joint configuration $\mathbf{q}_0 \mid \boldsymbol{f}(\mathbf{q}_0) = \boldsymbol{x}(t=0)$, using an analytical process [22] [23] [24][25] [26] [27] or search algorithm (denoted by $\mathcal{A}$ in Fig. 2.3).
2. Differentiate the task manipulation path with respect to normalized time to identify the desired task motion $\dot{\boldsymbol{x}}(t)$.
3. Evaluate the compliance extended Jacobian $\mathbf{J}$ and the weighting matrix $\mathbf{W} = \mathbf{J}_{\boldsymbol{\phi}}^T \mathbf{J}_{\boldsymbol{\phi}}$ at the current joint configuration. Evaluate the weighted pseudoinverse matrix $\mathbf{J}^{\dagger}$ using (2.6).
4. Identify the instantaneously optimal joint motion $\dot{\boldsymbol{q}}(t)$ using (2.5).
5. Numerically integrate the joint motion using a small step size in t.
6. Determine the actuator path using the actuator mapping: $\boldsymbol{\phi}(t) = \boldsymbol{h}(\boldsymbol{q}(t))$.

The integration loop (Steps 2-5) repeats, drawing information from the task manipulation path $\dot{\boldsymbol{x}}(t)$ each cycle until $t = 1$. Steps 2-5 describe the execution of (2.2) to generate the joint manipulation path.

The IK and RR problems are resolved in joint space rather than actuator space because $\boldsymbol{h}^{-1} : \Phi \rightarrow Q$ cannot be computed directly when gravitational loads are present; it may be approximated, however, by iterative calculation [10]. Evaluating the Jacobian $\mathbf{J}_{\phi}^{-1} = \partial \boldsymbol{h}^{-1}(\boldsymbol{\phi})/\partial \boldsymbol{\phi}$ is even more difficult. This difficulty prevents effective gravity compensation when the IK problem is resolved in the actuator space. The joint motion and *stiffness* resolution method [32] resolves the IK problem in the actuator space, but does not address gravity compensation (or redundant kinematics as stated previously).

2.4.1 Additional Details

The basic instantaneous manipulation planning approach, shown in Fig. 2.3, may exhibit drift in the task manipulation path due to the finite step size used in the integration. Drift may be mitigated by using an adaptive time step, or by using the forward map $\boldsymbol{f}$ to evaluate task error and correct the joint configuration using the Newton-Raphson method. The task error at each time instance is evaluated as

$$x_{\text{err}} = (\boldsymbol{f}(\mathbf{q}) - \boldsymbol{x}(t))^T (\boldsymbol{f}(\mathbf{q}) - \boldsymbol{x}(t)), \tag{2.34}$$

where $\mathbf{q}$ is the current joint configuration at time t.

The algorithm presented in Fig. 2.3 does not address joint limits. Joint limits, however, *cannot* be ignored for passive compliance control. Every joint must have a positive compliance value regardless of the VSA design. The domain of q_{c_i} is $(-\infty, \infty)$ but the feasible subset is given by $q_{c_i} \in [c_{\min}, c_{\max}]$, where $0 < c_{\min} < c_{\max} < \infty$ are the minimum and maximum compliances afforded by the VSA. Although not presented in Fig. 2.3, joint limits are addressed using the SNS algorithm [42], where redundancy is utilized to avoid joint limits only when the pseudoinverse method would produce a limit-violating joint motion. The SNS algorithm is used between Steps 3 and 4. The SNS algorithm was modified to accommodate any positive definite weighting matrix $\mathbf{W}$, with this modification in $\mathbf{J}_{\text{SNS}}^{\dagger}$:

$$\mathbf{J}_{\text{SNS}}^{\dagger} = \mathbf{S}\mathbf{W}^{-\frac{1}{2}}(\mathbf{J}\mathbf{W}^{-\frac{1}{2}}\mathbf{S})^{\#}, \tag{2.35}$$

where $(\cdot)^{\#}$ is the Moore-Penrose pseudoinverse and $\mathbf{S}$ is the saturation matrix (described in Section 2.1.2).

2.5 3R-VSA Manipulator Example

This section demonstrates the instantaneous manipulation planing approach for a 3R-VSA manipulator performing a particle planar task. Complete descriptions of the manipulator and task are provided. Details for constructing the compliance mapping function, compliance Jacobian, and actuator mapping function are also provided. Two joint manipulation paths are generated using different initial joint configurations.

Consider a 3R-VSA planar manipulator performing the particle planar constrained manipulation task in Fig. 2.4. The manipulator has a total length (reach) of L and a total weight of f_g. The normalized link lengths are dimensionless proportions of the total length: $l_1 = 0.46$, $l_2 = 0.43$, and $l_3 = 0.11$, (anthropomorphic ratios [50]). The gravitational force experienced by each link is expressed as proportions of the total weight f_g. The weight of each link is proportional to its length and its mass center is located at its geometric center as illustrated in Fig. 2.4.

The joint coordinates are given by

$$\mathbf{q} = [q_{p_1},\, q_{p_2},\, q_{p_3},\, q_{c_1},\, q_{c_2},\, q_{c_3}]^T.$$

The kinematic joint positions q_{p_1}, q_{p_2}, and q_{p_3} are expressed in radians; q_{c_1}, q_{c_2}, and q_{c_3} are joint compliances normalized by f_g. The normalized joint compliance matrix is

$$\mathbf{C}_q = \begin{bmatrix} q_{c_1} & 0 & 0 \\ 0 & q_{c_2} & 0 \\ 0 & 0 & q_{c_3} \end{bmatrix}. \tag{2.36}$$

The constrained manipulation task is to slide a block along a rigid surface and contact a rigid wall. The block center is to follow the motion path given by

$$\boldsymbol{x}_p(t) = \begin{bmatrix} 0.3 + 0.4(t) \\ 0.15 \end{bmatrix}, \tag{2.37}$$

where, for normalized time, $t = 0$ at the start and $t = 1$ at the end of the task. The task path is dimensionless, normalized by L.

Because the block is in contact with a horizontal rigid surface, the task compliance in the vertical direction is high to limit the constraint force that may result from error in the commanded vertical position of the task. In moving the block, friction is considered a disturbance impacting the desired motion. The task compliance in the horizontal direction is low to limit the end-effector

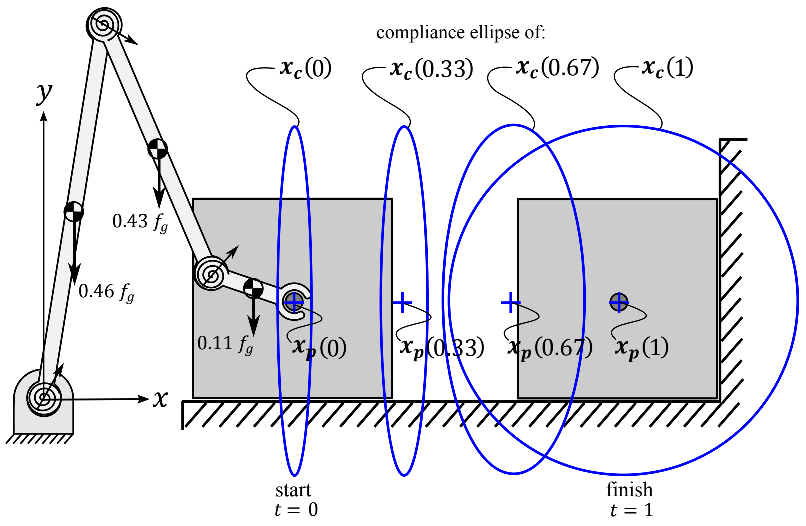


Figure 2.4: 3R-VSA manipulator and particle-planar task manipulation path. Four equally spaced instants in time illustrate the continuous end-effector motion path $\boldsymbol{x}_p(t)$ and the continuous compliance manipulation path $\boldsymbol{x}_c(t)$.

deflection from the planned path due to friction. The compliance in the horizontal direction increases as the block approaches the wall to limit the interaction force from wall contact. In Fig. 2.4, the desired compliance ellipse (see Appendix A) is shown at four instants in time, where the major and minor radii of the ellipse are eigenvalues of the task compliance matrix.

The time varying normalized task compliance matrix (task compliance normalized by L/f_g) is

$$\mathbf{C}_x(t) = \begin{bmatrix} \frac{\lambda_2}{\gamma(t)} & 0 \\ 0 & \lambda_2 \end{bmatrix}, \tag{2.38}$$

and the task configuration vector containing the upper triangular elements is

$$\boldsymbol{x}_c(t) = \begin{bmatrix} \frac{\lambda_2}{\gamma(t)} \\ 0 \\ \lambda_2 \end{bmatrix}, \tag{2.39}$$

where the larger eigenvalue is $\lambda_2 = 0.1$ and $\gamma = \lambda_2/\lambda_1$ is the ellipse aspect ratio that transitions from 10 to 1 according to the profile given by

$$\gamma(t) = 11 - 10^t. \tag{2.40}$$

The goal for implementing passive compliance control is to find an actuator path $\boldsymbol{\phi}(t)$ that executes the task manipulation path $\boldsymbol{x}(t) = [\boldsymbol{x}_p(t)^T, \boldsymbol{x}_c(t)^T]^T$. To do this, the compliance extended RIK problem is instantaneously resolved to generate a joint path $\boldsymbol{q}(t)$.

As a first step in this process, the task forward mapping function $\boldsymbol{f}(\mathbf{q})$ and task Jacobian $\mathbf{J}(\mathbf{q})$, needed in the compliance extended weighted pseudoinverse, are obtained below.

2.5.1 Mapping from Joint Space to Task Space

The task forward mapping functions are easily derived from the distal position of each link relative to its joint position. Let $\mathbf{p}_i = [p_{x_i}, p_{y_i}]^T$ be the position of link-i's distal point (end-effector or next joint) relative to joint-i's position. For the manipulator in this case study:

$$\mathbf{p}_1^T = [l_1 \cos(q_{p_1}), l_1 \sin(q_{p_1})]^T \tag{2.41a}$$

$$\mathbf{p}_2^T = [l_2 \cos(q_{p_1} + q_{p_2}), l_2 \sin(q_{p_1} + q_{p_2})]^T \tag{2.41b}$$

$$\mathbf{p}_3^T = [l_3 \cos(q_{p_1} + q_{p_2} + q_{p_3}), l_3 \sin(q_{p_1} + q_{p_2} + q_{p_3})]^T \tag{2.41c}$$

The forward kinematic mapping function is

$$\boldsymbol{f}_p(\mathbf{q}) = \mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3. \tag{2.42}$$

The kinematic Jacobian of this manipulator is found by taking the partial derivative of (2.42) with respect to each kinematic joint coordinate. Because the forward kinematic function is composed of sin and cos terms, $\partial p_{x_i}/\partial q_k = -p_{y_i}$ and $\partial p_{y_i}/\partial q_k = p_{x_i}$, the resulting Jacobian is

$$\mathbf{J}_{pp} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} (\mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3) & (\mathbf{p}_2 + \mathbf{p}_3) & \mathbf{p}_3 \end{bmatrix}. \tag{2.43}$$

This is also the geometric Jacobian ($\mathbf{J}_{pp} = \boldsymbol{J}_{pp}$).

The forward compliance function $\boldsymbol{f}_c(\mathbf{q})$ is readily obtained by substituting the kinematic Jacobian matrix, (2.43), and the joint compliance matrix, (2.36), into (2.16).

The compliance Jacobian matrix $\mathbf{J}_c$ is constructed one column at a time using (2.20) substituted into (2.19). The partition $\mathbf{J}_{cp}$ is readily evaluated with the partial derivatives of the kinematic Jacobian with respect to the kinematic joint variables:

$$\frac{\partial \mathbf{J}_{pp}}{\partial q_1} = \begin{bmatrix} -(\mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3) & -(\mathbf{p}_2 + \mathbf{p}_3) & -\mathbf{p}_3 \end{bmatrix}, \tag{2.44a}$$

$$\frac{\partial \mathbf{J}_{pp}}{\partial q_2} = \begin{bmatrix} -(\mathbf{p}_2 + \mathbf{p}_3) & -(\mathbf{p}_2 + \mathbf{p}_3) & -\mathbf{p}_3 \end{bmatrix}, \tag{2.44b}$$

$$\frac{\partial \mathbf{J}_{pp}}{\partial q_3} = \begin{bmatrix} -\mathbf{p}_3 & -\mathbf{p}_3 & -\mathbf{p}_3 \end{bmatrix}. \tag{2.44c}$$

Equations (2.44a-2.44c) are used in (2.20), for $k = 1, 2$, and 3 to construct the first three columns of the compliance Jacobian, respectively. The last three columns, $\mathbf{J}_{cc}$, are evaluated with (2.21) and used in (2.20), for $k = 4, 5$, and 6.

The compliance augmented Jacobian relating augmented joint motion to augmented task motion is sufficient for solving the instantaneous compliance extended inverse kinematics problem.

2.5.2 Mappings from Joint Space to Actuator Space

The actuator mapping and actuator Jacobian are defined below for gravity compensation and for joint redundancy resolution according to the minimum norm actuator motion criterion.

From the joint configuration, the kinematic actuator mapping $\boldsymbol{h}_p(\mathbf{q})$ is used to determine the actuator configuration that yields the joint configuration at static equilibrium. At static equilibrium, the external torques at each joint are balanced by the internal torque from the VSA elastic element. The normalized external torque is given by

$$\boldsymbol{g}(\mathbf{q}) = \begin{bmatrix} \frac{p_{x_1}}{2} & \left(p_{x_1} + \frac{p_{x_1}}{2}\right) & \left(p_{x_1} + p_{x_2} + \frac{p_{x_3}}{2}\right) \\ 0 & \frac{p_{x_2}}{2} & \left(p_{x_2} + \frac{p_{x_3}}{2}\right) \\ 0 & 0 & \frac{p_{x_3}}{2} \end{bmatrix} \begin{bmatrix} 0.46 \\ 0.43 \\ 0.11 \end{bmatrix}, \tag{2.45}$$

where $[0.46, 0.43, 0.11]^T$ are the link weights normalized by f_g. The kinematic actuator function $\boldsymbol{h}_p(\mathbf{q})$ is formed by substituting (2.45) and (2.36) into (2.24).

The VSAs of this example have an exponential compliance versus actuation profile like that of [49]. The VSA actuation lower-limit $\phi_{c_{\min}} = 0$ rad corresponds to the normalized joint compliance lower-limit $q_{c_{\min}} = 0.001$ and the VSA actuation upper-limit $\phi_{c_{\max}} = \pi/2$ rad corresponds to the normalized joint compliance upper-limit $q_{c_{\max}} = 10$. The compliance mapping function $h_{c_i}(q_{c_i})$ for each joint is given by (2.27). The constants ξ and c_0 are uniquely determined by the actuation and compliance limits, where $\xi = 5.86$ and $c_0 = 0.001$.

The compliance mapping function is

$$\boldsymbol{h}_c(\mathbf{q}) = \left[\frac{1}{\xi}\ln\left(\frac{q_{c_1}}{c_0}\right), \frac{1}{\xi}\ln\left(\frac{q_{c_2}}{c_0}\right), \frac{1}{\xi}\ln\left(\frac{q_{c_3}}{c_0}\right)\right]^T, \tag{2.46}$$

when each joint is equipped with the same VSA.

The actuator Jacobian is calculated by $\mathbf{J}_\phi = \partial \boldsymbol{h}(\mathbf{q})/\partial \mathbf{q}$ and is evaluated one column at a time similar to the task compliance Jacobian construction. The weighing matrix used in (2.6) is $\mathbf{W} = \mathbf{J}_\phi^T \mathbf{J}_\phi$. This weighing matrix resolves the redundancy by minimizing the norm of the actuator motion.

2.5.3 Results

Instantaneously resolved joint paths using the weighted pseudoinverse are generated with the numerical integration procedure presented in Section 2.4. An adaptive time step is used to ensure that the task error (evaluated using 2.34 at each time step) associated with the numerical integration is below a specified tolerance[6] and that the numerically resolved joint motion accurately follows the true instantaneously optimal path.

The first step in the procedure is to find an admissible initial joint configuration from the initial task configuration using an analytical procedure (or search algorithm) denoted by $\mathcal{A}$ in Fig. 2.3. For the 3R manipulator considered here (described at the beginning of Section 2.5), the set of admissible *kinematic* joint configurations is given by the analytical inverse kinematic solution of a 4-bar mechanism with the end-effector as a grounded joint at $\boldsymbol{x}(0)$. The orientation of the end-effector, given by $\psi = q_{p_1} + q_{p_2} + q_{p_3}$, is the self-motion parameter [51] that resolves the kinematic joint configuration. There are two distinct kinematic self-motion manifolds corresponding to the "elbow-up" pose and "elbow-down" pose, both are parametrized by ψ. The results here consider the "elbow-up" pose. The joint compliance configuration is uniquely determined by the desired task compliance and the kinematic joint configuration; formulas for the joint compliance are presented in [22]. Therefore the admissible initial joint configuration is completely defined by $\boldsymbol{x}(0)$ and ψ, where ψ defines the 1-dimensional set of admissible joint configurations.

[6] $x_{\text{err}} < 1 \times 10^{-16}$ is used for all instantaneously resolved paths in this work.

Consider two different initial joint configurations:

$$\mathbf{q}_{0_a} = [106°, -149°, 42.9°, 0.187, 0.0676, 5.86]^T$$

which corresponds to the self motion parameter $\psi_0 = 0$ rad, and

$$\mathbf{q}_{0_b} = [77.3°, -121°, -136°, 0.111, 0.0840, 7.16]^T$$

which corresponds to the self motion parameter $\psi_0 = \pi$ rad.

From these starting configurations, the joint manipulation paths are generated by integrating (2.2); the resulting paths are shown in Figs. 2.5 and 2.6. Figures 2.5a and 2.6a show equally time-spaced snapshots of the manipulator performing the task, including the initial and final task times. The color of each joint indicates the compliance actuator position ϕ_c for that joint. Because joint 1 does not translate, its compliance actuator position is indicated by the colored circles below the manipulator. Figures 2.5b and 2.6b show the continuous actuator position profiles for the task.

The generated joint manipulation path strongly depends on the initial joint configuration. Starting at $\mathbf{q}_{0_a}$, the manipulation path shown in Fig. 2.5a "pushes" the block to the wall, whereas starting at $\mathbf{q}_{0_b}$, the manipulation path shown in Fig. 2.6a "pulls" the block to the wall. Both actuator paths have smooth profiles and no compliance limits are encountered. The choice of a exponential compliance versus actuation profile for the VSA behaves as a soft joint limit avoidance criterion for the lower compliance limit, where greater actuator motion is required to produce a change in compliance near the lower limit.

Figure 2.5a and 2.6a show the manipulator *exactly* tracking the task compliance *when there is no disturbance.* Disturbance from uncharacterized friction would cause displacement of the end-effector from the commanded task position. The joint configuration, would also be displaced, yielding a slightly different task compliance because task compliance depends on joint kinematics. Because the direction of undesirable forces is known for this task, the *task* compliance is designed to be stiff in the direction of motion to keep the end-effector displacement small (as stated in this example).

Small end-effector position error generally corresponds to small joint position error. However, some joint configurations may result in a relatively large joint displacement for a small end-effector displacement. Consider joint 3 at the beginning of the task in Fig. 2.5a, where friction provides a "compressive load" on link 3. Because joint 3 has high compliance, a buckling

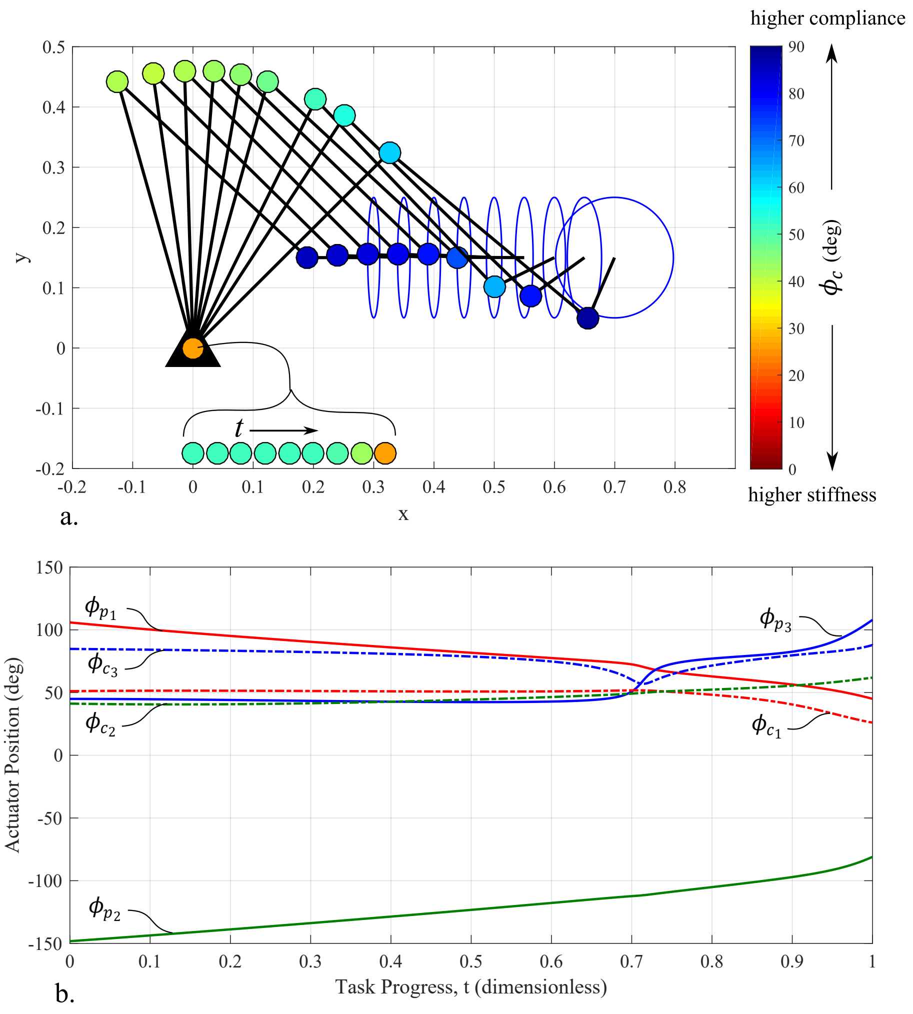
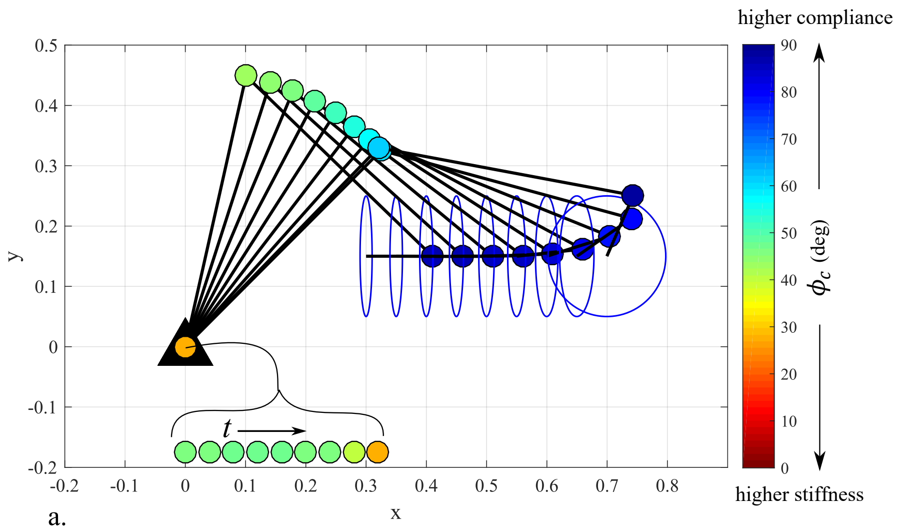


Figure 2.5: 3R-VSA joint manipulation path, instantaneously generated starting at $\mathbf{q}_{0_a}$. a). Stroboscopic image of the manipulator performing the task. The color of each joint indicates the compliance actuator position. The compliance ellipse is shown at each snap-shot. b) Position profiles of each joint's kinematic and compliance actuator.

phenomenon may occur and cause significant joint deflection and therefore significant task compliance error. The manipulation planning approach does not take this into account. However, the buckling phenomenon is not present for the "pulling" scenario in Fig. 2.6.

2.6 5R-VSA Manipulator Example

This section demonstrates the instantaneous manipulation planning approach for a 5R-VSA manipulator. Two tasks are considered, a particle planar task, and a general planar task. All joint paths are generated using the same starting joint configuration. Two joint paths are



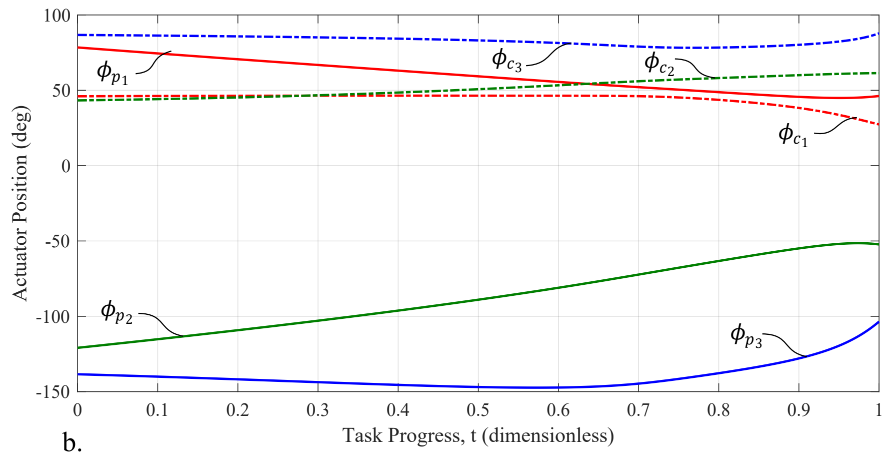


Figure 2.6: 3R-VSA joint manipulation path, instantaneously generated starting at $\mathbf{q}_{0_b}$. a). Stroboscopic image of the manipulator performing the task. The color of each joint indicates the compliance actuator position. The compliance ellipse is shown at each snap-shot. b) Position profiles of each joint's kinematic and compliance actuator.

generated for the general planar task using different redundancy resolution criteria.

Consider a 5R-VSA serial manipulator performing a task in the horizontal plane (no gravity compensation required). The manipulator has a total length (reach) of L. The normalized link lengths are dimensionless proportions of the total length: $l_1 = 0.4$, $l_2 = 0.3$, $l_3 = 0.15$, $l_4 = 0.1$, and $l_5 = 0.05$. The VSAs of this manipulator are the same used in Section 2.5. There are $n_p = 5$ kinematic joint coordinates, and $n_c = 5$ compliance joint coordinates; the total number of joint coordinates is $n = 10$. The total joint configuration is

$$\mathbf{q} = [q_{p_1}, q_{p_2}, q_{p_3}, q_{p_4}, q_{p_5}, q_{c_1}, q_{c_2}, q_{c_3}, q_{c_4}, q_{c_5}]^T.$$

Consider the linear transitioning task path:

$$\boldsymbol{x}(t) = (1-t)\mathbf{x}_0 + t\mathbf{x}_f \tag{2.47}$$

where $\mathbf{x}_0$ is the end-effector kinematic and compliance configuration at $t = 0$, and $\mathbf{x}_f$ is the end-effector kinematic and compliance configuration at $t = 1$.

A particle planar task has $m_p = 2$ kinematic task coordinates, $m_c = 3$ compliance task coordinates, and $m = 5$ total task coordinates; the degree of redundancy is $r = n - m = 5$. The total task configuration is given by

$$\mathbf{x} = [x, y, C_{x_{11}}, C_{x_{12}}, C_{x_{22}}]^T,$$

where (x, y) are the coordinates of the end-effector position relative to the base and the compliance coordinates are elements of the 2×2 compliance matrix:

$$\mathbf{C}_x = \begin{bmatrix} C_{x_{11}} & C_{x_{12}} \\ C_{x_{12}} & C_{x_{22}} \end{bmatrix}. \tag{2.48}$$

The particle planar task considered here has an initial configuration:

$$\mathbf{x}_0 = [0.1, 0, \pi/2, 0.1, 0, 0.1, 0, 0, 10]^T$$

and a final configuration:

$$\mathbf{x}_1 = [0.5, 0, 0.05, 0, 0.05, \pi/2, 0, 0, 10]^T.$$

A general planar task has $m_p = 3$ kinematic task coordinates, $m_c = 6$ compliance task coordinates, and $m = 9$ total task coordinates; the degree of redundancy is $r = n - m = 1$. The total task configuration is given by:

$$\mathbf{x} = [x, y, C_{x_{11}}, C_{x_{12}}, C_{x_{22}}, \psi, C_{x_{13}}, C_{x_{23}}, C_{x_{33}}]^T,$$

where ψ is the orientation of the end-effector relative to the x-axis. The compliance coordinates are elements of the 3×3 compliance matrix:

$$\mathbf{C}_x = \begin{bmatrix} C_{x_{11}} & C_{x_{12}} & C_{x_{13}} \\ C_{x_{12}} & C_{x_{22}} & C_{x_{23}} \\ C_{x_{13}} & C_{x_{23}} & C_{x_{33}} \end{bmatrix}, \tag{2.49}$$

where $C_{x_{33}}$ is the rotational compliance.

The general planar task considered here has initial and final configurations:

$$\mathbf{x}_0 = [0.1, 0, \pi/2, 0.1, 0, 0.1, 0, 0, 10]^T$$

and

$$\mathbf{x}_1 = [0.5, 0, \pi/2, 0.05, 0, 0.05, 0, 0, 10]^T,$$

where there is no coupling between rotational and translational compliance.

Two redundancy resolution criteria are used. The first criterion is minimizing the actuator velocity norm using the weighting matrix constructed from the actuator Jacobian. The second criterion is minimizing the joint velocity norm using an identity weighting matrix. The second criterion is used to show that instantaneous resolution methods need to account for joint limits and that instantaneous resolution methods can fail to complete the task.

2.6.1 Results

An admissible initial configuration was found using the a mechanism construction procedure [24] to find the unique 3R mechanism that realizes the 3 task compliance matrix. Two more joints were added and the joint compliances were set to the lower limit. The Newton-Raphson inverse kinematics method (extended to include compliance) was used to find the nearby joint configuration that maintained the desired task position and compliance.

The initial joint configuration is: $\mathbf{q}_0 = [\mathbf{q}_{p_0}^T, \mathbf{q}_{c_0}^T]^T$, where

$$\mathbf{q}_{p_0} = [62.7°, -137°, -46.0°, -68.4°, -82.7°]^T,$$

and

$$\mathbf{q}_{c_0} = [4.85, 0.654, 0.337, 3.76, 0.396]^T.$$

This joint configuration is used as the initial configuration for both the particle planar task and the general planar task.

With $\mathbf{q}_0$ as the starting configuration, three joint manipulation paths are generated: 1) a joint manipulation path satisfying the *particle* planar task using the minimum *actuator* velocity norm criterion, 2) a joint manipulation path satisfying the *general* planar task using the minimum *actuator* velocity norm criterion, and 3) a joint manipulation path satisfying the *general* planar task using the minimum *joint* velocity norm criterion. The joint manipulation paths are shown in Figs. 2.7, 2.8, and 2.9, respectively. Each figure show equally time-spaced snapshots of the manipulator performing the task. The color of each joint indicates the compliance actuator position ϕ_c for that joint. The task compliance of of each task manipulation path is tracked exactly. The realized particle planar compliance ellipse is shown at each instance. Rotational compliance components are not illustrated, but were obtained for manipulation paths in Figs. 2.8, and 2.9. Each figure also shows the continuous actuator position profiles for the task. The actuator profiles are separated into different plots to better show actuator position change with time.

As discussed in the 3R-VSA manipulation paths, minimizing the *actuator* norm for an exponential stiffness versus actuation relation behaves as a soft-joint limit criterion because greater actuator motion is required to produce a change in compliance near the lower limit of the selected VSA design. The joint manipulation paths in Figs. 2.7 and Fig. 2.8, generated using the minimum *actuator* norm criterion, do not encounter joint limits. However, minimizing the *joint* velocity norm does not demonstrate the soft-joint limit avoidance behavior. The joint manipulation path in Fig. 2.9 encounters a joint limit at $t = 0.037$ where joint-3's compliance reaches its minimum value. At this point, the SNS algorithm [52] uses redundancy to remain inside the joint limits instead of just minimizing the joint velocity norm. At time $t = 0.570$, the optimal joint velocity moves the joint configuration away from the saturation limit.

The joint manipulation path Fig. 2.9 terminates prematurely at time $t = 0.659$, at which point there is no admissible joint motion that advances the task. Figure 2.10 shows the condition number of the total Jacobian matrix over the joint manipulation path. The rapid increase in the condition number indicates that the total Jacobian matrix approaches a singularity. Although, the deired task manipulation path can be realized (e.g., Fig. 2.8), the path started in Fig. 2.9 fails because the minimum joint velocity norm criterion guided the instantaneously generated joint path into a problematic region in the RIKC solution space.

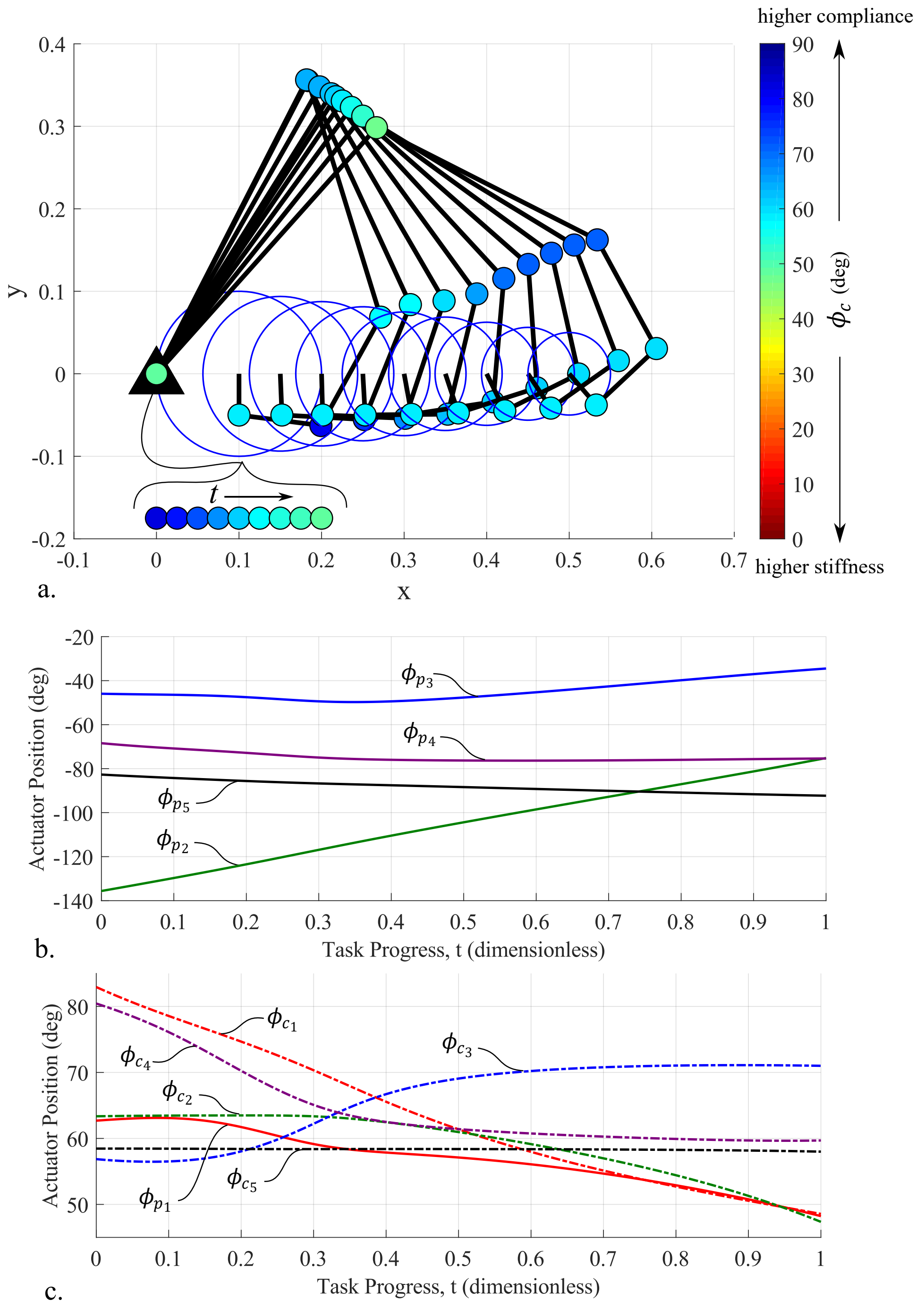
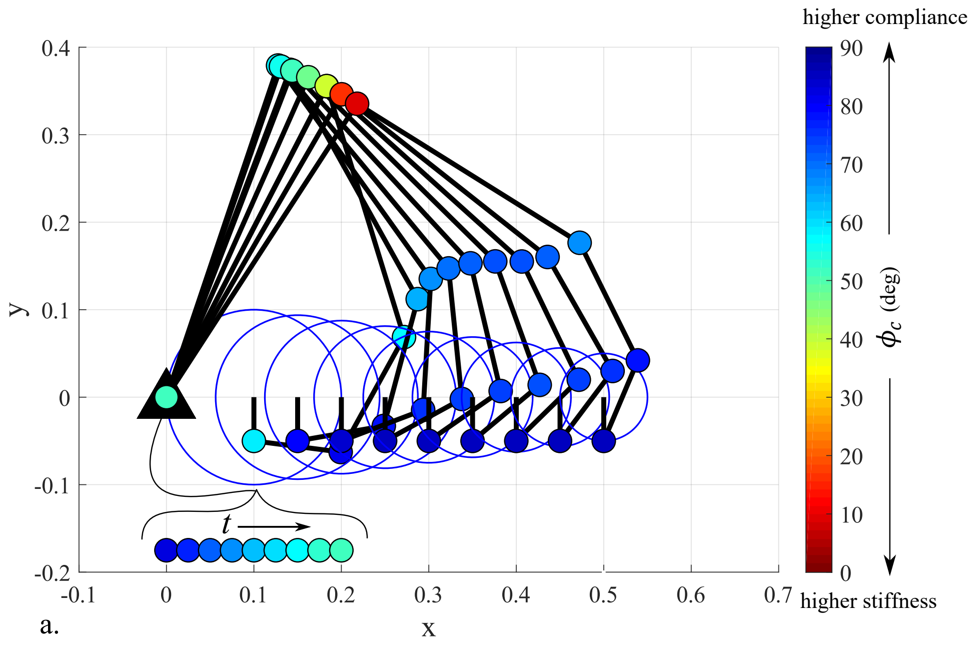
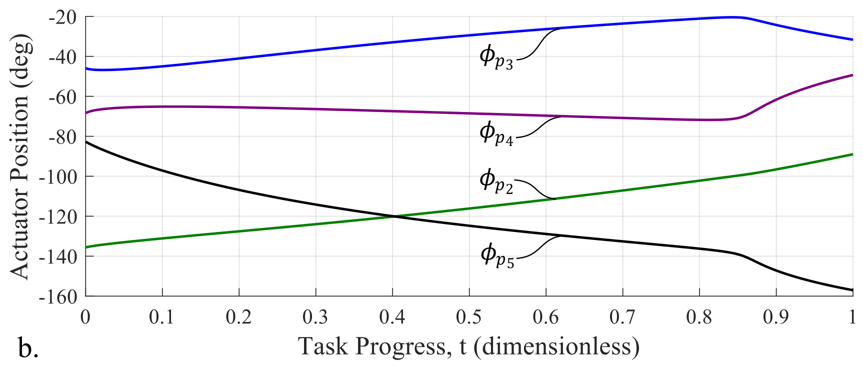


Figure 2.7: 5R-VSA joint manipulation path satisfying a particle planar task ($r = 5$), instantaneously generated using the minimum actuator velocity norm criterion. a). Stroboscopic image of the manipulator performing the task. The color of each joint indicates the compliance actuator position. The translational compliance ellipse is shown at each snap-shot. b) and c) Position profiles of each joint's kinematic and compliance actuator.





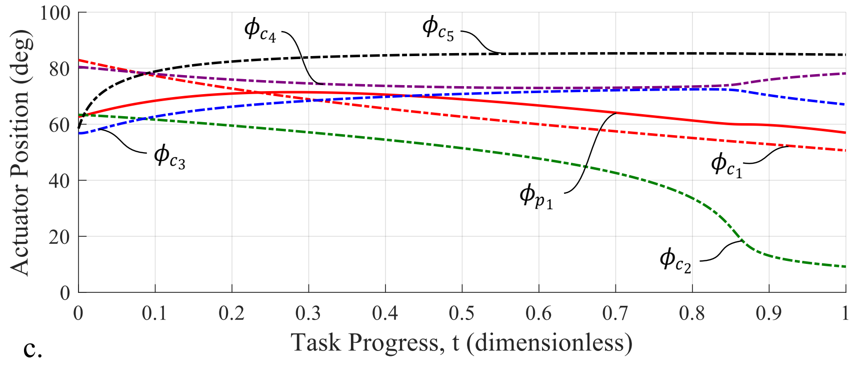


Figure 2.8: 5R-VSA joint manipulation path satisfying a general planar task ($r = 1$), instantaneously generated using the minimum actuator velocity norm criterion. a). Stroboscopic image of the manipulator performing the task. The color of each joint indicates the compliance actuator position. The translational compliance ellipse is shown at each snap-shot. b) and c) Position profiles of each joint's kinematic and compliance actuator.

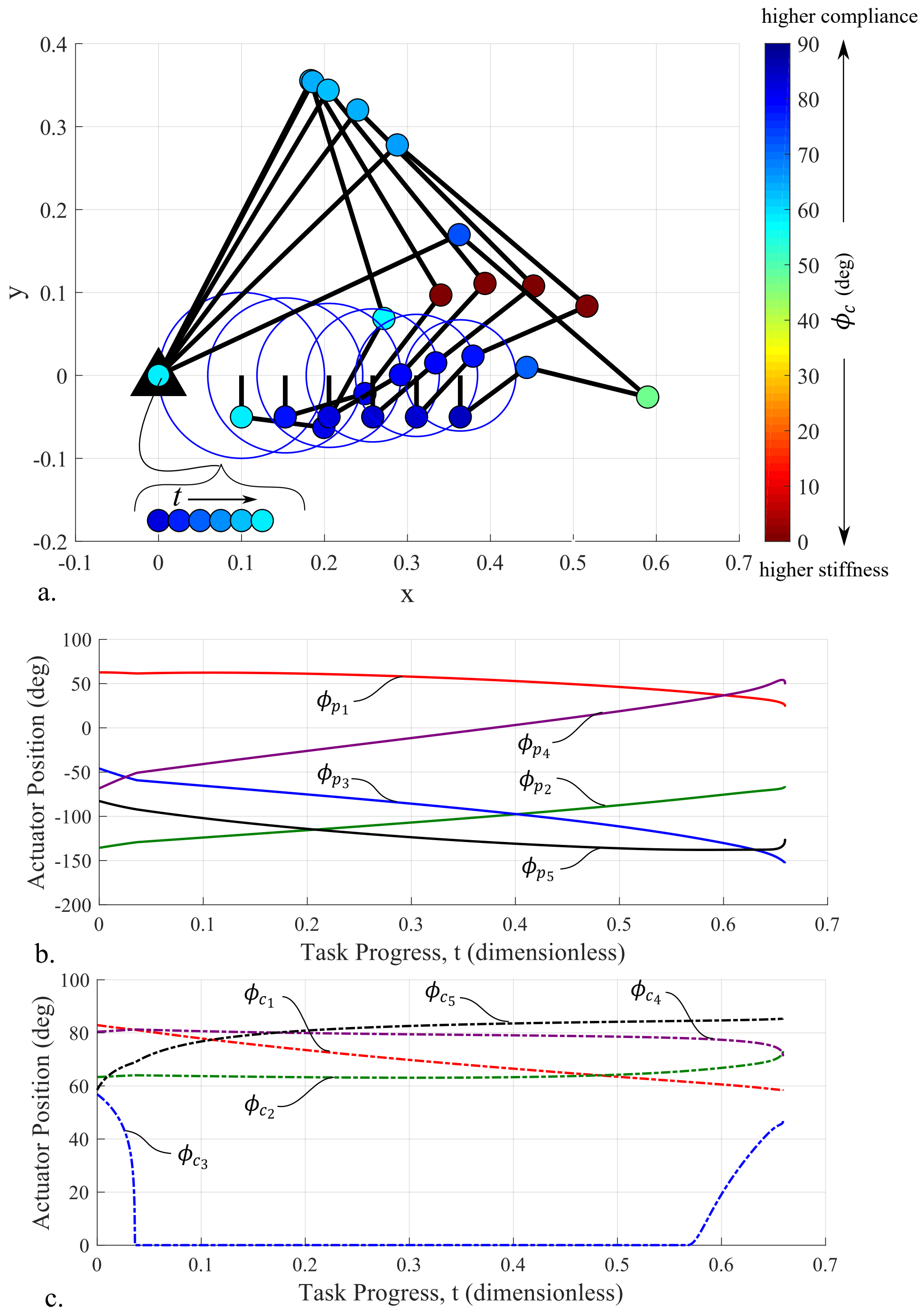


Figure 2.9: 5R-VSA joint manipulation path satisfying a general planar task ($r = 1$), instantaneously generated using the minimum joint velocity norm criterion. a). Stroboscopic image of the manipulator performing the task. The color of each joint indicates the compliance actuator position. The translational compliance ellipse is shown at each snap-shot. b) and c) Position profiles of each joint's kinematic and compliance actuator.

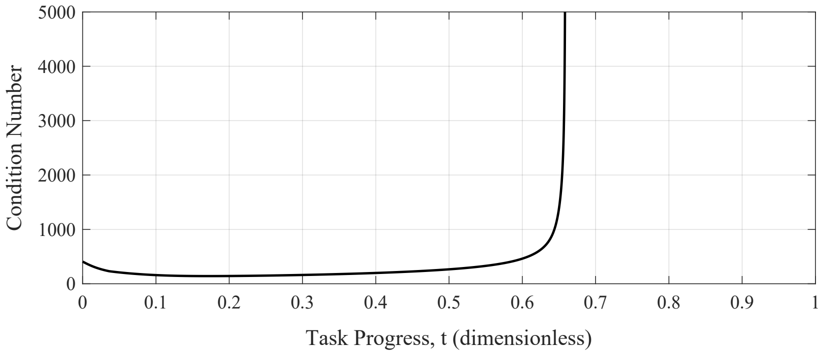


Figure 2.10: Condition number of the total Jacobian over joint task manipulation path in Fig. 2.9.

2.7 Chapter Summary

Passive compliance control is a strategy for regulating the contact forces present in constrained manipulation tasks. Here, a desired task manipulation path consisting of desired robot end-effector positions and end-effector compliances is implemented directly (without active control using sensor feedback) using redundant serial manipulators with real-time adjustable joint compliance. Actuator velocities were minimized to find the appropriate joint positions and joint compliances for implementing the task manipulation path.

Joint manipulation paths that produce the desired time-varying end-effector position and complinace were obtained by extending a standard redundant inverse kinematics (RIK) instantaneous resolution approach to include compliance (RIKC). Joint compliances were resolved simultaneously with the joint kinematics at the velocity level using the weighted pseudoinverse of the total Jacobian matrix (includes the kinematic Jacobian and the compliance Jacobian). Redundancy was resolved using the minimum norm actuator motion criterion, where the actuator coordinates include kinematic actuators and compliance actuators. Gravity compensation and joint limits were also addressed.

The instantaneous RIKC resolution method applies to any number of degrees of redundancy. Instantaneously generated joint paths depend on the initial joint configuration, the selected optimization criterion, and the presence of joint limits. Instantaneously generated joint paths can be generated in real-time, but may fail to find a complete solution to the RIKC path planning problem even when a solution exists.

CHAPTER 3

OPTIMAL GLOBAL RESOLUTION OF THE REDUNDANT INVERSE KINEMATIC PATH PLANNING PROBLEM WITH ONE DEGREE OF REDUNDANCY

The redundant inverse kinematic and compliance (RIKC) framework presented in Chapter 2 used an instantaneous (velocity-based) resolution method to identify joint manipulation paths for executing passive compliance control. The same framework can also be used with global resolution methods for identifying an optimal joint manipulation path for passive compliance control. Current global resolution methods do not identify the best joint path over the entire space of admissible joint paths, but only within a neighborhood of admissible joint paths. This limitation is not widely known. This chapter first describes the full scope of the optimal redundant inverse kinematic (RIK) path planning problem and the relevant prior work. The remainder of the chapter describes and demonstrates a new algorithm that identifies the globally optimal (best) joint path for RIK path planning problems with one degree of redundancy. Because this is a contribution to the state of the art in the RIK (kinematics only) path planning problem, compliance is not discussed here.

3.1 Introduction

Global resolution of the RIK path planning problem has been studied since the 1980's [53] [54] [55] [56] [57] [58]. Unfortunately, the terms used to describe the method of RIK resolution conflict with terms used to describe conventional optimization. Prior work on finding "globally optimal paths" present *global resolution* methods for finding a *local* minimum of the global cost function. These methods, in general, do not yield the global minimum. The existence of multiple locally optimal paths for the RIK path planning problem has been demonstrated in case studies [56], in which multiple locally optimal paths are shown to exist in different homotopy classes. Joint paths in different homotopy classes cannot be continuously deformed into each other without violating the task path or the boundary conditions. The existence of multiple homotopy classes complicates the problem of identifying the joint path that globally minimizes the global cost function.

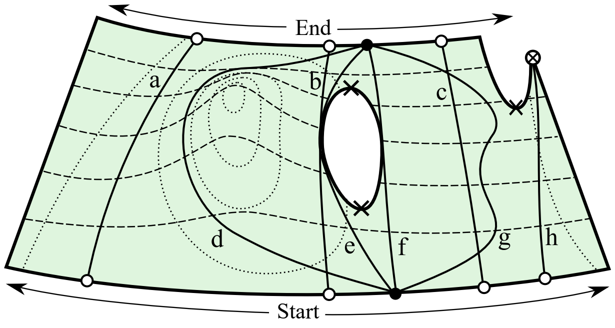


Figure 3.1: An RIK solution space with two complete homotopy classes separated by a single hole. Solid lines with circle endpoints are joint paths, dashed lines are self-motion paths, and dotted lines are contours showing surface topography.

3.1.1 Optimal Paths and Homotopy Classes

Locally optimal joint paths depend on the feasible RIK solution space and the selected boundary conditions. The *RIK solution space* is the set of joint configurations with end-effector configurations on the task path. The RIK solution space is represented as a time-indexed sequence of self-motion manifolds [51], which are sets (or groups of sets) of connected joint configurations that yield the same end-effector configuration. The *feasible* RIK solution space is the subset of realizable joint configurations that are within the hardware limits of the manipulator. A feasible RIK solution space for $r = 1$ is illustrated by the shaded surface in Fig. 3.1a, where dotted contour lines show the surface topography, dashed lines are feasible subsets of self-motion manifolds at various task instances, and solid lines with circle endpoints are alternative joint paths.

The joint path endpoints must be on the self-motion manifolds of the task path endpoints, regardless of the boundary conditions. For fixed boundary conditions, the endpoint joint configurations are user selected based on some other criterion and the optimization is constrained by them. For free (unconstrained) boundary conditions, the endpoint joint configurations are determined by the optimization. In general, locally optimal paths with free boundary conditions have lower global cost values than those with fixed boundary conditions. Paths a-f in Fig. 3.1a are paths that locally minimize the global cost function (corresponding to the length of the joint path), whereas path g is not locally optimal and path h is not a complete solution. The paths with free boundary conditions (a-c) are typically shorter than those with fixed boundary conditions (d-f).

The number of locally optimal joint paths depends on the connectivity of the RIK solution space and on its nonlinearities with respect to the global cost function.

The connection structure of the feasible RIK solution space can be complex. A connected subset of RIK solutions of a single task instance, as the task progresses, may 1) split (bifurcate) into two disjoint subsets or 2) may vanish entirely (prematurely terminate). Bifurcation or premature termination occurs in the RIK solution (feasible and infeasible) space at hyperbolic or saturation singularities [59], respectively. A *singularity* is a joint configuration for which arbitrary task motion is not possible. A hyperbolic singularity corresponds to saddle point in the RIK solution surface where self-motion manifolds split or converge at the singularity. A saturation singularity occurs at workspace boundaries, and is associated with self-motion manifolds reducing to a point before vanishing. Bifurcation and premature termination in the *feasible* RIK solution space is often associated with joint limits. The simple feasible RIK solution space in Fig. 3.1 shows bifurcation points ($\times$) where feasible self-motion subsets split or converge. It also shows a premature termination point ($\otimes$), where a feasible self-motion subset vanishes as the task progresses.

The connectivity of the RIK solution space impacts the connectivity of the infinite set of solution paths. The infinite set of possible joint paths is a collection of homotopy classes. The number of homotopy classes increases with the number of bifurcations. As a progressing joint path approaches a bifurcation point (e.g., $\times$ in Fig. 3.1) it must take one of two “branches” (each side of the saddle or hole). Joint paths that take different branches cannot be continuously deformed into each other. They are in different homotopy classes. For example, the paths on either side of the hole in Fig. 3.1 cannot be deformed into each other without leaving the shaded surface.

When a large number of homotopy classes exist, it is highly unlikely that the globally optimal path will be found without a multi-search strategy. Moreover, the obtained joint path may not be the best path even in its own homotopy class.

Each homotopy class may have multiple locally optimal paths due to cost function nonlinearities in the RIK solution space. This is illustrated in Fig. 3.1, where the hill in the surface induces locally optimal paths on opposite sides of the hill. Unlike the paths separated by the hole, joint paths on either side of the hill (e.g., paths a and b in Fig. 3.1) can be deformed into each other along the feasible self-motion subsets.

Three levels of “optimal paths” are described in this chapter and illustrated by the joint paths in Fig. 3.1:

1. A *locally optimal path* minimizes the global cost function such that any infinitesimal deformation of the path yields a higher global cost value. It is the best path in a

neighborhood of paths (e.g., paths a-c for free boundary conditions and paths d-f for fixed boundary conditions).

2. A *homotopy optimal path* is the locally optimal path with the lowest global cost of all locally optimal paths in its homotopy class. It is the best path in its homotopy class (e.g., paths a and c for free boundary conditions and paths e and f for fixed boundary conditions).
3. A *globally optimal path* is the homotopy optimal path with the lowest global cost of all homotopy optimal paths. It it the best path in the entire optimization domain (e.g., path c for free boundary conditions and path f for fixed boundary conditions).

Prior work in RIK path planning has not identified systematic procedures for identifying the best path when multiple homotopy classes exist or when multiple locally optimal paths exist in a homotopy class.

3.1.2 Prior Work in Global Resolution

Global resolution of the RIK path planning problem is usually framed as an optimal control problem which may be solved “indirectly” [53] [55] [56], “directly” [57] [58], or using dynamic programming [60]. Dynamic programming requires a fixed boundary condition and converges to the constrained globally optimal path, which is sub-optimal compared to the globally optimal path with free boundary conditions.

The optimal control problem, for any type of boundary condition, is solved “indirectly” by seeking to satisfy necessary conditions for optimality given by Pontryagin’s maximum principle [53] or the Euler-Lagrange equation [55] [56]. If the optimal joint path does not encounter a joint limit, it is the solution to a two-point-boundary-value problem. Shooting methods are frequently used to solve these problems, but often suffer from numerical instability when the Euler-Lagrange equation is stiff [61].

A more robust way to solve the optimal control problem, for any type of boundary condition, is to solve it “directly” using nonlinear programming [62] where the joint path is represented by a finite number of parameters (e.g., approximating the joint path with a spline curve with a finite number of nodes [58]). The joint path is iteratively improved by descending the gradient of the cost function with respect to the path parameters. This solution method requires an initial joint path and iteratively deforms the path over the RIK solution space into a locally optimal joint path.

None of these approaches first identifies the set of homotopy classes in which the globally optimal path may exist.

3.1.3 Prior Work in Homotopy Class Identification

Most work in homotopy class identification has addressed tasks without path constraints [63] [64]. The set of homotopy classes can be described using a *roadmap*, a graph with nodes corresponding to manipulator configurations and edges corresponding to feasible joint paths generated with an instantaneous path planner (often referred to as a local path planner). The discrete representation of a joint path is given by a *route*, a sequence of edges connecting nodes, where the start and terminal nodes are configurations corresponding to the start and end of the task. A discrete homotopy relation is used in [63] to simplify a roadmap such that each route corresponds to a joint path in a different homotopy class.

Roadmap nodes are commonly generated by randomly sampling the joint space [65]. These nodes, in general, do not satisfy the task path constraints, but nearby configurations on the task path can be found using Jacobian-based inverse kinematic methods [66] [67], or using rapidly-exploring-random-trees (RRTs) [68] [67] [69] [70]. Note, RRTs use a local planner to build the roadmap outward from existing nodes, whereas traditional roadmaps use an instantaneous planner to find edges between existing nodes. The instantaneous planner is usually a linear motion in the joint space that lifts off the nonlinear RIK solution surface resulting in task error between the nodes. Therefore, the nodes must be close together to limit task error.

3.1.4 Approach

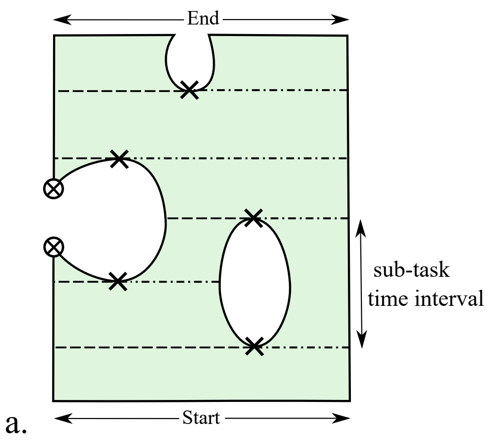
This chapter presents a 5-step algorithm for finding the globally optimal joint path satisfying a specified task path for manipulation with one degree of redundancy. The process involves: 1) decomposing the task into a set of sub-tasks, 2) finding multiple sub-optimal paths for each sub-task, 3) deforming these sub-optimal paths into multiple locally optimal solutions of the sub-tasks, 4) strategically concatenating sub-task solutions to obtain sub-optimal complete solutions, and 5) deforming these sub-optimal complete solutions into locally optimal complete solutions. The algorithm yields a set of many locally optimal paths that are continuous, smooth, and satisfy the equality constraints of the task path. The locally optimal path within this set having the lowest global cost is very likely, but is not guaranteed, to be the globally optimal path.

The algorithm first decomposes the task into sub-tasks by dividing the complete task path at instances when feasible self-motion paths bifurcate (split or converge) or prematurely terminate. Each sub-task RIK solution space has a relatively simple connection structure and is bounded by: 1) self-motion paths associated with the bifurcation points, 2) self-motion paths associated with the complete task endpoints, or 3) premature termination points. The connectivity of these sub-task RIK solution spaces is characterized by a new directed graph called the bifurcation branch roadmap (bb-roadmap). Each node of the bb-roadmap is a self-motion path of a sub-task endpoint (a self-motion path of either a bifurcation point or a complete task endpoint) or a premature termination point. Each edge of the bb-roadmap is a *sub-task homotopy class* (a homotopy class of joint paths that satisfy the sub-task). The number of sub-task homotopy classes in a single sub-task depends on whether the self-motion paths are open paths, or if they are closed manifolds (loops). For RIK solution spaces with open self-motion paths (e.g., Fig. 3.2a), there is only one sub-task homotopy class per sub-task and only one edge connecting the associated nodes (as in Fig. 3.2b). RIK solution spaces with closed self-motion manifolds, however, may have multiple sub-task homotopy classes per sub-task. Each route through the bb-roadmap (sequence of edges connecting endpoint nodes) corresponds to a (complete task) homotopy class and every relevant homotopy class has a corresponding route. If there is no complete route though the bb-roadmap, there is no solution to the RIK path planning problem (no feasible joint path satisfies the complete task path).

The remaining steps of the algorithm alternate between using a path planner based on *instantaneous* RIK resolution to obtain sub-optimal paths within desired sub-task homotopy classes and using a “direct” global RIK resolution method to deform the sub-optimal paths into locally optimal paths. The final step, in which complete paths are deformed into locally optimal paths, requires the most computation time, especially when there are many homotopy classes to investigate. The bifurcation branch algorithm uses an upper/lower bound method to eliminate a large number of homotopy classes that cannot contain the globally optimal path. For problems with many bifurcation points, the set of homotopy classes considered in the final step is reduced by a factor of 100 or more.

3.1.5 Chapter overview

This chapter presents an algorithm called the *bifurcation branch algorithm* (bb-algorithm) for identifying the globally optimal path for the optimal RIK path planning problem involving



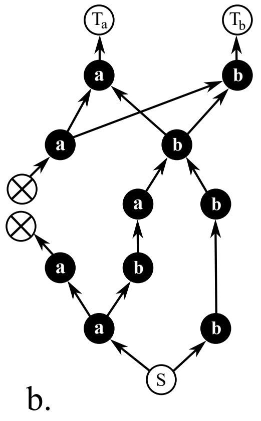


Figure 3.2: An RIK solution space and the bifurcation branch roadmap (bb-roadmap) characterizing its connection structure. a) RIK solution surface with self-motion in the horizontal direction and task progress in the vertical direction. b) Corresponding bifurcation branch roadmap. Each self-motion path of a complete task endpoint correspond to endpoint bb-nodes (white circles). Each self-motion path touching a bifurcation point ($\times$) is split into two separate paths (dashed and dashed-dotted lines). These paths correspond to bifurcation branch bb-nodes (black circles). Premature termination points ($\otimes$) are also used as bb-nodes. Homotopy classes of joint paths connecting sub-task endpoint self-motions correspond to directed edges (arrows).

multiple homotopy classes. Section 3.2 reviews relevant background of the RIK path planning problem. Section 3.3 describes the five steps of the bb-algorithm in greater detail. Section 3.4 describes the bifurcation branch roadmap (bb-roadmap) and identifies the homotopy classes for different RIK solution space structures with closed self-motion manifolds (no joint limits). Section 3.5 describes the instantaneous path planner used to generate sub-optimal joint paths in each homotopy class. Section 3.6 describes the procedure used to deform sub-optimal joint paths into locally optimal paths. Section 3.7 demonstrates the algorithm for a case study in which the number of homotopy classes is *very* high and the globally optimal path cannot be found using traditional methods. Section 3.8 summarizes the results.

3.2 Technical Background

This section reviews the technical background and terminology associated with the optimal redundant inverse kinematic (RIK) path planning problem.

3.2.1 Terminology

Path - A path in a topological space A is a continuous map $\boldsymbol{a}(\rho) : I \to A$, where I is the unit interval $[0,1]$, $\rho \in I$ is the indexing parameter, $\boldsymbol{a}(0)$ is the start point, and $\boldsymbol{a}(1)$ is the terminal point.

Three types of paths are used in this paper:

1. a *task path* $\boldsymbol{x}(t) : I \to X$,
2. a *joint path* $\boldsymbol{q}(t) : I \to Q$, and
3. a *self-motion path* $\boldsymbol{q}(\psi) : I \to Q$.

Task paths and joint paths are both parameterized by normalized time t denoting task progress, but are in different topological spaces; X is the m-dimensional task space and Q is the n-dimensional joint space. Joint paths and self-motion paths are in the same topological space Q, but are parameterized differently. A self-motion path corresponds to a connected set of joint configurations yielding the same end-effector configuration in task space. The self-motion parameter ψ is orthogonal to t.

Path homotopy - A path-homotopy is a continuous mapping $H(t, \psi) : I \times I \to Q$, such that $H(t, 0) = \boldsymbol{q}_0(t)$ and $H(t, 1) = \boldsymbol{q}_1(t)$, where $\boldsymbol{q}_0(t), \boldsymbol{q}_1(t) : I \to Q$ are arbitrary paths with the same endpoints[1]. A path homotopy describes the deformation of the joint path along self-motion paths.

Homotopy Class (of a joint path) - The homotopy class of a joint path $\boldsymbol{q}(t)$ is the set of all joint paths for which a path homotopy to $\boldsymbol{q}(t)$ exists.

3.2.2 Optimal Redundant Inverse Kinematic Path Planning Problem

Recall from Chapter 2, the forward mapping function $\boldsymbol{f}(\mathbf{q}) : Q \to X$ defined the relationship between the joint configuration $\mathbf{q} \in Q$ of the manipulator and the task configuration $\mathbf{x} \in X$ of its end-effector. For redundant manipulators ($n > m$) the mapping is not one-to-one and the RIK path planning problem does not have a unique solution. An admissible joint path can be obtained using instantaneous resolution (Ch. 2). The optimal redundant inverse kinematic path planning problem is to find the unique[2] best joint path, solving:

$$\begin{aligned} \text{min.} \qquad & G_{\text{global}}(\boldsymbol{q}(t)) \\ \text{s.t.} \qquad & \boldsymbol{f}(\boldsymbol{q}(t)) = \boldsymbol{x}(t) \\ & \mathbf{q}_{\min} \le \boldsymbol{q}(t) \le \mathbf{q}_{\max}, \end{aligned} \tag{3.1}$$

[1]The definition of path homotopy extends to paths with "free" endpoints as long as the "free" endpoint is path-connected to the original endpoint. In this context, free endpoints are path connected along self-motion paths.

[2]In RIK path planning problems with symmetry, two or more joint paths may "tie" for best path.

with optional boundary conditions, where $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$ are lower and upper limits of the joint coordinates, and $G_{\text{global}}(\boldsymbol{q}(t))$ is the global cost function.

A common choice for the global cost function is the integral of an instantaneous cost function:

$$G_{\text{global}}(\boldsymbol{q}(t)) = \int_{t=0}^{t=1} G_{\text{inst}}(\mathbf{q}(t), \dot{\mathbf{q}}(t))\, dt. \tag{3.2}$$

Higher order differential terms (e.g., $\ddot{\boldsymbol{q}}$) are not considered in this work.

A common choice of boundary conditions are "fixed" boundary conditions:

$$\boldsymbol{q}(0) = \mathbf{q}_0 \text{ and } \boldsymbol{q}(1) = \mathbf{q}_f$$

in which the initial and final joint configurations are specified joint configurations $\mathbf{q}_0$ and $\mathbf{q}_f$, respectively. If the task path is cyclic, it is common to use periodic boundary conditions:

$$\boldsymbol{q}(1) = \boldsymbol{q}(0) \text{ and } \dot{\boldsymbol{q}}(1) = \dot{\boldsymbol{q}}(0)$$

in which the initial and final joint configurations are the same and the initial and final joint velocities are the same. Using "free" boundary conditions communicates that the optimization (3.1) is not constrained by boundary conditions and the optimization is "free" to select the endpoint joint configurations.

A locally optimal joint path can be found using "direct" methods by deforming an initial joint path (a required input). An initial joint path can be obtained using instantaneous RIK resolution. The number of locally optimal joint paths obtained using global resolution depends on the RIK solution space structure.

3.2.3 RIK Solution Space Structure

The domain of the optimization is the feasible RIK solution space, where the RIK solution space is a sequence of sets of self-motion manifolds for every $\mathbf{x} \in \boldsymbol{x}(t)$ [51] given by

$$\boldsymbol{f}^{-1}(\mathbf{x}) = \bigcup_{i}^{n_s} \mathcal{M}_i(\boldsymbol{\psi}) \tag{3.3}$$

where n_s is the number of disjoint self-motion manifolds $\mathcal{M}_i(\boldsymbol{\psi})$, and $\boldsymbol{\psi}$ is a set of self-motion parameters $\boldsymbol{\psi} = \{\psi_1, \psi_2, \ldots, \psi_r\}$ (r is the degree of redundancy). For RIK problems with one degree of redundancy ($r = n - m = 1$), each self-motion manifold is a 1-dimensional path $\boldsymbol{q}_i(\psi)$.

The portions of the self-motion paths within the joint limits are feasible self-motion paths. The feasible RIK solution space is the collection of all feasible self-motion paths at each instance in time.

Technically, the manifold description of the preimage (3.3) is only valid for regular values, not critical values or coregular values [51], where

- a *regular value* is a task configuration $\mathbf{x}$ for which its preimage does not contain a singularity.
- a *singularity* is a joint configuration $\mathbf{q}_s$ for which the Jacobian matrix is rank deficient, i.e., $\text{rank}(\mathbf{J}(\mathbf{q}_s)) < m$,
- a *critical value* is a task configuration $\mathbf{x}_s$ for which its preimage is a singularity, i.e., $\boldsymbol{f}^{-1}(\mathbf{x}_s) = \mathbf{q}_s$, and
- a *coregular value* is a task configuration $\mathbf{x}_{cr}$ for which its preimage contains both singular and non-singular joint configurations. For $r = 1$, the preimage of a coregular value is a collection of *coregular self-motion paths*, but not manifolds.

The connectivity of self-motion manifolds of different task points is investigated in [51] by identifying homotopy classes *of self-motion manifolds* (as opposed to a homotopy class of joint paths)[3]. The workspace of the manipulator is divided into W-sheets, such that the self-motion manifolds of all task configurations within a W-sheet are in the same homotopy class (i.e., the self-motion manifolds can be continuously deformed into each other). Each W-sheet is a connected set of task configurations bounded by critical or coregular values. Critical values $\mathbf{x}_s$ exist at the workspace boundaries, whereas coregular values $\mathbf{x}_c$ exist at the interface between neighboring W-sheets. The RIK path planning problem is much simpler when the task is inside a single W-sheet for which the number of homotopy classes of joint paths (joint limits aside) is equal to the number of self-motion manifolds of the W-sheet. For task paths that cross coregular values, the self-motion manifolds do not deform continuously over the task, but bifurcate, resulting in a more complicated RIK solution space structure with multiple locally optimal paths.

3.3 Bifurcation Branch Algorithm

The existence of many locally optimal solutions to the RIK path planning problem greatly increases the difficulty of obtaining the globally optimal path. This section describes the

[3]A path homotopy of joint paths describes a continuous deformation of a joint path over self-motion paths. A path homotopy of self-motion paths describes a continuous deformation of a self-motion path over a task path.

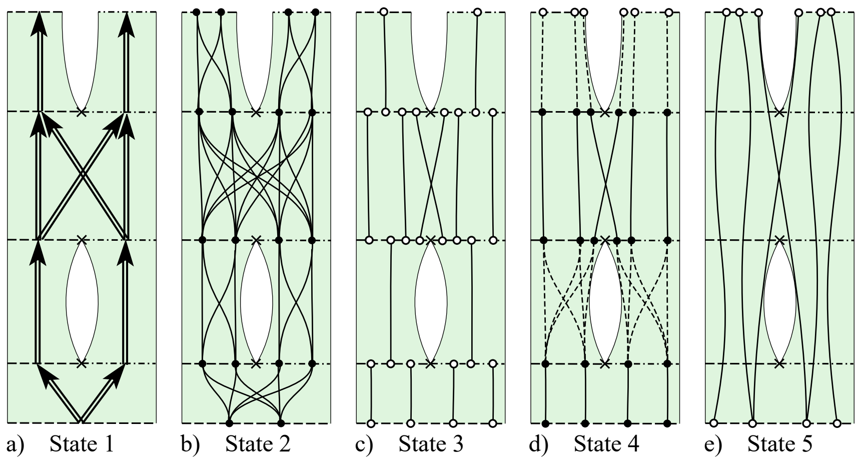


Figure 3.3: The states of the bifurcation branch algorithm. States are shown on a simplified abstract RIK solution space, after each step. The steps are: 1) generate the bb-roadmap, 2) generate a configuration roadmap of initial joint paths. 3) deform initial paths of sub-tasks into locally optimal paths with free boundary conditions, 4) join locally optimal paths together into a refined configuration roadmap, and 5) deform the complete initial paths into locally optimal paths.

bifurcation branch algorithm (bb-algorithm) that overcomes this difficulty for RIK problems with one degree of redundancy.

The algorithm is referred to as the bifurcation branch algorithm because it strategically uses the self-motion paths (branches) associated with the bifurcation points to effectively search the RIK solution space. The algorithm finds the globally optimal path using 5 steps. The steps are illustrated in Fig. 3.3 on a simple RIK solution space with open feasible self-motion paths in the horizontal direction and task progress in the vertical direction.

Step 1: Decompose the task into sub-tasks by dividing the task path at instances when feasible self-motion paths bifurcate (split or converge) or vanish. Points of bifurcation and vanishing associated with singularities are quickly identified numerically using known the geometric conditions of the singularities. Points of bifurcation and vanishing associated with joint limits are quickly identified numerically using a boundary edge path planner (described in Chapter 4.) Each sub-task path has endpoints that correspond to either a bifurcation point, or a task endpoint, or a premature termination point (not present in Fig. 3.3). The feasible RIK solution space of each sub-task is separated into regions covered by a single homotopy class and bounded at the sub-task endpoints by a feasible self-motion path (horizontal dashed or dashed-dotted lines in Fig. 3.3a). Each self-motion path is the largest possible, non-overlapping, feasible self-motion path that does not cross a bifurcation point. The endpoints (bounds) of each feasible self-motion path is either a bifurcation point or a joint configuration at the edge of the feasible subspace of the self-motion manifold. If the sub-task endpoint corresponds to a premature termination point, the feasible self-motion "path" is a single joint configuration that is the premature termination point.

The complete feasible RIK solution space is the union of the sub-task solution spaces. The connection structure is characterized by a new directed graph called the bifurcation branch roadmap (bb-roadmap). Figure 3.3a is a projection of the bb-roadmap onto the solution surface. Roadmap nodes are shown as self-motion horizontal lines. Directed edges are shown as double lined arrows pointing in the direction of forward task progress.

The bb-roadmap is constructed by identifying each bifurcation point and premature termination point and identifying the number and structure of self-motion manifolds (and of the feasible self-motion paths on them) before and after each of these points. The structure of a 1-dimensional self-motion manifold is either open or closed. The number and structure of self-motion manifolds before and after each bifurcation point provides critical information about how the bifurcation impacts the number of homotopy classes of joint paths. Section 3.4 provides this process for RIK solution spaces with one or two closed self-motion manifolds at each regular value task instance. The task instances of bifurcations and premature termination associated with singularities are quickly and deterministically identified using known geometric conditions for kinematic singularities and coregular values of the manipulator. Continuous self-motion paths associated with each bb-node are generated with a self-motion path planner.

If a complete route through the bb-roadmap (a sequence of edges connecting a task start node to a task end node) does not exist, then there is no solution to the RIK path planning problem. If there is one or more complete routes, the globally optimal joint path solving the RIK problem is identified using the following steps.

Step 2: Generate sub-optimal joint paths for each sub-task homotopy class using instantaneous RIK resolution. Conventional instantaneous resolution does not control the terminal endpoint joint configuration and therefore cannot control the homotopy class of the joint path. The bb-algorithm uses a new instantaneous path planner that bi-directionally generates a joint path between two specified endpoint joint configurations. The endpoint joint configurations are sampled from the self-motion paths of the associated bb-nodes. Complete homotopy classes cannot include sub-task homotopy classes associated with premature termination. Therefore, bb-nodes that are premature termination points or bb-nodes connected to a premature termination point by an edge are not relevant. The result of Step 2 is a *configuration roadmap* superimposed on the bb-roadmap, where each node is now a specific joint configuration and each edge is a specific joint path connecting two configuration nodes. The path that joins the specific configurations is generated using a modified instantaneous RIK resolution approach.

Because each sub-task homotopy class can have multiple locally optimal paths, multiple sub-optimal paths are generated by connecting different combinations of sampled node configurations. Figure 3.3b shows each bb-node with two sampled configurations (solid circles), and four initial joint paths (solid lines) in each sub-task homotopy class. Actual implementation uses three or more sampled joint configurations, equally spaced over the self-motion path, with two of the samples very close to the self-motion path bounds (i.e., near feasible limits or near bifurcation points).

Step 3: Deform the sub-optimal joint paths in each relevant sub-task homotopy class (obtained in Step 2) into locally optimal joint paths with free boundary conditions (solid lines with open circle endpoints in Fig. 3.3c). Because the RIK solution subspace corresponding to the sub-task homotopy class has relatively simple structure, with enough sampled points in Step 2, it is very likely that *all* the locally optimal joint paths of sub-tasks are found. The best path among all locally optimal paths in the same sub-task homotopy class is the *sub-task homotopy optimal path*. However, all of these locally optimal paths are important because they capture different regions of the RIK solution space of lower global cost. The globally optimal path of the complete task path will likely pass near some of these regions. This step is important for effectively searching the RIK solution space.

The locally optimal paths of adjoining sub-task homotopy classes do not have the same endpoint configurations and therefore cannot be combined into a continuous joint path, as shown by the discontinuous network of paths in Fig. 3.3c. A lower bound cost for the homotopy optimal path of each (complete task) homotopy class is obtained by summing the costs of the corresponding sequence of sub-task homotopy optimal paths.

Step 4: Generate a refined configuration roadmap that uses the locally optimal paths of sub-task homotopy classes as edges. The locally optimal paths of *every other* sub-task homotopy class in the sequence of sub-tasks are joined together using instantaneous path planning as shown by the dashed lines in Fig. 3.3d. The result is a refined *configuration* roadmap, in which each route through the roadmap is continuous, smooth, and satisfies the task constraints (equality and inequality constraints in (3.1)).

Step 5: Deform the sub-optimal joint paths of the refined configuration roadmap routes into locally optimal *complete* joint paths. The best of these paths is deemed to be the globally optimal joint path. Starting with the joint path from Step 4 with the lowest cost, this joint path is deformed into a locally optimal joint path. The cost of this locally optimal path is then used as an upper bound cost for the globally optimal path. Any homotopy class with a lower bound cost greater than the upper bound cost cannot contain the globally optimal path and is removed from the set of homotopy classes considered in this step. Only paths in promising homotopy classes are deformed into locally optimal paths. For tasks with many bifurcations, the set of promising homotopy classes considered is reduced by a factor of 100 or more, saving computation time associated with the path deformation procedure.

A "direct" optimal control method (based on nonlinear programming) is used to deform each sub-optimal complete path into a locally optimal complete path. The bb-algorithm uses a new "direct" method that reformulates the global resolution problem (3.1) into a reduced-order problem using self-motion paths. The modified direct method used in the bb-algorithm ensures the path deformation *reliably* converges to a locally optimal path within the same homotopy class and does not "jump" to a different homotopy class, which is of great concern, especially for sub-optimal paths passing near bifurcation points.

3.4 Bifurcation Branch Roadmap

Characterization of the manipulator's self-motion manifolds at all points on the task path precedes the construction of the bifurcation branch roadmap. The bb-roadmap for *open* self-motion manifolds with bifurcations caused by joint limits (like the RIK case illustrated in Fig. 3.2) has a relatively simple structure.

This section presents the more complex bb-roadmaps associated with manipulators with different numbers of *closed* self-motion manifolds (one or two depending on the task configuration). The bb-roadmaps presented here identify the more complex bifurcation structure caused by singularities at coregular values as opposed to those caused by joint limits. The impact of the coregular values on the RIK solution space structure is illustrated as well as their impact on the number of homotopy classes. The impact of crossing a single coregular value is quite different from crossing two coregular values. A method for quickly constructing the bb-roadmap of a general task that crosses many coregular values is introduced.

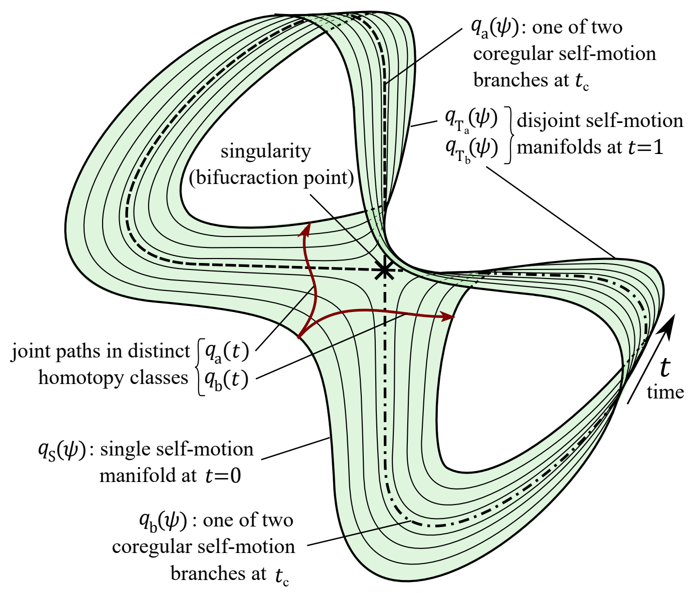

$q_a(\psi)$: one of two coregular self-motion branches at t_c
$q_{T_a}(\psi)$
$q_{T_b}(\psi)$
disjoint self-motion manifolds at t=1
singularity (bifucraction point)
joint paths in distinct homotopy classes
$q_a(t)$
$q_b(t)$
t
time
$q_S(\psi)$: single self-motion manifold at t=0
$q_b(\psi)$: one of two coregular self-motion branches at t_c

Figure 3.4: RIK solution space of a task path crossing a coregular value at time t_c. Solid paths are self-motion manifolds at different time instances. The coregular self-motion path intersects itself at the singularity; it is separated into a dashed line path $\boldsymbol{q}_a(\psi)$ and dashed-dotted line path $\boldsymbol{q}_b(\psi)$. The 2 heavier solid lines with an arrows are joint paths $\boldsymbol{q}_a(t)$ and $\boldsymbol{q}_b(t)$; they cannot be continuously deformed over the surface into each other.

3.4.1 Tasks Crossing a Single Coregular Value

Consider a task path crossing a single coregular value, starting at a task configuration with one self-motion manifold, $\boldsymbol{q}_S(\psi)$, and terminating at a task configuration with two self-motion manifolds, $\boldsymbol{q}_{T_a}(\psi)$ and $\boldsymbol{q}_{T_b}(\psi)$. The RIK solution surface of the task is illustrated in Fig. 3.4, where each solid line corresponds to a self-motion manifold (a connected set of joint configurations that yield the desired end-effector position at a specified time). As the task progresses, the self-motion manifold experiences a structural change at time t_c when the task path crosses the coregular value. The solution space structural change is due to a coregular self-motion path that is self-intersecting, shown by the dashed and dashed-dotted lines forming a figure-eight. The intersection point is a kinematic singularity, at which the Jacobian matrix is rank deficient and the dimension of the null space increases.

A singularity associated with a coregular value has hyperbolic characteristics [59] corresponding to a saddle point in the RIK solution space, as illustrated by the $\times$ in the center of Fig. 3.4. Joint paths $\boldsymbol{q}_a(t)$ and $\boldsymbol{q}_b(t)$ are on different sides of the saddle. Joint path $\boldsymbol{q}_a(t)$ cannot be continuously deformed along the self-motion manifolds into $\boldsymbol{q}_b(t)$. As such, $\boldsymbol{q}_a(t)$ and $\boldsymbol{q}_b(t)$ are in different homotopy classes.

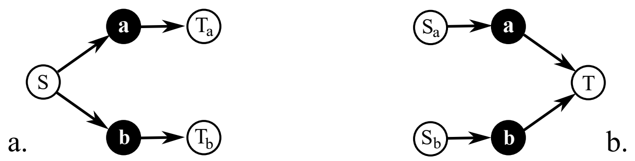


Figure 3.5: Bifurcation branch roadmap of a task path crossing a single coregular value. a) The roadmap of a task starting with 1 self-motion manifold and terminating with 2 self-motion manifolds. b) The roadmap of a task starting with 2 self-motion manifolds and terminating with 1 self-motion manifold.

Because joint paths cannot be deformed across a singularity, the coregular self-motion path is treated as two distinct open paths, $\boldsymbol{q}_a(\psi)$ and $\boldsymbol{q}_b(\psi)$, with endpoints adjacent to (but not including) the singularity. Feasible self-motion paths that are adjacent to each other at the bifurcation point are called *bifurcation branches.* The homotopy class of $\boldsymbol{q}_a(t)$ and $\boldsymbol{q}_b(t)$ directly depends on which of the two bifurcation branches (dashed or dashed-dotted lines in Fig. 3.4) is crossed. This bifurcation is captured in the bifurcation branch roadmap, where $\boldsymbol{q}_a(\psi)$ and $\boldsymbol{q}_b(\psi)$ are associated with bifurcation nodes.

The bb-roadmap of the RIK solution space of Fig. 3.4 is given by Fig. 3.5a, where white nodes correspond to regular self-motion paths at task endpoints, black nodes correspond to coregular bifurcation branches, and directed edges (arrows) correspond to sub-task homotopy classes connecting the nodes. Figure 3.5a corresponds to a single self-motion manifold bifurcating into two self-motion manifolds, whereas Fig. 3.5b corresponds to two self-motion manifolds converging into one self-motion manifold (such as the task of Fig. 3.4 with time reversed). There are two routes[4] that traverse the roadmap in Fig. 3.5a: 1) $[S, a, T_a]$ and 2) $[S, b, T_b]$, and two routes that traverse the roadmap in Fig. 3.5b: 1) $[S_a, a, T]$ and 2) $[S_b, b, T]$.

As a general task progresses, the self-motion manifolds alternate between splitting and joining at coregular values. Given the open manifolds of Fig. 3.3, one might assume that the number of homotopy classes, as the task progresses, doubles only at coregular values which split the self-motion manifolds. This assumption is not valid for closed self-motion manifolds as shown below.

3.4.2 Tasks Crossing Two Coregular Values

For a task path crossing two coregular values, consider the solution space between (and including) the coregular values. The task path passes through a single W-sheet. As such, the

[4]Routes can also be represented as a sequence of nodes (as opposed to a sequence of edges) when there is only one homotopy class of joint paths between nodes.

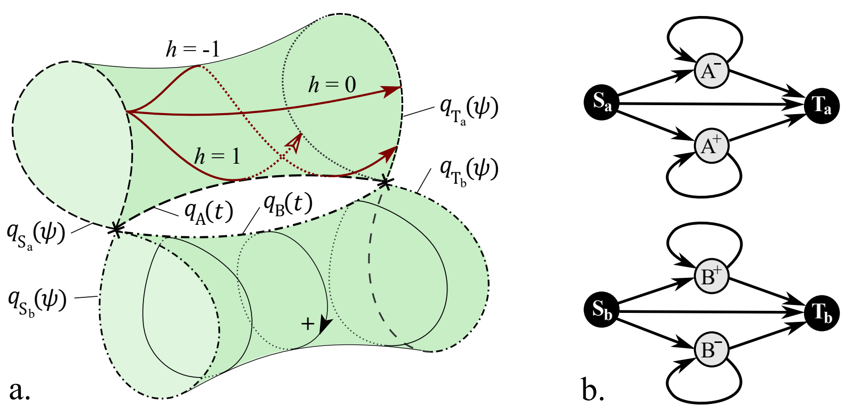


Figure 3.6: Structure of the RIK solution space for a task within a W-sheet with two self-motion manifolds. a) The RIK solution surface, two cylinders with "pinched" ends. Dashed and dash-dotted lines represent coregular self-motion paths and singularity connection paths used as roadmap nodes. b) Roadmap where black nodes are coregular self-motion paths of the bifurcation branches and gray nodes are crossings of a singularity connection path. Each route identifies a unique homotopy class.

self-motion manifolds of the interior task points are closed paths, homotopic to each other, but they are not homotopic to the coregular self-motion paths that are *open* paths. The structure of the RIK solution space for the task depends on the number of self-motion manifolds of task configurations inside the W-sheet. The bifurcation branch roadmap for W-sheets with two self-motion manifolds (Case R^{C_2}) is simpler than that for one (Case R^{C_1}) and is presented first.

Case R^{C_2}

For a task path through a W-sheet with 2 self-motion manifolds, the RIK solution space has the structure of Fig. 3.6a with two disjoint cylinders, each with "pinched" ends, joining only at the pinch points. The pinching effect is illustrated in the bottom cylinder by the three self-motion manifolds (solid closed curves, but dotted when the curve is hidden behind another surface). As a self-motion manifold approaches a coregular self-motion path, the smooth curve is "pinched" to have a sharp corner at the singularity.

Consider a joint path starting in coregular self-motion branch $\boldsymbol{q}_{S_a}(\psi)$. The joint path necessarily terminates in the coregular self-motion path $\boldsymbol{q}_{T_a}(\psi)$. The joint path may travel along the cylinder "directly" or by "wrapping around" the cylinder clockwise or counter-clockwise, as illustrated by the solid/dotted lines with arrows at the terminal endpoint in Fig. 3.6a. These paths are homotopically distinct because they cannot be continuously deformed into each other due to the sharp corner at the singularity.

The homotopy class of a joint path can be identified using the following method:

1. Identify a *singularity connection path*, a path over the RIK solution space that connects the singularities at each end of the task. For some cases, a path may be obtained from the inverse kinematic solution of an equivalent non-redundant mechanism with two twists (columns of the Jacobian matrix) constrained to be linearly dependent.
2. Assign a positive and negative direction for crossing the singularity connection path.
3. Identify all instances at which the joint path crosses the singularity connection path, keeping track of the crossing direction. (This information may be identified using root finding.)
4. Beginning with $h = 0$, for each crossing of the singularity connection path; $h = h + 1$ for a positive crossing, or $h = h - 1$ for a negative crossing. The net-value of h, along with start and terminal coregular bifurcation branches, identifies the homotopy class of the joint path. The h value of each path along the top cylinder is shown in Fig. 3.6a.

The RIK solution space of Fig. 3.6a has two singularity connection paths $\boldsymbol{q}_A(t)$ and $\boldsymbol{q}_B(t)$, one on each cylinder. The self-motion path on the lower cylinder with the arrow shows the assigned positive direction for crossing $\boldsymbol{q}_B(t)$.

The roadmap[5] in Fig. 3.6b completely captures the homotopy classes for this case. Gray nodes indicate the direction in which a singularity connection path is crossed. The node letter (A, B) identifies which path and the superscript sign ($+$, $-$) identifies the crossing direction. Cycles in the roadmap identify joint paths with self-motion cycles. For example, the route containing one cycle: $[S_a, A^+, A^+, T_a]$, corresponds to a path with $h_A = 2$ that makes at least one full self-motion cycle, but less than two self-motion cycles.

Because it is extremely unlikely that the globally optimal path for any fixed endpoint combination would contain a full self-motion cycle, the bb-algorithm uses a simplified roadmap without gray nodes associated with self-motion cycles. The bb-roadmap of this case is shown in Fig. 3.8e, where each edge corresponds to a homotopy class without a self-motion cycle. There are 6 different homotopy classes, 3 for each starting node. Each bb-edge number in Fig. 3.8e has a corresponding node sequence (summarized in Table 3.1).

Case R^{C_1}

For a task path through a W-sheet with a single self-motion manifold, the RIK solution space has the structure of Fig. 3.7a (one cylinder with "pinched" ends). The pinching effect is

[5]This roadmap is a bb-roadmap prior to its simplification. It identifies all homotopy classes, including those with full-cycle self-motions.

Table 3.1: Homotopy Classes of R^{C_2}

Edge in Fig. 3.8e	Node Sequence in Fig. 3.6b
1	S_a, T_a
2	S_a, A^+, T_a
3	S_a, A^-, T_a
4	S_b, T_b
5	S_b, B^+, T_b
6	S_b, B^-, T_b

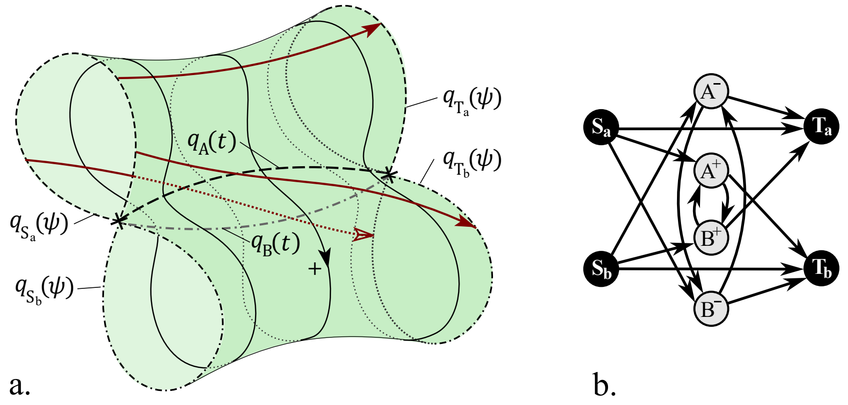


Figure 3.7: Structure of the RIK solution space for a task within a W-sheet with one self-motion manifold. a) The RIK solution surface, a cylinder with "pinched" ends. b) Roadmap where black nodes are coregular self-motion paths and gray nodes are crossings of a singularity connection path. Each route identifies a unique homotopy class.

illustrated by the three self-motion manifolds (solid/dotted closed curves). As a self-motion manifold approaches a coregular self-motion path, the manifold becomes pinched and eventually two opposite points on the manifold join.

Unlike the previous case in which two singularity connection paths existed in separate RIK solution surfaces, this case has two singularity connection paths ($\boldsymbol{q}_A(t)$ and $\boldsymbol{q}_B(t)$) on the same RIK solution surface.

Consider a joint path starting in a given coregular self-motion branch, $\boldsymbol{q}_{S_a}(\psi)$. As in the previous case, a "direct" joint path to $\boldsymbol{q}_{T_a}(\psi)$ exists $(h = 0)$. Unlike the previous case, the joint path here may also travel to the other coregular self-motion branch, $\boldsymbol{q}_{T_b}(\psi)$ by crossing singularity connection path $\boldsymbol{q}_A(t)$ or $\boldsymbol{q}_B(t)$, as illustrated in the example paths in Fig. 3.7a.

Table 3.2: Homotopy Classes of R^{C_1}

Edge in Fig. 3.8f	Node Sequence in Fig. 3.7b
1	S_a, T_a
2	S_a, A^+, B^+, T_a
3	S_a, B^-, A^-, T_a
4	S_a, A^+, T_b
5	S_a, B^-, T_b
6	S_b, T_b
7	S_b, A^-, B^-, T_b
8	S_b, B^+, A^+, T_b
9	S_b, A^-, T_a
10	S_b, B^+, T_a

The roadmap in Fig. 3.7b completely captures the homotopy classes for this case. Removing the self-motion cycles yields the bb-roadmap of this case, shown in Fig. 3.8f, where each edge corresponds to a unique homotopy class. There are 10 different homotopy classes, 5 for each starting node. Each bb-edge number in Fig. 3.8f has a corresponding node sequence (summarized in Table 3.2).

3.4.3 General Tasks

Consider a general task path that crosses multiple coregular values. The task is divided at the coregular values into a series of sub-task paths.

The bifurcation branch roadmap of a general task is a concatenation of the sub-task bb-roadmaps shown in Fig. 3.8. Each sub-task roadmap is labeled R^{S_w}, R^{T_w}, or R^{C_w}, where w is the number of distinct self-motion manifolds of the W-sheet containing the sub-task path, the superscripts S, T, and C correspond to start, terminal, and coregular endpoints, respectively.

The sub-task bb-roadmaps are always combined at the coregular self-motion nodes. For example, the simple bb-roadmap in Fig. 3.5a is constructed by combining R^{S_1} (Fig. 3.8a) and R^{T_2} (Fig. 3.8b). For a general task, the sub-task roadmaps alternate between $w = 1$ and $w = 2$. If the task has free boundary conditions, the first sub-task roadmap is R^{S_w} and the last sub-task roadmap is R^{T_w}. If the task has periodic boundary conditions, for path planning purposes, the start point can be selected to be at a coregular value such that every sub-task roadmap is R^{C_w}.

The number of homotopy classes for a general task path is given by the number of admissible routes through the bb-roadmap. For cyclic tasks, when periodic boundary conditions

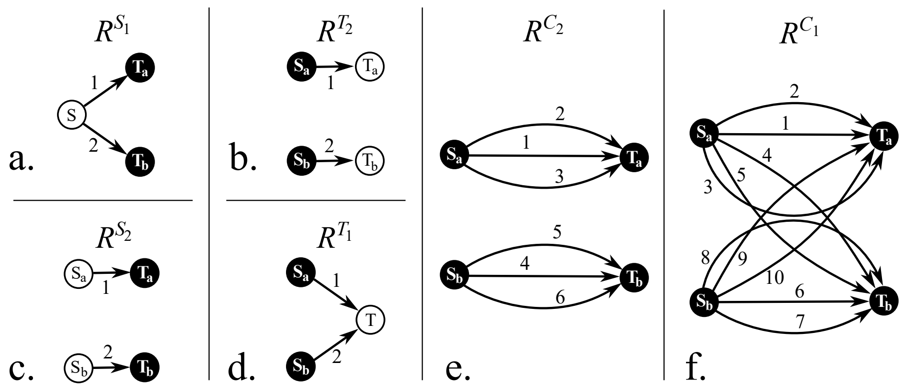


Figure 3.8: Bifurcation branch roadmaps of sub-tasks bounded by coregular values or a complete task endpoint. a-d) Sub-task bb-roadmaps with regular self-motion manifolds of a task endpoint. e-f) Sub-task bb-roadmaps with coregular self-motion paths at both endpoints.

are required, the start and terminal joint configuration must be in the same coregular self-motion branch. For free boundary conditions, there is no constraint on the start or terminal self-motion node, and because the bb-roadmap is symmetric, the number of homotopy classes is

$$N_H = 2 \prod_{k=2}^{K-1} N_h(R_k), \tag{3.4}$$

where K is the number of sub-tasks in a complete task sequence, and $N_h(R_k)$ is the number of homotopy classes per start node in the k^{th} sub-task. For the sub-task roadmaps for closed manifolds considered above, $N_h = 3$ for R^{C_2}, and $N_h = 5$ for R^{C_1}. The number of homotopy classes is independent of the type of sub-task roadmaps for $k = 1$ and $k = K$. The total number of homotopy classes increases rapidly as the number of coregular value crossings increases.

3.4.4 Generating Node Descriptions

Nodes of the bb-roadmap correspond to continuous self-motion paths. These paths must be mathematically defined so that representative samples of joint configurations can be obtained.

Some redundant robots have analytical expressions for identifying self-motion paths of bb-nodes. However, most robot structures do not have an analytical self-motion parameter that is valid over the *entire* workspace of the robot [71]. For this reason a "natural" parametrization corresponding to arc length on the joint configuration manifold is used to define the self-motion path.

The self-motion path $\boldsymbol{q}(\psi)$ for a closed manifold is numerically generated starting from an initial joint configuration $\mathbf{q} = \mathbf{q}_0$ at $\psi = 0$. The instantaneous direction of the self-motion path at a joint configuration is identified by the null space of the Jacobian matrix, which is a vector for RIK problems with $r = 1$ and when the joint configuration in non-singular. A finite joint motion along the null space is identified using:

$$\Delta \mathbf{q}_N = \mathbf{W}^{-1/2} \hat{\mathbf{n}} \Delta \psi_{\mathrm{d}}, \tag{3.5}$$

where $\hat{\mathbf{n}}$ is the unit null space vector of the Jacobian matrix of the current joint configuration $\mathbf{q}$ and $\Delta\psi_{\mathrm{d}}$ is the desired arc length step size. (Because the positive definite weighing matrix $\mathbf{W}$ is used to identify the weighted norm *squared*, the square root of $\mathbf{W}$ is used to identify a joint motion weighted norm with a magnitude equal to the arc length step size.)

A new joint configuration $\mathbf{q}' \approx \mathbf{q} + \Delta\mathbf{q}_N$ is refined using the Newton-Raphson method to eliminate task error. The arc length distance traveled in that step is estimated using

$$\Delta\psi = \sqrt{\Delta\mathbf{q}^T \mathbf{W} \Delta\mathbf{q}}, \tag{3.6}$$

where $\Delta\mathbf{q} = \mathbf{q}' - \mathbf{q}$. For small arc length step sizes $\Delta\psi_{\mathrm{d}} \approx \Delta\psi$.

The new joint configuration $\mathbf{q}'$ at $\psi + \Delta\psi$ is saved in a self-motion sequence. The current joint configuration is updated $\mathbf{q}' \to \mathbf{q}$ and the process repeats until a termination criterion is reached. Four useful criteria are: 1) when a joint limit is encountered, 2) when a singularity is encountered, 3) when the initial joint configuration is encountered again, 4) or when a maximum arc length is traveled. The first three criteria are identified using the distance estimate (3.6), where $\Delta\mathbf{q}$ is the difference between the current joint configuration of the self-motion path and the nearest joint configuration or interest, i.e., 1) nearest joint configuration with a saturated limit, 2) nearest singular joint configuration, or 3) initial joint configuration $\mathbf{q}_0$. Criterion 1 is relevant if joint limits are present; Criterion 2 is relevant if the task configuration is a coregular value; Criterion 3 is relevant for closed self-motion manifolds; and Criterion 4 is used to prevent infinite path generation of an open unbounded self-motion manifold.

The self-motion path, numerically generated as a sequence of configurations, is converted into a continuous path using a cubic spline fit.

For coregular self-motion paths, the singularity configuration, identified by a known geometric condition, is used as the initial point. However, (3.5) is not valid at the singularity, because the null space of the Jacobian is not a vector. The two directions for feasible finite self-motion must be identified using another method such as that presented in [72] or using an analytical parametrization. Once two nearby configurations are identified by taking a small step in the identified directions, the coregular self-motion paths can be generated using the method described above. Two paths are generated (e.g., $\boldsymbol{q}_a(\psi)$ and $\boldsymbol{q}_b(\psi)$ in Fig. 3.4), each path returns to the singularity.

3.4.5 Roadmap Construction Discussion

The bifurcation branch roadmap (bb-roadmap) depends on global characteristics of the manipulator, namely, the characterization of the self-motion manifold and the conditions for kinematic singularities and coregular values. When this information is available, the bb-roadmap of a general task path is easily constructed for RIK problems without joint limits.

The only analysis required for constructing the bb-roadmap for a general task is identifying the number of coregular values and the number of self-motion manifolds of task points between them. The coregular values can be identified numerically by finding the roots of geometric conditions associated with the coregular values (see example in Section 3.7). The number of complete homotopy classes is immediately identified as the number of complete routes through the bb-roadmap.

When the conditions for coregular values are known, the time instances of the coregular values along the task path are identified using root finding. The self-motion paths of the bb-nodes at these task instances, along with the task endpoints, are generated using the self-motion path generation method described above.

3.5 Bi-directional Instantaneous Path Planner

This section introduces a new instantaneous path planner used in the bifurcation branch algorithm in two different steps. It is used first in Step 2 to generate a path from a sampled configuration on one bb-node to a sampled configuration on an adjoining bb-node. The instantaneous planner ensures that the generated path is in the appropriate sub-task homotopy class.

The instantaneous path planner is again used in Step 4 in alternating sub-tasks to connect the terminal endpoint of one locally optimal path to the starting endpoint of another locally optimal path. The instantaneous path planner ensures that the concatenation of instantaneously generated paths with locally optimal paths yields a complete joint path that is continuous and satisfies the task constraints.

3.5.1 Path Calculation

Recall from Chapter 2, an instantaneously resolved joint path is generated by

$$\boldsymbol{q}(t) = \mathbf{q}_0 + \int_{t=0}^{t=1} \dot{\boldsymbol{q}}(t)\, dt, \tag{3.7}$$

where $\mathbf{q}_0$ is the starting joint configuration and $\dot{\boldsymbol{q}}(t)$ is the joint velocity at task time t. The joint velocity is instantaneously resolved to satisfy the task path $\boldsymbol{x}(t)$ using

$$\dot{\boldsymbol{q}}(t) = \mathbf{J}^{\dagger}(\boldsymbol{q}(t))\dot{\boldsymbol{x}}(t) + \mathbf{P}(\mathbf{q})\dot{\mathbf{q}}_N, \tag{3.8}$$

where $\dot{\boldsymbol{q}}(t)$ is the optimal admissible joint velocity for the current joint configuration $\boldsymbol{q}(t)$, $\mathbf{J}^{\dagger}$ (a function of the current joint configuration) is the weighted pseudoinverse of the Jacobian matrix, and $\mathbf{P} = (\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J})$ is a matrix that projects the "preferred" joint velocity $\dot{\mathbf{q}}_N$ into the null space of the Jacobian matrix. The Saturation in the Null Space (SNS) algorithm is used to accommodate inequality constraints associated with joint limits (see Chapter 2).

The bb-algorithm uses an instantaneous path planner that bi-directionally integrates (3.8), in which a non-zero "preferred" joint motion $\dot{\mathbf{q}}_N$ is used and continuously updated to guide the generated joint paths to meet.

Consider the desired endpoint joint configurations $\mathbf{q}_A$ and $\mathbf{q}_B$ at times t_A and t_B, respectively. The bi-directional planner simultaneously generates a forward-growing joint path $\boldsymbol{q}_a(\tau)$ starting at $\boldsymbol{q}_a(0) = \mathbf{q}_A$ and a backward-growing a joint path $\boldsymbol{q}_b(\tau)$ starting at $\boldsymbol{q}_b(0) = \mathbf{q}_B$ by integrating

$$\begin{bmatrix} \dot{\boldsymbol{q}}_a(\tau) \\ \dot{\boldsymbol{q}}_b(\tau) \end{bmatrix} = \begin{bmatrix} \mathbf{J}^{\dagger}(\mathbf{q}_a)\dot{\boldsymbol{x}}(\tau + t_A) + s(\tau)\mathbf{P}(\mathbf{q}_a)\dot{\mathbf{q}}_N \\ \mathbf{J}^{\dagger}(\mathbf{q}_b)(-\dot{\boldsymbol{x}}(t_B - \tau)) - s(\tau)\mathbf{P}(\mathbf{q}_b)\dot{\mathbf{q}}_N \end{bmatrix} \tag{3.9}$$

over $\tau \in [0, (t_A + t_B)/2]$ where, $s(\tau)$ is a monotonic scaling function that ranges from 0 to 1 over τ. The preferred joint motion $\dot{\mathbf{q}}_N$ is a direct motion in the joint space from path $\boldsymbol{q}_a(\tau)$'s current terminal point $\mathbf{q}_a$ to path $\boldsymbol{q}_b(\tau)$'s current terminal point $\mathbf{q}_b$ given by

$$\dot{\mathbf{q}}_N = \frac{\mathbf{q}_b - \mathbf{q}_a}{(t_B - t_A) - 2\tau}. \tag{3.10}$$

The scaling function $s(\tau)$ is used to cause the paths to meet. At the start of paths $\boldsymbol{q}_a(\tau)$ and $\boldsymbol{q}_b(\tau)$, the instantaneous joint motion minimizes the weighted joint velocity norm. By the end of the path generation, the joint motion is selected to go directly to the updated terminal point of the other path to connect the paths.

The resulting joint path is

$$\boldsymbol{q}_{a\to b}(t) = \begin{cases} \boldsymbol{q}_a(t - t_A), & \text{for } t_A \le t \le \frac{t_A+t_B}{2} \\ \boldsymbol{q}_b(t_B - t), & \text{for } \frac{t_A+t_B}{2} < t \le t_B. \end{cases} \tag{3.11}$$

The joint path is continuous if the terminal points of $\boldsymbol{q}_a(\tau)$ and $\boldsymbol{q}_b(\tau)$ are identical. The joint paths smoothly connect when $s(\tau)$ approaches 1 with a slope of zero. For instance, $s(\tau)$ may be a cubic-spline with slopes clamped at zero.

For RIK solution spaces with open feasible self-motion paths, each sub-task has a single sub-task homotopy class that is a simply connected space. Because the bi-directionally generated paths $\boldsymbol{q}_a(\tau)$ and $\boldsymbol{q}_b(\tau)$ are in a simply connected subspace, they are guaranteed to meet.

3.5.2 Challenges for Closed Self-Motion Manifolds

For RIK solution spaces with closed feasible self-motion paths, the bi-directionally generated paths $\boldsymbol{q}_a(\tau)$ and $\boldsymbol{q}_b(\tau)$ generated over a sub-task can fail to meet by terminating at opposite ends of the mid-task self-motion manifold. Failure of the paths to meet occurs under rare conditions when there is symmetry in the RIK solution space and in the starting joint configurations. In practice, the paths almost always meet. However, because there are multiple sub-task homotopy classes for each sub-task, the sub-task homotopy class in which the generated path exists is not directly controlled.

The sub-task homotopy class of the generated joint path is identified based on the net crossings of the relevant singularity connection path(s). This information can be obtained using root finding when the singularity connection path is generated with the inverse kinematic solution of an equivalent non-redundant mechanism with two twists (columns of the Jacobian matrix) constrained to be linearly dependent.

A single use of the bi-directional path planner between two fixed points generally finds the “direct” path through the RIK solution space between coregular values. However, finding paths that cross a singularity connection path, may require additional control. To control which singularity connection path is crossed, an intermediate point is selected on the relevant singularity connection path and two paths are generated (start to intermediate and intermediate to terminal) and pieced together.

3.5.3 Piecewise Construction of Complete Initial Paths

A complete initial joint path through the RIK solution space is constructed by piecing together instantaneously generated paths or locally optimal paths (of sub-tasks) associated with the route edges of the configuration roadmap produced in Step 4 of the bb-algorithm. Locally optimal paths are used from every other sub-task homtopy class starting from *either* the first sub-task homotopy class or from the second sub-task homotopy class. The decision on the starting sub-task is made such that the greatest number of locally optimal paths are used.

Consider a route with four sequentially connected sub-task homotopy classes $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$, each in a different adjoining sub-task. Let $\mathcal{A}$ and $\mathcal{C}$ be the sub-task homotopy classes with the greatest number of locally optimal paths. The bidirectional instantaneous path planner is used to generate a path $\boldsymbol{q}_B(t) \in \mathcal{B}$ to connect the terminal point of a locally optimal path $\boldsymbol{q}_{\mathcal{A}}(t) \in \mathcal{A}$ to the starting point of a locally optimal path $\boldsymbol{q}_{\mathcal{C}}(t) \in \mathcal{C}$. The piecewise concatenated joint path $\boldsymbol{q}_{\mathcal{ABC}}(t) = [\boldsymbol{q}_{\mathcal{A}}(t), \boldsymbol{q}_{\mathcal{B}}(t), \boldsymbol{q}_{\mathcal{C}}(t)]$ is continuous.

Depending on the selected boundary conditions, a joint path $\boldsymbol{q}_{\mathcal{D}}(t) \in \mathcal{D}$ generated by using either the normal (single-direction integration) instantaneous resolution approach or by using the bi-directional path planner. If the RIK problem has free boundary conditions, path $\boldsymbol{q}_{\mathcal{D}}(t)$ is generated using normal instantaneous resolution, starting at the terminal point of $\boldsymbol{q}_{\mathcal{C}}(t)$. The terminal point of $\boldsymbol{q}_{\mathcal{D}}(t)$ is determined by the instantaneous optimization criterion. If the RIK problem has fixed boundary conditions, $\boldsymbol{q}_{\mathcal{D}}(t)$ is generated using the bi-directional path planner to connect the terminal point of $\boldsymbol{q}_{\mathcal{C}}(t)$ to the fixed endpoint configuration. If the RIK problem has periodic boundary conditions, $\boldsymbol{q}_{\mathcal{D}}(t)$ is generated using the bi-directional path planner to connect the terminal point of $\boldsymbol{q}_{\mathcal{C}}(t)$ to the starting point point of $\boldsymbol{q}_{\mathcal{A}}(t)$.

The concatenated joint path $\boldsymbol{q}_{\mathcal{ABCD}}(t) = [\boldsymbol{q}_{\mathcal{A}}(t), \boldsymbol{q}_{\mathcal{B}}(t), \boldsymbol{q}_{\mathcal{C}}(t), \boldsymbol{q}_{\mathcal{D}}(t)]$ is a continuous joint path that exactly satisfies the complete task path and satisfies the selected boundary conditions. Additional joint paths in the same complete task homotopy class are obtained by connecting different combinations of locally optimal paths $\boldsymbol{q}_A(t) \in \mathcal{A}$ and $\boldsymbol{q}_C(t) \in \mathcal{C}$. These paths are then deformed into a locally optimal joint paths using a path deformation procedure.

3.6 Path Deformation Procedure

This section describes the procedure used for deforming a sub-optimal joint path into a locally optimal joint path. Although the procedure is restricted to problems with one degree of

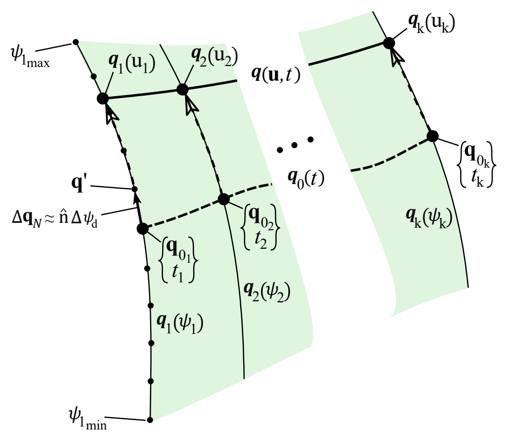

$\psi_{1_{\max}}$
$\boldsymbol{q}_1(u_1)$
$\boldsymbol{q}_2(u_2)$
$\boldsymbol{q}(\mathbf{u},t)$
$\boldsymbol{q}_k(u_k)$
$\mathbf{q}'$
$\Delta\mathbf{q}_N \approx \hat{n}\Delta\psi_d$
$\boldsymbol{q}_0(t)$
$\left\{\begin{matrix}\mathbf{q}_{0_1}\\ t_1\end{matrix}\right\}$
$\left\{\begin{matrix}\mathbf{q}_{0_2}\\ t_2\end{matrix}\right\}$
$\left\{\begin{matrix}\mathbf{q}_{0_k}\\ t_k\end{matrix}\right\}$
$\boldsymbol{q}_k(\psi_k)$
$\boldsymbol{q}_2(\psi_2)$
$\boldsymbol{q}_1(\psi_1)$
$\psi_{1_{\min}}$

Figure 3.9: Joint path deformation on an RIK solution surface. The initial joint path (thick dashed line) is discretized (large dots). The configurations (dots) are moved along the self-motion paths (thin solid lines), and the deformed path (thick solid line) is approximated with a cubic spline through the large dots.

redundancy, it reliably converges to a locally optimal joint path even if a very high degree of deformation is required. The procedure also ensures that the joint path will remain in the same homotopy class. The method is applicable for fixed, free, and periodic boundary conditions.

Given an initial joint path $\boldsymbol{q}_0(t)$ for the task path $\boldsymbol{x}(t)$, a locally optimal path is found using the following steps:

1. Discretize the initial joint path into a set of k configurations with corresponding time indicies:

$$\boldsymbol{q}_0(t) \rightarrow \begin{Bmatrix} \mathbf{q}_{0_1} & \mathbf{q}_{0_2} & \dots & \mathbf{q}_{0_k} \\ t_1 & t_2 & \dots & t_k. \end{Bmatrix} \tag{3.12}$$

   This set of joint configurations should include joint configurations associated with the bb-nodes: the joint configurations at task endpoints and the joint configurations intersecting any bifurcation branches. Additional joint configurations between bb-nodes are included to accurately approximate the task path.

2. For each joint configuration $\mathbf{q}_{0_i}$, $(i \in [1, 2, \dots, k])$, generate a self-motion path $\boldsymbol{q}_i(\psi_i) \mid \boldsymbol{f}(\boldsymbol{q}_i(\psi_i)) = \boldsymbol{x}(t_i),\ \mathbf{q}_{0_i} \in \boldsymbol{q}_i(\psi_i)$, where $\psi_i \in [\psi_{i_{\min}}, \psi_{i_{\max}}]$ is a bounded self-motion parameter. Bounds $\psi_{i_{\min}}$ and $\psi_{i_{\max}}$ are determined by the joint limits, bifurcation points, or a prescribed value. The self-motion paths are generated by integrating (3.5).

3. Select the optimization parameters as positions on the bounded self-motion paths $u_i \in [\psi_{i_{\min}}, \psi_{i_{\max}}]$. Every self-motion path parameter $(i \neq 1, k)$, is an independent optimization parameter $\psi_i \rightarrow u_i$. If the task endpoint $(i = 1, k)$ has a free boundary

condition, then $\psi_i \to u_i$. If the boundary condition is fixed, ψ_i is a fixed value (e.g., $\psi_i = 0$). If the boundary conditions are periodic, then $\psi_1 \to u_1$ and $\psi_k = u_1$.

4. Use nonlinear programming (NLP), such as an interior-point algorithm, to solve:

$$\begin{aligned} \text{min.} \quad & G_{\text{global}}(\boldsymbol{q}(\mathbf{u}, t)) \\ \text{s.t.} \quad & \mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}, \end{aligned} \tag{3.13}$$

where $\boldsymbol{q}(\mathbf{u}, t)$ is a joint motion path expressed as a cubic-spline with k nodes defined by $\mathbf{u}$ (the set of optimization parameters). For periodic boundary conditions, the start and terminal slopes of the joint path spline are constrained to be the same to enforce continuity in the joint velocities.

The novelty in this modified direct approach is the use of self-motion paths rather than using equality constraints in the optimization to ensure the joint path tracks the task path. This approach also reduces the number of dimensions needed to specify the joint configurations and allows for large, nonlinear deformation of the path in each iteration of the NLP solver. Because self-motion paths of bifurcation branches are used, the NLP solver can deform the path arbitrarily close to a bifurcation point (e.g., singularity) without crossing it. Also, because the self-motion paths are parameterized by arc length on the joint configuration manifold, the objective function is not highly sensitive to changes in the optimization parameters. The algorithm is therefore both fast and reliable and it is incapable of "jumping" to a different homotopy class (provided that the path is approximated with a sufficient number of nodes).

The self-motion paths generated in Step 2 of the path deformation procedure are generated bi-directionally outward from the initial joint configuration $\mathbf{q}_{0_i}$. One step of the self-motion path generation is illustrated at $\mathbf{q}_{0_1}$ in Fig. 3.9. Starting at the base-point, there is a "positive" motion direction given by $\hat{\mathbf{n}}$ and a "negative" motion direction given by $-\hat{\mathbf{n}}$. The self-motion sampled configurations are generated in both directions until bounds are reached at $\psi_{i_{\max}}$ and $\psi_{i_{\min}}$.

To reduce computation time in the generation of the self-motion paths, a maximum arc length s along the self-motion manifold is specified such that $\psi_{i_{\max}} \leq s$ and $\psi_{i_{\min}} \geq -s$. If the NLP solver converges to a joint path that does not touch a self-motion bound at s or $-s$, the resulting path is locally optimal. If the NLP solver converges to a path that touches a prescribed self-motion bound at s or $-s$, the resulting path is used as an initial path and Steps 1-4 are repeated until a locally optimal path is reached.

The equality constraints of the task path are accounted for by the self-motion paths. These constraints are satisfied at the discrete points, but not between them. The actual task error of a joint path is evaluated using

$$x_{\text{err}} = \int_{t=0}^{t=1} \left(\boldsymbol{x}(t) - \boldsymbol{f}(\boldsymbol{q}(\mathbf{u},t))\right)^T \left(\boldsymbol{x}(t) - \boldsymbol{f}(\boldsymbol{q}(\mathbf{u},t)\right)dt, \tag{3.14}$$

where the numerical integration uses a mesh at least 20 times more dense than the joint path discretization. If the task error exceeds a specified threshold value, more nodes are added to $\mathbf{u}$, those self-motion paths are evaluated, and the interior-point algorithm is restarted.

3.7 Case Study

This section demonstrates the bifurcation branch algorithm (bb-algorithm) for finding the globally optimal joint path for a 3R manipulator executing particle planar task paths. Joint limits are not present in this example. First, the self-motion and singularity characteristics are described to show that the sub-task roadmaps presented in Section 3.4 are valid for any task path in the manipulator's workspace. Next, the RIK solution spaces of three simple tasks are presented. These tasks have solution space structures like those in Figs. 3.4, 3.6, and 3.7. Lastly, a complex task path is presented for which there is a very high number of homotopy classes and the globally optimal path found using the bb-algorithm is not found using traditional methods.

3.7.1 Manipulator

Consider the 3R planar manipulator depicted in Fig. 3.10 performing particle planar tasks. The normalized link lengths (dimensionless proportions of its total length L) are $l_1 = 0.4$, $l_2 = 0.\overline{3}$, and $l_3 = 0.2\overline{6}$, the same proportions used in [56]. The manipulator configuration is described by $\mathbf{q} = [q_1, q_2, q_3]^T$ as illustrated in Fig. 3.10.

Each joint motion $\dot{q}_i$ is controlled by an actuator motion $\dot{\phi}_i$ related by a transmission ratio: $\dot{\phi}_i = \rho_i \dot{q}_i$. Consider identical actuators for each joint, but with different transmission ratios, $\rho_1 = l_1 + l_2 + l_3$, $\rho_2 = l_2 + l_3$, $\rho_3 = l_3$, each based on the total link length beyond the joint. The instantaneous cost function for minimizing the actuator velocity norm is $\frac{1}{2}\|\dot{\boldsymbol{\phi}}\|^2 = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{W}\dot{\mathbf{q}}$, where

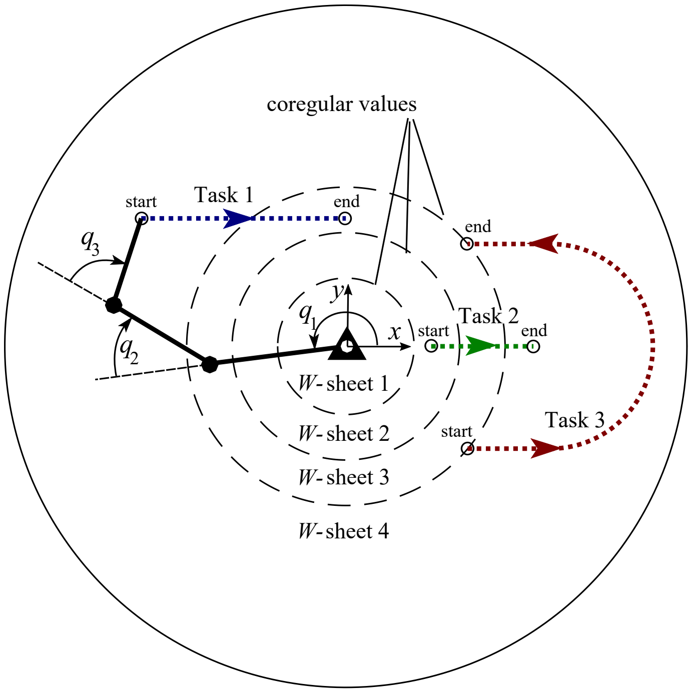


Figure 3.10: A 3R manipulator configuration in its workspace. The workspace is divided into W-sheets bounded by coregular values. Task points in W-sheets 2 and 4 have a single self-motion manifold, while task points in W-sheets 1 and 3 have two disjoint self-motion manifolds.

$$\mathbf{W} = \begin{bmatrix} \rho_1^2 & 0 & 0 \\ 0 & \rho_2^2 & 0 \\ 0 & 0 & \rho_3^2 \end{bmatrix}. \tag{3.15}$$

The global cost function is

$$G_{\text{global}}(\boldsymbol{q}(t)) = \int_{t=0}^{t=1} \frac{1}{2}\dot{\boldsymbol{q}}(t)^T \mathbf{W} \dot{\boldsymbol{q}}(t)\, dt. \tag{3.16}$$

The 3R manipulator configuration is singular if all three links are aligned. The singularities internal to the workspace have joint configurations at $\mathbf{q} = [q_1, \pi, 0]^T$, $\mathbf{q} = [q_1, \pi, \pi]^T$, and $\mathbf{q} = [q_1, 0, \pi]^T$, with any value for q_1. These singular cases yield end-effector configurations (coregular values) at distances $d_1 = |l_1 - l_2 - l_3|$, $d_2 = |l_1 - l_2 + l_3|$, and $d_3 = |l_1 + l_2 - l_3|$ from the base. The set of coregular values bounding the W-sheets are identified by the dashed rings in Fig. 3.10.

W-sheet 1 has two closed self-motion manifolds corresponding to the kinematics of a double-crank 4-bar mechanism with "elbow-up" and "elbow-down" poses. W-sheet 3 has two closed self-motion manifolds corresponding to the kinematics a crank-rocker 4-bar mechanism with "elbow-up" and "elbow-down" poses. W-sheet 2 and W-sheet 4 each have one closed self-motion manifold that corresponds to the kinematics of a 4-bar mechanism that cannot be fully driven by a single input link.

Each W-sheet is bounded by a W-sheet with a different number of self-motion manifolds. As such, for an arbitrary task path in the workspace, all regular task configurations have either one or two *closed* self-motion manifolds, and the number changes whenever a coregular value is crossed. The sub-task bifurcation branch roadmaps for closed self-motion manifolds presented in Section 3.4 are used in this case study.

3.7.2 Relatively Simple Tasks

Tasks 1, 2, and 3, shown in Fig. 3.10, were selected to show important features of the bb-algorithm and to illustrate real RIK solution spaces having the structural characteristics described in Section 3.4. Task 1 has a bifurcation singularity with a saddle point easily visible on its RIK solution space (the same structure as Figs. 3.4). Task 2 has the same RIK structure as Fig. 3.6 and has multiple locally optimal joint paths (and multiple within the same homotopy class), all of which are found by the bb-algorithm. Task 3 has the same RIK structure as Fig. 3.7 and is used to show multiple sub-optimal joint paths in different homotopy classes, each generated using the bi-directional instantaneous path planner.

Task 1

Task 1 in Fig. 3.10 crosses a single coregular value. It starts at a task instance with one self-motion manifold $\boldsymbol{q}_S(\psi)$ and ends at a task instance with two self-motion manifolds $\boldsymbol{q}_{T_a}(\psi)$, and $\boldsymbol{q}_{T_b}(\psi)$. The RIK solution space is shown as a sequence of self-motion manifolds in Fig. 3.11, where Fig. 3.11a is a top view, and Fig. 3.11b is a trimetric view. The top view looks like the structure in Fig. 3.4, in which the coregular self-motion (bold curve) is self-intersecting. The self-motion paths at time values after the coregular value, are pairs of "elbow-up"and "elbow-down" manifolds. These self-motion paths do not look like closed curves in Fig. 3.11b, but they are closed because when the manipulator performs a self-motion cycle, it returns to the same configuration after traveling 2π over q_3. The self-motion paths at time values before the coregular value are closed curves in Fig. 3.11b and are only shown for a single instance, though there are multiple instances separated by 2π over each joint coordinate q_i. For task paths crossing multiple coregular

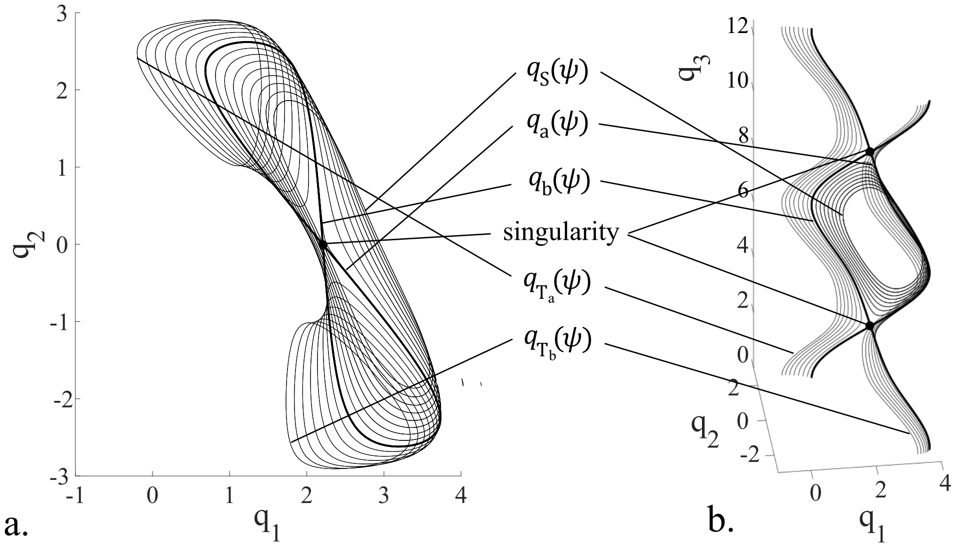


Figure 3.11: The RIK solution space of Task 1 shown as a sequence of self-motion paths. a) topview. b) trimetric view. Thin curves are regular self-motion paths. Thick curves are coregular self-motion paths.

values, it is important to account for equivalent self-motion manifolds separated by 2π over q_i.

Task 2

Task 2 in Fig. 3.10 crosses two coregular values and there are two disjoint self-motion manifolds for task points between the coregular values. The portion of the task between the coregular values has an RIK solution space with the same structure as that shown in Fig. 3.6 (if the “cylinders” in Fig. 3.6 were cut at the singularity connection paths and unrolled). Task 2, however, extends beyond the coregular values to better show different homotopy classes.

The RIK solution space of Task 2 is shown in Fig. 3.12. A joint path starting in the single self-motion manifold $\boldsymbol{q}_S(\psi)$ can either pass through coregular self-motion $\boldsymbol{q}_{1_a}(\psi)$ or $\boldsymbol{q}_{1_b}(\psi)$. A joint path passing through $\boldsymbol{q}_{1_a}(\psi)$, must also pass through $\boldsymbol{q}_{2_a}(\psi)$ traveling through the “elbow-up” RIK solution space. The dashed and dashed-dotted lines in Fig. 3.12 are the singularity connection paths obtained from reduced inverse kinematics constraining $q_3 = \pi$ to make twists 2 and 3 linearly dependent. There are two inverse kinematic solutions; one solution traces the singularity connection path on the “elbow-up” solution space and the other solution traces the singularity connection path on the “elbow-down” solution space. Both singularity connection paths are shown in two instances, separated by a difference of 2π in q_3.

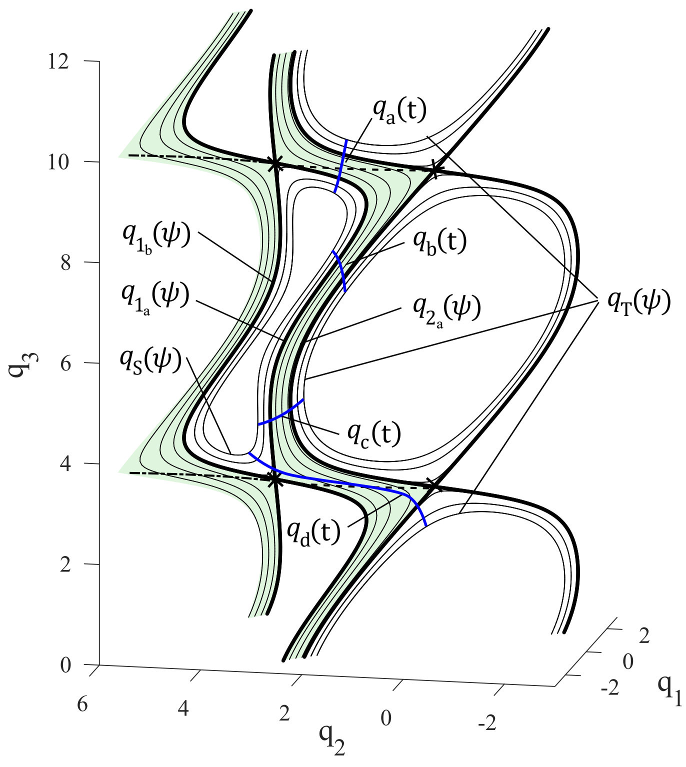


Figure 3.12: The RIK solution space of Task 2. The shaded regions are disconnected solution spaces for the sub-task between the coregular values. Thin curves are regular self-motion paths. Thick curves are coregular self-motion paths or locally optimal joint paths depending on the label. Dashed and dashed-dotted lines are singularity connection paths.

Four locally optimal paths ($\boldsymbol{q}_a(t)$, $\boldsymbol{q}_b(t)$, $\boldsymbol{q}_c(t)$, $\boldsymbol{q}_d(t)$) are shown on the RIK solution surface in Fig. 3.12. These paths were obtained using the bb-algorithm. Two of these paths, $\boldsymbol{q}_b(t)$, and $\boldsymbol{q}_c(t)$, do not cross the singularity connection path and are in the same homotopy class. It is easily seen that each task point in $\boldsymbol{q}_b(t)$ can be deformed along the self-motions into $\boldsymbol{q}_c(t)$. Paths $\boldsymbol{q}_a(t)$ and $\boldsymbol{q}_d(t)$ cross the singularity connection path in different directions; $\boldsymbol{q}_a(t)$ crosses the singularity connection path with an increasing q_3 value, whereas $\boldsymbol{q}_d(t)$ crosses the singularity connection path with a decreasing q_3 value. Although $\boldsymbol{q}_a(t)$, $\boldsymbol{q}_b(t)$, and $\boldsymbol{q}_d(t)$ terminate in the same self-motion manifold $\boldsymbol{q}_T(\psi)$, they terminate in manifolds separated by 2π in q_3. These paths cannot be continuously deformed into each other.

Task 3

Task 3 in Fig. 3.10 has task endpoints at coregular values, all other task configurations are inside W-sheet 4 and have a single self-motion manifold. The RIK solution space for this task is illustrated in Fig. 3.13 and has the same general structure as that of Fig. 3.7. Smooth joint paths

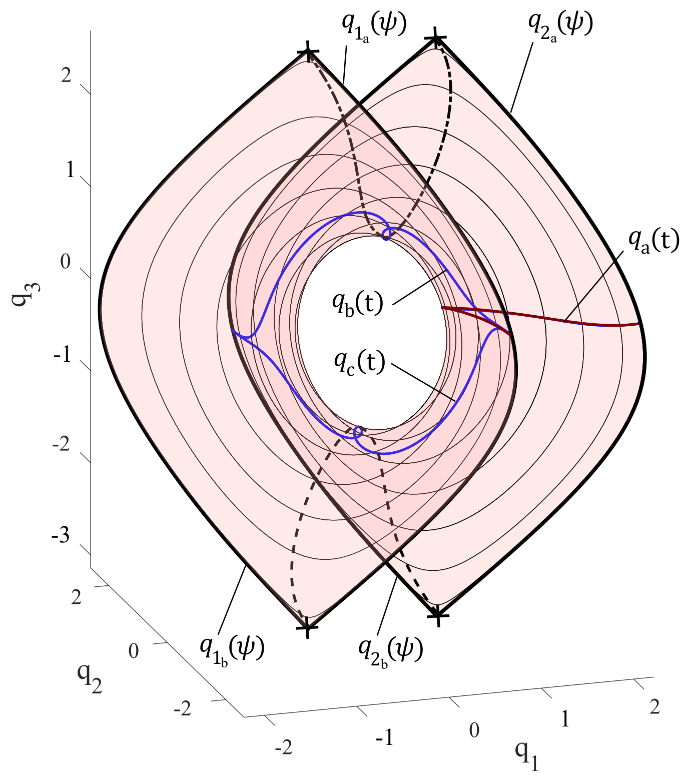


Figure 3.13: The RIK solution space of Task 3. Thin curves are regular self-motion paths. Thick curves are coregular self-motion paths or joint paths. Dashed and dashed-dotted lines are singularity connection paths.

$\boldsymbol{q}_a(t)$, $\boldsymbol{q}_b(t)$, and $\boldsymbol{q}_c(t)$ are generated using the bi-directional path planner, each starting from a joint configuration at the midpoint of the coregular self-motion path. Each of these paths has the same starting configuration and the same initial joint motion, but each tangentially diverges into a different homotopy class.

Path $\boldsymbol{q}_a(t)$ is a "direct path" connecting $\boldsymbol{q}_{1_a}(\psi)$ to $\boldsymbol{q}_{2_a}(\psi)$ it does not cross a singularity connection path. The dashed and dashed-dotted lines in Fig. 3.13 are the two singularity connection paths corresponding to reduced inverse kinematic solutions, where link 1 is oriented to point to the end-effector location (twists 1 and 3 are linearly dependent). There are two inverse kinematic solution cases corresponding to "wrist-up" and "wrist-down" cases of the two distal links. Paths $\boldsymbol{q}_b(t)$ and $\boldsymbol{q}_c(t)$ connect $\boldsymbol{q}_{1_a}(\psi)$ to $\boldsymbol{q}_{2_b}(\psi)$, but are in separate homotopy classes because they travel different directions around the solution surface, crossing different singularity connection paths.

Because of the symmetry of the solution space, attempting to bi-directionally generate a path between the midpoints of $\boldsymbol{q}_{1_a}(\psi)$ and $\boldsymbol{q}_{2_b}(\psi)$ yields bi-directionally generated paths that fail to meet, terminating at opposite ends of the same self-motion manifold. An intermediate point on the singularity connection path is used with the bi-directional path planner to control which way the joint path goes around the solution space.

3.7.3 General Task

Here the best locally optimal joint paths found by using the bifurcation branch algorithm (bb-algorithm) are compared to the best path found using a traditional multi-start method.

The multi-start method generates sub-optimal joint paths by integrating (3.8) with $\dot{\mathbf{q}}_N = \mathbf{0}$ from multiple equally spaced admissible joint configurations at the task start point. These sub-optimal paths are then deformed into locally optimal paths.

Consider a complex task with free boundary conditions and the path shown in Fig. 3.14 (defined by a cubic-spline) that crosses 12 coregular values. The bifurcation branch roadmap is constructed by linking 13 sub-task roadmaps (of Fig. 3.8) in the following order:

$$R_1^{S_1}, R_2^{C_2}, R_3^{C_1}, R_4^{C_2}, R_5^{C_1}, R_6^{C_2}, R_7^{C_1},$$

$$R_8^{C_2}, R_9^{C_1}, R_{10}^{C_2}, R_{11}^{C_1}, R_{12}^{C_2}, R_{13}^{T_1}$$

The total number of homotopy classes (without self-motion cycles in a single W-sheet), calculated using (3.4), is 4,556,250. The number of promising homotopy classes is reduced to 5,626 using the upper/lower bound method described in Sec. 3.3. Each initial joint path in the promising homotopy classes was deformed into a locally optimal path. The best of these, identified as the globally optimal path, has a cost value of 40.0 (evaluated using (3.16)).

Figure 3.14a shows snapshots of the manipulator tracking the best joint path found using the bifurcation branch algorithm. To avoid image clutter, the top image only shows the first half of the task and the bottom image only shows the second half of the task.

A multi-start method was used to generate 1,000 velocity-based (instantaneous) sub-optimal paths starting from equally spaced configurations on the starting self-motion manifold. The costs of the initial paths ranged from 57.3 to 140.7. Each path was deformed into a locally optimal joint path using the path deformation procedure described in Section 3.6 (using $k > 50$ sampled configurations to define the cubic spline of the joint path and using a task error threshold of 10^{-9}). With these inputs/tolerances, the path deformation procedure took about 250 times more computation time than the instantaneous path planner. The set of 1000 sub-optimal

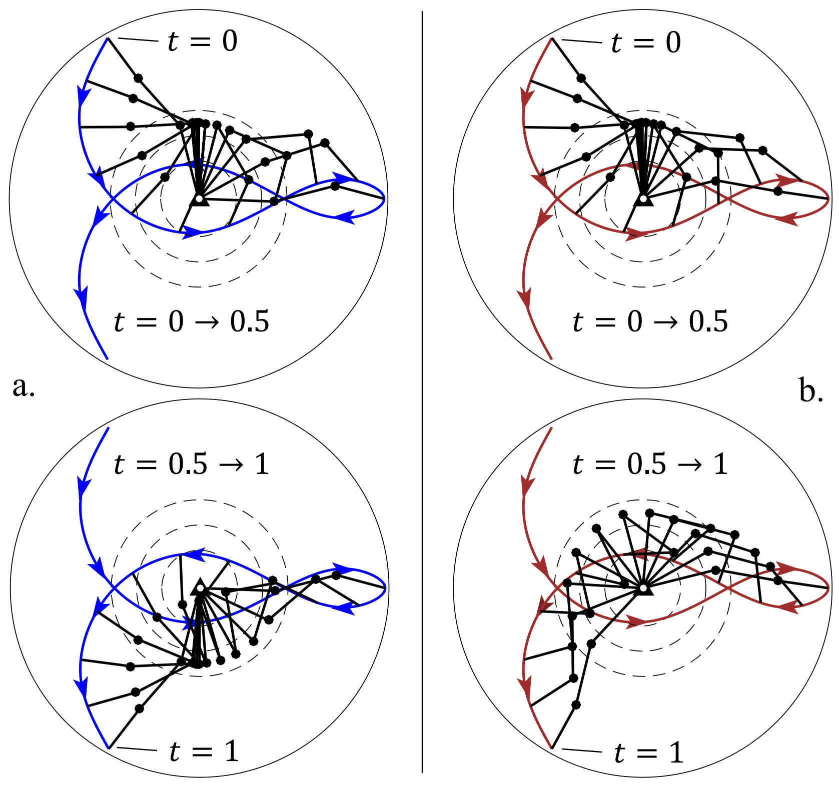


Figure 3.14: Stroboscopic image of the manipulator performing the best joint paths found using: a) bb-algorithm, and b) multi-start method (11th best path found by the bb-algorithm). The top images show manipulation of the first half of the task, the bottom images show manipulation of the second half of the task.

paths converged to a set of 7 unique locally optimal paths, each in a different homotopy class. The cost values of these locally optimal paths ranged from 44.7 to 67.6. The best path of these paths is shown in Fig. 3.14b. It is the same as the 11th best path found by the bifurcation branch algorithm. The bb-algorithm found 10 unique locally optimal paths that have lower cost values than the best path obtained from a rigorous multi-start method. The bb-algorithm took about 6 times longer than the multi-start method and found a joint path 10 percent better. Even with a very large number of seeds (starting configurations), the multi-start method failed to identify the globally optimal path.

The cost rates of the best paths found using the two methods are shown in Fig. 3.15. The best path (found by the bb-algorithm) requires that the manipulator switch from an "elbow-up" pose to an "elbow-down" pose over sub-task 7 ($R_7^{C_1}$) in W-sheet 4. However, crossing from $\boldsymbol{q}_{6_a}(\psi)$ to $\boldsymbol{q}_{7_b}(\psi)$ requires more joint motion as indicated by the higher cost rates in the middle of the tasks relative to the 11th best path, which remains in the "elbow-up" configuration. Although, the

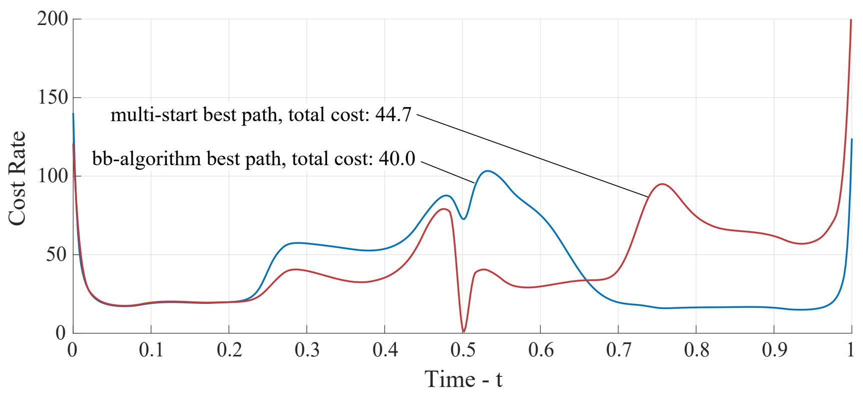


Figure 3.15: Cost Rates of the best joint paths found by the bifurcation branch algorithm and the multi-start method (11$^{\text{th}}$ best path found by the bifurcation branch algorithm).

multi-start optimal path has lower cost rates during the middle of the task $(0.22 < t < 0.66)$, it has higher cost rates at time greater than 0.66. The higher global cost value is reflected in Fig. 3.14, which shows greater joint motion for the multi-start optimal path during the second half of the task.

Paths generated using instantaneous optimization cannot find the homotopy class of the globally optimal path because they are unable to make local sacrifices to reap better returns later in the task.

3.8 Chapter Summary

This chapter presented an algorithm more capable than traditional methods for finding the globally optimal path of a redundant manipulator performing a task with a defined path. The globally optimal path is found using a custom "direct" method for solving optimal control problems. To find the globally optimal path, the direct solver must be initialized with a sub-optimal joint path in the same homotopy class as the globally optimal path. The homotopy class of the globally optimal joint path, however, can be extremely difficult to find without high-level knowledge of the solution space. The bifurcation branch algorithm identifies all homotopy classes for a task using high-level knowledge of the solution space, generates sub-optimal joint paths in each homotopy class, and deforms them into locally optimal paths. The set of possible homotopy classes in tasks with multiple bifurcations is reduced to a much smaller set of promising homotopy classes that are evaluated in detail.

CHAPTER 4
OPTIMAL GLOBAL RESOLUTION OF PASSIVE COMPLIANCE CONTROL WITH ONE DEGREE OF REDUNDANCY

This chapter addresses finding the optimal sequence of actuator commands for redundant serial manipulators with variable stiffness actuators (like that in Fig. 4.1) capable of passive compliance control. The compliance extended redundant inverse kinematic (RIKC) framework presented in Chapter 2 is combined with the bifurcation branch algorithm presented in Chapter 3 to identify the globally optimal joint path for manipulation with one degree of redundancy in the combined kinematic and compliance RIKC solution space.

4.1 Introduction

The bifurcation branch algorithm efficiently searches the entire feasible RIK(C) solution space (set of realizable joint configurations yielding end-effector configurations on the task path) using the bifurcation branch roadmap (bb-roadmap). The bb-roadmap is briefly reviewed below, the challenges of constructing the bb-roadmap for RIKC problems are discussed, and a chapter overview is provided.

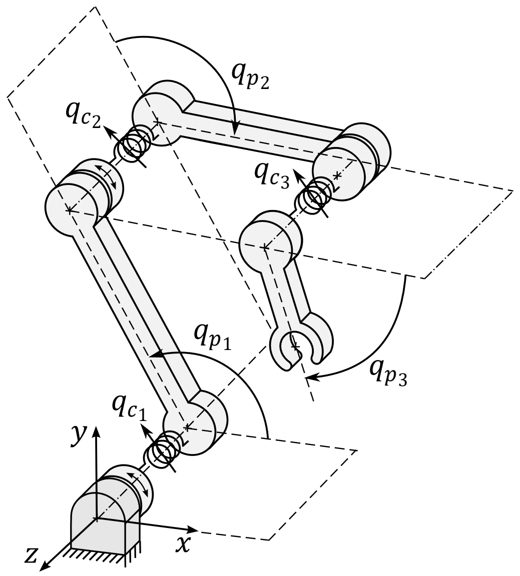


Figure 4.1: Planar serial manipulator with variable stiffness actuators (VSAs). For each joint i, the primary actuator controls the kinematic joint position q_{p_i} and the VSA controls the joint compliance q_{c_i}.

4.1.1 Bifurcation Branch Roadmap

The bifurcation branch roadmap (bb-roadmap) is a directed graph that compactly characterizes the connectivity of the RIK(C) solution space. Each node of the bb-roadmap is a feasible self-motion path of either a task endpoint, a bifurcation point, or a premature termination point. Each edge of the bb-roadmap is a homotopy class[1] of joint paths that satisfy the portion of the task path between the nodes. Each route (sequence of edges connecting a task start node to a task terminal node) corresponds to a different homotopy class of joint paths solving the RIK(C) path planning problem.

The bifurcation branch roadmap depends on the number and structure of the self-motion manifolds (connected sets of joint configurations yielding the same task configuration), and depends on joint limits. Figure 4.2a shows an example RIK solution space where a single closed self-motion manifold (each thin solid line is a self-motion) splits into two disjoint closed self-motion manifolds at the singularity ($\times$). When joint limits are present, the feasible RIK solution space is truncated such that each self-motion manifold has 0, 1, or multiple feasible self-motion paths (connected subsets of joint configurations attainable by the manipulator hardware). The simple feasible RIK solution space in Fig. 4.2b shows bifurcation points ($\times$) where feasible self-motion paths (shaded vertical cross-sections) split or converge. It also shows a premature termination point ($\otimes$), where a feasible self-motion path vanishes as time progresses. The corresponding bb-roadmaps of Figs. 4.2a and 4.2b are shown in Figs. 4.3a and 4.3b, respectively.

The bifurcation branch roadmap construction process presented in Chapter 3 for a class of RIK path planning problems with closed self-motion manifolds, is not adequate for RIKC path planning problems. This chapter provides means of constructing bb-roadmaps for RIKC path planning problems with joint limits.

4.1.2 Characterizing the Connectivity of RIKC Solution Spaces

The RIKC path planning problem has a more complex solution space than the RIK problem. The joint configuration subspace for compliance has a dramatically different structure than that for kinematics. Each kinematic joint variable with a revolute joint has a closed manifold (a full joint rotation returns the link to the same configuration); whereas, each compliance joint variable has an open manifold. The total joint configuration manifold combines these different manifold structures. Self-motion manifolds are sub-manifolds in the total joint configuration

[1]Joint paths that cannot be deformed into each other are in different homotopy classes.

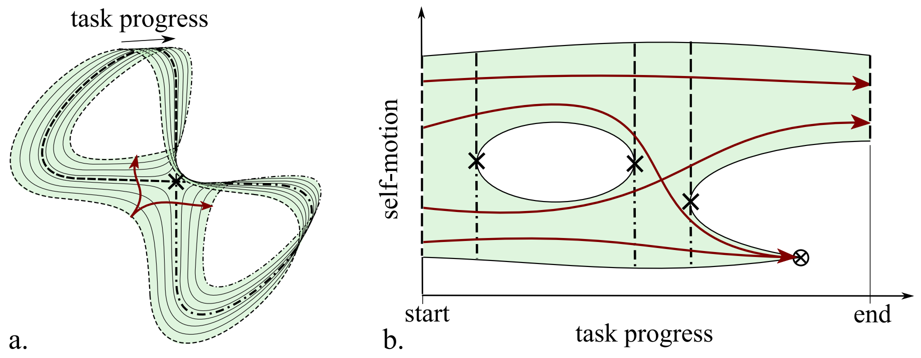


Figure 4.2: Bifurcations in feasible RIK solution spaces for one degree of redundancy. Each $\times$ is a bifurcation point, each $\otimes$ is a premature termination point. Dashed and dashed-dotted lines are feasible self-motion paths on one side of the bifurcation point or self-motion paths of task endpoints. Solid lines with arrow endpoints are joint paths in different homotopy classes. a) Bifurcation from a singularity on an RIK solution space with closed self-motion manifolds. b) Bifurcations and premature termination from joint limits on an RIK solution space with open self-motion manifolds.

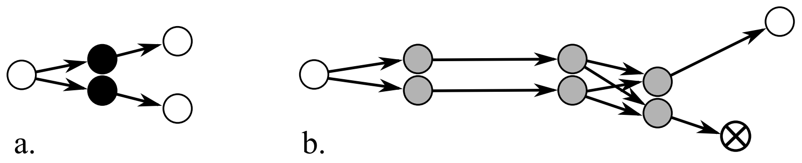


Figure 4.3: Bifurcation branch roadmaps (bb-roadmaps) corresponding to RIK solution spaces in Fig. 4.2. White nodes correspond to the feasible self-motion paths of task endpoints, pairs of black nodes correspond to the feasible self-motion paths on either side of a singularity bifurcation point, pairs of gray nodes correspond to the feasible self-motion paths on either side of a joint limit bifurcation point, and $\otimes$ corresponds to a premature termination point.

manifold. The number and structure[2] of self-motion manifolds (sub-manifolds in the total combined configuration space) are different from the number and structure of *kinematic* self-motion manifolds. The number of bifurcations from singularities is different too. Additionally, each VSA has hardware limits (a finite range of joint compliances) that remove many joint configurations from the feasible RIKC solution space, possibly adding more joint limit bifurcation points and premature termination points.

The challenges associated with developing the bb-roadmap in the combined solution space are: 1) identifying the number and structure of self-motion manifolds and the feasible self-motion paths on them, 2) identifying bifurcation nodes associated with singularities, 3) identifying bifurcation nodes associated with joint limits, and 4) identifying edges connecting nodes. The first challenge relates directly to identifying endpoint nodes, but also relates to identifying edges as this

[2]The "structure" of a 1-dimensional self-motion manifold is either open or closed.

information is important for understanding the bifurcation structure and its impact on homotopy classes.

The number of self-motion manifolds impacts the number of endpoint nodes in the bb-roadmap (white nodes). The structure of the self-motion manifolds impacts the number of edges connecting nodes. This information has been identified for the *kinematic* self-motion manifolds of many types of manipulators (e.g., [51] [73] [74]). However, the number and the structure of self-motion manifolds in the combined kinematic and compliance configuration space has not been identified for any type of manipulator. This chapter identifies this information by combining kinematic self-motion manifolds with compliance realization conditions (e.g., [22] [23] [24] [25] [26] [27]).

The identification of singular joint configurations (and their corresponding end-effector configurations) is necessary for finding the bifurcation branches of singularities (black nodes). In RIK problems, these features are identified using known geometric conditions for singularities (and for the task instances at which they occur). However, the geometric conditions for singularities and their task instances in the combined kinematic and compliance configuration (RIKC) spaces are not known for any type of manipulator. This chapter identifies this information using compliance realization conditions (e.g., [22] [23] [24] [25] [27]) for simpler manipulators that can achieve the same elastic behavior as the redundant manipulator.

The identification of bifurcations from joint limits is necessary for identifying joint limit bifurcation branches (gray nodes in Fig. 4.3). Chapter 3 did not address identifying joint limit bifurcation points. Because joint compliances are always lower and upper bounded, joint limits must be accounted for in the compliance subspace. Joint limit bifurcation points are identified here for differentiable task manipulation paths by tracking the boundary edges of the RIK(C) solution space using an instantaneous path planner based on a modified Saturation in the Null Space (SNS) algorithm [42]. The boundary edge path planner terminates at: 1) a task endpoint, 2) a premature termination point, or 3) a joint limit bifurcation point. The boundary edge path planner can be used in both RIK and RIKC path planning problems.

In addition to describing new procedures for identifying roadmap nodes, new procedures are also developed for identifying roadmap edges. The methods described in Chapter 3 for identifying edges of the bb-roadmap are limited to RIK problems with 1 or 2 closed self-motion manifolds at each task instance. New methods of identifying edges are needed to address the greater number of disjoint self-motion manifolds in RIKC problems. This chapter presents an algorithm for constructing the bb-roadmap edges for RIK(C) problems with any number of *open* self-motion manifolds and with bifurcations from both singularities and joint limits.

4.1.3 Chapter Overview

This chapter provides means of constructing the bifurcation branch roadmap (bb-roadmap) that identifies the connectivity of the solution space for RIKC path planning with one degree of redundancy. Strategies for identifying the number and the structure of self-motion manifolds, and their bifurcations from singularities and their bifurcations from joint limits are described. These strategies are used to generate the nodes of the bb-roadmap. Strategies for identifying the edges of the bb-roadmap are also provided.

Section 4.2 provides means of identifying the number and the structure of self-motion manifolds in the total joint space (kinematic and compliance coordinates). Section 4.3 describes procedures for identifying self-motion bifurcations from singularities in the total joint space. Section 4.4 provides a method of quickly identifying bifurcation points that result from joint limits. Section 4.5 describes a new algorithm for generating the bb-roadmap for manipulation with open self-motion manifolds and with bifurcations from both singularities and joint limits. Section 4.6 demonstrates the procedures and provides the results of a case study in which a 3R-VSA manipulator is used to turn a crank despite geometric uncertainties. Section 4.7 provides a brief summary and conclusion.

4.2 RIKC Self-Motion Manifolds

As stated previously, the main challenge in passive compliance control is to solve the redundant inverse kinematics and compliance (RIKC) path planning problem. There are multiple ways for a joint path to traverse the RIKC solution space's complicated connection structure. Characterizing the connectivity of the RIKC solution space is important for identifying the globally optimal joint path for passive compliance control.

The first aspect in characterizing the connectivity of the RIKC solution space is to identify the number and structure of the self-motion manifolds of regular task points on the task manipulation path. This information relates directly to generating the bb-roadmap nodes at the task endpoints (white start and terminal nodes). This section describes a strategy for identifying the number and structure of self-motion manifolds for RIKC problems with one degree of redundancy. Each disjoint self-motion manifold is distinguished using an identification number that is assigned based on a potential homotopy class. The self-motion manifold identification numbers are later used to identify bb-roadmap edges. The strategy is first described in general, then the concepts are demonstrated for a 3R-VSA manipulator like that in Fig. 4.1.

4.2.1 General Strategy

Because the compliance mapping depends on the kinematic joint configuration, but the kinematic mapping is independent of the joint compliance configuration, the kinematic self-motion manifolds are identified first. Although the kinematic joint configurations are connected on each kinematic self-motion manifold, the total joint configurations in the *combined* joint space are not continuously connected in the compliance subspace. The kinematic self-motion manifolds are divided into regions where the compliance configurations are connected.

Consider a total task configuration on the task manipulation path:

$$\mathbf{x} \in \boldsymbol{x}(t) = \begin{bmatrix} \mathbf{x}_p \in \boldsymbol{x}_p(t) \\ \mathbf{x}_c \in \boldsymbol{x}_c(t) \end{bmatrix}.$$

The preimage of the kinematic task configuration is

$$\boldsymbol{f}_p^{-1}(\mathbf{x}_p) = \bigcup_{i=1}^{n_s} \mathcal{M}_{p_i}(\boldsymbol{\psi}) \tag{4.1}$$

where n_s is the number of disjoint kinematic self-motion manifolds $\mathcal{M}_{p_i}(\boldsymbol{\psi})$. If available, analytical descriptions of the kinematic self-motion manifolds are used. Otherwise, numerical descriptions of the kinematic self-motion manifolds are used (e.g., parameterized by arc length using (3.5)).

The preimage of the compliance task configuration given the kinematic joint configuration,

$$\boldsymbol{f}_c^{-1}(\mathbf{x}_c) \mid \mathbf{q}_p \in \mathcal{M}_{p_i}(\boldsymbol{\psi}), \tag{4.2}$$

is unique for many manipulation cases ([22] [23] [25] [26]). The joint compliance values for these cases are readily identified by joint compliance synthesis formulas related to the compliance realization conditions. For manipulation cases with equality constraints in its compliance realization conditions, the equality constraints are used to identify a one-dimensional sub-manifold (path) $\boldsymbol{q}_{p_i}(\psi)$ of the kinematic self-motion manifold.

The joint synthesis formulas provided in [22], [23], [25], [26], evaluated over ψ, have discontinuities when any two twists align. The twists of any two joints (say joint i and joint j) align when

$$\mathbf{t}_j = \alpha \mathbf{t}_i, \tag{4.3}$$

where α is a scalar.

Because of the discontinuity, the total joint configurations are not continuously connected over the kinematic self-motion manifold. Instead, each set of continuously connected joint configurations on either side of the discontinuity is an *open* self-motion manifold. The number of self-motion manifolds for the task configuration $\mathbf{x} = [\mathbf{x}_p^T, \mathbf{x}_c^T]^T$ depends on the number of kinematic self-motion manifolds of $\mathbf{x}_p$ and on the number of twist alignment cases over each kinematic self-motion manifold. The bounding twist alignment cases of each self-motion manifold and the associated kinematic self-motion manifold determine the identification number.

The number of feasible self-motion paths (the connected subsets of *feasible* joint configurations on the self-motion manifolds) may be different. Each self-motion manifold may have 0, 1, or multiple disjoint feasible self-motion paths. The feasible self-motion manifolds are identified using a one dimensional search over each self-motion manifold, checking if the joint configuration is feasible. Bounds (endpoints) of the feasible self-motion paths are joint configurations with a joint variable saturated at its limit. The feasible self-motion paths at the task endpoints are used as end-point nodes in the bb-roadmap.

This strategy of identifying self-motion manifolds is illustrated below for a 3R-VSA manipulator capable of performing a task in which the end-effector follows a desired path in both position and compliance.

4.2.2 3R-VSA Manipulator

Consider any end-effector position inside a single kinematic W-sheet for which the distal link of the manipulator can fully rotate without changing the end-effector position. The *kinematic* self-motion manifolds are characterized by the analytical formulas of a four-bar Grashof mechanism, parameterized by the end-effector orientation $\psi = q_{p_1} + q_{p_2} + q_{p_3}$. There are two disjoint kinematic self-motion manifolds corresponding to the “elbow-up” and “elbow-down” solutions. Each of these kinematic self-motion manifolds is a *closed* manifold, where a manipulator performing a self-motion returns to it's original kinematic configuration after a full rotation of the end-effector.

Given the kinematic joint configuration (parameterized by ψ and the elbow pose, i.e., "up" or "down"), the joint compliance configuration is resolved by the joint compliance synthesis equations provided in [22]. Because the joint compliance synthesis equations yield a *unique* joint compliance configuration (for all regular task configurations in which two twists do not align), the kinematic self-motion parameter ψ is suitable as the self-motion parameter that resolves both kinematics and compliance.

Although the joint compliance configurations are unique for each kinematic configuration, the joint compliance configurations are not continuously connected over the *kinematic* self-motion manifolds. The joint compliance synthesis formulas [22] yield undefined joint compliance values at twist alignment cases.

There are four relevant twist alignment cases:

$$\begin{aligned}
\mathrm{a}^+: &\quad \mathbf{t}_2 || \mathbf{t}_3 \text{ and } \alpha > 0,\\
\mathrm{b}^+: &\quad \mathbf{t}_1 || \mathbf{t}_3 \text{ and } \alpha > 0,\\
\mathrm{a}^-: &\quad \mathbf{t}_2 || \mathbf{t}_3 \text{ and } \alpha < 0,\\
\mathrm{b}^-: &\quad \mathbf{t}_1 || \mathbf{t}_3 \text{ and } \alpha < 0.
\end{aligned}$$

Each kinematic self-motion manifold ("elbow-up" and "elbow-down") has kinematic configurations corresponding to these twist alignment cases. The four kinematic configurations for the "elbow-up" manifold are shown in Fig. 4.4a. The self-motion parameter value ψ^* for each twist alignment case ($\psi^* = \psi_{\mathrm{a}^+}$, ψ_{b^+}, ψ_{a^-}, or ψ_{b^-}) is shown in Fig. 4.4b.

The joint compliances associated with the aligned twists, approach $\pm\infty$ as ψ approaches ψ^* from the left, and approach $\mp\infty$ as ψ approaches ψ^* from the right. Because the compliance values diverge, the self-motion manifold is *open.* Therefore, the preimage of a regular task configuration is composed of eight open self-motion manifolds:

$$\boldsymbol{f}^{-1}(\mathbf{x}) = \bigcup_{i=1}^{8} \boldsymbol{q}_i(\psi), \tag{4.4}$$

where $\boldsymbol{q}_i(\psi)$, $i = 1$ to 4 is a self-motion manifold in the elbow-up pose (shown in Fig. 4.4b) and $\boldsymbol{q}_i(\psi)$, $i = 5$ to 8 is a self-motion manifold in the elbow-down pose. Each subscript value is the self-motion manifold identification number corresponding to a unique set of bounding twist

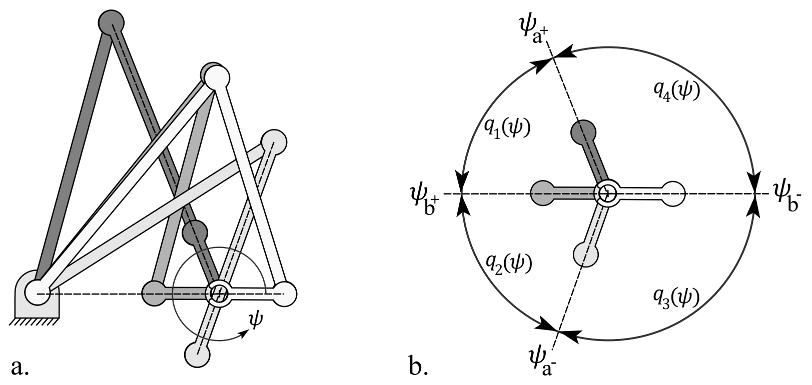


Figure 4.4: Twist alignment cases of the elbow-up self-motion manifold. a) Kinematic configurations of the twist alignment cases in order of darkest to lightest: a$^+$, b$^+$, a$^-$, b$^-$. b.) Elbow-up self-motion manifolds: $\boldsymbol{q}_i(\psi)$, $i = $1-4 separated by kinematic self-motion parameter values: ψ_{a^+}, ψ_{b^+}, ψ_{a^-}, ψ_{b^-}.

alignment cases and elbow pose. Note, the number and type of twist alignment cases is different for task configurations in W-sheets for which the end-effector cannot make a full rotation.

Although each self-motion manifold is unbounded in joint space, the manifold is fully captured by the bounded kinematic self-motion parameter $\psi \in (\psi_{\mathrm{a}^\pm}, \psi_{\mathrm{b}^\pm})$. Each self-motion manifold is unbounded in that each joint compliance may range between $\pm\infty$. However, the compliance of each joint-i has practical limits $q_{c_i} \in [q_{c_{i_{\min}}}, q_{c_{i_{\max}}}]$, $q_{c_{i_{\min}}} \geq 0$. Therefore, the feasible regions on self-motion manifolds are necessarily bounded by joint configurations with a joint compliance saturated at its limit.

4.3 Bifurcations from Singularities

To characterize the connectivity of the RIKC solution space, bifurcation nodes associated with singularities must be identified. This involves 1) identifying the task instances at which they occur, i.e., coregular values, 2) identifying the coregular self-motion paths at the coregular values, 3) identifying the singularity and splitting each coregular self-motion path into separate "branches", on either side of the singularity.

This section first describes the bifurcation structure of the self-motion manifolds presented in Section 4.2 and describes its relationship to homotopy classes of joint paths. The details for identifying each coregular value and its corresponding coregular self-motion paths and singularity are then provided.

4.3.1 Bifurcation Structure

For RIKC problems addressed here, the disjoint self-motion manifolds are open manifolds, each bounded by kinematic configurations associated with a twist alignment. Consider self-motion manifolds $\boldsymbol{q}_{\mathrm{i}}(\psi)$ and $\boldsymbol{q}_{\mathrm{j}}(\psi)$ that are bounded by different pairs of twist alignment cases, but with a shared case (e.g., $\boldsymbol{q}_1(\psi)$ and $\boldsymbol{q}_2(\psi)$ in Fig. 4.4b). As the task progresses, these disjoint self-motion manifolds converge/split at a coregular value. The bifurcation structure, associated with two independent open coregular self-motion paths, is illustrated by the RIKC solution space[3] shown in Fig. 4.5a. The bifurcation branch roadmap for this structure is shown in Fig. 4.5b. In the RIKC solution space, all self-motion manifolds (solid lines) to the left of the dashed-dotted plane correspond to self-motion manifolds $\boldsymbol{q}_{\mathrm{i}}(\psi)$, and those to the right correspond to $\boldsymbol{q}_{\mathrm{j}}(\psi)$, where i and j denote the identification numbers. Two independent open coregular self-motion paths exist, one with kinematic self-motion (dashed line) and one without kinematic self-motion (dashed-dotted line). The intersection point of these coregular self-motion paths is the singularity ($\times$). Each coregular self-motion path is separated into two paths at the singularity; these are bifurcation branches (black nodes) in the bb-roadmap. This bifurcation structure has four branches per singularity, whereas the structure associated with closed manifolds in Fig. 4.2a has two branches per singularity.

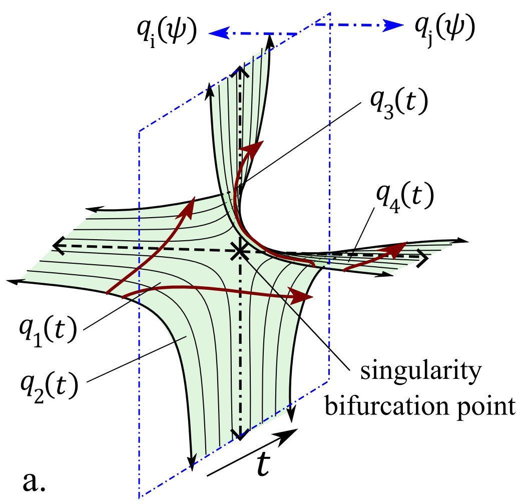
Figure 4.5a shows four joint paths (solid lines with arrow endpoints). Two paths start at a regular self-motion manifold $\boldsymbol{q}_{\mathrm{i}}(\psi)$ (left of plane), and two paths start at a regular self-motion manifold $\boldsymbol{q}_{\mathrm{j}}(\psi)$ (right of plane). The manipulator's joint configuration can switch from being in a self-motion manifold $\boldsymbol{q}_{\mathrm{i}}(\psi)$ to $\boldsymbol{q}_{\mathrm{j}}(\psi)$ (or $\boldsymbol{q}_{\mathrm{j}}(\psi)$ to $\boldsymbol{q}_{\mathrm{i}}(\psi)$) only if the joint path crosses the vertical dashed-dotted line corresponding to a coregular self-motion path for which there is no kinematic motion. Each of the joint paths shown has a different combination of start and terminal self-motion manifold identification numbers and is in a different homotopy class. For each case, the homotopy class of the joint path is identified by the coregular self-motion branch that the path crosses.

4.3.2 Identifying Coregular Values

The strategy for identifying coregular values is based on investigating twist alignment cases and the compliance realization conditions for a simpler manipulator with fewer joints.

The task compliance of a serial manipulator with n_p compliant joints is given by

[3]Figure 4.5 shows the RIKC solution space of the self-motion manifolds with the shared twist alignment case; the other self-motion manifolds have their own non-bifurcating RIKC solution regions (not shown).



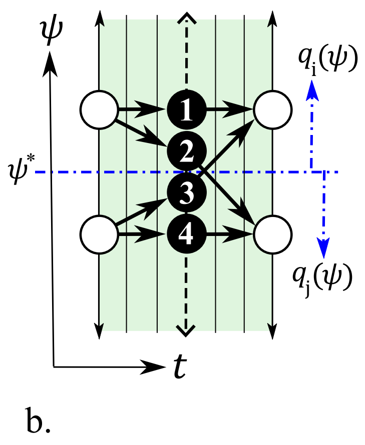


Figure 4.5: RIKC singularity bifurcation structure. a) RIKC solution space near a bifurcation point associated with a singularity. Solid lines are regular self-motion manifolds, dashed and dashed-dotted lines are coregular self-motion paths, and their intersection ($\times$) is the bifurcation point. Joint paths (solid lines with arrow endpoints) are in different homotopy classes. b) Rotated top view of Fig. 4.5a such that kinematic self-motion is in the vertical direction and time t is in the horizontal direction. The corresponding bb-roadmap is superimposed on the surface. Each joint path $\boldsymbol{q}_k(t)$ crosses a different bifurcation branch associated with a numbered black node.

$$\mathbf{C}_x = q_{c_i}\mathbf{t}_i\mathbf{t}_i^T + q_{c_j}\mathbf{t}_j\mathbf{t}_j^T + q_{c_k}\mathbf{t}_k\mathbf{t}_k^T + \cdots + q_{c_z}\mathbf{t}_z\mathbf{t}_z^T, \tag{4.5}$$

where $i, j, k, \cdots z$ are joints $1, 2, 3, \cdots, n_p$ in no particular order.

If twists i and j align, (4.3) is substituted into (4.5), yielding

$$\mathbf{C}_x = (q_{c_i} + \alpha^2 q_{c_j})\mathbf{t}_i\mathbf{t}_i^T + q_{c_k}\mathbf{t}_k\mathbf{t}_k^T + \cdots + q_{c_z}\mathbf{t}_z\mathbf{t}_z^T, \tag{4.6}$$

where $\mathbf{t}_k \cdots \mathbf{t}_z$ are the independent twists and

$$q_{c_c} = (q_{c_i} + \alpha^2 q_{c_j}) \tag{4.7}$$

is the combined compliance of joints i and j as seen from the twist associated with joint-i.

The task compliance expressed in (4.6) can be realized by a simpler manipulator with one less joint. The simpler manipulator has separate, more limiting, compliance realization conditions. Similar procedures are used if more than two joints align.

The twist alignment cases that can realize a task configuration on the task manipulation path are identified by searching (with root finding or optimization) for instances that satisfy the compliance realization conditions of the simpler manipulator over the task manipulation path $\boldsymbol{x}(t)$.

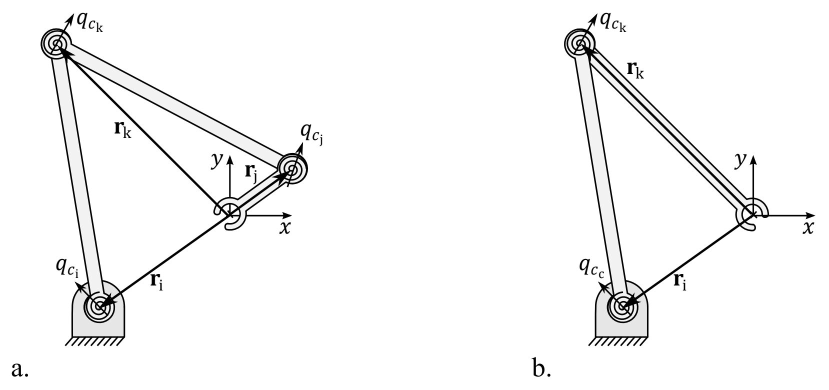


Figure 4.6: Twist alignments. a) 3R-VSA manipulator at twist alignment case b$^-$. b) 2R-VSA manipulator capable of achieving the same particle elastic behavior.

The compliance realization conditions are evaluated using the inverse kinematic solutions of each twist alignment case of the redundant manipulator. Each case may have 0, 1, or multiple task instances for which the compliance realization conditions are satisfied.

The steps used in identifying the coregular values of the 3R manipulator performing a particle planar task in the kinematic dexterous workspace are described below.

3R-VSA Manipulator

The particle-planar task compliance of a 3R-VSA manipulator end-effector is given by (4.5) where i, j, and k are joints 1, 2, and 3 in no particular order.

For a singularity to occur, two twists must align and the third twist must satisfy an additional requirement. Each singularity case must satisfy (4.6) and must satisfy a 2R compliance realization condition. For example, the 3R manipulator with an elbow-up pose and twist alignment case b$^-$ in Fig. 4.6a has the same compliant behavior as the 2R manipulator in Fig. 4.6b.

According to the compliance realization condition in [22], the 2R-VSA manipulator can achieve a specified task compliance $\mathbf{C}_x$ if and only if

$$\mathbf{r}_i^T \mathbf{C}_x \mathbf{r}_k = 0. \tag{4.8}$$

The calculated joint compliances q_{c_c} and q_{c_k} for the 2R realization are unique [22]. The compliance values for the 3R manipulator at this kinematic configuration are not unique. At these configurations a self-motion path exists in the compliance subspace.

The coregular values on the task manipulation paths are readily obtained by finding the roots of (4.8) over $\boldsymbol{x}(t)$ where $\mathbf{r}_i$ and $\mathbf{r}_k$ are defined by the unique inverse kinematic solutions of the twist alignment cases.

4.3.3 Coregular Self-Motion without Kinematic Motion

At each coregular value, there exists a normal self-motion path $\boldsymbol{q}(\psi)$ (described in Section 4.2) with motion in the combined kinematic and compliance space and there exists another self-motion path $\boldsymbol{q}(\psi_c)$ for which there is no kinematic self-motion. These paths are coregular self-motion paths intersecting at the bifurcation singularity. Coregular self-motion paths exist when a twist alignment occurs (4.3) and the compliance realization conditions of the effective simpler manipulator (e.g., condition (4.8) for the 2R manipulator) are satisfied.

For the case of two aligned joints, joint compliances of the aligned joints, q_{c_i} and q_{c_j} can linearly trade-off their value while keeping q_{c_c} constant. Let $\psi_c \in (-\infty, \infty)$ be the *compliance self-motion parameter* that resolves this trade-off:

$$q_{c_i} = \psi_c q_{c_c} \tag{4.9}$$

and

$$q_{c_j} = \frac{(1-\psi_c)}{\alpha^2} q_{c_c}. \tag{4.10}$$

The self-motion path $\boldsymbol{q}(\psi_c)$ is defined by the task configuration $\mathbf{x}$, the twist alignment case, and ψ_c. The compliances of the aligned twists q_{c_i}, q_{c_j} are resolved by ψ_c using (4.9) and (4.10). The self-motion path $\boldsymbol{q}(\psi_c)$ is linear in joint space.

The bounds of the self-motion parameter ψ_c are identified by substituting joint limits into (4.9-4.10). The self motion parameter bounds due to the compliance limits of joint-i and joint-j are

$$\left[\psi_{c_{i_{\min}}}, \psi_{c_{i_{\max}}}\right] = \left[\frac{q_{c_{i_{\min}}}}{q_{c_c}}, \frac{q_{c_{i_{\max}}}}{q_{c_c}}\right] \tag{4.11}$$

and

$$\left[\psi_{c_{j_{\min}}}, \psi_{c_{j_{\max}}}\right] = \left[1-\alpha^2 \frac{q_{c_{j_{\max}}}}{q_{c_c}}, 1-\alpha^2 \frac{q_{c_{j_{\min}}}}{q_{c_c}}\right], \tag{4.12}$$

respectively.

The bounds of the feasible self-motion path is given by

$$[\psi_{c_{\min}}, \psi_{c_{\max}}] = \left[\max(\psi_{c_{i_{\min}}}, \psi_{c_{j_{\min}}}), \min(\psi_{c_{i_{\max}}}, \ \psi_{c_{j_{\max}}})\right]. \tag{4.13}$$

A feasible self-motion path exists if $\psi_{c_{\min}} < \psi_{c_{\max}}$, and if the other joint coordinate values $(\mathbf{q}_p, q_k, \ldots, q_z)$ are all feasible. Similar results are obtained when more than two joints align.

4.3.4 Identifying the Singularity

The coregular self-motion $\boldsymbol{q}(\psi)$ intersects $\boldsymbol{q}(\psi_c)$ at ψ^* when (4.3) is satisfied. The compliance configuration, normally resolved over ψ by the compliance synthesis formulas are not valid at the twist alignment case. They are, however, valid at ψ values a small step to either side of ψ^*. The compliance configurations resolved on either side of ψ^* are used to approximate the intersecting joint configuration on $\boldsymbol{q}(\psi_c)$. The singular configuration is then identified by maximizing the condition number of the total Jacobian matrix over ψ_c. The singularity may or may not be a feasible joint configuration. If, the singularity is not a feasible joint configuration the bifurcation does not occur at the coregular value. Instead, the bifurcation results from a joint limit at a task time before (or after) the coregular value.

4.4 Bifurcations from Joint Limits

To characterize the connectivity of the RIKC solution space, bifurcation nodes associated with joint limits must be identified. This section describes a method to quickly identify bifurcation points (and premature termination points) resulting from joint limits in RIK and RIKC solution spaces with one degree of redundancy. The method involves using a *boundary edge path planner*, an instantaneous path planner that tracks the a boundary edge of the feasible RIK(C) solution space.

The boundaries of the feasible RIK(C) solution space result from truncation by a joint limit, as illustrated in Fig. 4.7a, by the joint-limit hyperplane (dashed-dotted plane) truncating the self-motion manifolds (solid lines). Figure 4.7b illustrates *conceptually*[4] a feasible RIKC solution space with several truncated regions, and the bifurcation ($\times$) at t_1 corresponds the bifurcation in Fig. 4.7a. Each continuous and smooth boundary edge (edge of the shaded region that is not a dotted line) of the feasible RIK(C) solution space corresponds to saturation of a single joint variable. Each path (thick solid line with arrows) tracking the boundary edge of the feasible solution space is generated from a starting configuration (circle dot) forward and backward in time (arrows indicate the direction of path generation). The starting configurations are boundary configurations of previously identified feasible self-motion paths (dotted lines). Each velocity-based generated path is generated forward in time until $t = 1$ (or backward in time until $t = 0$), or until the path can no longer be generated without leaving the boundary edge. If the path cannot be

[4]The illustrated surface is an abstract representation of the RIKC solution space, a 2-dimensional subspace of a much higher dimensional space (e.g., the 6 dimensional space of the 3R-VSA manipulator; 3 kinematic and 3 compliant). The 2-dimensional subspace is minimally parameterized using self-motion and time.

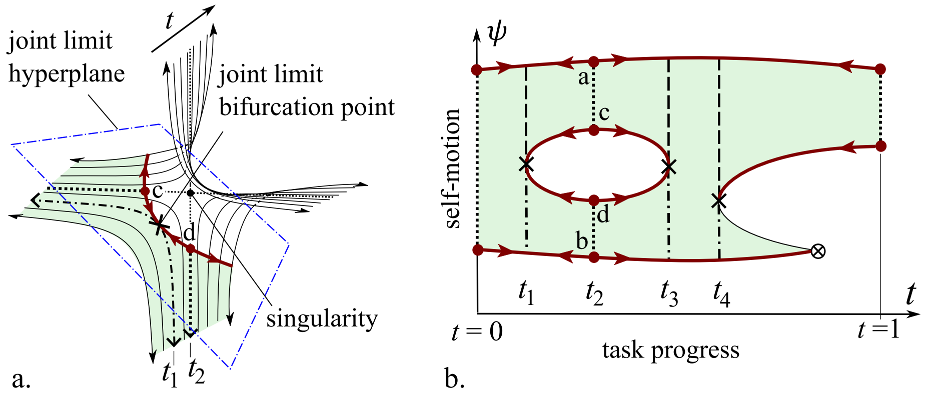


Figure 4.7: Boundary edges from joint limits. a) Joint limit hyperplane truncating the RIK(C) solution space (shaded surface is the feasible region). b) An RIK(C) solution space with several truncated regions. Thick solid lines are instantaneously generated paths starting from boundary configurations (circles) of previously identified self-motion paths (dotted lines)

generated further, the final joint configuration of the path is either a bifurcation point ($\times$) or a premature termination point ($\otimes$). The boundary edge path planner and the selection of starting points are described below.

4.4.1 Boundary Edge Path Planner

The boundary edge path planner is based on instantaneous RIK(C) resolution using the SNS algorithm (reviewed in Chapter 2).

Recall from Chapter 2, a joint path is generated by integrating

$$\dot{\boldsymbol{q}}(t) = \mathbf{J}^{\dagger}\dot{\boldsymbol{x}}(t) + (\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J})\dot{\mathbf{q}}_N, \tag{4.14}$$

where $\boldsymbol{x}(t)$ is the task manipulation path, $\mathbf{J}^{\dagger}$ is the weighted pseudoinverse, $\mathbf{I}$ is the identity matrix, and $\dot{\mathbf{q}}_N$ is the preferred joint velocity. To accommodate joint limits, $\mathbf{J}^{\dagger}$ is replaced by

$$\mathbf{J}^{\dagger}_{\text{SNS}} = \mathbf{S}\mathbf{W}^{-\frac{1}{2}}(\mathbf{J}\mathbf{W}^{-\frac{1}{2}}\mathbf{S})^{\#}, \tag{4.15}$$

where $(\cdot)^{\#}$ denotes the Moore-Penrose pseudoinverse, $\mathbf{W}$ is the selected weighting matrix, and $\mathbf{S}$ is the diagonal saturation matrix with elements equal to 1 for active joints and 0 for inactive joints.

A joint path along the boundary edge is generated using the SNS algorithm with *pre-saturation*. Normal implementation of the SNS algorithm initializes with all joints non-saturated ($\mathbf{S} = \mathbf{I}$). For boundary tracking, the joint path starts at a boundary configuration where joint variable i is saturated. Joint variable i is locked at its boundary value by initializing $\mathbf{S}$

with $S_{ii} = 0$ and setting the i^{th} value of $\dot{\mathbf{q}}_N$ to zero. The boundary path is then generated by integrating (4.14) with $\mathbf{J}^{\dagger}$ replaced by (4.15).

When the task path $\boldsymbol{x}(t)$ is continuous and differentiable, the boundary edge path planner terminates at the task endpoint, or terminates prematurely. The condition for premature termination is that the rank of the saturated Jacobian matrix is smaller than the dimension of the task, i.e., rank$(\mathbf{JS}) < m$. This occurs in the following two cases:

1. If rank$(\mathbf{S}) < m$, the termination point is the intersection point of two RIKC boundary edges.
2. If rank$(\mathbf{S}) = m$, the termination point is a joint limit tangent point where a self-motion manifold tangentially encounters a boundary edge path.

Case 1 always corresponds to a premature termination point. Case 2 corresponds to either a bifurcation point or a premature termination point.

If the boundary edge path planner terminates at a time other than the task start or task end, the terminal point $\mathbf{q}_f$ is either a premature termination point or a joint limit bifurcation point. If $\mathbf{q}_f$ has two saturated joint variables, then rank$(\mathbf{S}) < m$ and $\mathbf{q}_f$ is a premature termination point. If $\mathbf{q}_f$ has only one saturated joint variable, and a joint path can be generated beyond t_f (starting from the joint configuration generated just before $\mathbf{q}_f$) by unlocking the saturated joint, then $\mathbf{q}_f$ is a bifurcation point.

4.4.2 Boundary Edge Start Points

The initial joint configuration for each generated boundary edge path is selected from the previously identified feasible self-motion paths of task endpoints (which were identified when generating endpoint nodes, Sec. 4.2), and of coregular values (which were identified when searching for singularities, Sec. 4.3).

Let $\mathcal{S}$ and $\mathcal{T}$ be sets of feasible self-motion paths of the task start point and task terminal point, respectively. For each boundary configuration of each feasible self-motion path in $\mathcal{S}$, a boundary tracking joint path is generated forward in time (until $t = 1$). For each boundary configuration of each feasible self-motion path in $\mathcal{T}$, a boundary tracking joint path is generated backward in time (until $t = 0$). In Fig. 4.7b, $\mathcal{S}$ and $\mathcal{T}$ each have a single feasible self-motion path at $t = 0$ and $t = 1$, respectively. The boundary edge joint path generated from the lower boundary configuration in $\mathcal{S}$ terminates at $\otimes$. The boundary edge joint path generated from the lower boundary configuration in $\mathcal{T}$ terminates at $\times$ at t_4. A boundary tracking joint path generated from the upper configuration in $\mathcal{S}$ terminates at the upper configuration in $\mathcal{T}$.

Consider the bifurcation points $\times$ at t_1 and t_3 associated with the "hole" (closed path associated with the saturation of a single joint) in Fig. 4.7b. To find them with the boundary edge path planer, a starting configuration on a feasible self-motion at some time t_2 between the bifurcation points of the "hole" must be identified. For the RIKC problems, "holes" span coregular values for which the feasible coregular self-motion paths (identified in Sec. 4.3) are between joint limit bifurcation points. Let $\mathcal{C}$ be the set of feasible coregular self-motion paths. For each boundary configuration of each feasible self-motion path in $\mathcal{C}$, a boundary edge path is generated forward in time and backward in time. Boundary tracking paths starting from the top and bottom configuration at t_2 in Fig. 4.7b (points a and b) terminate at boundary configurations in $\mathcal{S}$ and $\mathcal{T}$. Boundary tracking paths starting from the remaining boundary configurations at t_2 (points c and d) terminate at $\times$ at t_1 for backward generated paths and terminate at $\times$ at t_3 for forward generated paths (not seen in Fig. 4.7a).

For all boundary tracking paths generated from the boundary configurations in $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{C}$, the path's termination case is checked, using the methods described in Sec. 4.4.1, to determine if the endpoint corresponds to a task endpoint, premature termination point, or joint limit bifurcation point. Multiple boundary edge paths may have the same terminal configuration. The set of unique premature termination points is identified and the set of unique joint limit bifurcation points is identified. For each joint limit bifurcation point, the feasible self-motion path touching the bifurcation point is generated. These self-motion paths are split into two branches (e.g., dashed and dashed-dotted lines in Fig. 4.7) and are used as nodes in the bb-roadmap.

4.5 Bifurcation Branch Roadmap Construction

The final step in characterizing the connectivity of the RIKC solution space is constructing the bifurcation branch roadmap (bb-roadmap). Means of identifying the nodes (feasible self-motion paths) were described in Section 4.2-4.4. Additional node characteristics are described below. The remaining challenge in constructing the bb-roadmap is identifying edges that correspond to homotopy classes of joint paths connecting feasible self-motion paths of nodes. This section describes a fast method to identify the bb-roadmap edges using the self-motion manifold identification numbers. The self-motion manifold identification numbers are assigned in a consistent manner for the entire RIKC solution space (described in Section 4.2).

4.5.1 Roadmap Node Characteristics

The nodes of the bifurcation branch roadmap are either feasible self-motion paths of task endpoints or bifurcation branches, i.e., each a feasible self-motion path on one side of a bifurcation point. There are subtle differences in the characteristics of the self-motion paths introduced in this paper. These differences are reviewed below using examples illustrated in Figs. 4.5a and 4.7a.

- A *self-motion path* is a path in the RIKC solution space for which the end-effector configuration does not change. Feasibility is not considered.
- A *coregular self-motion path* is a self-motion path at a coregular value (e.g., dashed and dashed-dotted lines in Fig. 4.5a and thin and thick dotted lines at $t = t_2$ in Fig. 4.7a).
- A *regular self-motion path* is a self-motion path at a regular value (e.g., thin solid lines).
- A *feasible self-motion path* is a (regular or coregular) self-motion path in the feasible RIKC solution space (paths completely on the shaded (feasible) surface). For example, the feasible coregular self-motion paths in Figs. 4.5a and 4.7a are the dashed and dashed-dotted lines in Fig. 4.5a and thick dotted lines in Fig. 4.7a.
- A *bifurcation branch* is a portion of a feasible (regular or coregular) self-motion path adjacent to a bifurcation point. A singularity bifurcation point has coregular bifurcation branches (e.g., the portions of the dashed and dashed-dotted lines left, right, above, and below the $\times$ in Fig. 4.5a). A joint limit bifurcation point has regular bifurcation branches (e.g., the dashed and dashed-dotted lines to either side the $\times$ in Fig. 4.7a).

The feasible self-motion paths in $\mathcal{S}$ and $\mathcal{T}$ correspond to endpoint nodes (white nodes in Fig. 4.3) of the bb-roadmap. Let $\mathcal{B}_{cr}$ be the set of coregular bifurcation branches identified in Section 4.3, these correspond to singularity bifurcation nodes (black nodes) of the bb-roadmap. Let $\mathcal{B}_r$ be the set of regular bifurcation branches identified in Section 4.4; these correspond to joint limit bifurcation nodes (gray nodes) of the bb-roadmap. Let $\mathcal{P}$ be the set of premature termination points identified in Section 4.4; these correspond to premature termination nodes ($\otimes$ nodes) in the bb-roadmap.

The feasible self-motion paths in $\mathcal{S}, \mathcal{T}, \mathcal{B}_{cr}, \mathcal{B}_r, \mathcal{P}$ are sorted by time into the set of all nodes of the bifurcation branch roadmap $\mathcal{N}$. There are multiple nodes with the same time value (two at each joint limit bifurcation point, four at each singularity bifurcation point).

Additional nodes are used in edge generation. When a joint limit bifurcation occurs, the bifurcation branches are not the nearby feasible coregular self-motion paths (e.g., the thick dotted lines in Fig. 4.7a). These coregular self-motion paths are needed, however, for edge identification. Let $\mathcal{C}_f$ be the set of feasible coregular self-motion paths (that are not bifurcation branches) that are needed for identifying roadmap edges.

4.5.2 Roadmap Edges

Because the RIKC solution space consists of open self-motion manifolds, there is exactly one homotopy class of joint paths connecting two nodes (not the case for closed self-motion manifolds in Chapter 3). A brute-force approach for identifying the edges is to attempt to generate a joint path between every two node combination. This would involve using the bi-directional instantaneous path planner (3.9) and checking if the generated joint path crosses the self-motion paths of any other nodes.

The edges of the bb-roadmap can be determined much faster by investigating the connectivity of the self-motion manifolds using the self-motion manifold identification numbers and the bifurcation structure. Consider first, an RIKC problem without joint limits. An edge between nodes A and B exists if a self-motion path $\boldsymbol{q}^+(\psi)$ immediately following a node A $\in \mathcal{N}$ deforms continuously along the feasible RIKC solution space until it reaches a self-motion path $\boldsymbol{q}^-(\psi)$ immediately preceding a later node B $\in \mathcal{N}$ (a later bifurcation node or task endpoint node).

Given node A at time t_A, node B is selected from a subset of $\mathcal{N}$ based on the following: 1) $t_B > t_A$ and 2) $\mathrm{Id}_{\mathrm{num}}(\boldsymbol{q}_B^-(\psi)) = \mathrm{Id}_{\mathrm{num}}(\boldsymbol{q}_A^+(\psi))$, where $\mathrm{Id}_{\mathrm{num}}(\cdot)$ is the identification number. Edges are generated to the nearest nodes (in time) in this subset.

This roadmap construction process is illustrated in Fig. 4.8 for a simple case without joint limits. Figure 4.8a shows a simplified representation of an RIKC solution space with 4 self-motion manifolds (identification numbers 1-4) at each time instance. The four manifolds are separated by 4 twist alignment cases (dashed lines) on a closed kinematic self-motion manifold[5] parameterized by ψ. The space contains 3 bifurcations from singularities at coregular values, each with a structure like that shown in Fig. 4.5. The bb-roadmap of the RIKC solution space near each singularity is overlayed on the surface to indicate the connectivity. The bifurcation branches (black nodes) of each singularity are in sets of four. Each of the four branches has self-motion manifolds $[\boldsymbol{q}^-(\psi),\ \boldsymbol{q}^+(\psi)]$ with a unique combination of self-motion manifold identification numbers (indicated by the white numbered nodes with dashed edges). The bb-roadmap of this RIKC solution space is given in Fig. 4.8b.

[5]The closed kinematic self-motion manifolds are "cut and unrolled" into a flat surface in Fig. 4.8a

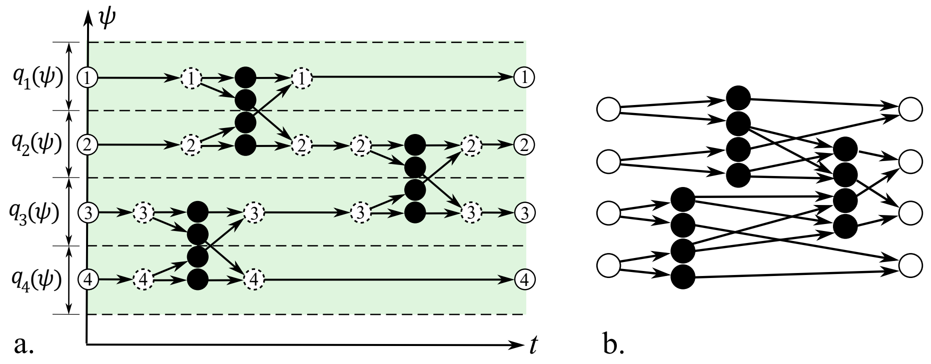


Figure 4.8: RIKC bifurcation branch roadmap construction. a) Simplified representation of the RIKC solution space parameterized by t on the horizontal axis and the kinematic self-motion parameter ψ on the vertical axis. b) Bifurcation branch roadmap.

The roadmap construction process is slightly modified when joint limits exist. Joint limit bifurcations cause multiple disjoint self-motion paths to exist in the same self-motion manifold. Although the feasible self-motion paths are no longer complete manifolds, they are assigned the same identification number as the manifolds they are in. An identification number reassignment procedure is used to distinguish disjoint feasible self-motion paths in the same manifold. Also, because of joint limit bifurcations, some feasible coregular self-motion paths are not in $\mathcal{N}$, but are in $\mathcal{C}_f$. These coregular self-motion paths must be considered because a continuously deforming feasible self-motion path that crosses a feasible coregular self-motion path without kinematic self-motion (vertical dotted line in Fig. 4.7a) changes from one self-motion identification number to another by crossing the twist alignment case.

Reassignment of the identification number is only required for joint limit bifurcation branches (nodes in $\mathcal{B}_r$) and for the feasible self-motion paths they deform into. If the bifurcation (splitting) occurs as time progresses forward, the $\boldsymbol{q}^+(\psi)$ self-motion identification number is reassigned a unique number k. Each of these feasible self-motion paths (say of Node A $\in \mathcal{N}$) continuously deforms into a feasible self motion path of a node B $\in \{\mathcal{N}, \mathcal{C}_f\}$ Node B is identified using the boundary edge path planner. The self-motion identification number of $\boldsymbol{q}^-(t)$ of B is reassigned as k. A similar process is used when the bifurcation occurs as time progresses backward.

The edge generation methodology is summarized in the following algorithmic procedure. For each node in $\mathcal{N}$ and not in $\mathcal{T}$, roadmap edges are identified using the following steps:

1. Denote the selected node as I, its task time t_I, and its forward self-motion $\boldsymbol{q}_\text{i}^+(\psi)$, where i is its self-motion identification number.

2. Identify all the nodes in $\mathcal{N}$ and $\mathcal{C}_f$ with time $t > t_{\mathrm{I}}$ each having a backward self-motion $\boldsymbol{q}_j^-(\psi) \mid j = i$.
3. From the set in Step 2, identify those with the smallest time $t > t_{\mathrm{I}}$ (there can be multiple). Let $t_{\mathcal{C}_f} > t_{\mathrm{I}}$ be the smallest time of the relevant nodes in $\mathcal{C}_f$; and let $t_{\mathcal{N}} > t_{\mathrm{I}}$ be the smallest time of the relevant nodes in $\mathcal{N}$.
4. If $t_{\mathcal{C}_f} < t_{\mathcal{N}}$, a coregular self-motion exists between the edge nodes. Because a twist alignment case can be crossed at $t_{\mathcal{C}_f}$, the forward self-motion $\boldsymbol{q}_{\mathrm{i}}^+(\psi)$ identification number i is changed to that of the node associated with $t_{\mathcal{C}_f}$, t_{I} is updated to $t_{\mathcal{C}_f}$, and Steps 2-3 are repeated.
5. If $t_{\mathcal{N}} < t_{\mathcal{C}_f}$ (or if there are no relevant nodes in $\mathcal{C}_f$ from Step 2), there is an edge from I to each of the relevant nodes in $\mathcal{N}$ with the time value $t_{\mathcal{N}}$.

The resulting directed graph fully captures the connectivity of the RIK solution space. This combined kinematic and compliance solution space bb-roadmap is used together with the bifurcation branch algorithm to identify the globally optimal solution to the RIKC path planning problem.

4.6 Case Study

This section demonstrates the algorithm for generating the bifurcation branch roadmap and obtaining the globally optimal joint path for a 3R-VSA planar manipulator. Two different task manipulation paths are considered. The kinematic motion path is the same in each; only the task compliance is different. Although the tasks are relatively simple, the connection structure of the two RIKC solution spaces are relatively complex and quite different from each other.

4.6.1 Manipulator and Task Description

Consider the 3R-VSA manipulator and particle planar compliance task illustrated in Fig. 4.9. The link lengths, normalized by the reach of the manipulator L, are: $l_1 = 0.46$, $l_2 = 0.43$, and $l_3 = 0.11$ (anthropomorphic ratios [50]). The kinematic joint positions q_{p_1}, q_{p_2}, and q_{p_3} are expressed in radians; the joint compliances q_{c_1}, q_{c_2}, and q_{c_3} are normalized by $(f_g L)^{-1}$ where f_g is the weight of the manipulator.

For 3R-VSA manipulators performing particle-planar compliance tasks, the total joint configuration is

$$\mathbf{q} = [q_{p_1},\, q_{p_2},\, q_{p_3},\, q_{c_1},\, q_{c_2},\, q_{c_3}]^T,$$

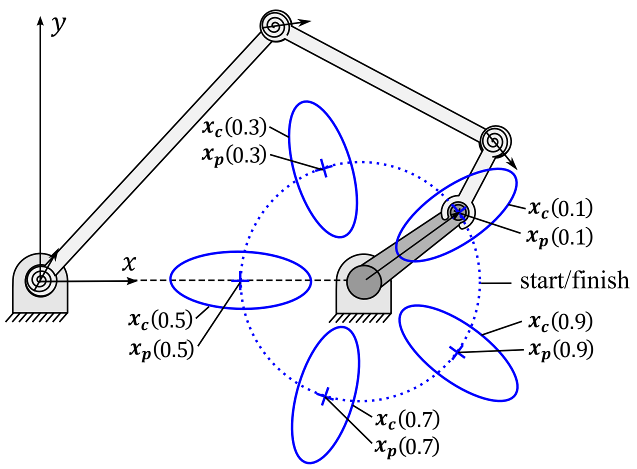


Figure 4.9: 3R-VSA manipulator performing a crank-turning task with an unknown work load. Five equally spaced instants in time illustrate the continuous end-effector path $\boldsymbol{x}_p(t)$ and the continuous compliance path $\boldsymbol{x}_c(t)$.

and the total task configuration is

$$\mathbf{x} = [x,\ y,\ C_{x_{11}},\ C_{x_{12}},\ C_{x_{22}}]^T.$$

The constrained manipulation task is to turn a crank. The end-effector motion path is given by

$$\boldsymbol{x}_p(t) = \begin{bmatrix} 0.5 + 0.2\cos(2\pi t) \\ 0.2\sin(2\pi t) \end{bmatrix}, \tag{4.16}$$

where, for normalized time, $t = 0$ at the start and $t = 1$ at the end of the task. The task motion path is dimensionless, normalized by L.

Because the crank handle has a fixed radial distance from the crank center, the task compliance in the direction along the crank arm is high to limit the constraint force that may result from error in the commanded relative position of the end-effector. The task compliance in the direction of crank motion is low to limit the end-effector deflection from the planned path due to the crank work-load. In Fig. 4.9, the desired compliance ellipse (see Appendix A) is shown at five instants in time, where the major and minor radii of the ellipse are eigenvalues of the task compliance matrix.

The normalized task compliance path (task compliance normalized by Lf_g^{-1}) in matrix form is

$$\mathbf{C}_x(t) = \mathbf{U} \begin{bmatrix} \frac{\lambda_2}{\gamma(t)} & 0 \\ 0 & \lambda_2 \end{bmatrix} \mathbf{U}^T \tag{4.17}$$

where $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2]$ are the normalized eigenvectors, $\lambda_2 = 0.1$ is the larger eigenvalue and $\gamma = \lambda_2/\lambda_1$ is the ellipse aspect ratio. The eigenvalues are constant in each task, but the eigenvectors change so that the orientation of the larger compliance eigenvector is always along the crank arm:

$$\mathbf{U} = \begin{bmatrix} -\sin(2\pi t) & \cos(2\pi t) \\ \cos(2\pi t) & \sin(2\pi t) \end{bmatrix}. \tag{4.18}$$

Each joint compliance is controlled using a VSA with an exponential compliance versus actuation profile. The feasible compliance range is $q_{c_i} \in [0.001, 10]$ corresponding to the actuator limits $\phi_i \in [0, \pi/2]$. The VSA actuator position is related to the joint compliance by: $\phi_{c_i} = \frac{1}{\xi}\ln(q_{c_i}/c_0)$, where $\xi = 5.86$ and $c_0 = 0.001$. The total actuator configuration is $\boldsymbol{\phi} = [\boldsymbol{\phi}_p^T, \phi_{c_1}, \phi_{c_2}, \phi_{c_3}]^T$, where $\boldsymbol{\phi}_p$ is a 3-vector of primary motion positions. The manipulator is oriented in the horizontal plane where gravity does not affect the equilibrium joint position such that $\boldsymbol{\phi}_p = \mathbf{q}_p$.

The globally optimal path is the one that minimizes the actuator velocities, i.e., minimizes

$$G_{\text{global}} = \int \frac{1}{2}\dot{\boldsymbol{q}}(t)^T \mathbf{W}(\boldsymbol{q}(t))\dot{\boldsymbol{q}}(t)dt, \tag{4.19}$$

where $\mathbf{W}(\cdot)$ is a joint configuration dependent weighing matrix, selected such that $\|\dot{\boldsymbol{\phi}}\|^2 = \dot{\mathbf{q}}^T\mathbf{W}\dot{\mathbf{q}}$:

$$\mathbf{W}(\mathbf{q}) = \operatorname{diag}\left(\left[1, 1, 1, \left(\frac{1}{\xi q_{c_1}}\right)^2, \left(\frac{1}{\xi q_{c_2}}\right)^2, \left(\frac{1}{\xi q_{c_3}}\right)^2\right]\right), \tag{4.20}$$

where diag$(\cdot)$ sorts the vector elements into the diagonal elements of a diagonal matrix and $1/(\xi q_{c_i})$ is the sensitivity of the VSA actuator position to variation in the compliance value of joint i.

4.6.2 Results

Results of the "elbow-up" poses are shown for two aspect ratios: $\gamma = 2$ and $\gamma = 10$. The bb-roadmaps for these tasks are shown in Fig. 4.10a and Fig. 4.11a, respectively. The routes through the bb-roadmaps identify the different joint path homotopy classes. For Task $\gamma = 2$, there are 9 routes that complete the task, of which, 5 are cyclic (listed in Table 4.1). For Task $\gamma = 10$, there are 120 routes that complete the task, of which, 30 are cyclic.

Table 4.1: Cyclic Homotopy Classes for Crank Task $\gamma = 2$

Cyclic Homotopy Class	Node Sequence in Fig. 4.10a
1	S_a, 3, 7, T_a
2	S_a, 4, 9, T_a
3	S_b, 1, 5, 8, 11, T_b
4	S_b, 1, 6, 10, 11, T_b
5	S_b, 2, 12, T_b

If the crank needs to be turned a number of times, periodic boundary conditions are used. For Task $\gamma = 2$, the upper/lower bound culling method of the bb-algorithm identified that only Homotopy Class 1 could contain the globally optimal cyclic joint path. However, for comparison purposes, all cyclic homotopy classes are presented for this task. Some homotopy classes had two locally optimal cyclic joint paths.

For Task $\gamma = 10$, the upper/lower bound culling method reduced the set of promising homotopy classes to four. Each homotopy class had a single locally optimal cyclic path.

The locally optimal cyclic paths found using the bb-algorithm are shown on the RIKC solution surfaces in Figs. 4.10b and 4.11b as thick solid lines. The cost value of each path, evaluated with (4.19), is shown in the figures. The globally optimal path cost values were 7.90 for Task $\gamma = 2$ and 32.23 for Task $\gamma = 10$. The actual RIKC solution space is a hyper surface in the 6-dimensional joint space. The surfaces in Figs. 4.10b and 4.11b are close representations of the hyper surface unrolled onto a 2-D plane, where normalized time (task progress) is on the horizontal axis and the kinematic self-motion parameter is on the vertical axis. The closed kinematic self-motion manifolds are "cut and unrolled" at the horizontal dashed lines. The curved dashed lines are the kinematic solutions of the twist alignment cases, $\boldsymbol{q}_{a^+}(t)$, $\boldsymbol{q}_{b^+}(t)$, $\boldsymbol{q}_{a^-}(t)$, and $\boldsymbol{q}_{b^-}(t)$, that separate the four (total configuration) self-motion manifolds.

The rasterized colored surface corresponds to feasible joint configurations, where the color of each point on the surface is given by RGB coordinate values corresponding to the compliance actuator configuration: $\boldsymbol{\phi}_c = [\phi_{c_1},\ \ \phi_{c_2},\ \ \phi_{c_3}]^T$ normalized by the actuation limits. Both surfaces have an overall blue hue indicating that the third joint is generally the most compliant.

Black circle markers on the simplified solution space correspond to bifurcation points from singularities. Gray circle markers on the surface correspond to bifurcation points from joint limits. Black triangle markers indicate the locations of feasible coregular self-motion paths $\boldsymbol{q}(\psi_c)$ in the

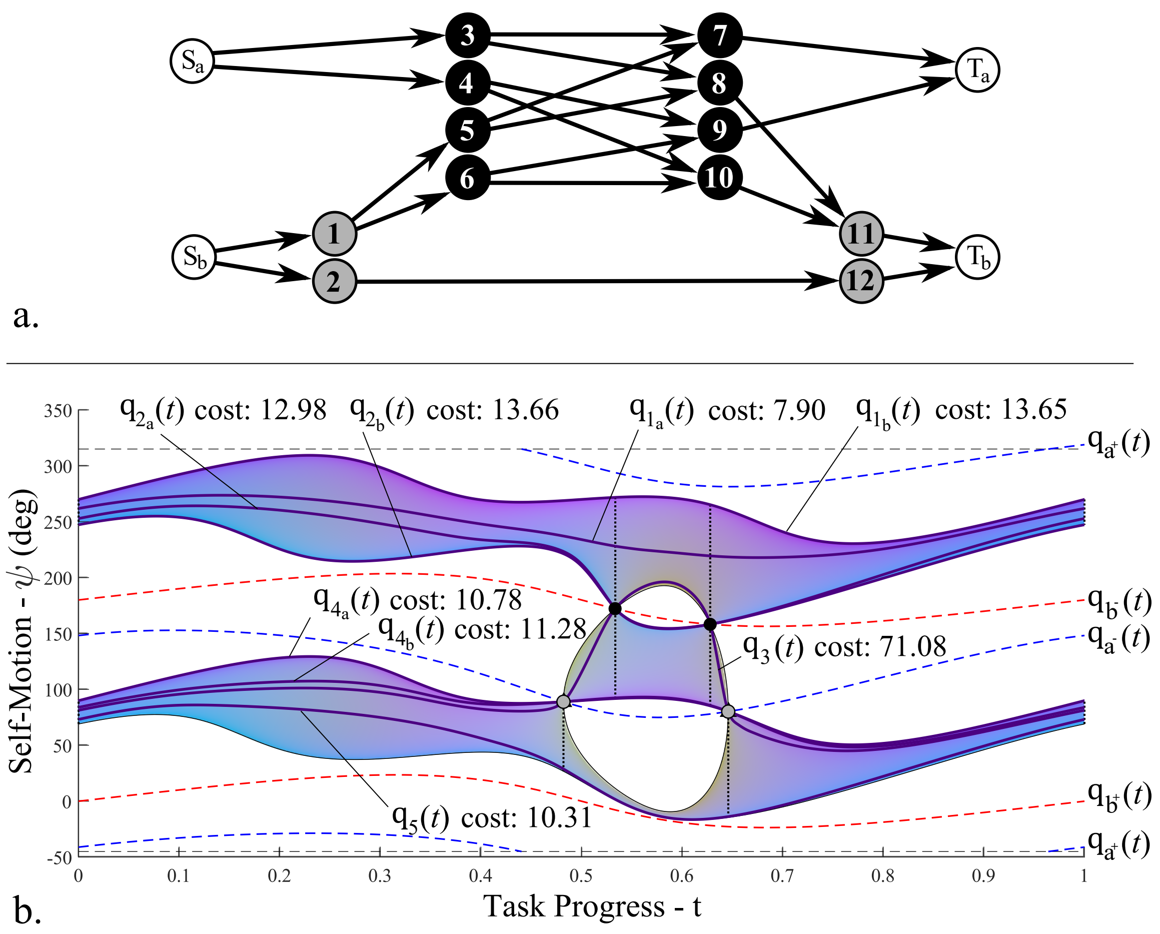


Figure 4.10: RIKC solution space connection structure for crank task $\gamma = 2$. a) Bifurcation branch roadmap. b) Sampled surface parameterized by t on the horizontal axis and the kinematic self-motion parameter ψ on the vertical axis.

compliance subspace that do not contain the singularity. Coregular paths in the compliance subspace are not visible on the plotted surface (which is parameterized by the *kinematic* self-motion parameter), but they can be imagined as lines going into or coming out of the page at the black circle and triangle marker locations. In some instances, the triangle markers are very close to (and are obscured by) gray circle markers. Vertical dotted lines are self-motion paths $\boldsymbol{q}(\psi)$ associated with bifurcation branches of the bb-roadmap. Each joint limit bifurcation point has two bifurcation branches. Each singularity bifurcation point has four bifurcation branches. Again, the two branches of $\boldsymbol{q}(\psi_c)$ are not visible on the plotted surface.

Although the two tasks are the same except for the compliance in the direction of crank's motion, the bb-roadmaps and the globally optimal cyclic joint paths are significantly different.

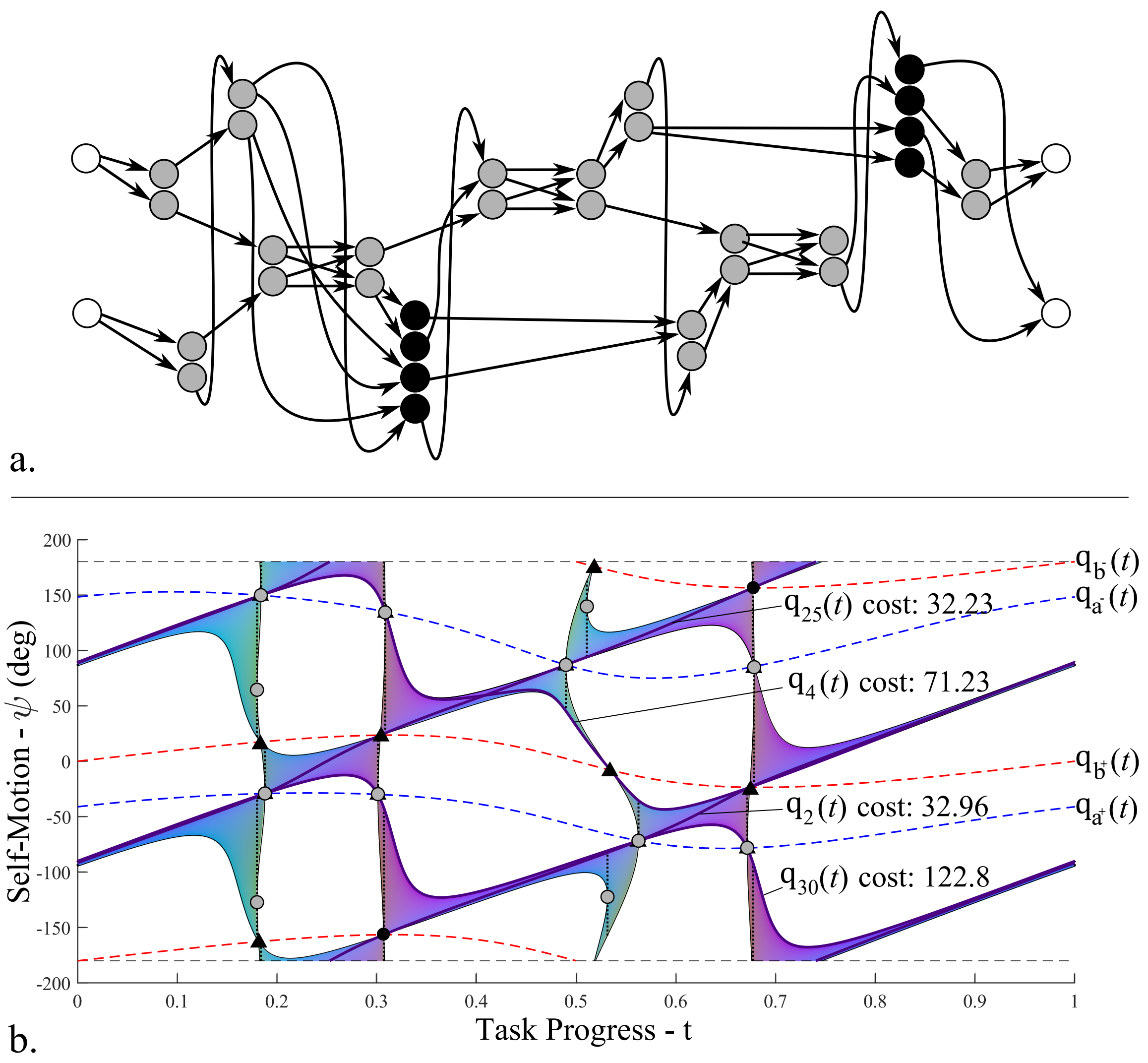


Figure 4.11: RIKC solution space connection structure for crank task $\gamma = 10$. a) Bifurcation branch roadmap. b) Sampled surface parameterized by t on the horizontal axis and the kinematic self-motion parameter ψ on the vertical axis.

4.7 Chapter Summary

This chapter provided a method for identifying the RIKC solution space connection structure (recorded in the bifurcation branch roadmap), for manipulation with one degree of redundancy. Methods for identifying the number and the structure of each self-motion manifold in the combined kinematic and compliance joint space were provided as well as methods to identify bifurcations from singularities and from joint limits. The bifurcation branch roadmap is used in the bifurcation branch algorithm to identify the globally optimal path in the combined kinematic and compliance joint space to achieve the optimal sequence of joint configurations for passive compliance control. The structure of the compliance-extended redundant inverse kinematic (RIKC) solution space is determined by the manipulator structure and the task. Procedures were demonstrated and results were presented for a 3R-VSA manipulator performing particle-planar compliance tasks.

CHAPTER 5

SUMMARY AND CONTRIBUTIONS

A robotic manipulator performing a constrained manipulation task may experience very high contact forces due to positioning errors. This work proposes passive stiffness/compliance control (a subset of impedance control) as an approach for simultaneously regulating the end-effector's contact forces and desired motion. In passive compliance control, the end-effector's multi-directional elastic behavior, described by its compliance (inverse of stiffness) is directly adjusted by utilizing kinematic redundancy to change the joint positions and variable stiffness actuation to change the joint compliances. The main difficulty of passive compliance control is in finding an appropriate sequence of joint positions and joint compliances that realize a desired sequence of end-effector positions and compliances. A list of this work's contributions related to this problem is provided below.

- The time-varying passive compliance control problem was framed as an extension of the redundant inverse kinematic (RIK) path planning problem to include compliance in the joint and task configuration spaces (RIK $\rightarrow$ RIKC). Aspects of this result include:
  - The total joint configuration space consisting of joint kinematic positions and joint compliances was defined.
  - The total task configuration space consisting of task position/orientation coordinates and task compliance coordinates was defined where the upper triangular elements of the task compliance matrix were identified as the minimum set of task compliance coordinates.
  - The forward compliance mapping for any serial manipulator with adjustable joint compliances was identified.
  - The compliance Jacobian relating infinitesimal changes in joint configuration to infinitesimal changes in the task compliance configuration was derived for any serial manipulation with adjustable joint compliance.
- A manipulation planning procedure using the weighted pseudoinverse and the Saturation in the Null Space (SNS) algorithm was used to instantaneously resolve the RIKC path planning problem.
- Redundancy resolution and gravity compensation were addressed using a separate configuration space for actuator positions. Aspects related to this include:

  - An actuator mapping function from the joint space to the actuator space was used to address gravity compensation.
  - The actuator mapping function for any serial manipulator with compliance actuators (for which each joint compliance depends only on the position of a single actuator) was provided.
  - The actuator Jacobian relating infinitesimal changes in joint configuration to infinitesimal changes in the actuator configuration was defined.
  - A weighting matrix constructed from the actuator Jacobian was used in the weighted pseudoinverse to resolve redundancy in the joint motion by minimizing the norm of the actuator motion.
- A new algorithm called the bifurcation branch algorithm (bb-algorithm) was developed for identifying the best solution to RIK and RIKC path planning problems with one degree of redundancy. The algorithm involves 1) characterizing the connectivity of the RIK(C) solution space such that all homotopy classes of solutions are identified, 2) generating sub-optimal solutions in each homotopy class using a novel instantaneous path planner, and 3) deforming sub-optimal solution into locally optimal solutions using a novel path deformation procedure. Aspects of this include:
  - A new directed graph called the bifurcation branch roadmap (bb-roadmap) was developed for strategically characterizing the connectivity of the RIK(C) solution space such that each homotopy class of joint paths solving the RIK(C) problem is identified. Each node of the bb-roadmap corresponds to a feasible self-motion path. Each edge of the bb-roadmap corresponds to a homotopy class of joint paths connecting two nodes.
  - Algorithmic procedures for generating the bb-roadmap were developed for two classes of RIK(C) problems: 1) RIK problems with either one or two *closed* feasible self-motion paths, 2) RIK(C) problems with any number of *open* feasible self-motion paths.
  - An upper/lower bound method was developed to eliminate from further considerations those homotopy classes having optimal paths with high cost function values. This significantly reduces the number of homotopy classes evaluated in the most computationally demanding stage of the bb-algorithm.
  - A novel bi-directional instantaneous path planner was developed to quickly generate continuous joint paths between specified joint configurations. A forward progressing

joint path from one staring configuration and a backward progressing joint path starting at the other joint configuration are simultaneously generated in which the instantaneous optimization criteria is updated to guide the paths to meet. If no joint limit is encountered, the continuous joint path is smooth.

- A novel "direct" optimal control solver (i.e., using nonlinear programming) was developed for deforming a sub-optimal joint path into a locally optimal joint path. The novelty of the path deformation procedure is in using self-motion paths to simplify the nonlinear programming problem by: 1) reducing the order of problem, 2) eliminating the need for equality constraints in the optimization, 3) and reducing the sensitivity of the objective function to changes in the optimization parameters.
- A self-motion path planner was provided.
- A novel method of identifying joint limit bifurcation points using a boundary edge path planner was developed.
- A boundary edge path planner was provided.
- A method of locating singularity bifurcations using root finding was described.
- The number and structure of self-motion manifolds of RIKC problems were identified. The corresponding self-motion paths were give parameterized descriptions.

BIBLIOGRAPHY

[1] M. H. Raibert and J. J. Craig, "Hybrid position/force control of manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 103, no. 2, pp. 126–133, 1981.

[2] J. Duffy, "The fallacy of modern hybrid control theory that is based on "orthogonal complements" of twist and wrench spaces," *Journal of Robotic Systems*, vol. 7, no. 2, pp. 139–144, 1990.

[3] N. Hogan, "Impedance control: An approach to manipulation: Part I - theory," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 1–7, 1985.

[4] D. Tsetserukou, R. Tadakuma, H. Kajimoto, N. Kawakami, and S. Tachi, "Intelligent variable joint impedance control and development of a new whole-sensitive anthropomorphic robot arm," in *International Symposium on Computational Intelligence in Robotics and Automation*, pp. 338–343, IEEE, 2007.

[5] G. Palli, C. Melchiorri, and A. De Luca, "On the feedback linearization of robots with variable joint stiffness," in *International Conference on Robotics and Automation*, pp. 1753–1759, IEEE, 2008.

[6] M. Howard, D. J. Braun, and S. Vijayakumar, "Transferring human impedance behavior to heterogeneous variable impedance actuators," *IEEE Transactions on Robotics*, vol. 29, pp. 847–862, Aug 2013.

[7] N. Hogan, "Impedance control: An approach to manipulation: Part II - implementation," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 8–16, 1985.

[8] C. Ott, *Cartesian Impedance Control of Redundant and Flexible-Joint Robots.* Springer, 2008.

[9] F. Caccavale, C. Natale, B. Siciliano, and L. Villani, "Six-DOF impedance control based on angle/axis representations," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 289–300, 1999.

[10] C. Ott, A. Albu-Schäffer, A. Kugi, and G. Hirzinger, "On the passivity-based impedance control of flexible joint robots," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 416–429, 2008.

[11] A. Albu-Schäffer, C. Ott, U. Frese, and G. Hirzinger, "Cartesian impedance control of redundant robots: Recent results with the DLR-light-weight-arms," in *International Conference on Robotics and Automation*, pp. 3704–3709, IEEE, 2003.

[12] A. Dietrich, C. Ott, and A. Albu-Schäffer, "An overview of null space projections for redundant, torque-controlled robots," *The International Journal of Robotics Research*, vol. 34, no. 11, pp. 1385–1400, 2015.

[13] F. Ferraguti, C. Secchi, and C. Fantuzzi, "A tank-based approach to impedance control with variable stiffness.," in *International Conference on Robotics and Automation*, pp. 4948–4953, IEEE, 2013.

[14] T. Lens, J. Kunz, O. Von Stryk, C. Trommer, and A. Karguth, "BioRob-Arm: A quickly deployable and intrinsically safe, light-weight robot arm for service robotics applications," in

41st International Symposium on Robotics and 6th German Conference on Robotics, pp. 1–6, VDE, 2010.

[15] L. Vo-Gia, N. Kashiri, F. Negrello, N. G. Tsagarakis, and D. G. Caldwell, "Development of a 7DOF soft manipulator arm for the compliant humanoid robot COMAN," in *International Conference on Robotics and Biomimetics*, pp. 1106–1111, IEEE, 2014.

[16] N. G. Tsagarakis, D. G. Caldwell, F. Negrello, W. Choi, L. Baccelliere, V. G. Loc, J. Noorden, L. Muratore, A. Margan, A. Cardellino, M. Ferrati, V. Varricchio, L. Pallottino, C. Pavan, A. Bicchi, A. Settimi, A. Rocchi, and A. Ajoudani, "WALK-MAN: A high-performance humanoid platform for realistic environments," *Journal of Field Robotics*, vol. 34, no. 7, pp. 1225–1259, 2017.

[17] M. G. Catalano, G. Grioli, M. Garabini, F. Bonomo, M. Mancini, N. Tsagarakis, and A. Bicchi, "VSA-CubeBot: A modular variable stiffness platform for multiple degrees of freedom robots," in *International Conference on Robotics and Automation*, pp. 5090–5095, IEEE, 2011.

[18] M. Grebenstein, A. Albu-Schäffer, T. Bahls, M. Chalon, O. Eiberger, W. Friedl, R. Gruber, S. Haddadin, U. Hagn, R. Haslinger, H. Höppner, S. Jörg, M. Nickl, A. Nothhelfer, F. Petit, J. Reill, N. Seitz, T. Wimböck, S. Wolf, T. Wüsthoff, and G. Hirzinger, "The DLR hand arm system," in *International Conference on Robotics and Automation*, pp. 3175–3182, IEEE, 2011.

[19] S. D. Eppinger and W. P. Seering, "Understanding bandwidth limitations in robot force control," in *International Conference on Robotics and Automation*, vol. 4, pp. 904–909, IEEE, 1987.

[20] B.-S. Kim, Y.-L. Kim, and J.-B. Song, "Preliminary experiments on robotic assembly using a hybrid-type variable stiffness actuator," in *International Conference on Advanced Intelligent Mechatronics*, pp. 1076–1080, IEEE/ASME, 2011.

[21] A. Albu-Schäffer, M. Fischer, G. Schreiber, F. Schoeppe, and G. Hirzinger, "Soft robotics: What cartesian stiffness can we obtain with passively compliant, uncoupled joints?," in *International Conference on Intelligent Robots and Systems*, pp. 3295–3301, IEEE, 2004.

[22] S. Huang and J. M. Schimmels, "Realization of point planar elastic behaviors using revolute joint serial mechanisms having specified link lengths," *Mechanism and Machine Theory*, vol. 103, pp. 1–20, 2016.

[23] S. Huang and J. M. Schimmels, "Synthesis of point planar elastic behaviors using three-joint serial mechanisms of specified construction," *Journal of Mechanisms and Robotics*, vol. 9, no. 1, p. 011005, 2017.

[24] S. Huang and J. M. Schimmels, "Geometric construction-based realization of planar elastic behaviors with parallel and serial manipulators," *Journal of Mechanisms and Robotics*, vol. 9, no. 5, p. 051006, 2017.

[25] S. Huang and J. M. Schimmels, "Geometric approach to the realization of planar elastic behaviors with mechanisms having four elastic components," *Journal of Mechanisms and Robotics*, vol. 10, no. 4, p. 041004, 2018.

[26] S. Huang and J. M. Schimmels, "Geometry based synthesis of planar compliances with redundant mechanisms having five compliant components," *Mechanism and Machine Theory*, (Submitted 2018).

[27] S. Huang and J. M. Schimmels, "Geometric construction-based realization of spatial elastic behaviors in parallel and serial manipulators," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 764–780, 2018.

[28] A. De Luca, B. Siciliano, and L. Zollo, "PD control with on-line gravity compensation for robots with elastic joints: Theory and experiments," *Automatica*, vol. 41, no. 10, pp. 1809–1819, 2005.

[29] H. Höppner, W. Wiedmeyer, and P. van der Smagt, "A new biarticular joint mechanism to extend stiffness ranges," in *International Conference on Robotics and Automation*, pp. 3403–3410, IEEE, 2014.

[30] F. P. Petit, *Analysis and Control of Variable Stiffness Robots.* PhD thesis, Eidgenössische Technische Hochschule ETH Zürich, Nr. 22134, 2014.

[31] F. Petit and A. Albu-Schäffer, "Cartesian impedance control for a variable stiffness robot arm," in *International Conference on Intelligent Robots and Systems*, pp. 4180–4186, IEEE, 2011.

[32] M. Howard, D. J. Braun, and S. Vijayakumar, "Constraint-based equilibrium and stiffness control of variable stiffness actuators," in *International Conference on Robotics and Automation*, pp. 5554–5560, IEEE, 2011.

[33] J. K. Salisbury, "Active stiffness control of a manipulator in cartesian coordinates," in *19th Conference on Decision and Control including the Symposium on Adaptive Processes*, pp. 95–100, IEEE, 1980.

[34] M. T. Mason and K. J. Salisbury, *Robot Hands and the Mechanics of Manipulation.* Artificial Intelligence, MIT Press, 1985.

[35] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control.* John Wiley & Sons, 2008.

[36] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on Man-Machine Systems*, vol. 10, pp. 47–53, June 1969.

[37] T. F. Chan and R. V. Dubey, "A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators," *IEEE Transactions on Robotics and Automation*, vol. 11, pp. 286–292, Apr 1995.

[38] C. A. Klein and C. H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, pp. 245–250, March 1983.

[39] A. A. Maciejewski and C. A. Klein, "The singular value decomposition: Computation and applications to robotics," *The International Journal of Robotics Research*, vol. 8, no. 6, pp. 63–79, 1989.

[40] T. Yoshikawa, "Manipulability of robotic mechanisms," *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[41] A. Liégois, “Automatic supervisory control of the configuration and behavior of multibody mechanisms,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, pp. 868–871, Dec 1977.

[42] F. Flacco, A. De Luca, and O. Khatib, “Control of redundant robots under hard joint constraints: Saturation in the null space,” *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 637–654, 2015.

[43] S. Huang and J. M. Schimmels, “The duality in spatial stiffness and compliance as realized in parallel and serial elastic mechanisms,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 124, no. 1, pp. 76–84, 2002.

[44] S.-F. Chen and I. Kao, “Conservative congruence transformation for joint and cartesian stiffness matrices of robotic hands and fingers,” *The International Journal of Robotics Research*, vol. 19, no. 9, pp. 835–847, 2000.

[45] H. Bruyninckx and J. De Schutter, “Symbolic differentiation of the velocity mapping for a serial kinematic chain,” *Mechanism and Machine Theory*, vol. 31, no. 2, pp. 135–148, 1996.

[46] F. Petit, M. Chalon, W. Friedl, M. Grebenstein, A. Albu-Schäffer, and G. Hirzinger, “Bidirectional antagonistic variable stiffness actuation: Analysis, design & implementation,” in *International Conference on Robotics and Automation*, pp. 4189–4196, IEEE, 2010.

[47] A. Jafari, N. G. Tsagarakis, and D. G. Caldwell, “AwAS-II: A new actuator with adjustable stiffness based on the novel principle of adaptable pivot point and variable lever ratio,” in *International Conference on Robotics and Automation*, pp. 4638–4643, IEEE, 2011.

[48] S. S. Groothuis, G. Rusticelli, A. Zucchelli, S. Stramigioli, and R. Carloni, “The variable stiffness actuator vsaUT-II: Mechanical design, modeling, and identification,” *IEEE/ASME Transactions on Mechatronics*, vol. 19, pp. 589–597, April 2014.

[49] J. M. Schimmels and D. R. Garces, “The arched flexure VSA: A compact variable stiffness actuator with large stiffness range,” in *International Conference on Robotics and Automation*, pp. 220–225, IEEE, 2015.

[50] C. C. Gordon, T. Churchill, C. E. Clauser, B. Bradtmiller, J. T. McConville, I. Tebbetts, and R. A. Walker, “Anthropometric survey of US army personnel: Summary statistics, interim report for 1988,” tech. rep., Anthropology Research Project Inc. Yellow Springs OH, 1989.

[51] J. Burdick, “On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds,” in *International Conference on Robotics and Automation*, pp. 264–270, IEEE, 1989.

[52] F. Flacco and A. De Luca, “Discrete-time redundancy resolution at the velocity level with acceleration/torque optimization properties,” *Robotics and Autonomous Systems*, vol. 70, pp. 191–201, 2015.

[53] Y. Nakamura and H. Hanafusa, “Optimal redundancy control of robot manipulators,” *The International Journal of Robotics Research*, vol. 6, no. 1, pp. 32–42, 1987.

[54] K. C. Suh and J. M. Hollerbach, “Local versus global torque optimization of redundant manipulators,” in *International Conference on Robotics and Automation*, IEEE, 1987.

[55] K. Kazerounian and Z. Wang, “Global versus local optimization in redundancy resolution of robotic manipulators,” *The International Journal of Robotics Research*, vol. 7, no. 5, pp. 3–12, 1988.

[56] D. P. Martin, J. Baillieul, and J. M. Hollerbach, “Resolution of kinematic redundancy using optimization techniques,” *IEEE Transactions on Robotics and Automation*, vol. 5, no. 4, pp. 529–533, 1989.

[57] M. Galicki, “The planning of robotic optimal motions in the presence of obstacles,” *The International Journal of Robotics Research*, vol. 17, no. 3, pp. 248–259, 1998.

[58] B. J. Martin and J. E. Bobrow, “Minimum-effort motions for open-chain manipulators with task-dependent end-effector constraints,” *The International Journal of Robotics Research*, vol. 18, no. 2, pp. 213–224, 1999.

[59] N. S. Bedrossian, “Classification of singular configurations for redundant manipulators,” in *International Conference on Robotics and Automation*, pp. 818–823, IEEE, 1990.

[60] A. Guigue, M. Ahmadi, M. Hayes, R. Langlois, and F. Tang, “A dynamic programming approach to redundancy resolution with multiple criteria,” in *International Conference on Robotics and Automation*, pp. 1375–1380, IEEE, 2007.

[61] L. F. Shampine, J. Kierzenka, and M. W. Reichelt, “Solving boundary value problems for ordinary differential equations in MATLAB with bvp4c,” *Tutorial notes*, pp. 1–27, 2000.

[62] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, vol. 19. Siam, 2010.

[63] E. Schmitzberger, J. L. Bouchet, M. Dufaut, D. Wolf, and R. Husson, “Capture of homotopy classes with probabilistic road map,” in *International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2317–2322 vol.3, IEEE, 2002.

[64] S. Bhattacharya, M. Likhachev, and V. Kumar, “Topological constraints in search-based robot path planning,” *Autonomous Robots*, vol. 33, no. 3, pp. 273–290, 2012.

[65] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, Aug 1996.

[66] M. Stilman, “Task constrained motion planning in robot joint space,” in *International Conference on Intelligent Robots and Systems*, pp. 3074–3081, IEEE, 2007.

[67] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, “Manipulation planning on constraint manifolds,” in *International Conference on Robotics and Automation*, pp. 625–632, IEEE, 2009.

[68] M. Stilman, “Global manipulation planning in robot joint space with task constraints,” *IEEE Transactions on Robotics*, vol. 26, pp. 576–584, June 2010.

[69] L. Jaillet and J. M. Porta, “Path planning under kinematic constraints by rapidly exploring manifolds,” *IEEE Transactions on Robotics*, vol. 29, pp. 105–117, Feb 2013.

[70] B. Kim, T. T. Um, C. Suh, and F. C. Park, "Tangent bundle RRT: A randomized algorithm for constrained motion planning," *Robotica*, vol. 34, no. 1, pp. 202–225, 2016.

[71] D. R. Baker and C. W. Wampler, "On the inverse kinematics of redundant manipulators," *The International Journal of Robotics Research*, vol. 7, no. 2, pp. 3–21, 1988.

[72] J. Kieffer, "Differential analysis of bifurcations and isolated singularities for robots and mechanisms," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 1, pp. 1–10, 1994.

[73] M. Shimizu, H. Kakuya, W.-K. Yoon, K. Kitagaki, and K. Kosuge, "Analytical inverse kinematic computation for 7-DOF redundant manipulators with joint limits and its application to redundancy resolution," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1131–1142, 2008.

[74] C. L. Luck and S. Lee, "Self-motion topology for redundant manipulators with joint limits," in *International Conference on Robotics and Automation*, pp. 626–631 vol.3, IEEE, 1993.

APPENDIX A

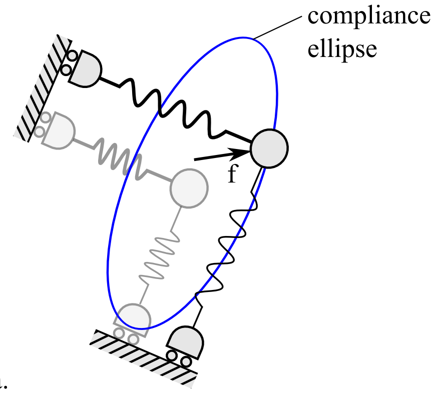
COMPLIANCE ELLIPSE

A particle planar compliance like that realized by the springs in Fig. A.1a has an elastic behavior given by

$$\Delta \mathbf{x} = \mathbf{C}_x \mathbf{f}, \tag{A.1}$$

where $\mathbf{C}_x$ is the 2×2 compliance matrix, $\Delta \mathbf{x}$ is the displacement of the particle from the equilibrium position, and $\mathbf{f}$ is an external force applied to the particle (illustrated in Fig. A.1a).

The compliance ellipse in this work is a particle's displacement-image from the unit force $(\mathbf{f} \mid \mathbf{f}^T \mathbf{f} = 1)$ applied in all directions. The directions of minimum and maximum displacement are along the eigenvectors $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively (shown in Fig. A.1b). The magnitudes of the minimum and maximum displacement are given by the eigenvalues λ_1 and λ_2, respectively.



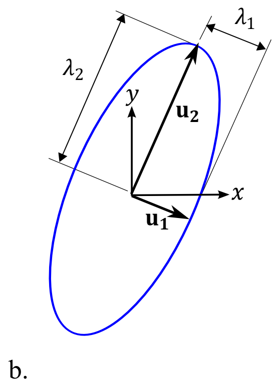


Figure A.1: Compliance ellipse: planar displacement image of a particle from a unit force $\mathbf{f}$. a) The elastic behavior realized by simple springs (the more transparent particle shows the equilibrium position). b) The minor and major radii are the eigenvalues of the compliance matrix: λ_1 and λ_2, respectively. The orientation of the ellipse is given by the minor and major eigenvectors of the compliance matrix: $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively.