Enhancement of the Robustness of Redundant Robot Arms Against Perturbations by Inferring Dynamical Systems Using Echo State Networks

Hiroshi Atsuta
Symbiotic Intelligent Systems Research Center,
Institute for Open and Transdisciplinary Research Initiatives,
Osaka University
Suita, Osaka, Japan
hiroshi.atsuta@otri.osaka-u.ac.jp

Yuji Kawai
Symbiotic Intelligent Systems Research Center,
Institute for Open and Transdisciplinary Research Initiatives,
Osaka University
Suita, Osaka, Japan
kawai@otri.osaka-u.ac.jp

Minoru Asada
International Professional University of Technology in Osaka
Kita-ku, Osaka, Japan
Symbiotic Intelligent Systems Research Center, Institute for Open and Transdisciplinary Research Initiatives, Osaka University
Suita, Osaka, Japan
Chubu University Academy of Emerging Sciences
Kasugai, Aichi, Japan
Center for Information and Neural Networks, National Institute of Information and Communications Technology
Suita, Osaka, Japan
asada@otri.osaka-u.ac.jp

2024 International Joint Conference on Neural Networks (IJCNN) | 979-8-3503-5931-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/IJCNN60899.2024.10649930

***Abstract*—A critical aspect in robot technology lies in ensuring that the robot arm executes the intended task or movement. The conventional teaching–playback technique for programming robot arm movements generates excessive torque when the initial posture differs from that during motion teaching or when external disturbances disrupt robotic arm execution. To address this problem, this study proposed a motion generation system with echo state networks (ESNs) for robot arms. These networks can reproduce trajectories including features such as attractors and bifurcations by training time-series data originating from a dynamical system. By leveraging this capability, the proposed approach enables an ESN to infer a dynamical system underlying demonstrated motions to perform a specific task on the basis of state feedback. Subsequently, the trained ESN generates reference signals to replicate the learned motion based on the current robot state. These reference signals can be used for linear feedback at the joint level. Redundant robots can easily transition to unobserved states because of their degrees of freedom. However, the trained ESN exhibits remarkable generalization capabilities, effectively guiding the robot back to its intended motion in those unobserved states. Numerical simulations confirmed that the proposed system dynamically adapts to robot behavior, effectively mitigating excessive acceleration, even in scenarios with initial posture deviations or external disturbances, thereby outperforming conventional teaching–playback methods.**

***Index Terms*—reservoir computing, echo state network, recurrent neural network, redundant robotic arm, learning from demonstration**

This work was supported by the project, JPNP16007, commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and JST, CREST (JPMJCR17A4).

I. Introduction

In the field of robotics technology, it is crucial to ensure that the robot arm can carry out its intended tasks or movements robustly. The teaching–playback technique is widely used for programming industrial robotic arms. In particular, direct teaching, also known as kinesthetic teaching, involves recording joint displacements as a time series by manually guiding the end effector of a robot in accordance with a work procedure. During the playback of the taught motion, a linear feedback control is used for each joint's position and velocity to adequately trace a trajectory smoothed by interpolating the recorded points. This approach has been theoretically proven to be effective [1].

Although direct teaching does not require specialized programming skills, its effectiveness is limited to scenarios in which the robot repeats identical motions in controlled, human-free environments. In environments in which robots coexist with humans, they are required to adapt to external forces from unexpected contact with humans or obstacles. The "learning from demonstration" is focused on generating motions required for task execution on the basis of demonstrated motions rather than just replaying them [2]. Numerous techniques based on this framework have been proposed [3]. In particular, in scenarios in which flexibility against disturbances is critical, generating motions through state feedback without explicitly including time is effective.

Ijspeert et al. [4] achieved robust and time-independent motion generation by optimizing parameters within an a priori assumed nonlinear equation based on numerous motion trajectories. Instead of assuming specific equations, Khansari-Zadeh and Billard [5] proposed a general method in which a Gaussian mixture model is optimized to ensure convergence at trajectory endpoints. Despite fluctuations in the initial posture or end-effector displacements caused by disturbances, this method effectively mitigates excessive accelerations by evolving the motion reference trajectory from the robot's current state. However, this method requires predefined constraints regarding task objectives, for stable dynamical system learning. Furthermore, it does not address the ill-posedness problem of the inverse kinematics for redundant robots [6] when computing reference values in the joint space from positions in the Cartesian space. A method for efficiently exploiting redundancy that involves estimating the appropriate stiffness parameters for joint-level impedance control was also proposed in a previous study [7]. However, both these methods [5], [7] incur high computational costs owing Gaussian mixture model learning.

By contrast, reservoir computing [8], [9] is an efficient and low-cost alternative for learning time-series patterns. Particularly, echo state networks (ESNs), a type of recurrent neural network with fixed connection weights known as a reservoir, have successfully replicated trajectories from time-series data sampled from a dynamical system. The trajectories have features of the original dynamical system, such as attractors and bifurcation structures [10]–[12]. Kim et al. [12] demonstrated that ESNs can predict the future states of a chaotic dynamical system. This capability can be applied to learn an underlying dynamical system to execute a specific task with a minimum number of motion instructions for robots.

In this study, we proposed an ESN-based system for inferring an underlying dynamical system that accomplishes a taught task from demonstrated motions. Training the ESN by using a time series of joint displacements from the demonstrated motions enables the output of the ESN to be used as a reference signal for linear feedback control at the joint level. The redundancy of the robot arm makes the robot prone to transit to unseen states in the presence of initial posture fluctuations or external disturbances. However, the reference signal continuously reflects the current state of the robot so that it guides the robot gradually back to the demonstrated motion from these unseen states. It results in effectively mitigating excessive acceleration. This study validated the proposed system by performing a reaching task to assess the system's adaptability and performance compared with conventional teaching–playback through numerical simulations.

II. Preliminary study: attractor estimation

A. ESN

An ESN is a recurrent neural network consisting of a reservoir layer with fixed connection weights and a trainable readout layer [9]. Given an input vector time series $\boldsymbol{u}(k) \in \mathbb{R}^{N_u}$, where $k \in \{0, 1, \dots, K\}$ denotes discrete time steps, this preliminary study is aimed to infer the attractor structure of an underlying dynamical system.

The dynamics of the reservoir state $\boldsymbol{r}(k) \in \mathbb{R}^{N_r}$ is recursively defined as follows:

$$\tilde{\boldsymbol{r}}(k+1) = \varphi\left(\boldsymbol{W}\boldsymbol{r}(k) + \boldsymbol{W}^{\text{in}}\boldsymbol{u}(k)\right) \tag{1}$$

$$\boldsymbol{r}(k+1) = (1-\alpha)\boldsymbol{r}(k) + \alpha\tilde{\boldsymbol{r}}(k+1), \tag{2}$$

where $\boldsymbol{W} \in \mathbb{R}^{N_r \times N_r}$ is the recurrent connection weight matrix, which is sampled with a connection probability p from a uniform distribution in the interval $[-1,\ 1]$ and scaled so that the spectral radius coincides with ρ. The input weight matrix $\boldsymbol{W}^{\text{in}} \in \mathbb{R}^{N_r \times N_u}$ is sampled from a uniform distribution in the interval $[-a,\ a]$, where $a > 0$ denotes input scaling. Additionally, $\varphi : \mathbb{R}^{N_r} \to \mathbb{R}^{N_r}$ represents the activation function, and α is the leakage rate.

The readout layer is defined as follows:

$$\boldsymbol{v}(k) = \boldsymbol{W}^{\text{out}}\boldsymbol{r}(k), \tag{3}$$

where $\boldsymbol{v}(k) \in \mathbb{R}^{N_v}$ denotes the network output, and $\boldsymbol{W}^{\text{out}} \in \mathbb{R}^{N_v \times N_r}$ is the output weight matrix.

Let the reservoir state be initialized as $\boldsymbol{r}(0) = [0, \dots, 0]^{\text{T}}$. After iteratively computing the reservoir state for the input sequence $\boldsymbol{u}(k)$ according to (1) and (2), $\boldsymbol{W}^{\text{out}}$ is obtained by solving the following optimization problem:

$$\boldsymbol{W}^{\text{out}} = \operatorname*{arg\,min}_{\boldsymbol{W}^{\text{out}}} \frac{1}{N_v}\sum_{i=1}^{N_v}\left(\sum_{k=0}^{K}\Big(d_i(k) - v_i(k)\Big)^2 + \beta\|\boldsymbol{w}_i^{\text{out}}\|^2\right), \tag{4}$$

where $\boldsymbol{w}_i^{\text{out}}$ denotes the ith row of $\boldsymbol{W}^{\text{out}}$, $\beta > 0$ represents a regularization coefficient, and $\|\cdot\|$ denotes the Euclidean norm. Furthermore, $\boldsymbol{d}(k) \in \mathbb{R}^{N_v}$ is the time series obtained from a dynamical system to be estimated and corresponds to the input time series $\boldsymbol{u}(k)$ in this section. The solution of the optimization problem expressed by (4) has the following closed form:

$$\boldsymbol{W}^{\text{out}} = \boldsymbol{D}\boldsymbol{R}^{\text{T}}\left(\boldsymbol{R}\boldsymbol{R}^{\text{T}} + \beta\boldsymbol{I}\right)^{-1}, \tag{5}$$

where $\boldsymbol{D} \in \mathbb{R}^{N_v \times (K+1)}$ are all $\boldsymbol{d}(k)$ and $\boldsymbol{R} \in \mathbb{R}^{N_r \times (K+1)}$ are all $\boldsymbol{r}(k)$ produced by presenting the reservoir with $\boldsymbol{u}(k)$, both collected into respective matrices by concatenating the column-vectors horizontally over the training period.

After training the output weight matrix $\boldsymbol{W}^{\text{out}}$, the estimated time series $\boldsymbol{x}(k)$ is obtained by evolving it over time under the autonomous dynamical system defined as follows:

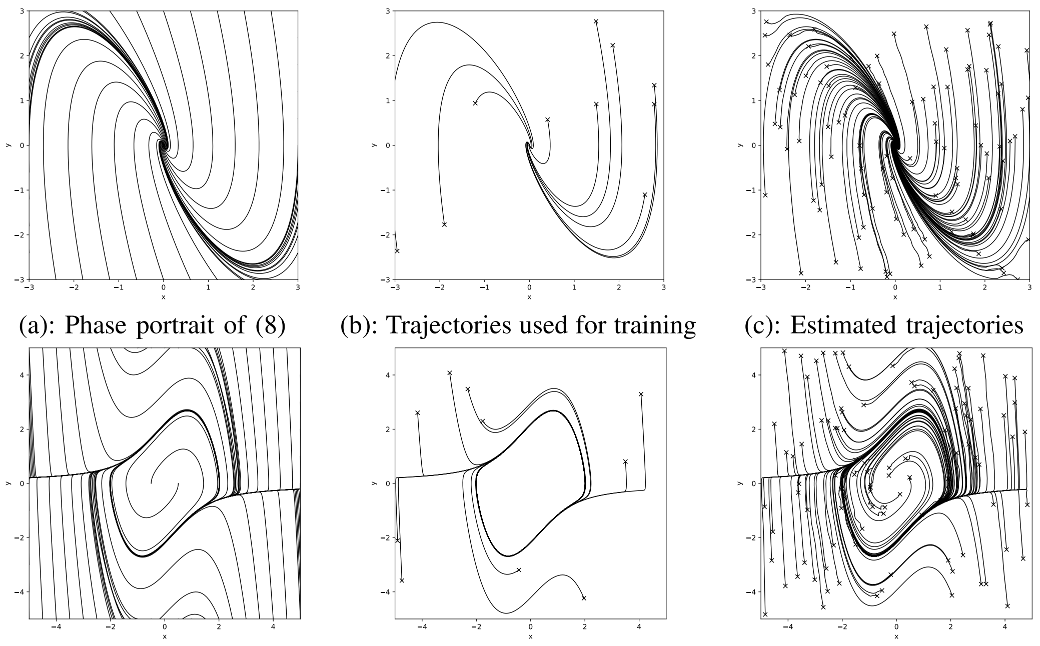
$$\boldsymbol{s}(k+1) = (1-\alpha)\boldsymbol{s}(k) + \alpha\varphi\left(\boldsymbol{W}\boldsymbol{s}(k) + \boldsymbol{W}^{\text{in}}\boldsymbol{x}(k)\right) \tag{6}$$

$$\boldsymbol{x}(k+1) = \boldsymbol{W}^{\text{out}}\boldsymbol{s}(k+1). \tag{7}$$

Here, $\boldsymbol{s}(k) \in \mathbb{R}^{N_r}$ represents the reservoir state for autonomous driving, which is initialized as $\boldsymbol{s}(0) = \boldsymbol{r}(0)$. The initial input $\boldsymbol{x}(0)$ is set to an arbitrary value prior to driving.

B. Inferring stable attractors with the ESN

The geometric features of chaotic attractors, such as the Macky–Glass and Lorenz attractors, can be reproduced using the method outlined in Section II-A [10], [12]. However, chaotic motion is not a prerequisite for early stage applications



(a): Phase portrait of (8) (b): Trajectories used for training (c): Estimated trajectories

(d): Phase portrait of (9) (e): Trajectories used for training (f): Estimated trajectories

Fig. 1: Estimation of attractors in dynamical systems using ESNs

TABLE I: Parameters used for training ESNs

Parameter	Description	Value (Sec. II)	Value (Sec. IV)
N_r	Number of neurons	300	400
a	Input scaling	0.1	0.001
p	Connect probability	0.1	0.1
ρ	Spectral radius	0.95	0.95
α	Leaking rate	0.99	0.4
φ	Activation function	tanh	tanh
β	Regularization coef.	0.0001	0.0001

in robot motion generation. Therefore, we assumed reaching and periodic motions to be fundamental motions and verified whether the attractor can be inferred from the time series for a dynamical system characterized by an equilibrium point or a limit cycle.

We consider the following second-order linear differential equation:

$$a\ddot{x} + b\dot{x} + cx = 0. \tag{8}$$

When appropriately parameterized, (8) presents an equilibrium point on the phase plane. Fig. 1(a) displays the phase portrait of (8) when the parameters are set to $a = 1, b = 3, c = 4$.

Fig. 1(b) displays the solution trajectories starting with 10 random initial points on the phase plane and evolving according to (8). Subsequently, the ESN in Table I was trained using these time-series data. Fig. 1(c) displays the results of autonomous driving according to (6) and (7), starting with 100 random initial points. These 100 initial points differed from those used during training.

For a dynamical system characterized by a stable limit cycle, we examined the van der Pol equation expressed as follows:

$$\ddot{x} = \mu(1 - x^2)\dot{x} - x. \tag{9}$$

TABLE II: Average and variance of root mean square error

	Average	Variance
Equation (8)	1.99×10^{-3}	4.15×10^{-6}
Equation (9)	1.15×10^{-1}	1.04×10^{-2}

Fig. 1(d) illustrates the phase portrait when $\mu = 1$. Fig. 1(e) displays the solution trajectories starting with 10 random initial points. The ESN described in Table I was trained with them. Fig. 1(f) shows a set of trajectories that evolved over time from 100 random initial points.

The root mean squared error (RMSE) was calculated for trajectory $[{}^{i}x(k) \ \ {}^{i}\dot{x}(k)]^{\mathrm{T}}$ evolved from the ith initial value ${}^{i}\boldsymbol{x}(0)$ by the trained ESN, and the solution trajectory $[{}^{i}x^{\mathrm{sc}}(k) \ {}^{i}\dot{x}^{\mathrm{sc}}(k)]^{\mathrm{T}}$ evolved from the same initial value ${}^{i}\boldsymbol{x}(0)$ by the original dynamical system as follows:

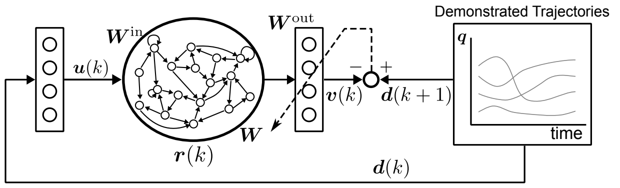
$$E({}^{i}\boldsymbol{x}, {}^{i}\boldsymbol{x}^{\mathrm{sc}}) = \frac{1}{2}\sum_{j=1}^{2}\left(\sqrt{\frac{1}{K+1}\sum_{k=0}^{K}\left({}^{i}x_j(k) - {}^{i}x_j^{\mathrm{sc}}(k)\right)^2}\right), \tag{10}$$

where ${}^{i}x_0 = {}^{i}x, {}^{i}x_1 = {}^{i}\dot{x}$, ${}^{i}x_0^{\mathrm{sc}} = {}^{i}x^{\mathrm{sc}}$ and ${}^{i}x_1^{\mathrm{sc}} = {}^{i}\dot{x}^{\mathrm{sc}}$.

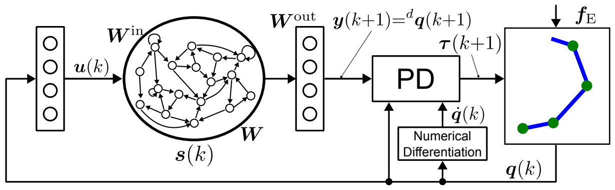
Table II summarizes the results calculated using (10) for both dynamical systems. Figures 1(c) and (f) and Table II confirm that the trained ESNs successfully reproduce the attractor of the original dynamical systems.

III. Adaptive motion generation using the ESN

We proposed a system in which the training of an ESN involves using a joint displacement time series acquired through direct teaching, with the expectation that the trained ESN reproduces the demonstrated movements. Fig. 2(a) illustrates the training process of the ESN. The time series of obtained



(a): ESN training scheme based on demonstrated motions



(b): Robot control scheme during motion generation

Fig. 2: Proposed adaptive motion generation using the ESN

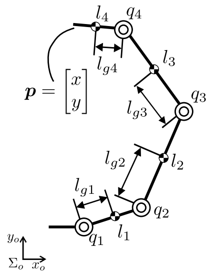


Fig. 3: Robot model

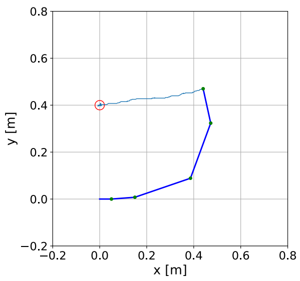


Fig. 4: Demonstrated trajectory

joint displacements correspond to $\boldsymbol{d}(k)$ in this framework. This vector is used as the input $\boldsymbol{u}(k)$ provided to the ESN input layer, which is expressed as follows:

$$\boldsymbol{u}(k) = \boldsymbol{d}(k). \tag{11}$$

When motions are generated for an n-degree-of-freedom robot using the trained ESN, the current joint displacement $\boldsymbol{q}(k)$ of the robot is applied to the input layer of the ESN, as depicted in Fig. 2(b). Subsequently, the network output $\boldsymbol{y}(k+1)$ is obtained using the following equation:

$$\boldsymbol{u}(k) = \boldsymbol{q}(k) \tag{12}$$

$$\boldsymbol{s}(k+1) = (1-\alpha)\boldsymbol{s}(k) + \alpha\varphi\left(\boldsymbol{W}\boldsymbol{s}(k) + \boldsymbol{W}^{\mathrm{in}}\boldsymbol{u}(k)\right) \tag{13}$$

$$\boldsymbol{y}(k+1) = \boldsymbol{W}^{\mathrm{out}}\boldsymbol{s}(k+1), \tag{14}$$

where the network output $\boldsymbol{y}(k+1)$ is the desired joint displacement, ${}^{d}\boldsymbol{q}(k+1)$, for controlling the robot. Consequently, the driving torque $\boldsymbol{\tau}(k+1)$ of the robot is computed using the proportional–derivative (PD) controller as follows:

$${}^{d}\boldsymbol{q}(k+1) = \boldsymbol{y}(k+1) \tag{15}$$

$$\boldsymbol{\tau}(k+1) = \boldsymbol{K}_P\left({}^{d}\boldsymbol{q}(k+1) - \boldsymbol{q}(k)\right) - \boldsymbol{K}_D\dot{\boldsymbol{q}}(k), \tag{16}$$

where $\boldsymbol{K}_P \in \mathbb{R}^{n\times n}$ and $\boldsymbol{K}_D \in \mathbb{R}^{n\times n}$ represent the proportional and differential gains, respectively.

In contrast to autonomous driving using (6) and (7), the ESN's input layer incorporates the robot's current joint displacement $\boldsymbol{q}(k)$. This configuration can achieve adaptive motion generation that responds to the robot's current state.

IV. Numerical simulation setup

A. Robot model

To assess the effectiveness of the adaptive motion generation system outlined in Section III, we implemented a forward dynamics simulator of a planar 4-degree-of-freedom (DOF) robot arm (Fig. 3). The equation of motion of the robot is expressed as follows (omitting time t):

TABLE III: Parameters of the 4-degrees-of-freedom robot

	Link 1	Link 2	Link 3	Link 4
Link length l[m]	0.1	0.25	0.25	0.15
COM position l_g[m]	0.05	0.1	0.1	0.05
Link mass m[kg]	0.5	1.0	1.0	0.5
Inertia moment I[kgm^2]	0.005	0.01	0.007	0.003

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} = \boldsymbol{\tau} + \boldsymbol{J}_E^{\mathrm{T}}(\boldsymbol{q})\boldsymbol{f}_E - \boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}), \tag{17}$$

where $\boldsymbol{q} \in \mathbb{R}^4$ is the vector representing joint displacements, $\boldsymbol{\tau} \in \mathbb{R}^4$ is the driving torque vector acting on the joints, $\boldsymbol{H}(\boldsymbol{q}) \in \mathbb{R}^{4\times 4}$ is the inertia matrix, $\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^4$ signifies the centrifugal and Coriolis forces, $\boldsymbol{f}_E \in \mathbb{R}^2$ denotes the contact force acting on the tip, and $\boldsymbol{J}_E \in \mathbb{R}^{2\times 4}$ is the Jacobian matrix mapping joint velocity $\dot{\boldsymbol{q}}$ to the tip velocity $\boldsymbol{v}_E$. External forces act only on the tip in this simulation.

The parameters for the robot listed in Table III were used for the numerical simulations. The proportional and derivative gains of the PD controller were configured as $\boldsymbol{K}_P = diag\{35.0, 35.0, 35.0, 35.0\}$ and $\boldsymbol{K}_D = diag\{0.4, 0.4, 0.4, 0.4\}$, respectively. Numerical integration was performed using the fourth-order Runge–Kutta method with an integration time step of $\Delta t = 10$ ms.

B. Reaching motion demonstration

To instruct a motion, we implemented the simulator to allow the tip of the robot to be manipulated using a mouse, and joint angles at each sampling time were recorded as a time series.

The reaching motion was selected as the teaching task for the robot. The robot was initialized in a random posture and the target was set to $\boldsymbol{p}_{\mathrm{g}} = [0.0\ 0.4]^{\mathrm{T}}$ (Fig. 4). A reaching motion that guided the tip of the robot to the target for approximately 10 s and kept it within a circle with a radius of 3 cm for approximately 5 s was demonstrated.

The ESN, configured with the parameters in Table I, was trained using the acquired time series of the joint displacements, by employing the method outlined in Section III. For performance assessment, we conducted simulations in which the desired joint displacement ${}^{d}\boldsymbol{q}(k+1)$ in (16), was replaced with the recorded time-series data. However, the recorded data were smoothed over time using spline interpolation since the sampling time for recording and controlling differed from each other. This technique is commonly used in conventional teaching–playback methods.

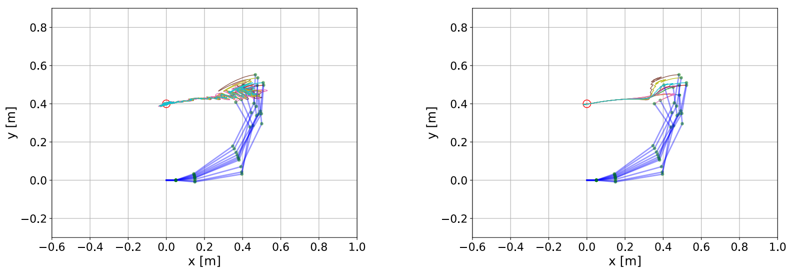

(a): Controlled by spline interpolation. (b): Controlled by trained ESN.

Fig. 5: Results of performing reaching motions from random initial postures

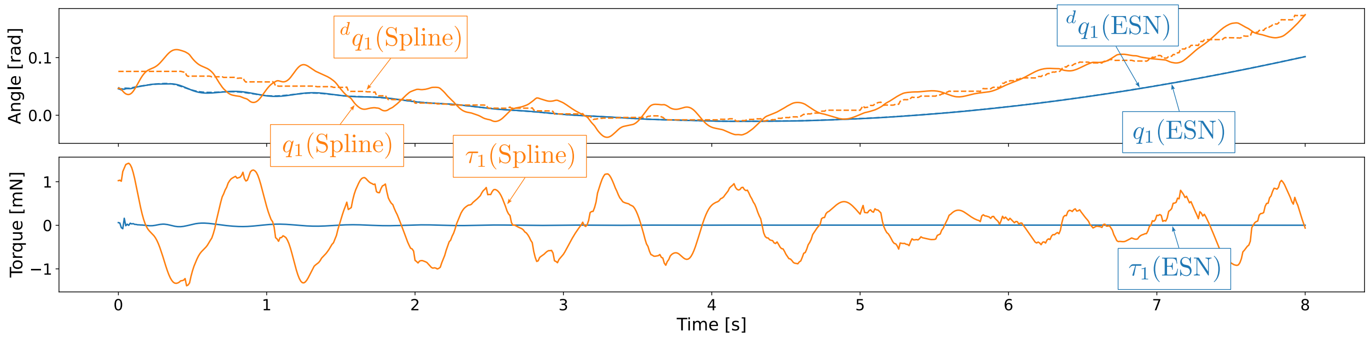


Fig. 6: Comparison of desired and actual joint displacements as well as driving torques when starting from the same initial posture for spline interpolation and the proposed ESN-based system. In the upper row, the desired joint displacements are represented by dotted lines, whereas the actual joint displacements are indicated by solid lines. The lower row displays the driving torques for both methods. This image displays the first joint of the robot and excludes the data beyond the initial 8 s.

V. Results

A. Starting from deviated initial postures

To compare behaviors controlled under conventional teaching–playback method and the proposed motion generation using ESN, simulations were conducted starting from various initial postures that deviated slightly from that used during the motion teaching. Simulations were initiated from ten postures. The results are displayed in Fig. 5, demonstrating successful arrivals at the target in all cases. However, while spline interpolation generated jerky motions, the proposed system exhibited smooth movements. We compared the two methods using specific motions starting from the same initial posture chosen from the ten postures. The upper row of Fig. 6 displays the desired and actual joint displacements of the first joint of the robot, with those after 8 s being excluded for clarity. The driving torques generated at this joint are presented in the lower row. The orange and blue lines correspond to spline interpolation and the proposed system, respectively.

In spline interpolation control, an error existed between the initial joint displacement and the reference, generating considerable driving torque. This phenomenon resulted in an oscillatory behavior that persisted until the motion ended. By contrast, the overlap of the solid and dotted blue lines in the upper row of Fig. 6 shows that the proposed ESN-based control generated reference signals based on the robot's current state. Therefore, it resulted in mitigation of substantial torque generation throughout the motion.

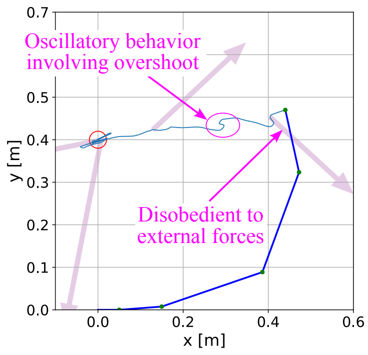
B. Applying external forces during motion

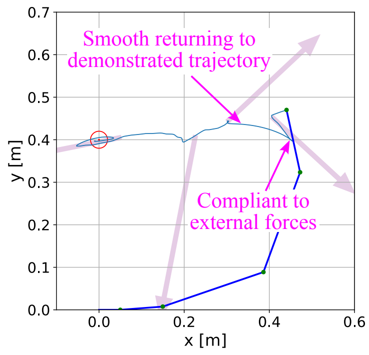
Subsequently, we compared the behavior of the robot induced by spline interpolation and the proposed system when external forces were applied during motion. Fig. 7 displays the trajectories of the robot's tip when external forces of identical directions and magnitudes were applied along the same time course. The purple arrows indicate the directions and magnitudes of the external forces. The controller using spline interpolation resisted the external forces, causing overshooting on removal. The proposed system demonstrated compliant responses to the external forces, gradually readjusting towards the target on their removal.

Fig. 8 displays the external forces, desired and actual joint displacements of the first joint, and the generated driving torques for both control methods. Spline interpolation generated substantial torque because of the deviation from the reference after the first external force was applied, perpetuating the oscillatory behavior even after the force was removed. By contrast, ESN control avoided excessive torque generation because the desired joint displacement was generated from the actual displacement even under external force application.

VI. Conclusion

In this study, we developed a robust system to perform a demonstrated task for a redundant robotic arm. We used ESNs to successfully infer the underlying dynamical system from time-series data. The system integrates an ESN trained on





(a): Controlled by spline interpolation. (b): Controlled by trained ESN.

Fig. 7: Trajectories of robot's tip when external forces are applied during motion. In both cases, the directions and magnitudes of these forces were identical and applied at the same timings. The initial two forces correspond to the shaded purple regions depicted in Fig. 8.

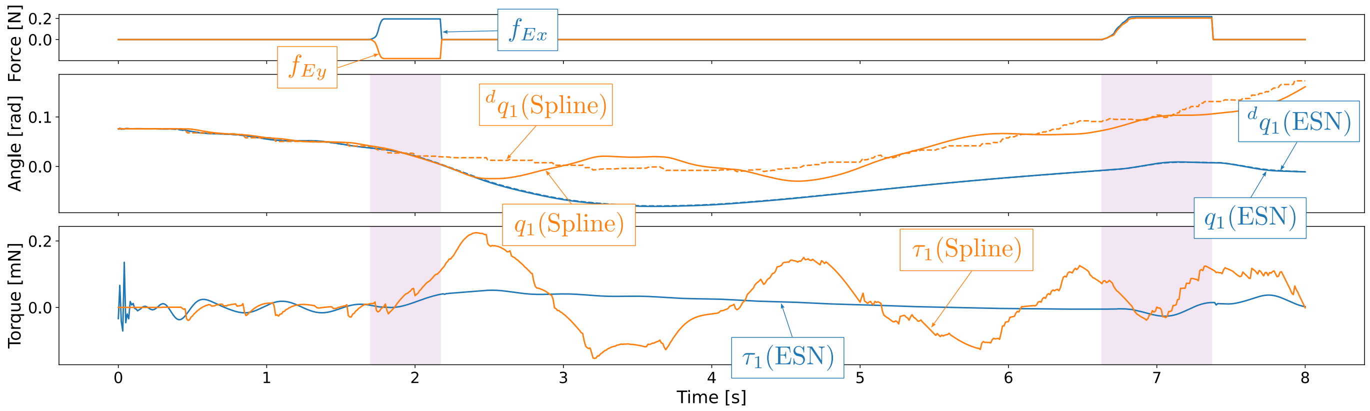


Fig. 8: Comparison of desired and actual joint displacements, along with driving torques when external forces are applied during motion for spline interpolation and the proposed ESN-based system. The upper row displays the time course of the applied external forces. In the middle row, the desired joint displacements are represented by dotted lines, whereas the actual joint displacements are indicated by solid lines. The lower row displays the driving torques for both methods. The image displays the first joint of the robot and excludes the data beyond the initial 8 s.

demonstrated motions, using its output as the control reference for the robot. The robot could learn reaching motion towards a target with minimal motion instructions. The adaptability of the system to slight deviations in the initial posture and random external forces during motion was demonstrated.

References

[1] S. Arimoto and F. Miyazaki, "Asymptotic Stability of Feedback Control Laws for Robot Manipulator," *IFAC Proceedings Volumes*, vol. 18, no. 16, pp. 221–226, Nov. 1985.

[2] S. Schaal, "Learning from Demonstration," in *Advances in Neural Information Processing Systems*, vol. 9. MIT Press, 1996, pp. 1040–1046.

[3] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent Advances in Robot Learning from Demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020.

[4] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning Attractor Landscapes for Learning Motor Primitives," in *Advances in Neural Information Processing Systems*, vol. 15. MIT Press, 2002, pp. 1545–1552.

[5] S. M. Khansari-Zadeh and A. Billard, "Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, Oct. 2011.

[6] S. Arimoto, J.-H. Bae, H. Hashiguchi, and R. Ozawa, "Natural Resolution of Ill-Posedness of Inverse Kinematics for Redundant Robots Under Constraints," *Communications in Information and Systems*, vol. 4, no. 1, pp. 1–28, 2004.

[7] M. Saveriano, F. J. Abu-Dakka, and V. Kyrki, "Learning stable robotic skills on Riemannian manifolds," *Robotics and Autonomous Systems*, vol. 169, p. 104510, Nov. 2023.

[8] W. Maass, T. Natschläger, and H. Markram, "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, Nov. 2002.

[9] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks – with an Erratum note," German National Research Center for Information Technology, Tech. Rep. 148, 2001.

[10] H. Jaeger and H. Haas, "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication," *Science*, vol. 304, no. 5667, pp. 78–80, Apr. 2004.

[11] A. Hart, J. Hook, and J. Dawes, "Embedding and approximation theorems for echo state networks," *Neural Networks*, vol. 128, pp. 234–247, Aug. 2020.

[12] J. Z. Kim, Z. Lu, E. Nozari, G. J. Pappas, and D. S. Bassett, "Teaching recurrent neural networks to infer global temporal structure from local examples," *Nature Machine Intelligence*, vol. 3, no. 4, pp. 316–323, Apr. 2021.