Variable Interaction Control with Dynamical Systems

handed in
MASTER'S THESIS

Xiao Chen

born on the 10.03.1994
living in:
Kellerstrasse 35D
81667 Munich
Tel.: +491788256807

Human-centered Assistive Robotics
Technical University of Munich

Univ.-Prof. Dr.-Ing. Dongheui Lee

Supervisor:	Youssef Michel
Start:	18.12.2019
Intermediate Report:	16.06.2020
Delivery:	08.12.2020

Abstract

Many of the existing dynamical system (DS) methods learned from demonstrations do not enable the robot to adhere to a reference path during execution. And the traditional open loop control configuration which is used to track a reference trajectory does not endow the robot with compliance behavior in interaction with environment. Suppose that a desired DS learned from demonstrations is given, as well as a desired stiffness profile. This thesis proposes a method to generate a new DS, called Variable Stiffness DS (VSDS), that encodes desired stiffness profile. This VSDS follows the desired DS motion, and has the spring like symmetrical attraction towards a reference trajectory which is decided by original DS and initial position. The VSDS is fed in a closed loop control configuration, which enables a compliance and safe behavior in interaction with environment. During the motion, robot will resist the small perturbations and stick to the reference path because of the symmetrical attraction. When the robot is far away from the reference path e.g because of a human perturbation, the method generates another VSDS online based on current robot state and drive the robot towards goal point. This method is tested on 7DOF KUKA LWR robot arm, and it performs well in motion execution, human robot interaction, trajectory regeneration and control with variable stiffness profile. A task of inserting a charger into socket is also accomplished using VSDS.

Contents

Chapter 1

Introduction

Since robots entered industry, they relieve human from lots of tasks that are repetitive, dangerous or that demand high precision and high power. Hence, they play an irreplaceable role in modern industry. Unlike the working environment in industry which is usually certain and clear for robot, the environment in real life is unpredictable and complex. So industrial robots controlled in open loop configurations, unlike humans, are inherently stiff, unable to interact with an uncertain environment and unable to collaborate safely with humans. However, endowing robot with compliance behavior is becoming more and more significant in robot research. On one hand, industrial robots are expected to take over all manipulation tasks that need a mass of interactions with environments. On the other hand, safe interactions with environment is one basic requirement for service robots as health care or work assistance.

Compliance is exhibited at three different levels ([KB19]): 1) Compliance at the force-level: the robot can fulfill a specific task, but it only keeps compliant toward small perturbations. 2) Compliance at the motion-level: the robot can finish a particular task, even there is a perturbation resulting in variation of motions, the robot can still fulfill the task. 3) Compliance at the task-level: the robot can switch between tasks based on the intention of human partner. This thesis focuses on the compliance at both force-level and motion-level.

To realize the compliance, impedance control ([Hog85]) is applied to robots. A robot is usually modelled as a mass-spring-damper system. Impedance controller controls the relationship between external forces (torques) and resulting velocity (angular velocity) in terms of impedance of the system. In general, a high impedance means stiff behavior because the robot tries to rejects external perturbation and stay in its original position. A low impedance means compliant behavior and robot tend to comply with the external forces. The benefits of varying the impedance according to the task requirements has been shown by many works ([KB13], [RCC$^+$13]). Defining impedance profile is achieved by many different methods including Learning from Demonstration (LfD). Among the three impedance parameters, i.e. inertial, damping and stiffness, the stiffness is most related to robot compliance behavior.

Therefore, this thesis focuses on encoding desired stiffness profile into robot motion tasks that is described by a dynamical system with a global attractor.

Dynamical Systems (DS) are becoming increasingly popular to model tasks or generate motions in robotics. Autonomous DS is a DS that does not depends on time and extra control input. It takes the state variable, e.g. the robot Cartesian position or joint position, as input, and returns the change rate of the state variable, e.g. the Cartesian velocity or joint velocity. With its advantages of convergence to an attractor point or a limit cycle, it performs well on modeling the periodic movements or reaching movements, and its application also extends to LfD area. LfD endows robots with the capability to learn a motion or a task from human demonstrations. Different DS under the paradigm LfD are proposed, such as DMP ([Sch06]), SEDS ([KZB11]) and LMDS ([KKB15]). These DS will be introduced later in Chapter 2.

Although autonomous DS generates the change rate of state variable only depending on state variable itself, traditional impedance controller configuration cannot benefit from this property. Traditional configuration of impedance control is open loop. In this configuration, a reference trajectory is generated by integrating the DS. The actual position of robot is not applied to the DS except initialization at beginning of motion. If there is a inconsistency of robot actual state and the reference trajectory, for instance, the robot is stopped by obstacle or human in the middle of reference trajectory, the error between actual position and integrated position increases. As the impedance controller pulls the robot from actual position to desired position, an increasing force is generated by the controller. This can damage the robot or the environment. Therefore a passive controller in closed loop configuration is proposed in [KB15]. The DS is updated with actual position of robot by the feedback, and the controller send control command based on velocity error.

The main contribution of this thesis is encoding a desired stiffness profile into the DS under the closed loop control configuration. With any DS and a start point inside the workspace of robot, a reference path and a new modified DS, called Variable Stiffness DS (VSDS), encoded with the desired stiffness profile are generated. Using the closed loop motion generation, we get rid of the notion tracking a trajectory, which tracks the time indexed reference motion. This method realize force-level compliance and motion-level compliance simultaneously. Without any perturbations, robot will stick to the reference path. If a deviation happens, robot will be dragged back to reference path. The compliance behavior is defined by stiffness profile, which means the higher the stiffness is, the harder to drag robot from reference path and the faster robot returns to the path. When a large perturbation happens, this approach regenerates a DS and fulfills the task.

The rest of this report is structured as follow: Chapter 2 is the literature review of three topics: variable impedance control (VIC), dynamical systems (DS) and closed loop motion generation (CLMG). Chapter 3 introduces the approach of this thesis and illustrates with the simulation results. Experimental results on a 7 DOF KUKA LWR are shown in Chapter 4. Finally, Chapter 5 presents the conclusion and future work of this thesis.

Chapter 2

Related Work

In this chapter, relevant literature is reviewed in three aspects: variable impedance control, dynamical systems and closed loop motion generation. This thesis is aimed at endowing robot with compliant behavior, which can be modeled as different stiffness profiles in variable impedance control. The stiffness profiles are encoded in dynamical systems. Relevant works of variable impedance control and dynamical systems are reviewed in Section 2.1 and Section 2.2 respectively. The framework of this thesis is under a closed loop configuration, which is introduced in Section 2.3

2.1 Variable Impedance Control

It is still a difficult problem to control the robot to have physical contact with the environment. One typical control class which controls robot in physical contact is impedance control ([Hog85]). Hogan uses the physical concept effort (force) and flow (velocity) and models a second order linear dynamical system as a virtual inertia, damping and stiffness system. The causality clarifies a mechanical system can either take velocity or force as output, but not both. A system uses impedance control when the input is velocity and output is force. In contrast, a system uses admittance control when the input is force and output is velocity. In general, impedance control is more robust in rigid contact, while admittance control provides higher accuracy in non-contact tasks [OMN10].

Although impedance control with invariant impedance parameters performs well in some situations ([Hog87, Par01]), constant impedance parameters may not be ideal in all situations. For instance, a high stiffness makes robot try to maintain the current states, resulting a performance with higher accuracy. But high stiffness means low compliance, when interacting with environment, robot applies higher force to the environment, this may cause danger. [KB12] shows that without varying the impedance, task like lighting a match is not possible to succeed.

Many researches show that human use varying impedance in task execution ([BOF$^+$01, YGH$^+$11]). For both human and robots, using variable impedance parameters is more flexible and provides better performance in many tasks. The variable impedance profile can be obtained by different methods, one of which is Learning from Demonstrations (LfD). Free motion task information like desired motion trajectory or velocity are learned from human demonstrations. However, for tasks requiring specific contact forces, or tasks requiring compliance when interacting, additional information like impedance should be learned. In [CSC10], a method deriving stiffness variations for a compliant controller from trajectory demonstrations is proposed. The demonstrations is fitted to Gaussian based DMP model, and the stiffness profile is defined proportional to the inverse of the observed covariance. [RCC$^+$13] uses weighted least square method to estimate stiffness matrix. By assuming the robot is driven by a set of virtual springs, each stiffness matrix is estimated from demonstrated positions and sensed forces.

Besides learning impedance profile from demonstrated trajectory data, robot can also learn stiffness parameters from human robot interaction. [KB12] proposes an online incremental algorithm that allows the user to interactively teach a stiffness profile through human-robot interaction. The user teaches the scale of decreasing stiffness by wiggling the end-effector around the equilibrium point. The stiffness has a linear relationship to the standard deviation of the perturbations, which is easy for user to identify the influence of the teaching. This work is extended in [KB13] with learning the stiffness with haptic signals from human. A frequency domain is used to separate the interactions. And it extends the framework with a mode for increasing the stiffness and for stiffness in joint space.

Other approaches to obtain impedance profile including optimal control, reinforcement learning and biologically inspired approaches. [HWWAS11] presents a solution which use optimal control to find the stiffness of variable impedance actuators which maximizes the link side velocity. [BTSS11] uses reinforcement learning for learning variable impedance policies. The approach formulates the varying stiffness as a differential equation which is an additional state in the dynamic movement primitives framework. However, a well-designed cost function is indispensable in both reinforcement learning and optimal control. Biologically inspired approaches are also intuitive because humans and animals performs perfectly in terms of compliance. [YGH$^+$11] presents a human-like controller by investigating and modeling human motor control. The controller adapts position trajectory, feedforward force and impedance in the presence of unknown dynamics.

2.2 Dynamical Systems

In robotics, Dynamical System (DS) has been advocated as a powerful method for motion generation and task modeling ([KZB11, KZB14]). As one of the most general and flexible methods modeling motion plans, numerous DS formulations have been proposed. Complex tasks such as playing table tennis [KMK$^+$10] or peeling a zucchini [FB17] are well approximated by DS from human demonstrations.

A popular DS framework is Dynamic Movement Primitives (DMPs), which is first proposed in [Sch06] and updated in [INH$^+$13]. DMPs performs well in both rhythmic and discrete motions. In this framework, a nonlinear forcing term is introduced to modify the behavior. The forcing term is a nonlinear combination of simple canonical systems, which are simple dynamical systems converging to zero or a limit circle. The state of canonical system is called phase variable, as it decays from initial value to zero, the system converges to global attractor. The phase variable is a function of time, so DMP depends on time implicitly. DMP shows robustness and flexibility in application, because it has capacity in both spatial and temporal scaling. It gains popularity in both reinforcement learning [BTSS11] and imitation learning [SIB03].

Instead of using forcing term in DMP, another class of DS depends only on the kinematic state variable, e.g. joint or Cartesian positions or velocities. One example is the Stable Estimator of Dynamical Systems (SEDS) in [KZB11], which is a time invariant and globally asymptotically stable DS. This method uses GMM for modeling demonstrations of position and velocity. Through rearrangement of the GMM parameters, a robot motion can be formulated as a non-linear combination of several linear DS, whose equilibrium point is at the goal point. The parameters are learned from demonstrations using optimization problem to minimize the likelihood or mean square error. To ensure global asymptotic stability of the DS, some constraints are added to the optimization. A continue work introduces another methods to ensure stability of the existing DS, i.e. SEDS-II in [KZB14]. This approach firstly learns a valid Lyapunov function from demonstrations by solving optimization problem, then generates the stabilizing commands which forces the motion in the direction which the Lyapunov function decreases.

[KKB15] proposed a Locally Modulated Dynamical Systems (LMDS), which locally reshapes an existing DS while preserving the stability. An incremental learning algorithm for LMDS is introduced, which is learned from training data using Gaussian processes. [FFB18] proposes a Locally Active Global Stable DS (LAGS-DS), which is a sum of one global DS which defines the desired trajectory and several local linear DS which decomposes the desired trajectory and is encoded with spring-like behavior. The LAGS-DS provides not only global convergence but also a stiffness-like symmetric attraction behavior around a reference-trajectory.

2.3 Closed Loop Motion Generation

DS as motion generators have the capacity to model a task and generate smooth trajectories ([KB19], [KZB11]). Traditional motion generator is placed in an open loop configuration (Figure 2.1). The DS is integrated in a separate loop to generate a reference trajectory. Then the trajectory is tracked by the classical impedance control. The actual position of robot is not applied to the DS except initialization at beginning of motion. However, this makes DS unable to react to the environmental disturbance, thus the robot behavior does not consist with task model. When a mismatch between actual robot motion and planned motion happens, the accumulated

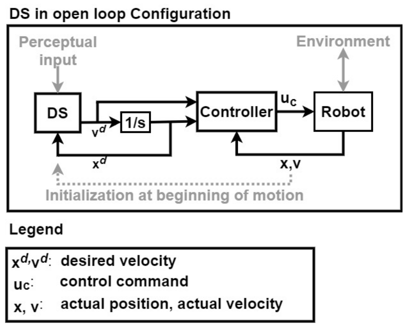

DS in open loop Configuration
Perceptual input
Environment
DS
1/s
Controller
Robot
v^d
x^d
u_c
x,v
Initialization at beginning of motion
Legend
x^d,v^d: desired velocity
u_c: control command
x, v: actual position, actual velocity

Figure 2.1: An open loop control configuration with DS. [KB15]

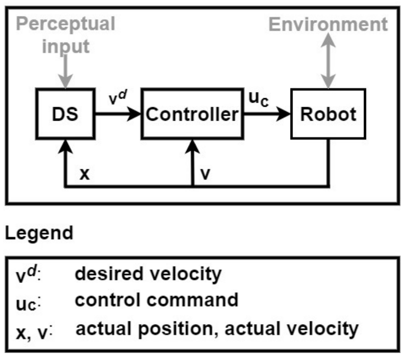


Figure 2.2: A closed loop control configuration with DS. [KB15]

error may results in large interaction force or harsh motion.

A robot with closed loop motion generation configuration can respond to physical perturbations and guarantee a safe interaction. In [KB15], a novel control architecture is proposed, which uses DS in a closed loop configuration and guarantees stable interaction with any passive environment. As illustrated in Figure 2.2, the DS is updated with actual position of robot by the feedback, and the controller send control command based on velocity error. This means the generated motion is affected by the actual robot state.

[KB19] also proposes a closed loop configuration which has DS as motion generator. The adaptive motion generator can combine several DS, in order to comply to the human intention of transiting from one task to another. In Figure 2.3, b_i are the task-beliefs. $\hat{x}_r$ and $\dot{\hat{x}}_r$ are the actual robot position and velocity, respectively. $f_i\left(x_r\right)$ are the corresponding task velocity. The adaptation mechanism of b_i is based on the similarities between each task DS and the actual robot velocity. This work realizes a task-level compliance behavior of robot.

[KZKB14] proposes a single-loop architecture which performs feedback motion generation and state varying impedance control at once, meanwhile the stability is guaranteed (Figure 2.4). The robot motion is modeled as an autonomous DS, which is formulated from Gaussian Mixture Regression (GMR). Each Gaussian function

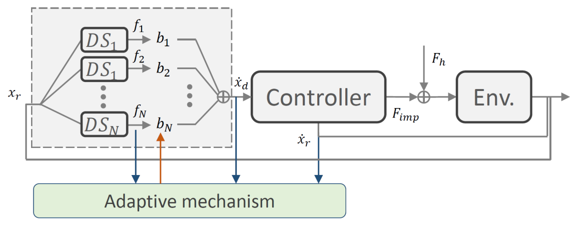


Figure 2.3: Task adaptation with closed loop configuration. [KB19]

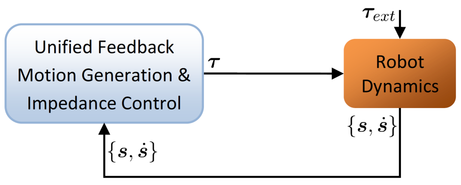


Figure 2.4: Single loop configuration. [KZKB14]

in GMR is seen as a linear spring-damper system. This DS takes robot position and velocity $(s, \dot{s})$ as input and generates the robot control torque (τ) as output. A stability proof based on Lyapunov function is also provided in this work.

Similarly, [KZK17] introduces a unified motion and variable impedance control policy, which regulates both robot motion and stiffness profile. The control policy is learned from demonstration data including position, velocity, torque and stiffness. The learning process is done by solving two optimization problems, which guarantees the convergence and preservation of velocity profile.

Chapter 3

Technical Approach

This chapter describes the general approach of this thesis. Some preliminaries like Dynamical Systems, Lyapunov functions, close loop motion generation and robot dynamics are firstly introduced in Section 3.1. Then the approach to construct the new DS is described in Section 3.2.

3.1 Preliminaries

3.1.1 Dynamical systems

Dynamical Systems (DS) describes how a system will change in the next step. Let $\boldsymbol{\xi} \in \Re^d$ be the d-dimensional state vector, which can be defined arbitrarily to represent the system. In this thesis, $\boldsymbol{\xi}$ is defined as the position in joint coordinates or Cartesian coordinates. The DS usually depends not only on the state vector, but also on time and input vector. A DS is called autonomous if it is not an explicit function of time:

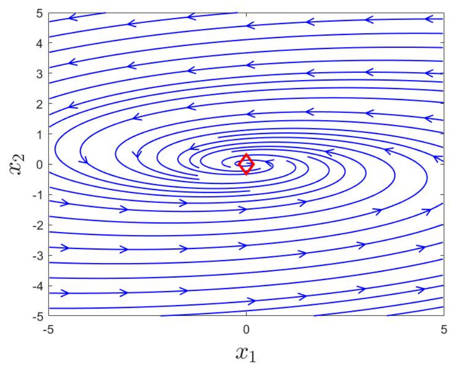
$$\dot{\boldsymbol{\xi}} = \mathbf{f}(\boldsymbol{\xi})$$

where $\mathbf{f}(\boldsymbol{\xi})$: $\Re^d \mapsto \Re^d$ is the function that maps current system state to state changing rate.

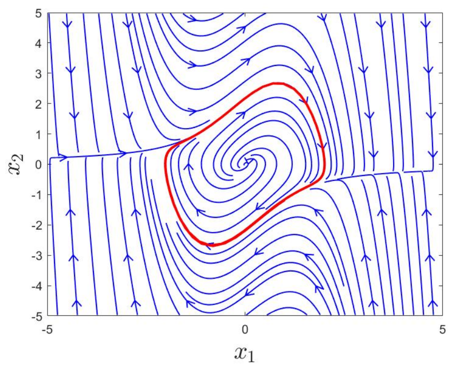
A crucial property of DS is stability, which means a well-designed DS converges to an equilibrium point or a limit cycle. In this thesis we focus on the stable DS which converges to equilibrium point, we also call it attractor. Consider an autonomous DS: $\dot{\boldsymbol{\xi}} = \mathbf{f}(\boldsymbol{\xi}(t))$, $\boldsymbol{\xi}(0) = \boldsymbol{\xi}_0$, where $\boldsymbol{\xi}(t) \in \Re^d$ is the state variable. Suppose the equilibrium of $\mathbf{f}$ is $\boldsymbol{\xi}^* \in \Re^d$, i.e. $\mathbf{f}(\boldsymbol{\xi}^*) = \mathbf{0}$.

- If for every $\epsilon > 0$, there exits a $\delta > 0$ such that, if $\| \boldsymbol{\xi}(0) - \boldsymbol{\xi}^* \| < \delta$, then for every $t \geq 0$ we have $\| \boldsymbol{\xi}(t) - \boldsymbol{\xi}^* \| < \epsilon$, then equilibrium $\boldsymbol{\xi}^*$ is globally Lyapunov stable.

- If there exists $\delta > 0$ such that if $\| \boldsymbol{\xi}(0) - \boldsymbol{\xi}^* \| < \delta$, then $\lim_{t \to \infty} \| \boldsymbol{\xi}(t) - \boldsymbol{\xi}^* \| = 0$, then equilibrium $\boldsymbol{\xi}^*$ is globally asymptotically stable.



(a) Dynamical System converges to a point (marked with red diamond)



(b) Dynamical System converges to a limit circle (marked with red line)

Figure 3.1: Convergence behavior of Dynamical Systems

As will be analyzed in section 3.1.1, the stability of linear DS can be easily seen from the parameters in the function. Meanwhile, for non-linear DS, the stability analysis is not trivial. But the Lyapunov function provides a strong tool for stability analysis. Assume $V(\boldsymbol{\xi})$: $\Re^d \mapsto \Re$ is a continuously differentiable function, which is called Lyapunov function.

- If V satisfies following conditions:
  - $V(\boldsymbol{\xi}^*) = 0$
  - $V(\boldsymbol{\xi}) > 0, \forall \boldsymbol{\xi} \neq \boldsymbol{\xi}^*$
  - $\dot{V}(\boldsymbol{\xi}) \leq 0, \forall \boldsymbol{\xi} \neq \boldsymbol{\xi}^*$
  - $\| \boldsymbol{\xi} \| \to \infty \Rightarrow V(\boldsymbol{\xi}) \to \infty$

  then the DS $\dot{\boldsymbol{\xi}} = \mathbf{f}(\boldsymbol{\xi}(t))$ is globally Lyapunov stable.

- If V satisfies following conditions:
  - $V(\boldsymbol{\xi}^*) = 0$
  - $V(\boldsymbol{\xi}) > 0, \forall \boldsymbol{\xi} \neq \boldsymbol{\xi}^*$
  - $\dot{V}(\boldsymbol{\xi}) < 0, \forall \boldsymbol{\xi} \neq \boldsymbol{\xi}^*$
  - $\| \boldsymbol{\xi} \| \to \infty \Rightarrow V(\boldsymbol{\xi}) \to \infty$

  then the DS $\dot{\boldsymbol{\xi}} = \mathbf{f}(\boldsymbol{\xi}(t))$ is globally asymptotically stable.

If we limit the state variable in a region which contains the origin, i.e. $\boldsymbol{\xi}(t) \in \mathcal{D} \subseteq \Re^d$, and we leave the boundlessness of Lyapunov function, then the stability is said to be locally stable.

The approach in this thesis is basically a combination of linear DS. This simple but powerful DS takes the form:

$$\dot{\boldsymbol{\xi}} = \mathbf{A}\left(\boldsymbol{\xi} - \boldsymbol{\xi}^{*}\right) \tag{3.1}$$

where $\mathbf{A} \in \Re^{d \times d}$ is a constant matrix, indicating the stability of the linear DS. $\boldsymbol{\xi}^{*}$ is the global attractor if the DS is stable. The stability is related to eigenvalues $\lambda_i, i = 1 \dots d$ of $\mathbf{A}$. If and only if the real part of all eigenvalues are strictly negative, i.e. $re\left(\lambda_i\right) < 0, \forall i = 1 \dots d$, the linear DS in equation 3.1 is stable and converges to attractor $\boldsymbol{\xi}^{*}$. The good properties of linear DS are that 1) there exists only one global attractor, 2) stability is easy to achieve by tuning the sign of eigenvalues of matrix $\mathbf{A}$, 3) the shape of DS is easy to vary by tuning the magnitude of eigenvalues of matrix $\mathbf{A}$.

The effects of eigenvalues on stability and shape of linear DS are here illustrated simply with 2-dimensional case. For more detailed explanation please refer to Chapter 5 of book [Str01].

Without loss of generality, the equilibrium point (also referred to attractor below) is set to origin. Other attractors can be reached by shifting the equilibrium point. A linear DS with 2-D state variable $\mathbf{x}$ is written as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{3.2}$$

To get the eigenvalue, a quadratic equation below should be solved:

$$\lambda^2 - \tau\lambda + \Delta = 0$$

where

$$\tau = trace\left(\mathbf{A}\right) = a + d$$

$$\Delta = det\left(\mathbf{A}\right) = ad - bc$$

Then

$$\lambda_1 = \frac{\tau + \sqrt{\tau^2 - 4\Delta}}{2}, \lambda_2 = \frac{\tau - \sqrt{\tau^2 - 4\Delta}}{2}$$

With different values of τ and Δ, DS from formula 3.2 has different stability properties and different shapes:

- If $\Delta < 0$, the eigenvalues are both real but with opposite signs, then the attractor is a saddle point (Figure 3.2(a)).
- If $\Delta > 0$:
  - If $\tau^2 - 4\Delta > 0$, the eigenvalues are real with same sign, the attractor is called nodes (Figure 3.2(b), 3.2(c)).

  - If $\tau^2 - 4\Delta < 0$, the eigenvalues are complex conjugate, the attractor is spirals or centers (Figure 3.2(d), 3.2(e), 3.2(f)).
  - If $\tau^2 - 4\Delta = 0$, on the parabola are star nodes and degenerate nodes (Figure 3.2(g), 3.2(h)).
  - If $\tau < 0$, both eigenvalues have negative real parts, so the attractor is stable (Figure 3.2(b), 3.2(d)).
  - If $\tau > 0$, both eigenvalues have positive real parts, so the attractor is unstable spirals or nodes (Figure 3.2(c), 3.2(e)).
  - If $\tau = 0$, the eigenvalues are pure imaginary, the attractor is centers (Figure 3.2(f)).

- If $\Delta = 0$, at least one of the eigenvalues is zero, the attractor is non-isolated.

Figure 3.3 summaries the type and stability.

This thesis focused mainly on the stable node linear DS type. As Figure 3.2(b) shows, the state in this DS has a behavior that it firstly converges to a "center line" symmetrically, then converge to the attractor. This is similar to the goal of this thesis, which is a symmetrical convergence to a desired path. More details of the stable node DS type will be discussed in the following.

The constant matrix in 3.2 can be written as:

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$$

where $\mathbf{Q}$ is a direction matrix, here with a rotation matrix form:

$$\mathbf{Q} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

and $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues, i.e.

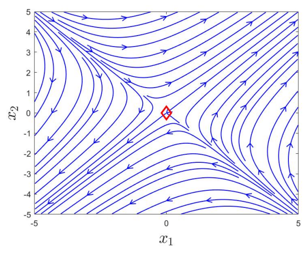
$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

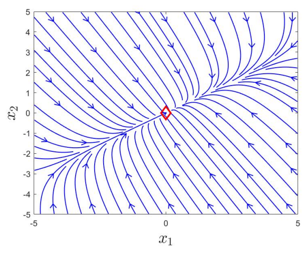
The ratio $\gamma = \frac{\lambda_2}{\lambda_1}$ controls the extent of the state variable convergence to the center line. And the rotation angle α defines the orientation of the center line. The influence of γ and α can be seen in Figure 3.4.

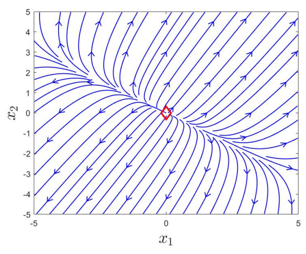
Then the analytical solution of 3.1 can be written as:

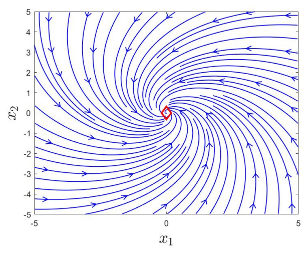
$$\mathbf{x}(t) = c_1 e^{\lambda_1 t}\mathbf{v}_1 + c_2 e^{\lambda_2 t}\mathbf{v}_2 \tag{3.3}$$

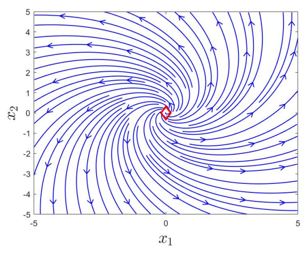
where the c_1 and c_2 are constants that are decided by the initial state of 3.1, λ_1, λ_2 are eigenvalues of matrix $\mathbf{A}$, and $\mathbf{v}_1, \mathbf{v}_2$ are the eigenvectors of $\mathbf{A}$, i.e. the columns of direction matrix $\mathbf{Q}$.

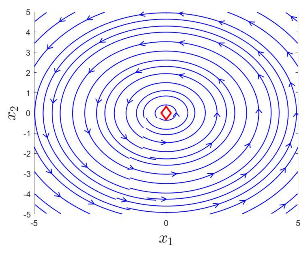

(a) Saddle

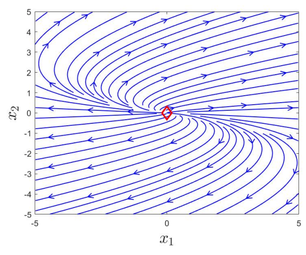

(b) Stable node


(c) Unstable node


(d) Stable spiral


(e) Unstable spiral


(f) Center


(g) Star node

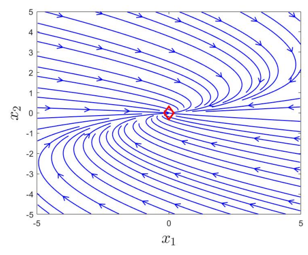

(h) Degenerate node

Figure 3.2: Different linear dynamical systems

3.1.2 Stable Estimator of Dynamical Systems

The Stable Estimator of Dynamical Systems (SEDS) is a learning method based on Gaussian Mixture Model (GMM). It presents robot motion as a non-linear autonomous DS. SEDS is ensured to be global asymptotically stable at the target ([KZB11]). In this report, SEDS is used extensively as the desired original DS. It is expressed as a nonlinear sum of linear DS:

$$\dot{\boldsymbol{\xi}} = \mathbf{f}\left(\boldsymbol{\xi}\right) = \sum_{i=1}^{N} \tilde{h}_i\left(\boldsymbol{\xi}\right)\left(\mathbf{A}_i\boldsymbol{\xi} + \mathbf{b}_i\right) \tag{3.4}$$

where $\mathbf{A}_i \in \Re^{d\times d}$, $\mathbf{b}_i \in \Re^d$ and $\mathbf{A}_i\boldsymbol{\xi} + \mathbf{b}_i$ is the linear DS. N denotes the number of linear DS. $\tilde{h}_i\left(\boldsymbol{\xi}\right) \in \Re$ is the nonlinear weighting function and satisfies $\sum_{i=1}^{N} \tilde{h}_i\left(\boldsymbol{\xi}\right) = 1$.

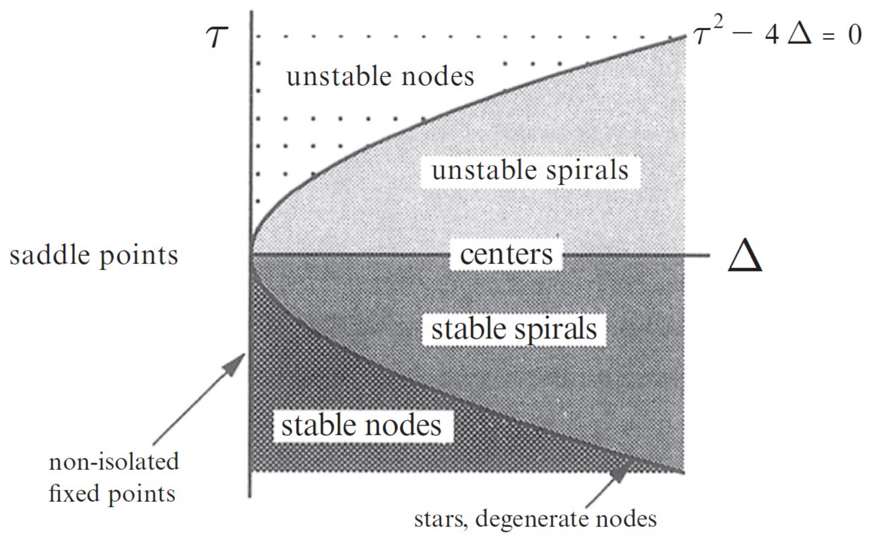


Figure 3.3: Type and stability of different attractors. [Str01]

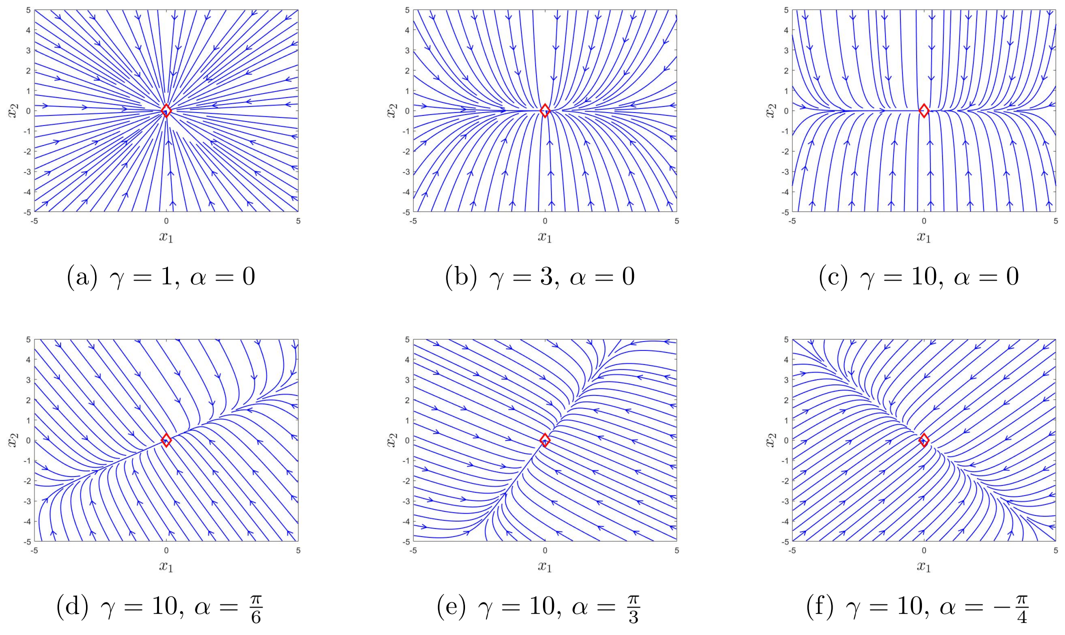
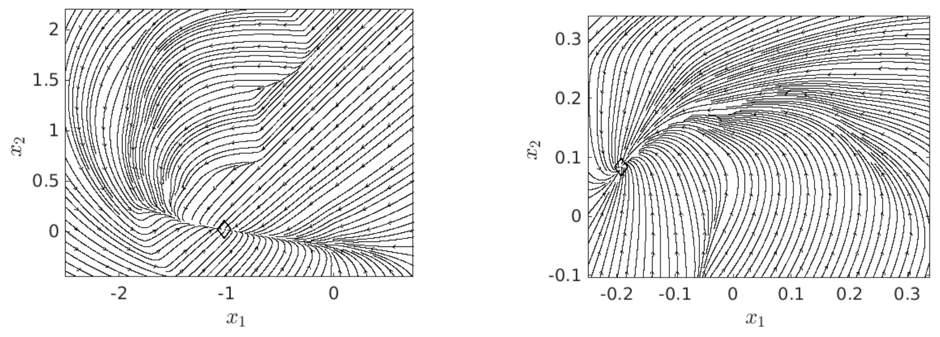


Figure 3.4: Different linear dynamical systems with stable node attractor

$$h_i(\boldsymbol{\xi}) = p_i \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma}_i)}} e^{-\frac{1}{2}(\boldsymbol{\xi}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{\xi}-\boldsymbol{\mu}_i)}$$

where the parameters p_i, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are priors, mean and covariance matrix of GMM. And



(a) C shape SEDS (b) Convex shape SEDS

Figure 3.5: Visualization of two SEDS used as desired DS. The black diamond denotes the global attractor. The starting point is not marked here, for it can be any point on the plane in our case.

$$\tilde{h}_i(\boldsymbol{\xi}) = \frac{h_i(\boldsymbol{\xi})}{\sum_{j=1}^{N} h_j(\boldsymbol{\xi})}$$

The above mentioned parameters are learned by solving an optimization problem which minimizes the likelihood or mean square error. To guarantee the global asymptotic stability of 3.4, the optimization problem also has the following constraints:

$$\begin{cases} \mathbf{b}_i = -\mathbf{A}_i \boldsymbol{\xi}^* \\ \mathbf{A}_i + \mathbf{A}_i^T \prec 0 \end{cases} \quad \forall i = 1 \dots N \tag{3.5}$$

where $\boldsymbol{\xi}^*$ is the global attractor.

In this report, 2 different SEDS are used (Figure 3.5). One with "C" shape is learned from hand drawn demonstration in matlab GUI. Onother with convex curve shape is learned from a real robot motion demonstration. The parameters of these 2 SEDS can be found in Appendix B.

3.1.3 Passive DS control configuration and compliant behaviors

This section introduces robot rigid body dynamics under the closed loop passive DS control configuration (Figure 2.2) of [KB15]. In this thesis, variable $\boldsymbol{\xi}$ is a generalized state variable, which can be robot joint position or Cartesian position. Consider a gravity compensated robot dynamics with state variable $\boldsymbol{\xi}$:

$$\mathbf{M}(\boldsymbol{\xi})\ddot{\boldsymbol{\xi}} + \mathbf{C}\left(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}\right)\dot{\boldsymbol{\xi}} = \mathbf{u}_c + \mathbf{u}_e \tag{3.6}$$

where $\mathbf{M}(\boldsymbol{\xi}) \in \Re^{d\times d}$ corresponds to inertia matrix and $\mathbf{C}\left(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}\right) \in \Re^{d\times d}$ is the Coriolis matrix. $\mathbf{u}_c \in \Re^d$ and $\mathbf{u}_e \in \Re^d$ are control torque and external torque respectively. $\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, \ddot{\boldsymbol{\xi}} \in \Re^d$ denote the position, velocity and acceleration in either joint or Cartesian space.

In open loop control configuration:

For a tracking problem, an impedance control is written as:

$$\mathbf{u} = \mathbf{M}\left(\ddot{\boldsymbol{\xi}}_d - \ddot{\boldsymbol{\xi}}\right) + \mathbf{D}\left(\dot{\boldsymbol{\xi}}_d - \dot{\boldsymbol{\xi}}\right) + \mathbf{K}\left(\boldsymbol{\xi}_d - \boldsymbol{\xi}\right) \tag{3.7}$$

where $\mathbf{u} \in \Re^d$ is force as output of impedance controller. $\boldsymbol{\xi}_d$, $\dot{\boldsymbol{\xi}}_d$, $\ddot{\boldsymbol{\xi}}_d \in \Re^d$ are desired position, velocity and acceleration, respectively. $\mathbf{M} \in \Re^{d\times d}$, $\mathbf{D} \in \Re^{d\times d}$, $\mathbf{K} \in \Re^{d\times d}$ are the impedance parameters, i.e. inertia, damping and stiffness.

To be more specific, here we choose the PD+ controller proposed in [PP88]. This can be seen as an impedance controller without inertia shaping (i.e compliance control) . The control torque $\mathbf{u}_c \in \Re^d$ is defined as:

$$\mathbf{u}_c = \mathbf{u}_d + \mathbf{D}\left(\dot{\boldsymbol{\xi}}_d - \dot{\boldsymbol{\xi}}\right) + \mathbf{K}\left(\boldsymbol{\xi}_d - \boldsymbol{\xi}\right) \tag{3.8}$$

where $\mathbf{u}_d \in \Re^d$ is a dynamic compensation torque defined as:

$$\mathbf{u}_d = \mathbf{M}(\boldsymbol{\xi})\,\ddot{\boldsymbol{\xi}}_d + \mathbf{C}\left(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}\right)\dot{\boldsymbol{\xi}}_d \tag{3.9}$$

The closed loop behavior becomes:

$$\mathbf{u}_e = \mathbf{M}(\boldsymbol{\xi})\,\ddot{\mathbf{e}} + \left(\mathbf{D} + \mathbf{C}\left(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}\right)\right)\dot{\mathbf{e}} + \mathbf{K}\mathbf{e} \tag{3.10}$$

where $\mathbf{e} = (\boldsymbol{\xi} - \boldsymbol{\xi}_d)$, $\dot{\mathbf{e}} = \left(\dot{\boldsymbol{\xi}} - \dot{\boldsymbol{\xi}}_d\right)$ and $\ddot{\mathbf{e}} = \left(\ddot{\boldsymbol{\xi}} - \ddot{\boldsymbol{\xi}}_d\right)$ are the tracking errors during the motion.

In the open loop control configuration in Figure 2.1, the desired trajectory is generated from a separate integration loop. This means the motion generation loop cannot receive the actual robot state. When an interaction between robot and the environment happens, or other reasons which causes the mismatch between planned motion and actual trajectory happens, an error occurs and starts to accumulate. This can result in a large contact force due to the accumulated error in stiffness part. For instance, a human stops the robot in the middle of a motion, as the the difference between desired position/velocity increases, the robot applies rising force to human. When the robot is released again, it will acquire a large acceleration to track the reference trajectory. The rising force and the large acceleration are not safe for both human and robot.

In closed loop control configuration:

In closed loop control configuration in [KB15], the robot motion planner is aware of the robot current state all the time, hence robot motion is generated in real time. This increases the safety in possible uncertainty in the environment or unforeseen interaction with human. It ensures a compliant behavior of robot, i.e. when the robot is perturbed, the robot follows the perturbation compliantly. Because the robot motion is generated from DS based on current position (Figure 2.2), unlike in open loop, there is no incremental error, thus, no huge jerk when the perturbation stops. When the robot is again released, the robot starts moving towards the global attractor smoothly from the position where the perturbation ends. The controller generate a control signal $\mathbf{u}_c$ by tracking the desired velocity which is generated from nominal motion plan DS $\mathbf{f}(\boldsymbol{\xi})$. This controller ensures stable interaction with a passive environment by using a negative velocity error feedback control law:

$$\mathbf{u}_c = -\mathbf{D}(\boldsymbol{\xi})\left(\dot{\boldsymbol{\xi}} - \mathbf{f}(\boldsymbol{\xi})\right) \tag{3.11}$$

where $\mathbf{D}(\boldsymbol{\xi}) \in \Re^{d\times d}$ is a state-varying damping matrix, which enables selective energy dissipation in desired and undesired directions. The $\mathbf{D}(\boldsymbol{\xi})$ is chosen as:

$$\mathbf{D}(\boldsymbol{\xi}) = \mathbf{Q}(\boldsymbol{\xi})\boldsymbol{\Lambda}\mathbf{Q}(\boldsymbol{\xi})^T$$

where $\boldsymbol{\Lambda} \in \Re^{d\times d}$ is a diagonal matrix with $\lambda_1, \cdots, \lambda_d \geq 0$ indicating damping values. The columns of $\mathbf{Q}(\boldsymbol{\xi}) \in \Re^{d\times d}$ are orthonormal basis $\mathbf{e}_1, \cdots, \mathbf{e}_d$. The energy dissipation direction is related to the direction of desired trajectory, i.e. through the transform by $\mathbf{Q}(\boldsymbol{\xi})$, the damping matrix dissipates energy in the direction perpendicular to the motion, and provide driving force along the motion, hence, $\mathbf{e}_1$ is defined as $\mathbf{e}_1 = \frac{\mathbf{f}(\boldsymbol{\xi})}{\|\mathbf{f}(\boldsymbol{\xi})\|}$, which indicates the direction of the desired motion. The remaining $\mathbf{e}_2, \cdots, \mathbf{e}_d$ are directions orthogonal to desired motion.

With the definition of $\mathbf{D}(\boldsymbol{\xi})$, DS $\mathbf{f}(\boldsymbol{\xi})$ is eigenvector of $\mathbf{D}(\boldsymbol{\xi})$, corresponding eigenvalue is λ_1 (see A.3). Thus the controller in (3.11) is rewritten as:

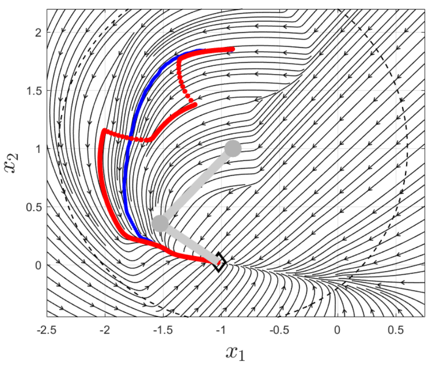
$$\mathbf{u}_c = -\mathbf{D}(\boldsymbol{\xi})\dot{\boldsymbol{\xi}} + \lambda_1\mathbf{f}(\boldsymbol{\xi}) \tag{3.12}$$

Combining (3.6) and (3.11), the closed loop dynamics becomes:

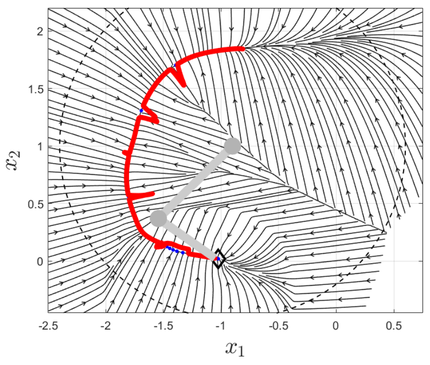
$$\mathbf{u}_e = \mathbf{M}(\boldsymbol{\xi})\ddot{\boldsymbol{\xi}} + \left(\mathbf{D}(\boldsymbol{\xi}) + \mathbf{C}\left(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}\right)\right)\dot{\boldsymbol{\xi}} - \lambda_1\mathbf{f}(\boldsymbol{\xi}) \tag{3.13}$$

Compare equation 3.13 with impedance control formulation in a open loop control configuration (equation 3.10), $\mathbf{M}(\boldsymbol{\xi})$ and $\mathbf{D}(\boldsymbol{\xi}) + \mathbf{C}\left(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}\right)$ are mass and damping matrix, respectively. $-\lambda_1\mathbf{f}(\boldsymbol{\xi})$ can be seen as a non-linear stiffness term in controller (3.13) whose stiffness is:

$$\mathbf{K}(\boldsymbol{\xi}) = -\lambda_1\frac{\partial\mathbf{f}(\boldsymbol{\xi})}{\partial\boldsymbol{\xi}} \tag{3.14}$$



(a) DS without attraction behavior



(b) DS with attraction behavior

Figure 3.6: Different DS stiffness-like attraction behaviors. Blue lines are the reference path and red lines are simulated robot motion. In (a), robot generates a new path after perturbation. In (b), the DS is encoded with high stiffness, the robot return back to the reference path after perturbation.

This shows that the stiffness depends not only on the damping $\mathbf{D}(\boldsymbol{\xi})$, but also on the properties of DS. [KB19] derives the stiffness in a particular direction $\boldsymbol{\xi}_s$ using Rayleigh quotient is:

$$K(\boldsymbol{\xi}, \boldsymbol{\xi}_s) = -\lambda_1 \boldsymbol{\xi}_s^T \frac{\partial \mathbf{f}(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \boldsymbol{\xi}_s \tag{3.15}$$

However, the non-linear stiffness cannot provide the behavior which is able to follow a reference path in the closed loop control configuration. The behavior which is similar to a stiffness attraction around the reference path is desired (Figure 3.6). But because of the complexity and non-linearity of the learned DS, it is difficult to modify the DS to obtain such desired stiffness behavior. Thus, [FFB18] and this thesis uses sum of linear DS to encode the stiffness-like attraction behavior.

In [FFB18], a Locally Active Globally Stable DS (LAGS-DS) is proposed under this passive DS control configuration, which generates the locally active stiffness-like behavior and ensures global asymptotic stability (Figure 3.7). The LAGS-DS is formulated as:

$$\dot{\boldsymbol{\xi}} = \alpha(\boldsymbol{\xi}) \mathbf{f}_g(\boldsymbol{\xi}) + \bar{\alpha}(\boldsymbol{\xi}) \mathbf{f}_l(h(\boldsymbol{\xi}), \boldsymbol{\xi})$$

where $\mathbf{f}_g(\boldsymbol{\xi})$ is a global DS learned from demonstrated data. $\alpha(\boldsymbol{\xi}) \in [0, 1]$ is an activation function, indicating that the local dynamics is active at the regions in state space. $\bar{\alpha}(\boldsymbol{\xi}) = 1 - \alpha(\boldsymbol{\xi})$. When $\alpha(\boldsymbol{\xi}) = 1$ the global DS $\mathbf{f}_g(\boldsymbol{\xi})$ is activated. $\mathbf{f}_l(h(\boldsymbol{\xi}), \boldsymbol{\xi})$ is locally active DS also learned from demonstrated data by clustering them. $h(\boldsymbol{\xi}) \in [0, 1]$ is a hyper-plane partitioning function, indicating the region of the state-space which each DS belongs to.

However, how to shape the robot stiffness in LAGS-DS is not clear. And the stiffness of this DS is coupled with the damping value (3.14). As Figure 3.7 shows,

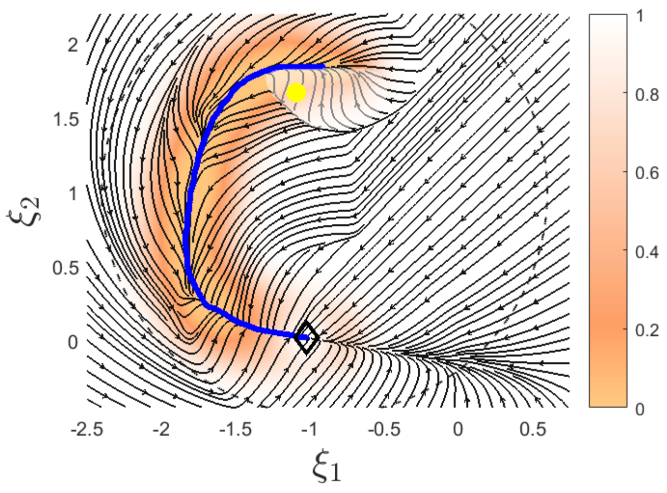


Figure 3.7: Illustration of LAGS-DS. The orange area denotes the locally activated area. Demonstrated data is shown with blue line. The LAGS-DS shows a symmetrical attraction towards the reference path, and global asymptotic stability.

the attraction behavior is only locally around the path, when the robot is dragged out of the active area, the symmetrical attraction no longer exists. And LAGS-DS is learned from demonstrated data, therefore we cannot endow an existing DS with symmetrical attraction behavior unless we have the original demonstration data. This thesis proposes a method under the passive DS control configuration to generate a modified DS with encoded stiffness profile purely based on an existing original global DS. The robot is locally attracted to a reference path and globally follows the original DS motion.

Another method is proposed in [KZK17]. The robot motion and its stiffness behavior are encoded by devising potential function, whose gradient indicates the motion generation and curvature indicates the stiffness profile (Figure 3.8). The potential function $\Phi(\boldsymbol{\xi})$ is learned from demonstrations.

The Unified Motion and Impedance Control policy is obtained by taking the gradient of the learned potential function. It takes the demonstrated data sequence as center points. A nonlinear weights function is used to transit between the center points. The control policy $\boldsymbol{\tau}_c$ consists three parts:

$$\begin{aligned}\boldsymbol{\tau}_c &= \sum_i \boldsymbol{\tau}^i_{nominal} + \sum_i \boldsymbol{\tau}^i_{attract} + \sum_i \boldsymbol{\tau}^i_{damp} \\ &= \boldsymbol{\tau}_{nominal} + \boldsymbol{\tau}_{attract} + \boldsymbol{\tau}_{damp}\end{aligned}$$

where $\boldsymbol{\tau}_{nominal}$ generate the nominal motion, $\boldsymbol{\tau}_{attract}$ attracts the state variable to the closest center point and $\boldsymbol{\tau}_{damp}$ damps the unwanted energy. The control policy is illustrated in Figure 3.9.

In this thesis, the controller in [KB15] is also adjusted. Note that in 3.14, the stiffness depends on the first eigenvalue of damping. By encoding the stiffness into

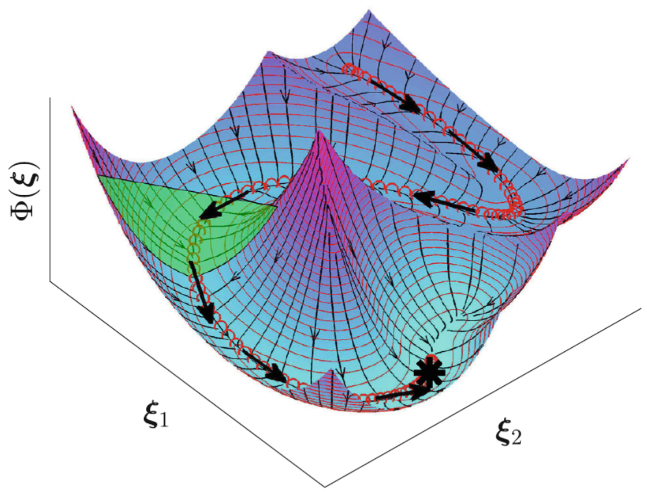


Figure 3.8: Illustration of potential function learned from demonstrations data (red circles). The arrows denote the motion due to the slope. The green surface visualizes the stiffness behavior.

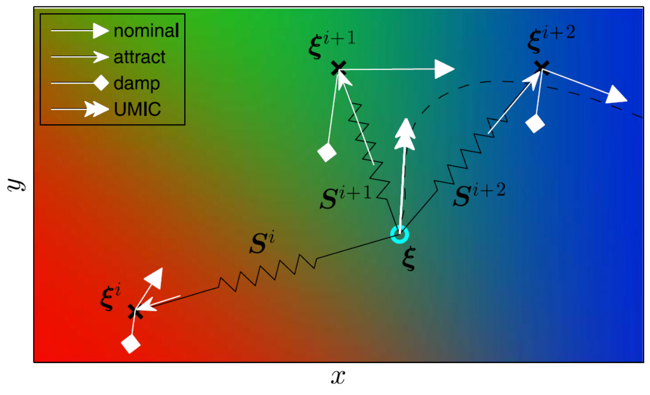


Figure 3.9: Illustration of control policy. The cross points are center points and the circle point is the query points. The RGB color map indicates the value of nonlinear weights.

DS, we decouple the stiffness from damping. The controller used in this thesis is:

$$\mathbf{u}_c = -\mathbf{D}(\boldsymbol{\xi})\,\dot{\boldsymbol{\xi}} + \mathbf{f}(\boldsymbol{\xi}) \tag{3.16}$$

the closed loop dynamics then becomes:

$$\mathbf{u}_e = \mathbf{M}(\boldsymbol{\xi})\,\ddot{\boldsymbol{\xi}} + \left(\mathbf{D}(\boldsymbol{\xi}) + \mathbf{C}\left(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}\right)\right)\dot{\boldsymbol{\xi}} - \mathbf{f}(\boldsymbol{\xi}) \tag{3.17}$$

and the stiffness of the closed loop dynamics is:

$$\mathbf{K}(\boldsymbol{\xi}) = -\frac{\partial \mathbf{f}(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$$

3.2 Approach

3.2.1 Problem statement

As discussed in 3.1.3, because the actual robot state is not feedback to the motion planner, the open loop configuration introduces an error accumulation when the actual robot motion mismatches the time-indexed reference trajectory as time increases. Using closed loop control configuration will mitigate the problem, for the motion planner is aware of actual robot state. This eliminates the notion tracking reference trajectory. Instead of tracking a reference time indexed position, velocity and acceleration sequence (a reference trajectory), the robot in this thesis try to follow a path which is generated from desired DS. [FFB18] did not show a clear way to encode the desired stiffness profile into the DS. [KZK17] did not reflect the stiffness along the motion, which is not encoding the stiffness totally.

The contribution of this thesis is that under the closed loop control configuration, this methods encodes a desired stiffness profile into a robot motion which is defined by an original desired DS. By encoding the stiffness, the stiffness part in the controller is decoupled from damping. And the robot performs force-level compliance locally and motion-level compliance globally.

Assume that a desired DS describing a motion plan i.e. desired position and velocity, as well as a desired stiffness profile are given. The goal of the thesis is to follow this motion plan in a closed loop control configuration while the interaction behavior of the robot is described by the desired stiffness profile.

To be more explicit, as shown in Figure 3.10, if the robot initial position is $\mathbf{x}_{0,1}$ (for simplicity, $\mathbf{x}$ is used in this section to represent a 2 DOF position, instead of $\boldsymbol{\xi}$) in the state space, the robot will follow the path defined by the given desired DS from $\mathbf{x}_{0,1}$ to reach the global attractor. Meanwhile, a stiffness behavior, i.e. a symmetrical attraction around the path in a certain tube area is encoded. The attraction tube area is corresponding to the user defined stiffness profile. When the robot is perturbed to a new position out of the attraction tube, i.e. $\mathbf{x}_{0,2}$, instead of attracted to the old path, the robot will start a new path define by the desired DS with stiffness-like attraction behavior.

As seen in equation 3.14, it is difficult to modify the given complex non-linear DS for a desired stiffness profile. Therefore, to realize the goal, a new DS has to be built. This new DS should encode the desired stiffness profile.

3.2.2 Variable Stiffness DS with linear DS

As shown in Figure 3.4, a linear DS with stable node attractor shows a symmetrical attraction behavior. The state variable firstly converges to a ”center line” then reaches the equilibrium point. Therefore, to obtain a stiffness-like attraction behavior around a reference path, it is possible to interpolate the path into several segments, build a stable node linear DS for each segment and concatenate those

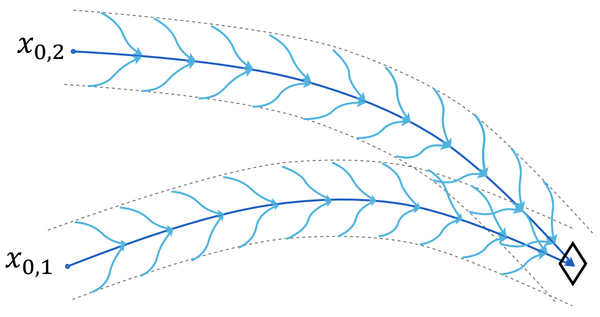


Figure 3.10: Illustration of the goal of thesis

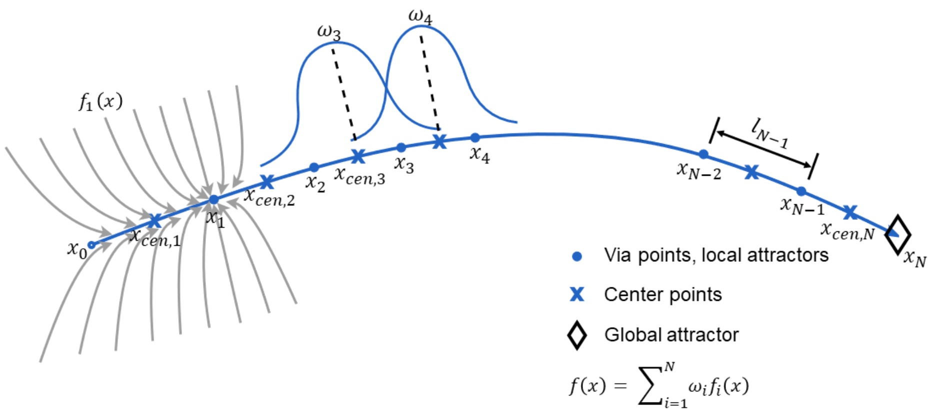


Figure 3.11: Illustration of the VSDS algorithm

linear DS. This is the basic idea of the VSDS in this thesis.

The approach is illustrated in Figure 3.11. In this figure, $\mathbf{x}_i$ are the sampled via points. $\mathbf{x}_{cen,i}$ are centers for each local DS. The local DS are $\mathbf{f}_i(\mathbf{x})$. ω_i are weighting functions or transfer functions between local DS. l_i is the length between via points $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$. $i = 1 \dots N$ holds for above mentioned. N denotes the number of local DS, as well as number of via points. The approach is still under the closed loop control configuration in [KB15] but with some modification as shown in Figure 3.13(b). This approach can be easily generalized to higher DOF. In this configuration, a VSDS $\mathbf{f}(\mathbf{x})$ is generated from the given original DS, with the parameters desired stiffness profile $\mathbf{K}_{des}$ and initial position $\mathbf{x}_{init} = \mathbf{x}_0$. In the following, an original desired DS is denoted as $\mathbf{f}_g(\mathbf{x})$ whose global attractor is $\mathbf{x}^* = \mathbf{x}_N$.

Each local DS can be seen as a spring whose stiffness is the desired stiffness. The physical interpretation is that, a sequence of springs are attached to the via

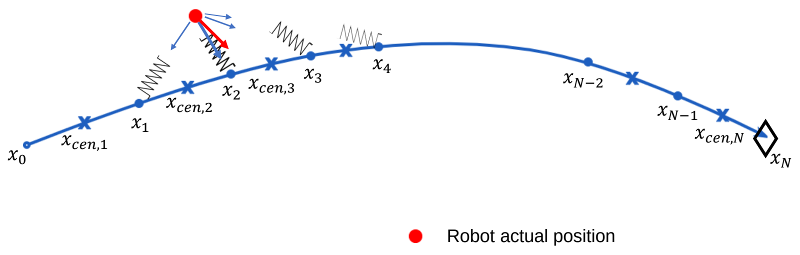
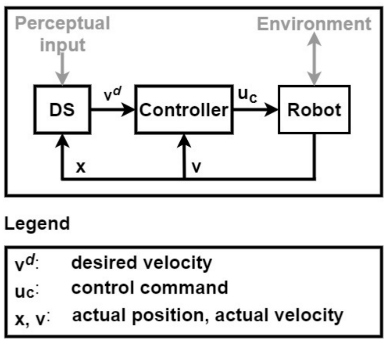
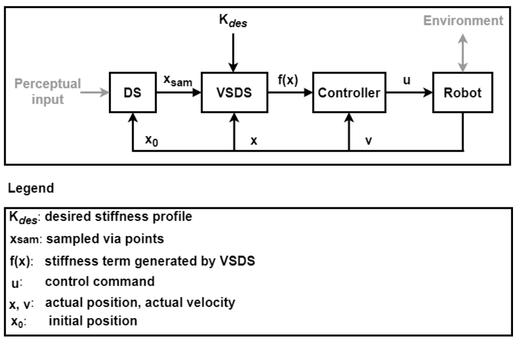


Figure 3.12: Illustration of the physical interpretation of control force with encoded stiffness. The thickness of springs denotes the effective stiffness, which is the actual stiffness multiplies with distance depending weight. The blue arrows indicate the attraction force of active springs, the width of arrow roughly shows the magnitude of force. The red arrow is the resultant force which drags robot towards reference path and the next via point.



(a) Closed loop configuration [KB15]



(b) Modified closed loop configuration

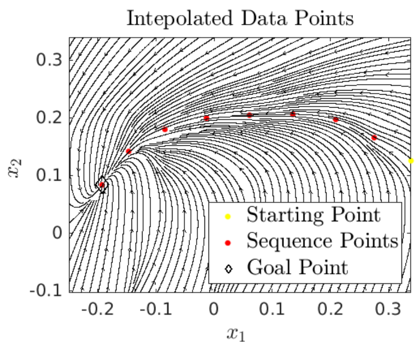
Figure 3.13: Original and modified closed loop configuration

points, which drags the robot towards the reference path and the global attractor, as Figure 3.12 shown.

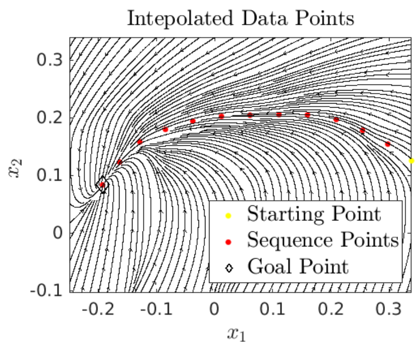
The approach is described in the following step by step.

Sampling a path with via points:

The first step is to interpolate the original DS $\mathbf{f}_g(\mathbf{x})$ in order to obtain a sequence



(a) $N = 8$



(b) $N = 12$

Figure 3.14: Generated via points with fixed points number. These results shows that with less number of via points, the distance between adjacent points are larger.

with $N+1$ via points $\mathbf{x}_{sam} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N\}$ from initial point $\mathbf{x}_0$ to global attractor $\mathbf{x}^*$.

The basis of the sampling method is a numerical integration method called Euler's method, described in Algorithm 1. Here the given original DS is $\dot{\mathbf{x}} = \mathbf{f}_g(\mathbf{x})$, initial position is $\mathbf{x}_0$, $\mathbf{x}^*$ represents the global attractor, step size is Δt and a threshold is defined as ϵ. Euler's method returns a sequence of temporary points on the path, then the sampling method chooses the via points from the temporary points.

Algorithm 1: Numerical integration with Euler's method

```
Result: $\{\mathbf{x}_{temp,0} \ldots \mathbf{x}_{temp,n}\}$
1 $n = 0$;
2 while $\|\mathbf{x}_n - \mathbf{x}^*\| \geqslant \epsilon$ do
3     $n = n + 1$;
4     $\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{f}_g(\mathbf{x}_{n-1}) \Delta t$;
5 end
```

The number of points N can be set to generate sequence of via point with equal distance. After sampling the a temporary sequence $\mathbf{x}_{temp}$ using basic Euler numerical integration with a small stepsize Δt, the total distance d_{temp} of the path is summed up. Then a distance d_s is determined by the total path length and number of points. The first via point is set as the initial point $\mathbf{x}_0$. Starting from the second point in the temporary sequence, check if the distance between current point and last via point along the path equals to the distance d_s. If so, mark the current point as a new via point, otherwise, continue to next point.

The point number N in sampling with point in equidistance need to be tuned carefully (Figure 3.16(a), 3.16(b)). According to the approach of constructing the VSDS, the robot control force directly depends on desired stiffness profile. So this method reflects the stiffness along the motion directly. When robot interacts with

Algorithm 2: Sampling with fixed points number

```
Result: $\mathbf{x}_0 \dots \mathbf{x}_N$
$n = 0$;
$\mathbf{x}_{temp,0} = \mathbf{x}_0$;
while $\|\mathbf{x}_{temp,0} - \mathbf{x}^*\| \geqslant \epsilon$ do
    $n = n + 1$;
    $\mathbf{x}_{temp,n} = \mathbf{x}_{temp,n-1} + \mathbf{f}_g\left(\mathbf{x}_{temp,n-1}\right)$;
end
$n = n + 1$;
$\mathbf{x}_{temp,n} = \mathbf{x}^*$;
$d_{temp} = 0$;
for $i = 1$ to $n$ do
    $d_{temp} = d_{temp} + \|\mathbf{x}_i - \mathbf{x}_{i-1}\|$
end
$d_s = \frac{d_{temp}}{N}$;
for $i = 1$ to $N - 1$ do
    $d_i = i * d_s$
end
$d_n = d_{temp}$;
$i = 1$;
$j = 1$;
$d_a = 0$;
while $i < N$ do
    if $d_a < d_{temp}$ then
        $d_a = d_a + \|\mathbf{x}_i - \mathbf{x}_{i-1}\|$;
        $i = i + 1$;
    else
        $\mathbf{x}_j = \mathbf{x}_{temp,i}$;
        $d_a = d_a + \|\mathbf{x}_i - \mathbf{x}_{i-1}\|$;
        $i = i + 1$;
        $j = j + 1$;
    end
end
$\mathbf{x}_N = \mathbf{x}^*$;
```

the environment along the motion direction, the stiffness determines the interaction force. However, with different stiffness profile, the VSDS constructed from same sequence will have different force profiles, which decides the robot velocity profile. Then the robot cannot follow the velocity profile of the original DS well.

To preserve the velocity profile as much as possible, another sampling method with greedy algorithm respect to an excepted force F_{exp} is used in generating the via points sequence. Greedy algorithm is heuristic method to make locally optimal decision at each step. It cannot guarantee global optimal solution, but it provides an acceptable solution to our problem. We know that the distance between via points and the stiffness determine the force $\mathbf{f}(\mathbf{x})$ together. And the property of the $\mathbf{A}$ matrix we construct with 3.19 is:

$$\frac{\parallel \mathbf{A}(\mathbf{x}_2 - \mathbf{x}_1) \parallel}{\parallel \mathbf{x}_2 - \mathbf{x}_1 \parallel} = k_1 \tag{3.18}$$

Detailed explanation can be found in A.3. Note that the first vector in $\mathbf{Q}$ should be parallel to vector $\mathbf{x}_2 - \mathbf{x}_1$. But we know from practice that there is not much difference between the direction of $\mathbf{x}_2 - \mathbf{x}_1$ and that of $\mathbf{f}(\mathbf{x}_1)$. With this property, we can define an expected scalar force F_{exp}. The idea is by adjusting the distance between two adjacent via points to make the resulted force $\mathbf{f}_i(\mathbf{x})$ as close as possible to F_{exp}.

In this sampling approach, a temporary sequence $\mathbf{x}_{temp}$ is generated using basic Euler numerical integration with relative small step size. Save the first point in temporary sequence as first point in the via points sequence. Then starting from the second point in temporary sequence, check if the current point minimize the difference between the F_{exp} and the spring force generated by the desired stiffness and the current point and the last point in via point sequence (Line 13). If so, add the current point to via points sequence, otherwise, move to next point in temporary sequence. A loop is carried out until the last point in temporary sequence are reached. This algorithm is described in Algorithm 3. By increasing F_{exp}, less via points with larger distance are generated. By decreasing the first value of stiffness profile $\mathbf{K}_{des,1,1}$, more via points with smaller distance are generated. The result of this algorithm can be seen in Figure 3.15.

The parameter F_{exp} needs to be chosen carefully, because it decides the velocity profile shape 3.16.

Constructing linear DS with desired stiffness profile:

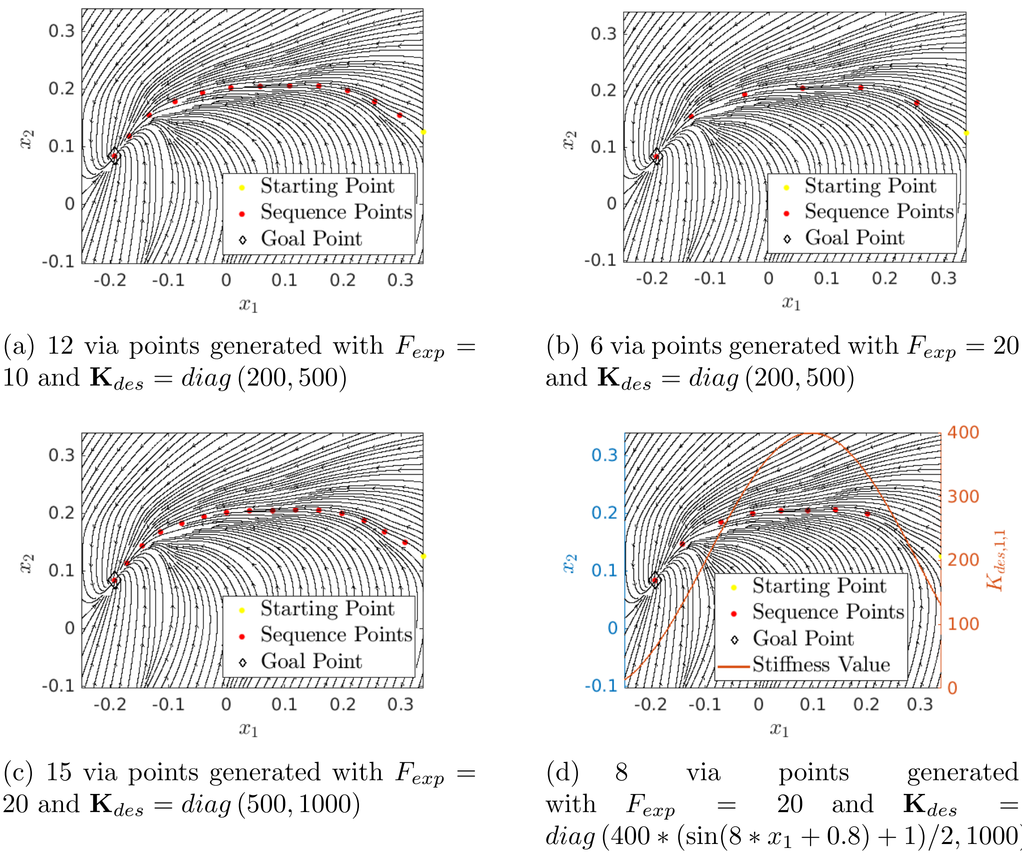
Although the stiffness in equation 3.14 is complex with the non-linear DS, the stiffness is simple for a linear DS $\mathbf{f}(\mathbf{x}) = \mathbf{A}(\mathbf{x} - \mathbf{x}^*)$, since:

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial \mathbf{A}(\mathbf{x} - \mathbf{x}^*)}{\partial \mathbf{x}} = \mathbf{A}$$

In this case, the stiffness is the matrix $\mathbf{A}$. So we are able to encode the stiffness

Algorithm 3: Sampling with greedy algorithm with respect to F_{exp}

```
Result: $\mathbf{x}_0 \ldots \mathbf{x}_N$
$n = 0$;
$\mathbf{x}_{temp,0} = \mathbf{x}_0$;
while $\|\mathbf{x}_{temp,0} - \mathbf{x}^*\| \geqslant \epsilon$ do
    $n = n + 1$;
    $\mathbf{x}_{temp,n} = \mathbf{x}_{temp,n-1} + \mathbf{f}_g(\mathbf{x}_{temp,n-1})$;
end
$n = n + 1$;
$\mathbf{x}_{temp,n} = \mathbf{x}^*$;
$i = 2$;
$j = 0$;
$t = 100000$;
while $i \leqslant n$ do
    if $\|\mathbf{K}_{des,1,1}(\mathbf{x}_j - \mathbf{x}_{temp,i}) - F_{exp}\| < t$ then
        $t = \|\mathbf{K}_{des,1,1}(\mathbf{x}_j - \mathbf{x}_{temp,i}) - F_{exp}\|$;
        $i = i + 1$;
    else
        $j = j + 1$;
        $\mathbf{x}_j = \mathbf{x}_{temp,i-1}$;
        $t = 1000000$;
    end
end
$j = j + 1$;
$\mathbf{x}_j = \mathbf{x}^*$;
```



(a) 12 via points generated with $F_{exp} = 10$ and $\mathbf{K}_{des} = diag\,(200, 500)$

(b) 6 via points generated with $F_{exp} = 20$ and $\mathbf{K}_{des} = diag\,(200, 500)$

(c) 15 via points generated with $F_{exp} = 20$ and $\mathbf{K}_{des} = diag\,(500, 1000)$

(d) 8 via points generated with $F_{exp} = 20$ and $\mathbf{K}_{des} = diag\,(400 * (\sin(8 * x_1 + 0.8) + 1)/2, 1000)$

Figure 3.15: Generated via points with different F_{exp} and $\mathbf{K}_{des}$. These results shows that with less via points are generated with larger F_{exp} and smaller $\mathbf{K}_{des}$. (d) shows the distance between via points decreases with higher stiffness.

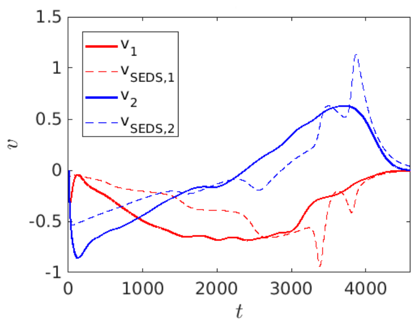
into $\mathbf{A}$. As introduced in Section 3.1.1, the constant matrix linear DS is defined as a multiplication of direction matrix and eigenvalue matrix, i.e.

$$\mathbf{A}_i = -\mathbf{Q}_i \mathbf{K}_{des,i} \mathbf{Q}_i^T \tag{3.19}$$

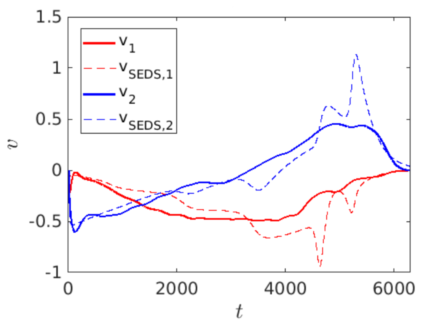
The eigenvalue matrix is desired stiffness profile which is a positive definite diagonal matrix:

$$\mathbf{K}_{des,i} = \begin{pmatrix} k_{i,1} & 0 \\ 0 & k_{i,2} \end{pmatrix}$$

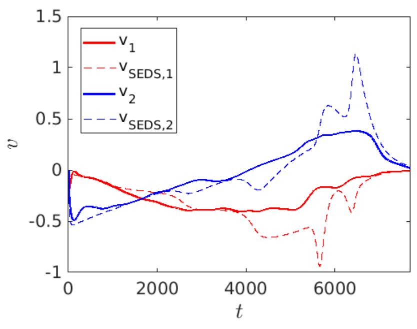
where $k_{i,2} > k_{i,1} > 0$. Here $\mathbf{K}_{des,i}$ is the desired stiffness profile. The stiffness profile can either be constant or variable. A variable stiffness depending on robot state $\mathbf{x}$ is denoted as $\mathbf{K}_{des}(\mathbf{x})$, whose eigenvalues satisfy $k_2 > k_1 > 0$. For simplicity, in this report, the desired stiffness profile is written as $\mathbf{K}_{des}$. From equation 3.3, the first component k_1 is the stiffness along the motion, and can be seen as a velocity scale factor along the path. k_2 indicates the stiffness orthogonal to the motion, which is



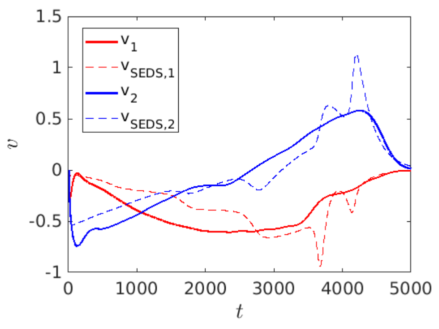
(a) 25 via points



(b) 35 via points



(c) Greedy algorithm respect to F



(d) 80 fixed points

Figure 3.16: Velocity profile simulated with 2 links robot in robot toolbox, whose control force is generated from different sampling methods. The dash line shows the velocity profile of original DS (SEDS is used here). Desired stiffness profile is $\mathbf{K}_{des} = diag\,(300, 500)$. Damping matrix is chosen as $\mathbf{D} = diag\,(25, 30)$.

the stiffness of the attraction behavior. i.e. how hard to drag the robot away from the path.

The direction matrix of the i_{th} linear DS is defined as:

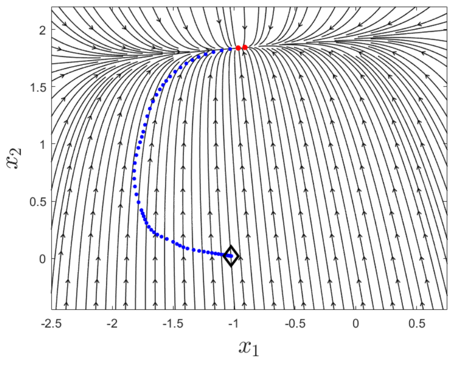
$$\mathbf{Q}_i = [\mathbf{e}_{i,1}, \mathbf{e}_{i,2}]$$

where $\mathbf{e}_{i,1}$ is parallel to the velocity at state $\mathbf{x}_i$, i.e. $\mathbf{e}_{i,1} = \frac{\mathbf{f}_g(\mathbf{x}_i)}{\|\mathbf{f}_g(\mathbf{x}_i)\|}$. And $\mathbf{e}_{i,2}$ is calculated such that $\mathbf{e}_{i,1} \perp \mathbf{e}_{i,2}$.

As mentioned in Section 3.1.1, a linear DS is constructed of a constant matrix $\mathbf{A}$ and an attractor $\mathbf{x}^*$. Except the initial point $\mathbf{x}_0$, each via points in $\mathbf{x}_{sam}$ is an attractor for corresponding linear DS, i.e. $\mathbf{x}_i^* = \mathbf{x}_i, \forall i = 1 \dots N$. Thus, the i_{th} linear DS is defined as:

$$\mathbf{f}_i\,(\mathbf{x}) = \mathbf{A}_i\,(\mathbf{x} - \mathbf{x}_i^*)$$

Now the linear DS is constructed and partially illustrated in Figure 3.17. Next step is to concatenate these DS. But different numerical integration methods will be introduced.


(a) The 1st linear DS $\mathbf{f}_1(\mathbf{x})$

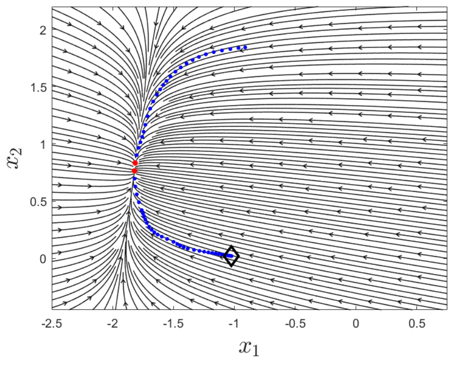

(b) The 26th linear DS $\mathbf{f}_{26}(\mathbf{x})$

Figure 3.17: Illustration of single linear DS. The bigger red points indicate the begin and end of the illustrated linear DS. Blue smaller points are interpolated via points. And diamond is the global attractor.

Scale function around initial point:

As the initial force at $\mathbf{x}_0$ can be large, it can introduce a large robot acceleration and a very jerky motion at the beginning. To avoid this, a simple scale function $\alpha(\mathbf{x})$ is defined, whose value depends on the distance relative to $\mathbf{x}_0$:

$$\alpha(\mathbf{x}) = \begin{cases} 1 & \|\mathbf{x}-\mathbf{x}_0\| > d \\ \sin\left(\arcsin(1-b)\frac{\|\mathbf{x}-\mathbf{x}_0\|}{d}\right) + b & \|\mathbf{x}-\mathbf{x}_0\| < d \end{cases} \tag{3.20}$$

where b is a value to scale down the initial force, but the value of b cannot be too small, otherwise the robot keeps still at the initial position because the initial force is not large enough to overcome the friction. As for d, it is the effective area of this scale function. Usually it is set as a small portion of the total path length, in this report. d is usually between 0.01 and 0.1 times of total path length and b is set to 0.6. Figure 3.18 shows one dimensional examples of the scale function.

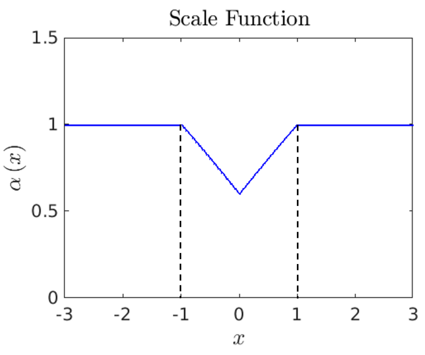
Constructing weighting functions

To define the weighting function, a distance dependent Gaussian kernel is first calculated as:

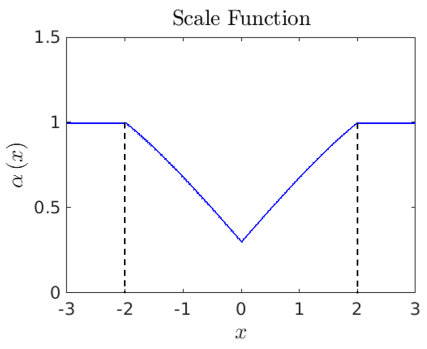
$$\omega_i(\mathbf{x}) = e^{-\frac{\left(\mathbf{x}-\mathbf{x}_{cen,i}\right)^T\left(\mathbf{x}-\mathbf{x}_{cen,i}\right)}{2(\sigma^i)^2}} \tag{3.21}$$

Where the $\mathbf{x}_{cen,i}$ denotes the center of i_{th} linear DS:

$$\mathbf{x}_{cen,i} = \frac{1}{2}\left(\mathbf{x}_i + \mathbf{x}_{i-1}\right)$$



(a) $x_0 = 0$, $b = 0.6$, $d = 1$



(b) $x_0 = 0$, $b = 0.3$, $d = 2$

Figure 3.18: One dimensional illustration of scale function around the beginning point $x_0 = 0$ with different effective area t.

And $\sigma^i \in \mathbb{R}^+$ is the smoothing parameter that controls the region of influence of each linear DS. σ^i is calculated according the distance l_i:

$$\sigma^i = \delta l_i$$

$$l_i = \|\mathbf{x}_i - \mathbf{x}_{i-1}\|$$

where δ is a fraction number to scale the smoothing parameter. Then the weighting function is a normalization of these kernels:

$$\tilde{\omega}_i(\mathbf{x}) = \frac{\omega_i(\mathbf{x})}{\sum_{j=1}^{N} \omega_j(\mathbf{x})} \tag{3.22}$$

The value of δ need to be chosen carefully. If each linear DS has a small area of influence with small δ, then there can be no transition between linear DS (Figure 3.19(a)), or the force profile is not smooth, which means vibration and jerky motion of robot(Figure 3.19(d)). With large δ, the influence area is large, so that linear DS with opposite direction impact on the desired velocity, this drags the attractor a little backwards in the motion direction (Figure 3.19(q)). Detailed analysis is in Section 3.2.4. Both situations results in local attractors that destroy the global asymptotic stability at the desired global attractor $\mathbf{x}^*$ (Figure 3.19(c), 3.19(s)).

Constructing DS as weighted sum of each local linear DS

With all the preparation introduced above, now the VSDS can be simply constructed as a weighted sum of local linear DS:

$$\mathbf{f}(\mathbf{x}) = \alpha(\mathbf{x}) \sum_{i=1}^{N} \tilde{\omega}_i(\mathbf{x}) \mathbf{f}_i(\mathbf{x}) \tag{3.23}$$

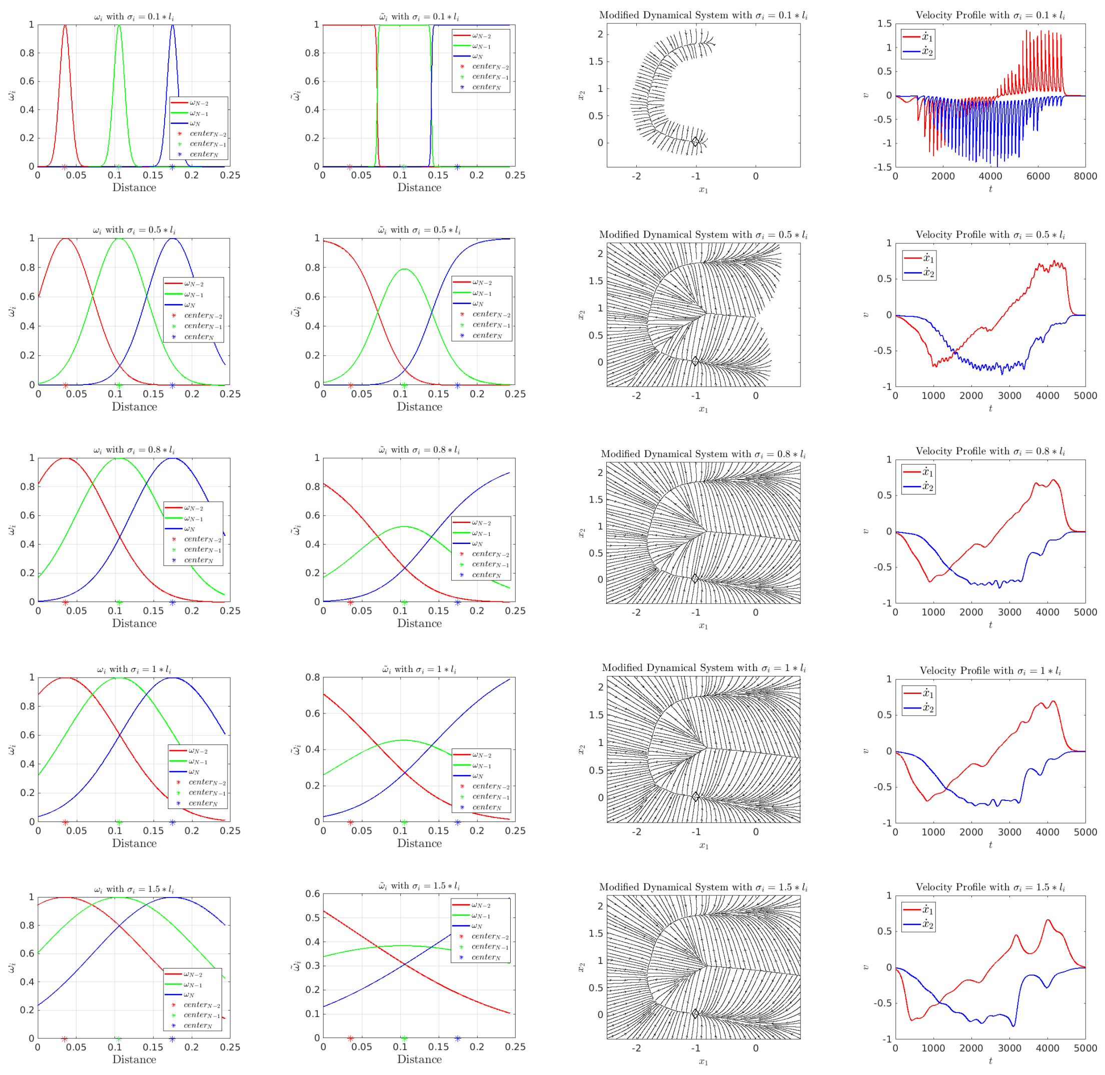
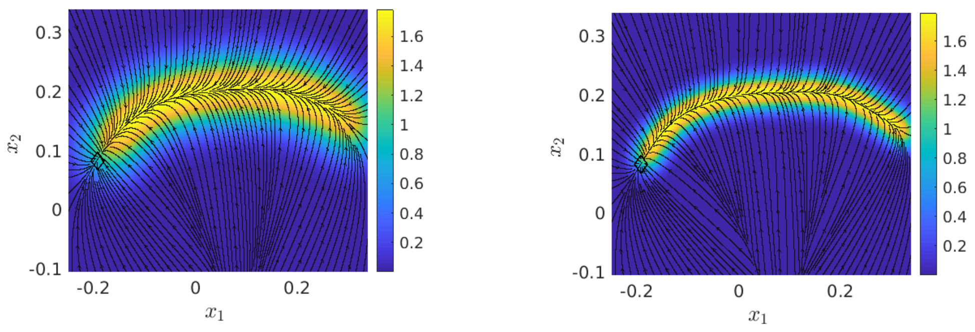


Figure 3.19: Influence of fraction value δ. The left column shows the weighting function of the last three via points in $\mathbf{x}_{sam}$. The right column displays the resulted VSDS with different δ. The blank area in the figure is because the weights decays to a extremely small value.

Regenerating the VSDS:

With the VSDS in 3.23, the robot has a behavior of returning to reference path. However, the intrinsic property of the VSDS is that, the further the robot away from reference path is, the large control torque the robot need. For the safety of human, the robot is endowed another motion level compliance behavior, i.e. when human keep dragging the robot out of an attraction tube area, the robot gives up the old



(a) $N = 10$ (b) $N = 20$

Figure 3.20: Visualization of *poscheck* $= \sum_{i=1}^{N} \omega_i(\mathbf{x})$, a tube like shape can be seen clearly. And with different number of via points, the size of tube changes.

path and generate a new path and corresponding VSDS online.

Therefore, we still need to regenerate the VSDS when the robot is deviated from the attraction tube area. During the execution of robot, a real-time position check is applied. If current position is out of the effect area, which is defined as the sum of Gaussian kernel ω_i, the approach restarts from sampling via points in original DS $\mathbf{f}_g(\mathbf{x})$, with $\mathbf{x}_0 = \mathbf{x}_{current}$.

This effect tube is defined using the distance based weighting function 3.21. The value *poscheck* $= \sum_{i=1}^{N} \omega_i(\mathbf{x})$ is compared with a threshold value. With larger threshold value, we have a smaller the effect tube around the reference path. However, the sum value *poscheck* will change as the number of via points differs or as the distance between via points differs (Figure 3.20), then the threshold value need to decided according to the situation.

Summary

In the end, the above mentioned approach can be summarized in Algorithm 4. This approach is also applied on 6 motions from a library of human handwriting motions, see Figure 3.21. The motions are firstly learned by SEDS, which are the desired original DS. And the VSDS and the original SEDS are compared. The sampling method is fixed number of points with $N = 50$.

3.2.3 Stiffness behavior of closed loop DS

As shown in equation 3.14, the stiffness is proportional to the Jacobian of DS. The Jacobian of the VSDS is:

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \sum_{i=1}^{N} \left(-\frac{\tilde{\omega}_i(\mathbf{x})}{(\delta_i)^2} (\mathbf{x} - \mathbf{x}_{cen,i}) (\mathbf{x} - \mathbf{x}_i^*)^T \mathbf{A}_i^T + \frac{\tilde{\omega}_i(\mathbf{x})}{(\delta_i)^2} (\mathbf{x} - \mathbf{x}_{cen,i}) \mathbf{f}(\mathbf{x})^T + \tilde{\omega}_i(\mathbf{x}) \mathbf{A}_i \right) \tag{3.24}$$

Algorithm 4: Algorithm generation of VSDS

```
Result: $\mathbf{f}(\mathbf{x})$
1  while $\|\mathbf{x} - \mathbf{x}^* > \epsilon\|$ do
2      if $\sum_{i=1}^{N} (\omega_i) < thres$ or NotExist( $\mathbf{f}(\mathbf{x})$) then
3          $\mathbf{x}_0 = \mathbf{x}_{current}$ Interpolate $\mathbf{f}_g(\mathbf{x})$ for via points sequence
            $\mathbf{x}_{sam} = \{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_N\}$;
4          $\mathbf{e}_{i,1} = \frac{\mathbf{f}_g(\mathbf{x}_i)}{\|\mathbf{f}_g(\mathbf{x}_i)\|}$;
5          $\mathbf{e}_{i,2}$ = FindPerpendicularBasis($\mathbf{e}_{i,1}$);
6          $\mathbf{Q}_i = [\mathbf{e}_{i,1}, \mathbf{e}_{i,2}]$;
7          $\mathbf{A}_i = -\mathbf{Q}_i \mathbf{K}_{des} \mathbf{Q}_i^T$;
8          $\mathbf{x}_i^* = \mathbf{x}_i$;
9          $\mathbf{f}_i(\mathbf{x}) = \mathbf{A}_i(\mathbf{x} - \mathbf{x}_i^*)$;
10         $\mathbf{x}_{cen,i} = \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_{i-1})$;
11         $l_i = \|\mathbf{x}_i - \mathbf{x}_{i-1}\|$;
12         $\sigma_i = \delta l_i$;
13         $\omega_i(\mathbf{x}) = e^{-\frac{(\mathbf{x}-\mathbf{x}_{cen,i})^T \mathbf{\Sigma}_i^{-1} (\mathbf{x}-\mathbf{x}_{cen,i})}{2}}$;
14         $\tilde{\omega}_i(\mathbf{x}) = \frac{\hat{\omega}_i}{\sum_{j=1}^{N} \omega_j(\mathbf{x})}$;
15         $\mathbf{f}(\mathbf{x}) = \alpha(\mathbf{x}) \sum_{i=1}^{N} \tilde{\omega}_i(\mathbf{x}) \mathbf{f}_i(\mathbf{x})$;
16     end
17 end
```

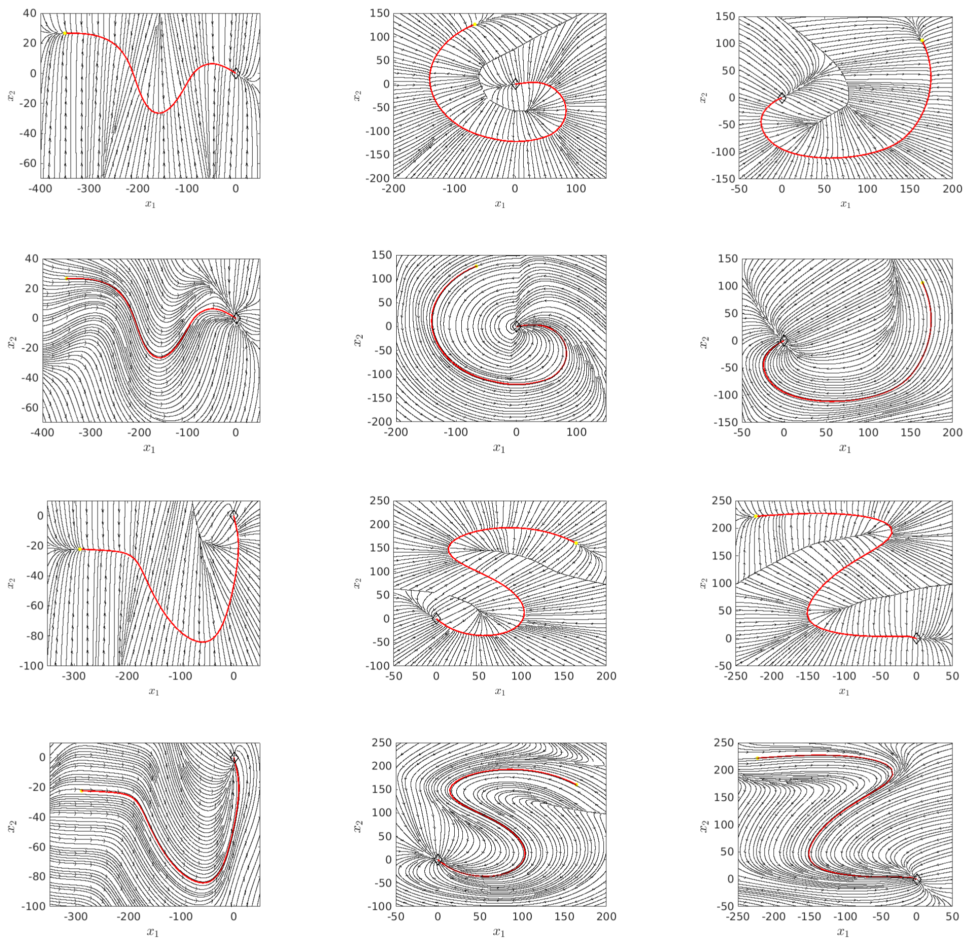
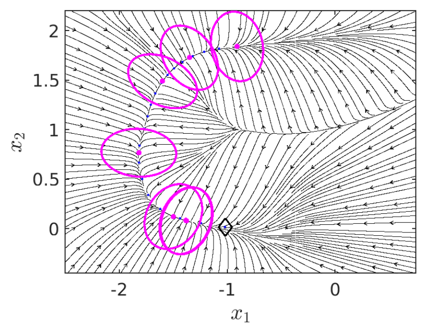


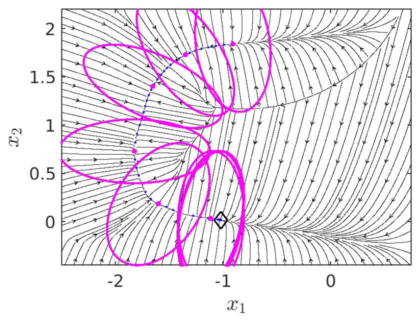
Figure 3.21: Handwriting library simulation. The initial point is marked as yellow point. Red line denotes the reference path and black diamond is the goal point. Row 1 and 3 are VSDS, row 2 and 4 are SEDS.

Detailed derivation is shown in Appendix A. The scale function around initial point is not considered here, since it has effect only on a small area at the beginning of the path. Then the stiffness can be decomposed using equation 3.15 into two directions: along the motion K_1 and perpendicular to the motion K_2. It can be visualized in Matlab simulation, as shown in Figure 3.22.

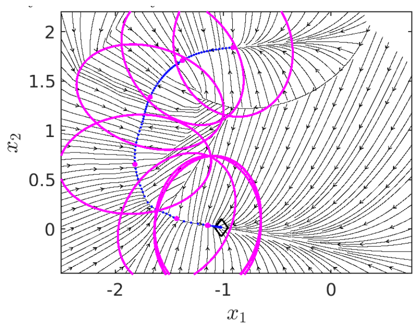
The ellipses are not identical for constant desired stiffness profile, the possible reason is that the gradient of $\mathbf{A}_i$ matrix. As shown in the definition of $\mathbf{A}_i$, it is depends on the robot state $\mathbf{x}$. However, calculating the gradient is not trivial, so the $\mathbf{A}_i$ is seen as constant matrix here. This will introduce the errors in the stiffness analysis.



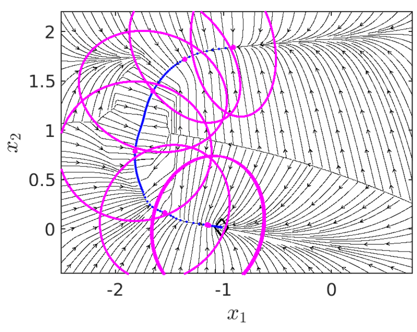
(a) $\mathbf{K}_{des} = diag\,(5, 25)$



(b) $\mathbf{K}_{des} = diag\,(10, 100)$



(c) $\mathbf{K}_{des} = diag\,(25, 100)$



(d) $\mathbf{K}_{des} = diag\left(50\frac{\sin(2x_2)+1}{2}, 100\right)$

Figure 3.22: Stiffness ellipses in simulation

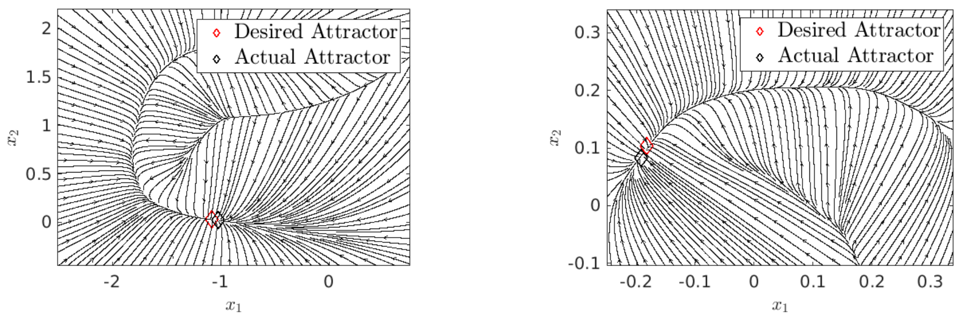


Figure 3.23: DS attractor locates mostly before the desired global attractor of original DS.

3.2.4 Discussion of stability

The VSDS $\mathbf{f}\,(\mathbf{x})$ in equation 3.23 is interpreted as concatenated stable linear DS by a transfer function, which is a distance dependent weighting function. From the illustration in Figure 3.11, one can imagine that one linear DS $\mathbf{f}_i\,(\mathbf{x})$ drives robot to the local attractor $\mathbf{x}_i$ and hand it over to next linear DS $\mathbf{f}_{i+1}\,(\mathbf{x})$ through the weighting function $\tilde{\omega}_{i+1}$. Therefore, it is intuitive to see that the DS is stable and will converge to an attractor.

However, the attractor of VSDS can be different from the global attractor (Figure 3.23). This is caused by the instinct of weighting function. To explain this, we can see the value of VSDS at global attractor $\mathbf{x}^*$. Note that the last local attractor is the global attractor, i.e. $\mathbf{x}_N = \mathbf{x}^*$.

$$\begin{aligned} \mathbf{f}\left(\mathbf{x}^*\right) &= \alpha\left(\mathbf{x}^*\right)\sum_{i=1}^{N}\tilde{\omega}_i\left(\mathbf{x}^*\right)\mathbf{f}_i\left(\mathbf{x}^*\right) \\ &= \sum_{i=1}^{N-1}\tilde{\omega}_i\left(\mathbf{x}^*\right)\mathbf{f}_i\left(\mathbf{x}^*\right) + \mathbf{0} \end{aligned} \tag{3.25}$$

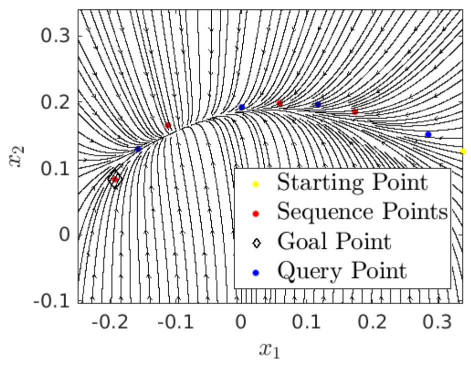
We know that $\mathbf{f}_i\left(\mathbf{x}^*\right) \neq \mathbf{0}$ for all $i = 1 \cdots N-1$, because the local attractors of these linear DS are not $\mathbf{x}^*$. To make the attractor of DS identical with the global attractor, $\mathbf{f}\left(\mathbf{x}^*\right) = \mathbf{0}$ is required. However, we can see that the weights $\tilde{\omega}_i\left(\mathbf{x}^*\right)$ is not always zero. As the weighting function is based on Gaussian kernel, so only when the distance is large enough, i.e. out of the influence area, the value of weighting function can become zero. Remember that the influence area is decided by the proportional parameter δ. Larger δ means larger influence area. Usually, in our case, the distance between $\mathbf{x}^*$ and the last several center points is not large enough to deactivate the linear DS, because we cannot set δ too small. So the last several linear DS contribute velocities against the desired motion direction. This drags the attractor a little backward. Also, the normalization of weighting function decreases the difference between the weights.

To see the effect of weighting function on the attractor more clearly, we will look at an small example with only 4 via points (Figure 3.24), here we set $\delta = 1$, with which each local DS will have a large influence on other DS. And we see the weighting function values of 4 query points on the reference path (Table 3.1), the result of each linear DS $\mathbf{f}_i\left(\mathbf{x}\right)$ and the total velocity $\mathbf{f}\left(\mathbf{x}\right)$ (Table 3.2). Figure 3.25 illustrates the velocity component in each desired velocity.

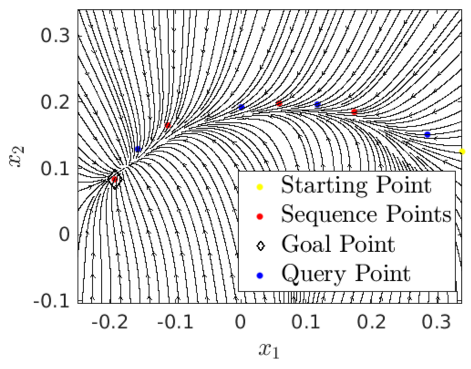
- Point $\mathbf{x}_{que,1}$ has the maximum at the first linear DS. And the weights of rest linear DS are not small enough to ignore, except the last one. So the desired velocity at this point is the sum of the first 3 linear DS. This results in a large force at $\mathbf{x}_{que,1}$ (Figure 3.25(a)).

- See the ω_2 of point $\mathbf{x}_{que,2}$, it has max value at the second linear DS, however, the value for the 1st and 3rd linear DS is also not small. And after normalization, $\tilde{\omega}_2$ shows the 2nd weight doesn't stand out from the rest. This decrease the desired velocity, because the 1st and the 3rd linear DS compensate each other to some extend (Figure 3.25(b)).

- $\mathbf{x}_{que,3}$ has the similar situation. The sum of velocity still points to the goal point. But the magnitude of the desired velocity is quite small compared to that of $\mathbf{x}_{que,1}$ (Figure 3.25(c)).

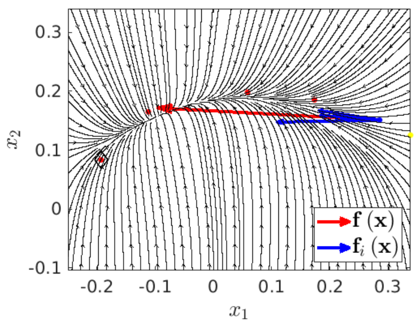


(a) $\delta = 1$

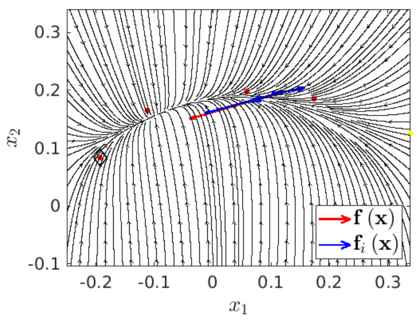


(b) $\delta = 0.5$

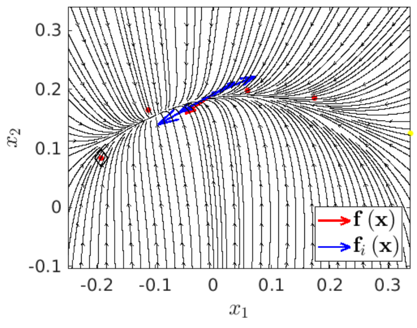
Figure 3.24: Illustration of the 4 via points and the DS generated from them with different δ value. The blue points are 4 query points, from right to left are $\mathbf{x}_{que,1} \cdots \mathbf{x}_{que,4}$. Because of the characteristic of the weighting function, such few number of via points will not be used in this thesis. As can be seen in this figure, the generated DS is not exactly following the reference path.



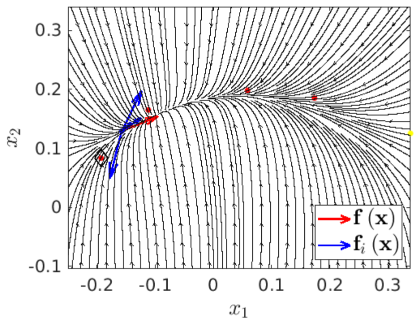
(a) $\mathbf{x}_{que,1}$



(b) $\mathbf{x}_{que,2}$



(c) $\mathbf{x}_{que,3}$



(d) $\mathbf{x}_{que,4}$

Figure 3.25: Illustration of control force of query points in the simple example.

- However, $\mathbf{x}_{que,4}$ has the wrong direction of desired velocity. We can see from Table 3.1, although it has the maximum weight at the 4th linear DS, however, the third weights are also not trivial, the sum of desired velocity will be pointing backwards (Figure 3.25(d)).

In this simple example, the attractor of DS is far from goal point. However, by increasing the number of via points, or decrease the δ, i.e. decreasing the influence between adjacent linear DS, this problem is relieved but not solved (Figure 3.23). The analysis above explains why our DS cannot converge to the desired global attactor.

i		weights			
$\mathbf{x}_{que,1}$	ω_1	0.9858	0.3232	0.1955	0.0007
	$\tilde{\omega}_1$	0.6549	0.2147	0.1299	0.0005
$\mathbf{x}_{que,2}$	ω_2	0.7111	0.9991	0.7065	0.0526
	$\tilde{\omega}_2$	0.2880	0.4046	0.2861	0.0213
$\mathbf{x}_{que,3}$	ω_3	0.3393	0.6099	0.9856	0.3455
	$\tilde{\omega}_3$	0.1488	0.2675	0.4322	0.1515
$\mathbf{x}_{que,4}$	ω_4	0.0614	0.0533	0.7193	0.9981
	$\tilde{\omega}_4$	0.0335	0.0291	0.3926	0.5448

Table 3.1: Weighting function value of the query points, $\delta = 1$.

i	$\mathbf{f}_i(\mathbf{x})$				$\mathbf{f}(\mathbf{x})$
$\mathbf{x}_{que,1}$	-17.5595	-10.1155	-10.3822	-0.0571	-38.1383
	-0.3572	0.7961	1.6023	0.0157	2.0535
$\mathbf{x}_{que,2}$	4.1177	-5.2831	-13.0370	-1.5154	-15.7333
	0.7837	-1.6328	-3.6054	-0.1544	-4.6146
$\mathbf{x}_{que,3}$	7.0616	3.5929	-9.6231	-5.9709	-4.9493
	2.9635	2.0112	-5.2257	-3.1402	-3.3966
$\mathbf{x}_{que,4}$	3.3573	1.5758	3.4045	-2.0812	6.2546
	2.2597	1.4700	6.7965	-8.0304	2.4936

Table 3.2: Velocity of each linear DS and the sum of the velocities, i.e. the desired velocity, $\delta = 1$.

One solution is to reduce the δ value. As we see in Figure 3.19(g) and 3.19(c), reduced δ results in less influence area of each linear DS. However, a too small δ causes the oscillation of $\mathbf{f}(\mathbf{x})$, and causes the oscillation of the robot velocity furthermore (Figure 3.19(d)). From practice, we choose $\delta = 0.5$ in robot execution.

Compare 3.24(a) and 3.24(b), we can see that by decreasing the δ, the attractor of new DS is closer to desired attractor. For each query point, the maximum weight stand more out from the rest. Then the force from other local DS has less impact on the query point.

Chapter 4

Evaluation

The algorithm is applied on 7 joints KUKA LWR. From Section 4.1 to 4.4, a SEDS in $y - z$ plane (B.2) is learned from a human demonstrated data. In Section 4.5, another SEDS (B.3) is learned from human demonstration. Then our algorithm takes them as original DS to generate our VSDS.

4.1 Motion execution

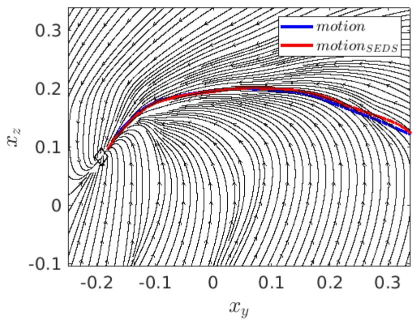
We first tested how the robot moves under our algorithm, and how it follows the desired original DS. The algorithm is compared with SEDS in open loop control configuration. A PD controller is chosen for the controller in open loop configuration, we take $\mathbf{K} = diag\,(300, 500)$ and $\mathbf{D} = diag\,(30, 30)$ as stiffness and damping, respectively. In contrast, our algorithm is applied with closed loop configuration (Figure 3.13(b)), and the controller we use is 3.16. For our algorithm, we take the same stiffness and damping. The δ in this experiment is 0.5. And the via points are generated using greedy algorithm with respect to F_{exp}. $F_{exp} = 20$ is used.

From Figure 4.1 we can see that the motion of the robot follows the original DS and the velocity profile is similar to that of SEDS.

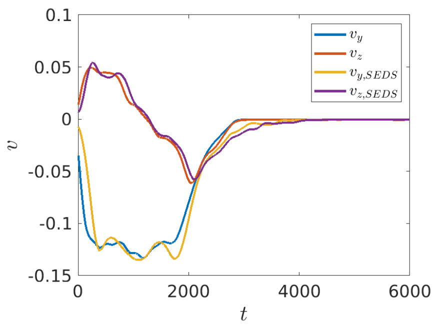
Now we put SEDS into closed loop control configuration, and take the controller in 3.12, with dampling values $\lambda_1 = 150$ and $\lambda_2 = 230$. Then the performance of SEDS and our VSDS in closed loop control configuration is compared. Figure 4.2 shows that when a perturbation happens in SEDS, the robot cannot follow the reference path anymore. In our algorithm, however, the robot shows a spring like behavior and goes back to reference path after perturbation.

4.2 Human robot collision

A drawback of the open loop control configuration (Figure 2.1) is a non-safe behavior at human robot collision case. In open loop configuration, the reference motion is integrated separately. So the motion planner does not get the robot actual

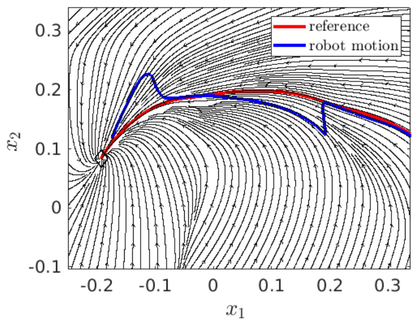


(a) Motion in Cartesian space

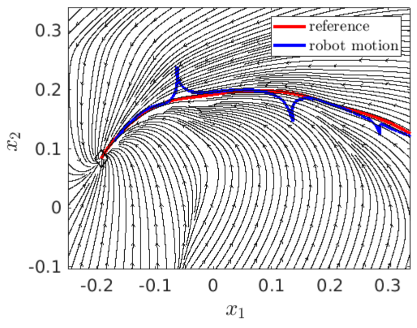


(b) Velocity profile

Figure 4.1: Motion execution comparision between SEDS and VSDS. Both use the impedance value $\mathbf{K} = diag\,(300, 500)$ and $\mathbf{D} = (30, 30)$. The background DS in (a) is the learned SEDS.



(a) Motion of SEDS



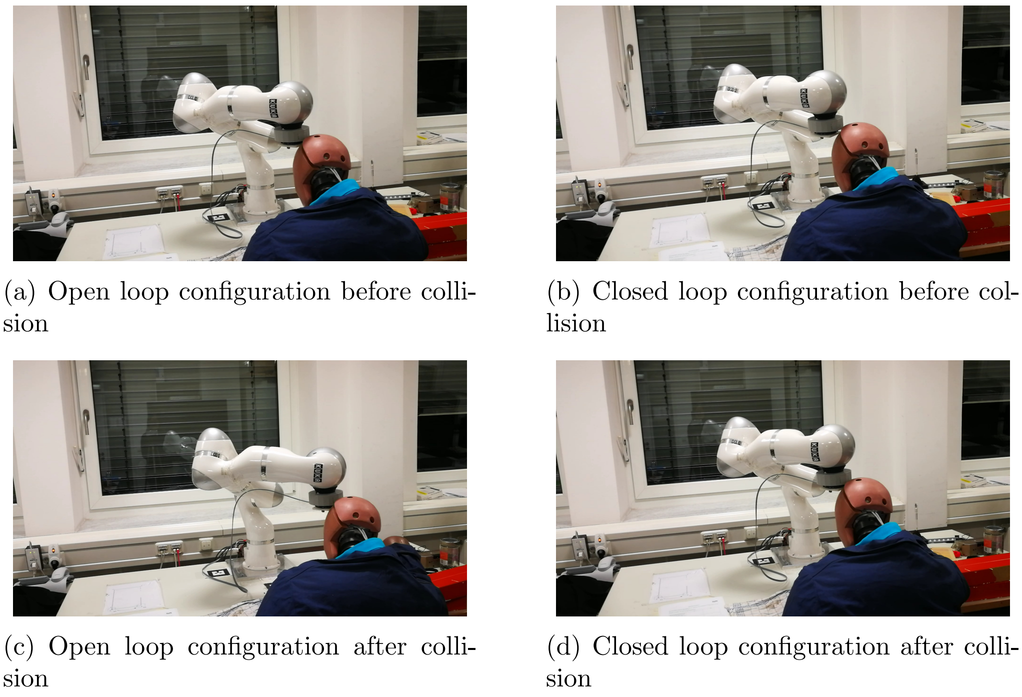
(b) Motion of VSDS

Figure 4.2: Comparision of motion with perturbation between SEDS and VSDS. The background DS is the learned SEDS. The red lines indicate the reference path starting from initial point $\mathbf{x}_0$. The blue lines are the actual robot motion.

position. Then when the robot actual state does not match the reference motion, the accumulated error can cause two main effects in human robot collision: 1) When the robot is blocked by human, the difference between actual robot state and reference state increases. This results in an increasing force that robot applies to the human. 2) When the robot is released again, a large acceleration is generated. This results in a sudden movement which may cause danger to human or robot.

Meanwhile, in the closed loop configuration (Figure 3.13(b)), the robot motion is generated online according to robot current state. When robot is blocked by human, the reference motion keeps the same. The interaction force between robot and human stays in low level, which is safe for human. When the robot is released again, the robot continues the motion smoothly, without any sudden acceleration.

The setup of an experiment for human robot collision is a 7DOF KUKA LWR and a "human" who is sitting inside the robot workspace (Figure 4.3). Robots moves


(a) Open loop configuration before collision

(b) Closed loop configuration before collision

(c) Open loop configuration after collision

(d) Closed loop configuration after collision

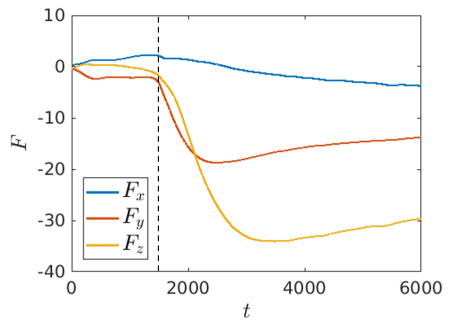
Figure 4.3: Human robot collision experiment.

and hit the head of human. Comparing Figure 4.3(a) and 4.3(c), we see that when collision happens, the robot using open loop configuration keeps pushing human's head. In contrast to this, Figure 4.3(b) and 4.3(d) show that the robot stops when it hits human head. Figure 4.4 shows the force profile of the experiment. Using open loop configuration, the force increases to $38N$ after collision (Figure 4.4(b)). Meanwhile, Figure 4.4(d) shows the force in closed loop configuration keeps a low level ($5.4N$).

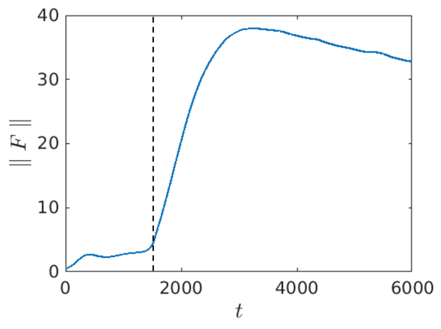
4.3 Path regeneration

The motion level compliance behavior path regeneration is also realized during the robot movement. Here we consider a scenario that human and robot work in the same workspace, and human blocks the robot motion and push the robot away to find another path to arrive the goal point.

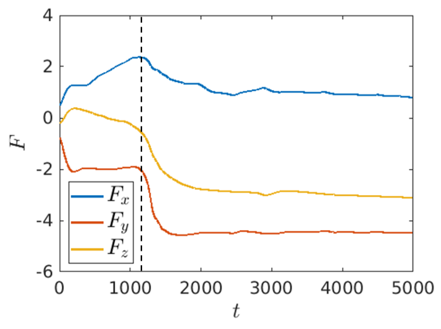
Figure 4.5(a) shows that, if the perturbation is small, robot will go back to the reference path. But after human pushes it far from the path, robot generates a new path and reaches the final goal point. If a small perturbation happens again after regeneration, the robot still goes back to the new reference path (Figure 4.5(b)).



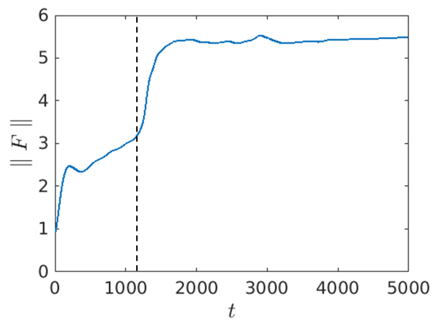
(a) Force profile of SEDS in open loop control configuration



(b) Force norm profile of SEDS in open loop control configuration

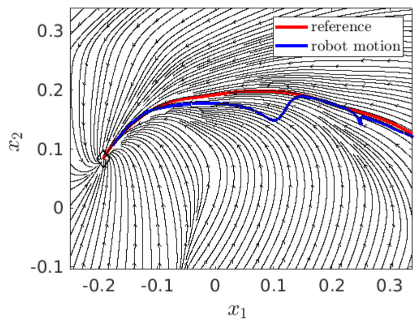


(c) Force profile of VSDS in closed loop control configuration

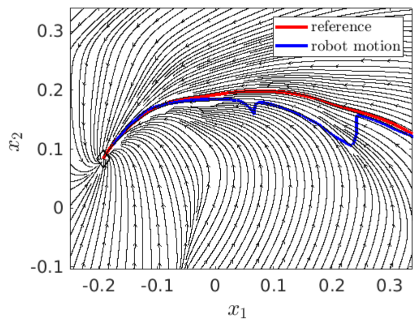


(d) Force norm profile of VSDS in closed loop control configuration

Figure 4.4: Force profile at crash situation. The dash line marks roughly when the collision happens.



(a) First small perturbation then large perturbation

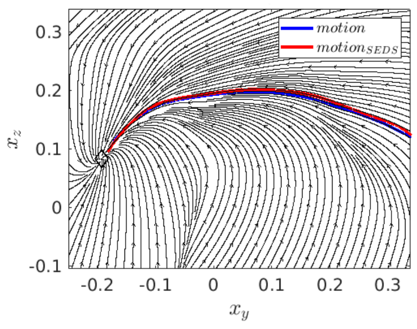


(b) First large perturbation then small perturbation

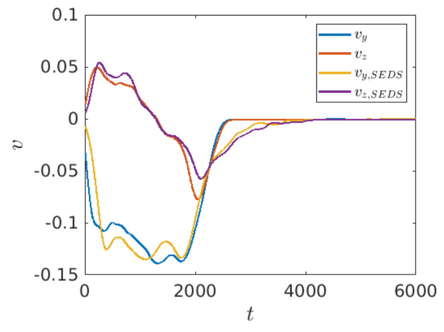
Figure 4.5: Path regeneration after large perturbation. The DS in background is the original DS. The red lines indicate the reference path starting from initial point $\mathbf{x}_0$. The blue lines are the actual robot motion.

4.4 Variable stiffness

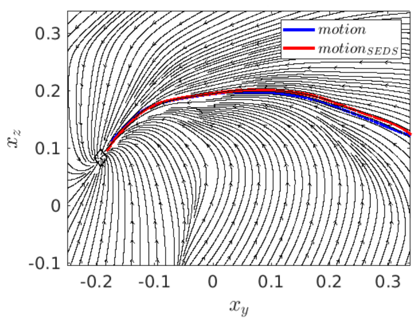
Variable stiffness profile can also be endowed to robot. Here we take the stiffness profile variable depending on robot state, i.e. $\mathbf{K}_{des} = diag\,(400(\sin(8x_y + 0.8) + 2)/2, 1000)$ and $\mathbf{K}_{des} = diag\,(300, 400(\sin(8x_y + 0.8) + 1.5)/2)$. The greedy algorithm with respect to F_{exp} is used to generate the via points, and F_{exp} is 20. The robot motion and velocity profile are shown in Figure 4.6, and are compared with motion of SEDS in open loop control configuration. The robot motion and velocity basically follows the desired original motion.



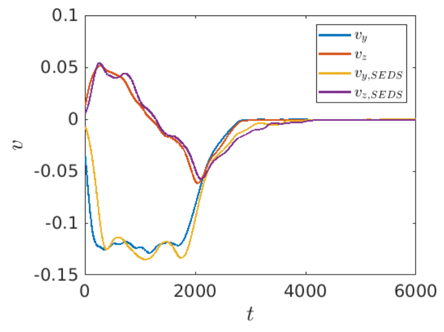
(a) Motion in Cartesian space



(b) Velocity profile



(c) Motion in Cartesian space



(d) Velocity profile

Figure 4.6: Motion execution with variable stiffness. The via points are generated using greedy algorithm with respect to $F_{exp} = 20$. (a) and (b) use stiffness profile $\mathbf{K}_{des} = diag\,(400(\sin(8x_y + 0.8) + 2)/2, 1000)$. (c) and (d) are with stiffness of $\mathbf{K}_{des} = diag\,(300, 400(\sin(8x_y + 0.8) + 1.5)/2)$.

4.5 Inserting charger into socket

The VSDS is also tested in an assignment of plugging a charger into a socket. A charger is fixed to robot as end-effector, the socket is fixed on the table (Figure 4.7). A demonstration from an initial position to the socket is demonstrated. Then

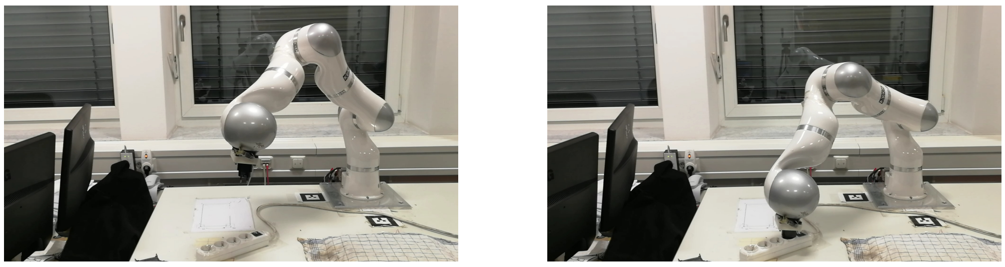

(a) Robot at initial position (b) Robot inserts charger into socket

Figure 4.7: Experiment of the charger insertion.

a SEDS in $y - z$ plane is learned from it as desired original DS. Sampling method of fixed number of point is used with $N = 9$.

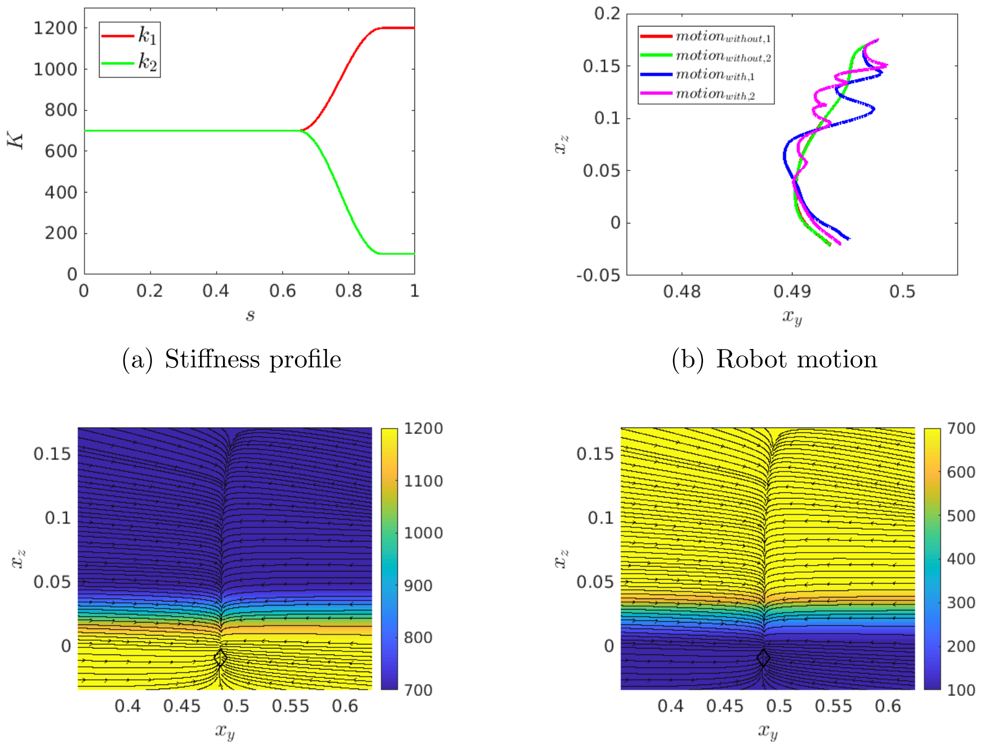
Variable stiffness is necessary in this task. When the robot is in free motion, i.e. approaching the socket, the stiffness along the motion is relatively low, the stiffness perpendicular to the motion is relatively high, this makes the robot adhere to the path and resist perturbations. During insertion, the stiffness along the motion is increased, for the robot need to apply a larger force to insert the charger, meanwhile the stiffness in the other direction decreases, because the holes constrain the charger, the robot need to be compliance to this constraint. The stiffness profile we use in this insertion task is shown in Figure 4.8(a). s denotes the relative robot position on the path.

The generated VSDS from the motion is clear (Figure 4.8(c) and 4.8(d)). And the stiffness along the motion is shown in the colormap. In the experiments we see that the robot can execute this task successfully. Even when perturbations happens during approaching the socket, the robot can return to the reference path and finish the task (Figure 4.8(b)). This would not happen if SEDS in closed loop configuration is used. Since sticking to a path is critical in this case, otherwise the robot will hit the side of socket, and be blocked there.

4.6 Discussion

From the robot experiments, we see several advantages of our algorithm. First is that our approach can finish the tasks which require a specific path even with perturbations. For instance in the charger insertion task in Section 4.5, it can be succeeded only when the robot follows the path. Otherwise, the robot will miss the holes on socket, or even hit the socket from side. Robot using our approach sticks to the specific path even under small perturbations, while other DS like SEDS do not have this property, see Section 4.1.

Another advantage of our algorithm is safety. In VSDS, when the task fails, for instance the robot stops on the socket instead of going into the hole in this insertion case, the contact force keeps at low level, which will not destroy other objects in



(a) Stiffness profile

(b) Robot motion

(c) VSDS with stiffness k_1

(d) VSDS with stiffness k_2

Figure 4.8: Experiment parameters and result of the charger insertion. The stiffness profile changes in the near of socket, $s = 0.65$. The robot motions are two without perturbation and two with perturbation.

the environment. In contrast, robot using open loop configuration will apply an increasing force to the environment as time goes, which is not safe. We can also see this in Section 4.2.

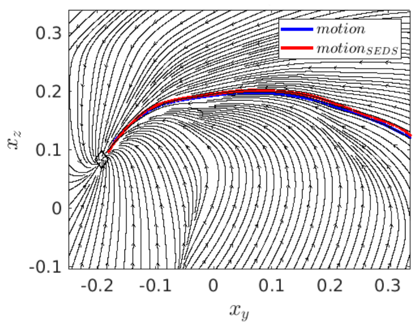
In addition, the VSDS provides a clear way to shape the stiffness. Compare VSDS with the LAGS-DS in [FFB18], both methods shows a behavior that returns to the reference path under perturbation. But the LAGS-DS does not have ability to shape the stiffness. Meanwhile, the desired stiffness can be encoded to VSDS easily.

From the principle of the VSDS, we know that as long as the stiffness profile is positive definite, i.e. $\mathbf{K}_{des} + \mathbf{K}_{des}^T \succ 0$, the VSDS will converge to a point, since it is concatenation of several stable linear DS. And the VSDS will follow the sampled via points one by one. By rotating the stiffness profile in 3.19, taking a two dimensional stiffness profile as example, k_1 denotes the stiffness along the motion, and the k_2 is the stiffness perpendicular to the motion. This helps us to define the stiffness profile more easily. If a behavior such that the robot come back to the PATH fast is desired, then the k_2 should set larger than k_1. If the robot need to stick to a motion, then we can increase k_2 to make the robot resist the perturbation perpendicular to motion direction.

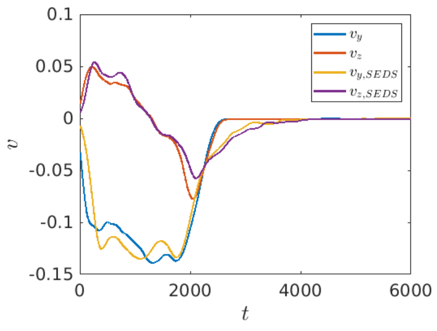
Using the fixed number points method makes the robot follows the stiffness profile well along the motion, see Section 4.5. But there is a compromise on the velocity profile (Figure 4.9(d)). So another method to sample the via point is also proposed in this report.

Using greedy algorithm with respect to F_{exp}, the robot motion is similar as the SEDS in open loop control configuration, and the robot shows the spring like stiffness behavior (Figure 4.9(a) and 4.9(b)). However, note that robot stiffness is not only the spring like behavior in the direction that perpendicular to the motion, but also the resistance force it exerts to the perturbation along the motion. With higher stiffness, the robot should apply higher force to the environment in contact during the movement. However, if we use the greedy algorithm with respect to F_{exp}, the robot is expected to be controlled by a constant force F_{exp}, this means that the force at contact case still keeps the same, even with higher stiffness. This can be solved by generating fixed number of via points, which results in a higher force at higher stiffness, but then the velocity profile is not that similar compared to the other (Figure 4.9(d)). So there is a trade-off between velocity and the stiffness behavior.

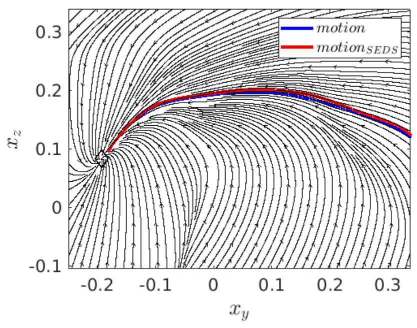
Another drawback of this algorithm is that, the variable stiffness is not encoded continuously, because only the stiffness at the via points ($\mathbf{K}_{des}\left(\mathbf{x}_i\right)$) are considered in the algorithm.



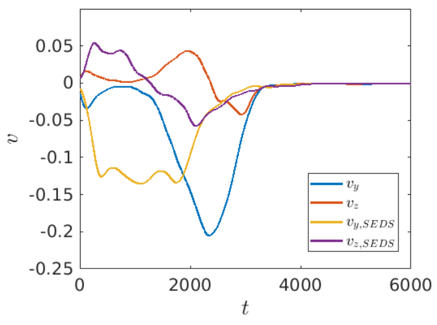
(a) Motion in Cartesian space



(b) Velocity profile



(c) Motion in Cartesian space



(d) Velocity profile

Figure 4.9: Motion execution with variable stiffness. The stiffness profile $\mathbf{K}_{des} = diag\,(400(\sin(8x_y + 0.8) + 2)/2, 1000)$. The via points of (a) and (b) are generated using greedy algorithm with respect to $F_{exp} = 20$. (c) and (d) use fixed number of points to generate the via points. But both methods result in same size of via points, $N = 14$

Chapter 5

Conclusion

This thesis proposes a method of generating a new Variable Stiffness Dynamical System (VSDS) based on a given desired original DS. The VSDS follows the motion of original DS, and is encoded with a desired stiffness profile. The stiffness is decomposed into two directions, along the motions and orthogonal to the motion. The stiffness along the motion results in different magnitudes of force when interacting with environment along the motion. The stiffness orthogonal to the motion indicates how much effort it takes to make robot deviate from the reference path. The VSDS generates the motion based on the actual robot state, which endows robot with a compliant behavior in interaction with environment. This VSDS not only shows a symmetrical attraction around the reference path, but also has a global motion-level compliance behavior, i.e. when the perturbation is large, the robot generates a new reference path according to the original DS with locally attraction towards it as well.

Two limitations are in this thesis work. The first limitation of this method is that, the attractor of the VSDS is not identical with the global attractor in original DS. It converges always a little earlier than the original one. This is caused by the weighting function. As the weighting function is not zero outside the linear DS area, other linear DS in the near always has impact on the final values. The final linear DS is dragged a little back by several linear DS before it.

The second limitation is that, there is always a trade-off between velocity profile and stiffness behavior along the motion. It depends on the generation of via points. Choosing the greedy algorithm respect to F_{exp} guarantees the preserve of velocity profile, as long as the F_{exp} is set properly. But then the stiffness behavior along the motion does not exactly follow the stiffness profile. On the other hand, using fixed number of via point makes the robot present a proper stiffness behavior. But the velocity profile then changes. In addition, the stiffness profile is encoded discretely, only the stiffness at the via points are considered.

In the future, a better transfer function should be found. This transfer function should have the following properties: 1) the transfer from one linear DS to the next DS should be smooth, which guarantees a smooth velocity profile. 2) the transfer function only has impact on the corresponding linear DS, so that the linear DS do not influence the others. Then this transfer function ensures the attractor of VSDS is identical with original DS.

Another direction is to use variable damping in the controller. As the control command is consisted of VSDS and damping term, it is possible to adjust the control force by adjusting the damping. High stiffness results in high VSDS, which increases the robot velocity. If the damping is also variable and has the same trend with variable stiffness, then it damps more velocity at higher stiffness. In this way, the desired stiffness profile and the desired velocity profile can be achieved at the same time.

Appendix A

A.1 Gradient of Transfer Function

The distance dependent Gaussian kernel is defined as:

$$\omega_i(\mathbf{x}) = e^{-\frac{\left(\mathbf{x}-\mathbf{x}_{cen,i}\right)^T\left(\mathbf{x}-\mathbf{x}_{cen,i}\right)}{2(\sigma^i)^2}} \tag{A.1}$$

The transfer function between adjacent DS is a normalization of Gaussian kernel in A.1:

$$\tilde{\omega}_i(\mathbf{x}) = \frac{\omega_i(\mathbf{x})}{\sum_{j=1}^{N}\omega_j(\mathbf{x})} \tag{A.2}$$

Then the gradient of A.2 is:

$$\begin{aligned}
\frac{\partial\tilde{\omega}_i(\mathbf{x})}{\partial\mathbf{x}} &= \frac{\frac{\partial\omega_i(\mathbf{x})}{\partial\mathbf{x}}\sum_{j=1}^{N}\omega_j(\mathbf{x}) - \omega_i(\mathbf{x})\frac{\partial\sum_{j=1}^{N}\omega_j(\mathbf{x})}{\partial\mathbf{x}}}{\left(\sum_{j=1}^{N}\omega_j(\mathbf{x})\right)^2} \\
&= \frac{-\frac{1}{(\delta_i)^2}\left(\mathbf{x}-\mathbf{x}_{cen,i}\right)\omega_i(\mathbf{x})\sum_{j=1}^{N}\omega_j(\mathbf{x}) - \omega_i(\mathbf{x})\sum_{j=1}^{N}\frac{-1}{(\delta_j)^2}\left(\mathbf{x}-\mathbf{x}_{cen,j}\right)\omega_j(\mathbf{x})}{\left(\sum_{j=1}^{N}\omega_j(\mathbf{x})\right)^2} \\
&= -\frac{1}{(\delta_i)^2}\left(\mathbf{x}-\mathbf{x}_{cen,i}\right)\tilde{\omega}_i(\mathbf{x}) + \tilde{\omega}_i(\mathbf{x})\sum_{j=1}^{N}\frac{1}{(\delta_j)^2}\left(\mathbf{x}-\mathbf{x}_{cen,j}\right)\tilde{\omega}_j(\mathbf{x})
\end{aligned} \tag{A.3}$$

A.2 Close-loop Stiffness

The close-loop stiffness of the robot is:

$$\begin{aligned}
\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} &= \sum_{i=1}^{N} \frac{\partial \tilde{\omega}_i(\mathbf{x}) \mathbf{A}_i (\mathbf{x} - \mathbf{x}_i^*)}{\partial \mathbf{x}} \\
&= \sum_{i=1}^{N} \left(\frac{\partial \tilde{\omega}_i(\mathbf{x})}{\partial \mathbf{x}} (\mathbf{x} - \mathbf{x}_i^*)^T \mathbf{A}_i^T + \tilde{\omega}_i(\mathbf{x}) \mathbf{A}_i \right) \\
&= \sum_{i=1}^{N} \left(-\frac{1}{(\delta_i)^2} (\mathbf{x} - \mathbf{x}_{cen,i}) \tilde{\omega}_i(\mathbf{x}) (\mathbf{x} - \mathbf{x}_i^*)^T \mathbf{A}_i^T \right) \\
&\quad + \sum_{i=1}^{N} \left(\tilde{\omega}_i(\mathbf{x}) \sum_{j=1}^{N} \frac{1}{(\delta_j)^2} (\mathbf{x} - \mathbf{x}_{cen,j}) \tilde{\omega}_j(\mathbf{x}) (\mathbf{x} - \mathbf{x}_i^*)^T \mathbf{A}_i^T \right) + \sum_{i=1}^{N} \tilde{\omega}_i(\mathbf{x}) \mathbf{A}_i \\
&= \sum_{i=1}^{N} \left(-\frac{\tilde{\omega}_i(\mathbf{x})}{(\delta_i)^2} (\mathbf{x} - \mathbf{x}_{cen,i}) (\mathbf{x} - \mathbf{x}_i^*)^T \mathbf{A}_i^T + \frac{\tilde{\omega}_i(\mathbf{x})}{(\delta_i)^2} (\mathbf{x} - \mathbf{x}_{cen,i}) \mathbf{f}(\mathbf{x})^T + \tilde{\omega}_i(\mathbf{x}) \mathbf{A}_i \right)
\end{aligned} \tag{A.4}$$

A.3 Eigenvector proof

We have a vector $\mathbf{x} \in \mathbb{R}^{2\times1}$, whose norm is $\|\mathbf{x}\| = \alpha$. Then we define $\mathbf{e}_1 = \frac{\mathbf{x}}{\|\mathbf{x}\|}$ and we find the vector $\mathbf{e}_2$ orthogonal to $\mathbf{e}_1$, i.e. $\mathbf{e}_2^T\mathbf{e}_1 = 0$. Now we have:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{pmatrix}$$

the eigenvalues are:

$$\mathbf{K} = \begin{pmatrix} k_1 & 0 \\ 0 & k_2 \end{pmatrix}$$

now they can construct a $\mathbf{A}$ matrix:

$$\mathbf{A} = -\mathbf{Q}\mathbf{K}\mathbf{Q}^T$$

The product of $\mathbf{A}$ and $\mathbf{x}$ is:

$$\begin{aligned}
\mathbf{A}\mathbf{x} &= -\mathbf{Q}\mathbf{K}\mathbf{Q}^T\mathbf{x} \\
&= \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{pmatrix} \begin{pmatrix} k_1 & 0 \\ 0 & k_2 \end{pmatrix} \begin{pmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \end{pmatrix} \alpha \mathbf{e}_1 \\
&= \alpha \begin{pmatrix} k_1\mathbf{e}_1 & k_2\mathbf{e}_2 \end{pmatrix} \begin{pmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \end{pmatrix} \mathbf{e}_1 \\
&= \alpha \left(k_1 \mathbf{e}_1 \mathbf{e}_1^T + k_2 \mathbf{e}_2 \mathbf{e}_2^T \right) \mathbf{e}_1 \\
&= \alpha k_1 \mathbf{e}_1 \mathbf{e}_1^T \mathbf{e}_1 + \alpha k_2 \mathbf{e}_2 \mathbf{e}_2^T \mathbf{e}_1 \\
&= k_1 \alpha \mathbf{e}_1 \\
&= k_1 \mathbf{x}
\end{aligned}$$

We see that $\mathbf{x}$ is an eigenvector of $\mathbf{A}$, and it is obvious that

$$\frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} = k_1$$

Here we take a 2D example, this relationship in higher dimensional vectors can be easily get from it.

Appendix B

B.1 C shape SEDS parameters

The SEDS with a C shape is learned from hand drawn data in Matlab GUI. The parameters are listed below:

$\mathbf{A}_1$		$\mathbf{A}_2$		$\mathbf{A}_3$	
-0.4766	-0.2808	-0.5051	-0.2904	-0.5466	-0.3903
0.2170	-0.0413	0.2546	-0.0630	0.3294	-0.1318
$\mathbf{A}_4$		$\mathbf{A}_5$		$\mathbf{A}_6$	
-0.7044	-0.7874	-8.2000	-26.8481	-1.5980	-1.9378
0.5909	-0.2221	-4.4394	-29.8678	-2.6798	-10.5546

Table B.1: $\mathbf{A}$ of C shape SEDS

$\mathbf{b}_1$	$\mathbf{b}_2$	$\mathbf{b}_3$	$\mathbf{b}_4$	$\mathbf{b}_5$	$\mathbf{b}_6$
-0.4819	-0.5108	-0.5512	-0.7045	-7.8351	-1.5950
0.2229	0.2618	0.3398	0.6092	-3.9248	-2.5237

Table B.2: $\mathbf{b}$ of C shape SEDS

$\boldsymbol{\xi}^*$
-1.0232
0.0207

Table B.3: $\boldsymbol{\xi}^*$ of C shape SEDS

$\boldsymbol{\mu}_1$	$\boldsymbol{\mu}_2$	$\boldsymbol{\mu}_3$	$\boldsymbol{\mu}_4$	$\boldsymbol{\mu}_5$	$\boldsymbol{\mu}_6$
-1.0814	-1.4525	-1.7104	-1.8106	-1.2349	-1.6178
1.8413	1.7174	1.3311	0.7280	0.0596	0.2026

Table B.4: $\boldsymbol{\mu}$ of C shape SEDS

$\mathbf{\Sigma}_1$		$\mathbf{\Sigma}_2$		$\mathbf{\Sigma}_3$	
0.0100	0.0002	0.0120	0.0051	0.0077	0.0037
0.0002	0.0035	0.0051	0.0096	0.0037	0.0170
$\mathbf{\Sigma}_4$		$\mathbf{\Sigma}_5$		$\mathbf{\Sigma}_6$	
0.0181	-0.0000	0.0214	-0.0029	0.0070	-0.0024
-0.0000	0.0516	-0.0029	0.0083	-0.0024	0.0043

Table B.5: $\mathbf{\Sigma}$ of C shape SEDS

p_1	p_2	p_3	p_4	p_5	p_6
0.0700	0.1167	0.1067	0.1067	0.0533	0.0467

Table B.6: p of C shape SEDS

B.2 Convex shape SEDS parameters

The SEDS which we used in Chapter 4 as original DS is learned from demonstrated data. The parameters are listed below:

$\mathbf{A}_1$		$\mathbf{A}_2$		$\mathbf{A}_3$	
-0.16022	-0.62291	-0.76097	-0.34418	-0.10991	-0.86913
0.1679	-0.52685	1.3589	-1.9955	0.31421	-0.73881

Table B.7: $\mathbf{A}$ of Convex shape SEDS

$\mathbf{b}_1$	$\mathbf{b}_2$	$\mathbf{b}_3$
0.021045	-0.11823	0.051308
0.076391	0.42896	0.12233

Table B.8: $\mathbf{b}$ of Convex shape SEDS

$\boldsymbol{\xi}^*$
-0.19311
0.083455

Table B.9: $\boldsymbol{\xi}^*$ of Convex shape SEDS

$\boldsymbol{\mu}_1$	$\boldsymbol{\mu}_2$	$\boldsymbol{\mu}_3$
0.29428	-0.14886	0.087785
0.16028	0.13009	0.20769

Table B.10: $\boldsymbol{\mu}$ of Convex shape SEDS

$\boldsymbol{\Sigma}_1$		$\boldsymbol{\Sigma}_2$		$\boldsymbol{\Sigma}_3$	
0.0015358	-0.00058954	0.0022244	0.0010006	0.0057907	0.00011249
-0.00058954	0.0010855	0.0010006	0.0019663	0.00011249	0.0020313

Table B.11: $\boldsymbol{\Sigma}$ of Convex shape SEDS

p_1	p_2	p_3
0.016857	0.020286	0.012857

Table B.12: p of Convex shape SEDS

B.3 SEDS parameters of insertion task

The SEDS which we used in Section 4.5 as original DS is learned from demonstrated data. The parameters are listed below:

$\mathbf{A}_1$		$\mathbf{A}_2$		$\mathbf{A}_3$	
-2.6166	0.0909	-0.3267	-0.0252	-2.0391	0.0296
-1.9464	-0.8132	0.0220	-0.2177	1.1166	-0.4913

Table B.13: $\mathbf{A}$ of insertion task SEDS

$\mathbf{b}_1$	$\mathbf{b}_2$	$\mathbf{b}_3$
1.2679	0.1580	0.9877
0.9350	-0.0127	-0.5452

Table B.14: $\mathbf{b}$ of insertion task SEDS

$\boldsymbol{\xi}^*$
0.4842
-0.0093

Table B.15: $\boldsymbol{\xi}^*$ of insertion task SEDS

$\boldsymbol{\mu}_1$	$\boldsymbol{\mu}_2$	$\boldsymbol{\mu}_3$
0.4853	0.4927	0.4869
0.0142	0.1551	0.0938

Table B.16: $\boldsymbol{\mu}$ of insertion task SEDS

$\mathbf{\Sigma}_1$		$\mathbf{\Sigma}_2$		$\mathbf{\Sigma}_3$	
0.0001260	0.0000017	0.0000445	0.0000150	0.0001868	0.0000152
0.0000017	0.0003598	0.0000150	0.0001155	0.0000152	0.0005311

Table B.17: $\mathbf{\Sigma}$ of insertion task SEDS

p_1	p_2	p_3
0.0265	0.0098	0.0138

Table B.18: p of insertion task SEDS

List of Figures

Acronyms and Notations

DS Dynamical Systems

VSDS Variable Stiffness DS

DOF Degree Of Freedom

LfD Learning from Demonstration

RK Runge-Kutta method

VIC Variable Impedance Control

CLMG Closed Loop Motion Generation

DMP Dynamic Movement Primitives

SEDS Stable Estimator of Dynamical Systems

GMM Gaussian Mixture Model

LMDS Locally Modulated Dynamical Systems

LAGS-DS Locally Active Global Stable DS

GMR Gaussian Mixture Regression

Bibliography

[BOF$^+$01] Etienne Burdet, Rieko Osu, David W Franklin, Theodore E Milner, and Mitsuo Kawato. The central nervous system stabilizes unstable dynamics by learning optimal impedance. *Nature*, 414(6862):446–449, 2001.

[BTSS11] Jonas Buchli, Evangelos Theodorou, Freek Stulp, and Stefan Schaal. Variable impedance control a reinforcement learning approach. *Robotics: Science and Systems VI*, pages 153–160, 2011.

[CSC10] Sylvain Calinon, Irene Sardellitti, and Darwin G Caldwell. Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 249–254. IEEE, 2010.

[FB17] Nadia Figueroa and Aude Billard. Learning complex manipulation tasks from heterogeneous and unstructured demonstrations. In *IROS Workshop on Synergies between Learning and Interaction*, 2017.

[FFB18] Nadia Barbara Figueroa Fernandez and Aude Billard. Modeling compositions of impedance-based primitives via dynamical systems. In *In Proceedings of the Workshop on Cognitive Whole-Body Control for Compliant Robot Manipulation (COWB-COMP)*, number CONF, 2018.

[Hog85] Neville Hogan. Impedance control: An approach to manipulation: Part i – theory. 1985.

[Hog87] Neville Hogan. Stable execution of contact tasks using impedance control. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 1047–1054. IEEE, 1987.

[HWWAS11] Sami Haddadin, Michael Weis, Sebastian Wolf, and Alin Albu-Schäffer. Optimal control for maximizing link velocity of robotic variable stiffness joints. *IFAC Proceedings Volumes*, 44(1):6863–6871, 2011.

[INH$^+$13] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[KB12] Klas Kronander and Aude Billard. Online learning of varying stiffness through physical human-robot interaction. In *2012 IEEE International Conference on Robotics and Automation*, pages 1842–1849. Ieee, 2012.

[KB13] Klas Kronander and Aude Billard. Learning compliant manipulation through kinesthetic and tactile human-robot interaction. *IEEE transactions on haptics*, 7(3):367–380, 2013.

[KB15] Klas Kronander and Aude Billard. Passive interaction control with dynamical systems. *IEEE Robotics and Automation Letters*, 1(1):106–113, 2015.

[KB19] Mahdi Khoramshahi and Aude Billard. A dynamical system approach to task-adaptation in physical human–robot interaction. *Autonomous Robots*, 43(4):927–946, 2019.

[KKB15] Klas Kronander, Mohammad Khansari, and Aude Billard. Incremental motion learning with locally modulated dynamical systems. *Robotics and Autonomous Systems*, 70:52–62, 2015.

[KMK$^+$10] Jens Kober, Katharina Mülling, Oliver Krömer, Christoph H Lampert, Bernhard Schölkopf, and Jan Peters. Movement templates for learning of hitting and batting. In *2010 IEEE International Conference on Robotics and Automation*, pages 853–858. IEEE, 2010.

[KZB11] S Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[KZB14] S Mohammad Khansari-Zadeh and Aude Billard. Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6):752–765, 2014.

[KZK17] Seyed Mohammad Khansari-Zadeh and Oussama Khatib. Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors. *Autonomous Robots*, 41(1):45–69, 2017.

[KZKB14] S Mohammad Khansari-Zadeh, Klas Kronander, and Aude Billard. Modeling robot discrete movements with state-varying stiffness and damping: A framework for integrated motion generation and impedance control. *Proceedings of Robotics: Science and Systems X (RSS 2014)*, 10:2014, 2014.

[OMN10] Christian Ott, Ranjan Mukherjee, and Yoshihiko Nakamura. Unified impedance and admittance control. In *2010 IEEE International Conference on Robotics and Automation*, pages 554–561. IEEE, 2010.

[Par01] Jong Hyeon Park. Impedance control for biped robot locomotion. *IEEE Transactions on Robotics and Automation*, 17(6):870–882, 2001.

[PP88] Brad Paden and Ravi Panja. Globally asymptotically stable 'pd+' controller for robot manipulators. *International Journal of Control*, 47(6):1697–1712, 1988.

[RCC$^+$13] Leonel Dario Rozo, Sylvain Calinon, Darwin Caldwell, Pablo Jiménez, and Carme Torras. Learning collaborative impedance-based robot behaviors. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[Sch06] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer, 2006.

[SIB03] Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):537–547, 2003.

[Str01] Stephen Strogatz. Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering (studies in nonlinearity). 2001.

[YGH$^+$11] Chenguang Yang, Gowrishankar Ganesh, Sami Haddadin, Sven Parusel, Alin Albu-Schaeffer, and Etienne Burdet. Human-like adaptation of force and impedance in stable and unstable interactions. *IEEE transactions on robotics*, 27(5):918–930, 2011.

License