2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Limit Cycle Generation with Pneumatically Driven Physical Reservoir Computing

Hiroaki Shinkawa[1*], Toshihiro Kawase[2*], Tetsuro Miyazaki[1],
Takahiro Kanno[3], Maina Sogabe[1] and Kenji Kawashima[1]

***Abstract*— One of the recent developments in physical reservoir computing, which uses the complex dynamics of a physical system as a computational resource, is the use of a pneumatic pipeline system as a computational resource. This uses the dynamics of air for computation, and because it is lightweight and power-saving, it is used for gait-assist control using a soft exoskeleton with pneumatic rubber artificial muscles. In this study, we verified that by feeding back the estimated information to a pneumatic pipeline system, the pneumatic physical reservoir computing can generate periodic pressure changes as a stable limit cycle, such as those seen in walking. A pneumatic reservoir with feedback loops was modeled to generate limit cycles in the simulation, and it was confirmed that the system could generate limit cycles with high accuracy even from initial positions far from the target limit cycle. This system is expected to be applied to assist walking movements with a soft exoskeleton with a lightweight computational device.**

I. Introduction

In advanced robots and exoskeletons, the realization of lightweight computational devices is an important challenge. In recent years, large-scale machine learning frameworks, such as deep learning, have enabled pattern recognition that was previously unattainable. However, training large-scale models often requires optimization of hundreds of millions of parameters, which takes an enormous amount of time and computational resources. They require not only hardware for computation, but also batteries to supply power. Therefore, lighter and more power-efficient computational methods are needed for controlling robots that need to be lightweight, especially those worn on the body, such as prosthetic limbs and exoskeletons.

Reservoir computing exists as an effective machine learning method that efficiently handles time series data with a relatively small computational complexity [1]. Reservoir computing is one of the neural network methods that were originally proposed as the echo state network [2] and liquid state machine [3]. In feed-forward neural networks, which have been widely used in recent years, the connections between neurons in multiple layers are all updated in training. On the other hand, reservoir computing consists of an input layer, a reservoir layer, and an output layer, as shown in Fig. 1(a). The connections between the input and reservoir layers and within the reservoir layer, as well as their weights, are first randomly determined and thereafter fixed. Only the weights of the connections between the reservoir and output layers are updated during training. This makes it possible to perform training with a lower computational cost than a neural network that updates all parameters. Furthermore, since the reservoir layer has recurrent connections, it is effective for time series data where the state depends not only on the current input but also on the past input history.

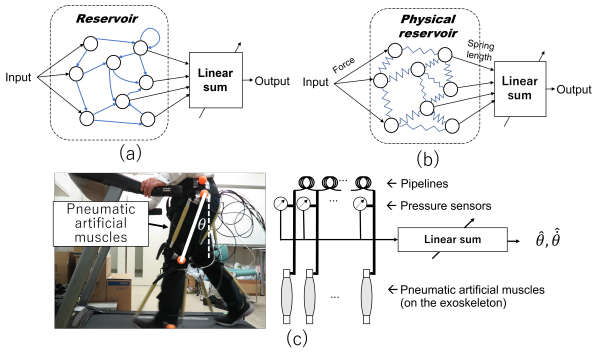

Fig. 1. Reservoir computing (a), physical reservoir computing (b) and a soft exoskeleton using pneumatic reservoir computing (c). Details of the pneumatic reservoir computing and exoskeleton can be found in [11].

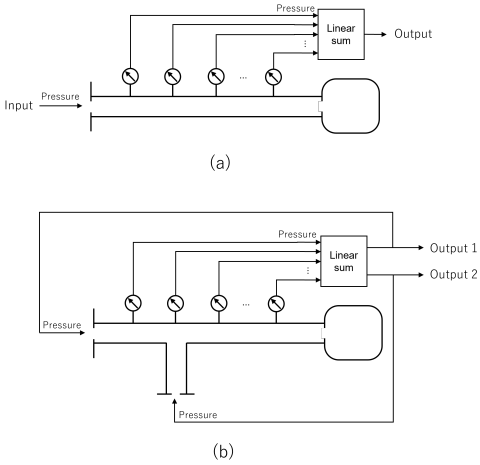

Fig. 2. Pneumatic reservoir computing with no feedback (a) and with feedback (b).

*Authors equally contributed this work. This work was supported by JSPS KAKENHI Grant Numbers 20K04375, 21H04544.

[1]Hiroaki Shinkawa, Tetsuro Miyazaki, Maina Sogabe and Kenji Kawashima are with Department of Information Physics and Computing, The University of Tokyo, Tokyo 113-8656, Japan

[2]Toshihiro Kawase is with Department of Information and Communicatin Engineering, School of Engineering, Tokyo Denki University, Tokyo 120-8551, Japantkawase@mail.dendai.ac.jp

[3]Takahiro Kanno is with Riverfield Inc., Tokyo 160-0017, Japan

Corresponding author: Toshihiro Kawase

Since the connections between the input and reservoir layers and within the reservoir layer are randomly determined and then fixed, this part does not necessarily need to be a neuron with artificially specified nonlinear functions and connections. Therefore, it is possible to replace the reservoir layer with a physical system that exhibits complex dynamics (Fig. 1(b)). This is called physical reservoir computing, and the physical systems that can be used here are spring-mass systems [4], [5], optical systems [6], analog circuits [7], soft robots [8], spintronics systems [9], etc.

Recently, a method using a pneumatic pipeline system as a physical reservoir (hereinafter referred to as pneumatic reservoir computing) was proposed. Kawase et al. [10] showed through simulations that it is possible to predict nonlinear time series data by using a single pneumatic pipeline system as a physical reservoir and using the pressure values in the pipeline as input/output to the physical reservoir, as shown in Fig. 2(a). An example of an application of pneumatic reservoir computing is a walking assist. This is because the compressed air used as part of the computation can also be used to drive the pneumatic artificial muscles on the walking assist device. These have been implemented in actual assistive suits, which use pneumatic reservoir computing to recognize walking movements and control the assistive suit online [11] (Fig. 1(c)).

In the physical reservoir computing, it is known that the output can be fed back to the physical reservoir to improve its computational power. Hauser et al. [5] reported that in a system using a spring-mass system as a physical reservoir, a feedback loop is constructed in which the output is part of the input to the reservoir and the physical reservoir. They reported that the computational power can be dramatically improved. As a computational example, the generation of periodic oscillations by limit cycles was shown.

This study shows that when a pneumatic pipeline system is used as a physical reservoir, it is possible to generate limit cycles by incorporating a feedback loop as shown in Fig. 2(b). The generation of limit cycles is of great importance in gait assistance. This is because if a self-excited motion can be generated by the assistive suits, it is possible to assist the user to return to the original gait even when the gait is disturbed. In this study, the simulation of a single pneumatic pipeline system used to validate the pneumatic reservoir computing was first modified to accommodate the case with confluence of two flows. Using this, we implemented an algorithm for pneumatic reservoir computing that incorporates feedback loops. By learning the weights of the output layer of the reservoir, it is possible to generate limit cycles that follow the pre-determined Van der Pol equation, and it was shown that limit cycles can be generated with high accuracy even from initial conditions far from the target.

Section 2 describes the details of the conventional pneumatic pipeline system simulation; Section 3 describes the proposed method, which extends the simulations of the previous studies and incorporates feedback loops; Section 4 presents the results of the limit cycle generation. Finally, Section 5 summarizes the paper and discusses future work.

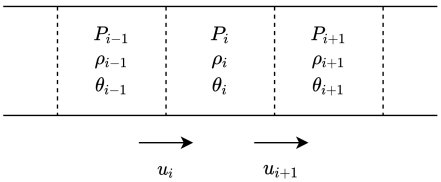

Fig. 3. Staggered grid system for the simulation.

II. Simulation model for pneumatic reservoir computing

This section describes the model used by Kawase et al. [10] to simulate pneumatic reservoir computing. This simulation is based on Li et al. [12], which numerically computes the pressure, temperature, density, velocity, etc. at each lattice in a tube, assuming a one-dimensional flow in the tube.

A. Governing equation

Let D be the inner diameter L be the length of the tube, u be the flow velocity at a point in the tube, ρ be the density, P be the pressure, x be the displacement from the origin, and t be the time, the equations of motion are as follows:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = -\frac{1}{\rho}\frac{\partial P}{\partial x} - \frac{\lambda}{2D}u^2, \tag{1}$$

where λ is the friction coefficient of the tube.

The equation of continuity then becomes

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} = 0. \tag{2}$$

Let θ be the temperature at a certain point in the tube, and the equation of energy conservation becomes

$$\frac{\partial \theta}{\partial t} = -\frac{4h(\theta - \theta_a)}{\rho C_v D} - u\frac{\partial \theta}{\partial x} - \frac{R\theta}{C_v}\frac{\partial u}{\partial x} + \frac{1}{C_v}\frac{\lambda |u| u^2}{2D}, \tag{3}$$

where θ_a is the atmospheric temperature, h is the heat transfer coefficient, R is the gas constant, and C_v is the specific heat of the gas at a constant volume.

Finally, the equation of state becomes

$$P = \rho R\theta. \tag{4}$$

From these four equations, the pressure, temperature, density, and velocity at any point in the tube can be obtained.

When actually performing numerical calculations using (1),(2) and (3), the equations are discretized using the staggered grid shown in Fig. 3. The resulting difference equations are described in [10].

The Reynolds number Re is expressed as (5), from which the friction coefficient λ can be obtained. For laminar flow, (6) is used, and for turbulent flow, (7) (Blasius' equation) is used, where μ is the viscosity of air.

$$\mathrm{Re} = \frac{\bar{\rho}|u|D}{\mu}. \tag{5}$$

$$\lambda_{\text{laminar}} = \frac{64}{\mathrm{Re}}. \tag{6}$$

$$\lambda_{\text{turbulence}} = 0.3164\mathrm{Re}^{-0.25}. \tag{7}$$

In our simulations, we assumed laminar flow when $\mathrm{Re} < 3000$ and turbulent flow otherwise. When $\mathrm{Re} = 0$, $\lambda = 0$ was assumed. However, when Re is very small (which happens when the flow velocity u is very small), λ diverges, so in this study, λ is set to 0 in the case of $\mathrm{Re} < 10^{-10}$.

B. Boundary conditions

First, we consider the boundary conditions at the inlet of the tube. In this simulation, we assume a situation where the pressure P_s at the inlet fluctuates at a specified sequence (Fig. 2(a)), and we consider a combined element of a orifice and a tube lattice. The formula for the mass flow G through the orifice varies depending on the ratio of the upstream pressure to the downstream pressure. The threshold value is called the critical pressure ratio and is obtained as follows:

$$\frac{P}{P_s} = \left(\frac{2}{\kappa+1}\right)^{\frac{\kappa}{\kappa-1}} \approx 0.5283, \tag{8}$$

where κ is the specific heat ratio of air, which is about 1.4. If the ratio of the downstream pressure to the upstream pressure is less than the critical pressure ratio, the flow velocity just after the orifice reaches the speed of sound, resulting in a sonic flow where the flow cannot go any faster. On the other hand, if the pressure ratio is greater than the critical pressure ratio, G becomes smaller as the pressure ratio approaches 1. This part of the flow is called subsonic flow. The mass flow rate G in each flow situation is expressed as follows.

$$G = \begin{cases} S_e P_s \sqrt{\frac{2\kappa}{R\theta(\kappa-1)}\left[\left(\frac{P}{P_s}\right)^{\frac{2}{\kappa}} - \left(\frac{P}{P_s}\right)^{\frac{\kappa+1}{\kappa}}\right]} \\ \qquad\qquad \left(\frac{P}{P_s} \le 0.5283\right), \\ S_e P_s \sqrt{\frac{\kappa}{R\theta}\left(\frac{2}{\kappa+1}\right)^{\frac{\kappa+1}{\kappa-1}}} \\ \qquad\qquad \left(\frac{P}{P_s} > 0.5283\right). \end{cases} \tag{9}$$

Dividing this by the cross-sectional area and density ρ yields the flow velocity u:

$$u = \frac{G}{\left(\frac{D}{2}\right)^2 \cdot \pi \cdot \rho}. \tag{10}$$

where S_e is the cross-sectional area of constricted flow that occurs immediately after the orifice.

Next, we consider the boundary conditions at the outlet. In this simulation, the outlet section consists of an orifice and an isothermally isotropic tank, as shown in Fig. 2(a). The total mass m of air in the tank at the next time is

$$m = \rho \cdot V + \rho \cdot \left(\frac{D}{2}\right)^2 \cdot \pi \cdot u \cdot dt, \tag{11}$$

where V is the volume of the tank. Thus, the density ρ_{new} at the next time is

$$\rho_{\text{new}} = \frac{m}{V}. \tag{12}$$

From the equation of state, the pressure P_{new} at the next time is

$$P_{\text{next}} = \rho_{\text{new}} R\theta. \tag{13}$$

From the pressure ratio between this and the pressure P_N in the previous tube lattice, the flow velocity u_{next} at the next time can be calculated as in (9) and (10).

III. Pneumatic reservoir computing with feedback loops

A. Feedback loops for physical reservoir

Hauser et al. showed that, theoretically, a nonlinear time-invariant system with arbitrary fading memory can be emulated by constructing a network of connected nonlinear spring-mass systems and using them as physical reservoirs [4]. Here, fading memory is the property that the state of the current reservoir is more strongly affected by inputs in the near past than by inputs in the far past. However, for learning periodic locomotion such as walking, not only fading memory but also persistent memory, in which the influence does not disappear over time, is an important factor. For example, in gait assistance, the average speed and foot swing generally remain unchanged over time. If this information can be retained in the reservoir, the risk that predicted speeds, angles and their associated state quantities deviate significantly from actual values can be reduced.

Maass et al. showed that any nonlinear dynamic system can be emulated by attaching appropriate static nonlinear feedback and readouts to nonlinear dynamic systems that can be feedback linearized [13]. Using this argument, Hauser et al. proposed to attach feedback loops and readouts to the physical reservoir [5]. Maass's argument assumed the installation of appropriate static, nonlinear feedback and readouts, whereas Hauser et al. argued that if the reservoir is sufficiently complex, the proper nonlinearization has already been done and a static linear feedback loop and readout alone can reproduce a dynamic nonlinear system. They have indeed successfully generated several limit cycles by linearly feeding back estimated information to a network consisting of a nonlinear spring-mass system, and have shown its robustness.

This study examined the possibility of improving reservoir performance by attaching linear feedback loops to the pneumatic physical reservoir.

B. Simulation design including confluence

In general, there is more than one output to be fed back. In this study, in order to feed back two outputs used in the generation of limit cycles, the tubes were branched as shown in Fig. 2(b), and the outputs were fed back to the two inlets.

The state at each point in the tube at a given time can basically be calculated numerically by the simulation described in Section 2, but special boundary conditions must be considered where the flows from the two inlets merge. This section describes these boundary conditions. The derivation of the boundary conditions is based on Sakamoto et al. [14]. In the study, the boundary conditions at branching

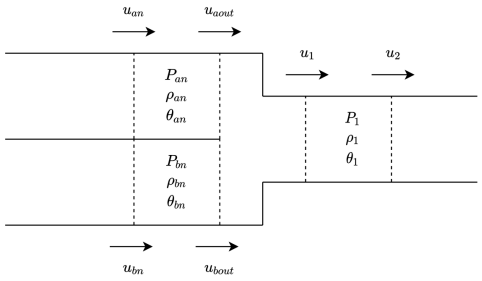

Fig. 4. Model of confluence at branching point of pipeline.

point of pipeline were limited to flow in one direction. In the present study, we consider flow in both directions so that both branching and confluence of flows can be handled. In order to consider the boundary conditions at the confluence, we consider a situation such as that shown in Fig. 4. For simplicity, the energy loss due to confluence is not considered.

First, we consider the update equation for the flow velocity u_1. Since u_1 is expressed by (14), it is sufficient to know the update equations for u_{aout} and u_{bout}.

$$u_{1,j+1} = u_{\mathrm{aout},j+1} + u_{\mathrm{bout},j+1}. \tag{14}$$

For u_{aout}, the following holds:

$$u_{\mathrm{aout},j+1} = u_{\mathrm{aout},j} - \frac{\Delta t}{\Delta x} u_{\mathrm{conv}} - \frac{1}{\bar{\rho}} \frac{\Delta t}{\Delta x} \left(P_{1,j} - P_{\mathrm{a}n,j}\right) - \frac{\lambda \Delta t}{2D} \left|u_{\mathrm{aout},j}\right| u_{\mathrm{aout},j}. \tag{15}$$

The convective term u_{conv} is expressed as follows:

$$u_{\mathrm{conv}} = \frac{u_{\mathrm{aout},j} + \left|u_{\mathrm{aout},j}\right|}{2} \left(u_{\mathrm{aout},j} - u_{n,j}\right) + \frac{u_{\mathrm{aout},j} - \left|u_{\mathrm{aout},j}\right|}{2} \left(u_{2,j} - u_{\mathrm{aout},j}\right). \tag{16}$$

Note that $u_{2,j}$, not $u_{1,j}$, is downwind of the second term. Similarly, since $u_{\mathrm{bout},j+1}$ is also obtained, $u_{1,j+1}$ is obtained from (14). In the case of u_{an} and $u_{\mathrm{b}n}$, the same equations as in the normal case without confluence can be used, but the downwind values are assumed to be $u_{\mathrm{aout},j}$ and $u_{\mathrm{bout},j}$ before confluence, respectively.

Next, we consider the update equation for the density ρ_1. Basically, it is the same as [10] described in Section 2, but when considering the upwind of the first term of the convective term ρ_{conv}, we should consider two sides, a and b. In this study, after calculating $\rho_{1,j+1}$ with a as upwind, $\rho_{1,j+1}$ is calculated with b as upwind, and $\rho_{1,j+1}$ is set to the average of the two.

$$\rho^{\mathrm{a}}_{\mathrm{conv}} = \frac{\bar{u}_{1,j} + \left|\bar{u}_{1,j}\right|}{2} \left(\rho_{1,j} - \rho_{\mathrm{a}n,j}\right) + \frac{\bar{u}_{1,j} - \left|\bar{u}_{1,j}\right|}{2} \left(\rho_{2,j} - \rho_{1,j}\right), \tag{17}$$

$$\rho^{\mathrm{a}}_{1,j+1} = \rho_{i,j} - \frac{\Delta t}{\Delta x} \rho_{1,j} \left(u_{2,j} - u_{1,j}\right) - \frac{\Delta t}{\Delta x} \rho^{\mathrm{a}}_{\mathrm{conv}}. \tag{18}$$

$$\rho^{\mathrm{b}}_{\mathrm{conv}} = \frac{\bar{u}_{1,j} + \left|\bar{u}_{1,j}\right|}{2} \left(\rho_{1,j} - \rho_{\mathrm{b}n,j}\right) + \frac{\bar{u}_{1,j} - \left|\bar{u}_{1,j}\right|}{2} \left(\rho_{2,j} - \rho_{1,j}\right), \tag{19}$$

$$\rho^{\mathrm{b}}_{1,j+1} = \rho_{i,j} - \frac{\Delta t}{\Delta x} \rho_{1,j} \left(u_{2,j} - u_{1,j}\right) - \frac{\Delta t}{\Delta x} \rho^{\mathrm{b}}_{\mathrm{conv}}. \tag{20}$$

$$\rho_{1,j+1} = \frac{\rho^{\mathrm{a}}_{1,j+1} + \rho^{\mathrm{b}}_{1,j+1}}{2}. \tag{21}$$

Finally, we consider the update equation for the temperature θ_1. This is basically the same as in [10], but as in the case of density ρ_1, there are two patterns, one on the a side and the other on the b side. we first calculate $\theta_{1,j+1}$ for each pattern, and then average the two values to obtain the final $\theta_{1,j+1}$. Note that not only the convective term θ_{conv}, but also the mean value of density $\bar{\rho}$, Reynolds number Re, friction coefficient λ, Nusselt number Nu, thermal conductivity h, etc. must be considered separately for a side and b side.

IV. Generation of limit cycle

Following the method in the previous section, feedback loops attached to a pneumatic physical reservoir was simulated to generate a limit cycle.

A. Limit cycle

This study deals with the Van der Pol equation. This is the governing equation for an oscillator with nonlinear damping as follows:

$$\frac{d^2x}{dt^2} - \mu\left(1 - x^2\right)\frac{dx}{dt} + x = 0. \tag{22}$$

where x is the position, t is the time, and μ is the magnitude of damping. Oscillators following the Van der Pol equation have limit cycles in phase space. If we let x_1 denote the position and x_2 denote the velocity, then (22) is rewritten as follows and converges to a closed orbit shown in Fig. 5. In this study, μ was set to 1. The limit cycle expressed by the Van der Pol equation is also treated in the limit cycle generation by the physical reservoir using the spring-mass system by Hauser et al. [5].

$$x_1' = x_2, \tag{23}$$

$$x_2' = -x_1 + (1 - x_1^2)x_2. \tag{24}$$

B. Generation algorithm

Limit cycles were generated by learning the weights of the output layer of the reservoir with pre-computed data, similar to Hauser et al. [5]. First, $\{x_1(k), x_2(k)\}_{k=1}^{k=1680}$ was prepared as training data. For the limit cycle study, about 50 cycles were prepared. Before learning, x_1 and x_2 were normalized to have the same range of values (Fig. 5). The model of the pneumatic pipeline system used as the pneumatic reservoir was in the form of Fig. 6, where P_s was varied according to $x_1(k)$ and P_f according to $x_2(k)$ as follows:

$$P_s = P_0(1 + 6 \times x_1(k)) = f(x_1(k)), \tag{25}$$

$$P_f = P_0(1 + 6 \times x_2(k)) = g(x_2(k)). \tag{26}$$

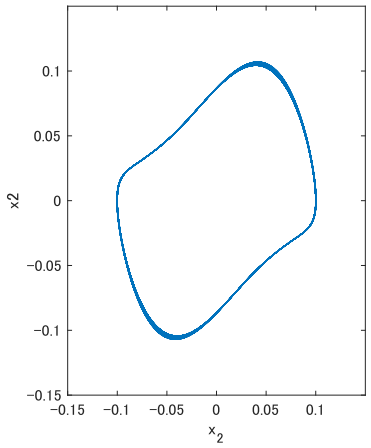

Fig. 5. Limit cycle of the Van der Pol equation. x_1 and x_2 are normalized to have the same range (see Section IV-B).

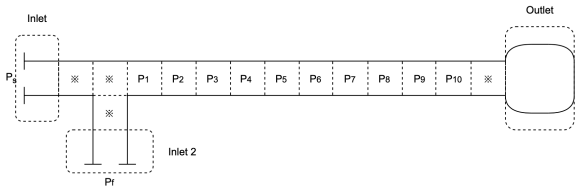

Fig. 6. Simulation model of pneumatic pipeline system used to generate limit cycles.

Here, P_0 is atmospheric pressure, which was set to 1.073×10^5 Pa. Values for other parameters are shown in Table I.

The algorithm is shown in Algorithm 1. The weights of the output layer were obtained by the recursive least squares method used by Caluwaerts et al. in their control of a tensegrity robot by reservoir computing [15]. The details of the algorithm is described below.

First, the prediction of the values of $x_1(k), x_2(k)$ was done by multiplying the pressure values $\boldsymbol{P}(k) = [P_1(k), P_2(k), \cdots, P_{100}(k)]^{\mathrm{T}}$, whose elements were 10 points in the tube at 10 different times by the weight $\boldsymbol{w}_1, \boldsymbol{w}_2$ (n-dimensional row vectors), respectively. This is the multiplexing technique used in a physical reservoir computing [8]. In learning using the recursive least squares method, the output was computed by a weighted average of the correct and predicted data. The weight of the correct data is assumed to decrease with learning. In this experiment, the weight α^{rls} of the correct data at time k was calculated as follows.

$$\alpha^{\mathrm{rls}} = 1 - \frac{k}{1680}. \tag{27}$$

The estimated output $\tilde{x}_1(k)$ at each time during learning was computed by a weighted average of the correct and predicted data as follows:

$$\tilde{x}_1(k) = \alpha^{\mathrm{rls}} x_1(k) + (1 - \alpha^{\mathrm{rls}}) \boldsymbol{w}_1(k) \boldsymbol{P}(k), \tag{28}$$

$$\tilde{x}_2(k) = \alpha^{\mathrm{rls}} x_2(k) + (1 - \alpha^{\mathrm{rls}}) \boldsymbol{w}_2(k) \boldsymbol{P}(k). \tag{29}$$

Based on the $\tilde{x}_1(k), \tilde{x}_2(k)$ calculated in this way, the inlet pressures P_s, P_f were varied according to (25) and (26) and the weights were updated as follows. Here, only the update of $\boldsymbol{w}_1$ is shown, but $\boldsymbol{w}_2$ was updated as well.

$$\boldsymbol{L}^{\mathrm{rls}}(k) = \frac{\boldsymbol{S}^{\mathrm{rls}}(k)\boldsymbol{P}(k)}{1 + \boldsymbol{P}(k)^{\mathrm{T}} \boldsymbol{S}^{\mathrm{rls}}(t) \boldsymbol{P}(k)}, \tag{30}$$

$$\boldsymbol{S}^{\mathrm{rls}}(k+1) = \boldsymbol{S}^{\mathrm{rls}}(k) - \frac{\boldsymbol{S}^{\mathrm{rls}}(k)\boldsymbol{P}(k)\boldsymbol{P}^T(k)\boldsymbol{S}^{\mathrm{rls}}(k)}{1 + \boldsymbol{P}^T(k)\boldsymbol{S}^{\mathrm{rls}}(k)\boldsymbol{P}(k)}, \tag{31}$$

$$e^{\mathrm{rls}}(k) = x_1(k) - \boldsymbol{w}_1(k-1)\boldsymbol{P}(k), \tag{32}$$

$$\boldsymbol{w}_1(k) = \boldsymbol{w}_1(k-1) + \boldsymbol{L}^{\mathrm{rls}}(k) e^{\mathrm{rls}}(k). \tag{33}$$

where $\boldsymbol{S}^{\mathrm{rls}}$ is the covariance matrix and the initial value was an n-dimensional identity matrix. The evaluation was performed by stopping the update of the weights $\boldsymbol{w}_1, \boldsymbol{w}_2$ and feeding back only the predictions with $\alpha^{\mathrm{rls}} = 0$.

TABLE I

PARAMETERS FOR THE SIMULATION

Parameter	Value
Sampling time Δt	0.00025 s
Length of each grid Δx	4 m
Inner diameter of the tube D	4 mm
Volume of the tank V	60 ml
Effective orifice area of the inlets	1.8 mm^2
Effective orifice area of the outlet	2 mm^2

Algorithm 1 Generating Limit Cycles Using Recursive Least Squares

%% Learning %%
Let $\boldsymbol{P}$ be $mN \times 1$ vector. Let $\boldsymbol{S}_1^{\mathrm{rls}}, \boldsymbol{S}_2^{\mathrm{rls}}$ be n dimensional identity matrix.
for $k = 1, 2, \cdots, T$ **do**
  $P_s \leftarrow f(\tilde{x}_1(k-1))$
  $P_f \leftarrow g(\tilde{x}_2(k-1))$
  for $i = 1, 2, \cdots, m$ **do**
    Wait for $\frac{0.125}{m}$ seconds
    $\boldsymbol{P}((i-1)N + 1 : iN) \leftarrow [P_1, P_2, \cdots, P_N]$
  end for
  $\alpha \leftarrow 1 - \frac{k}{T}$
  $\tilde{x}_1(k) \leftarrow \alpha^{\mathrm{rls}} x_1(k) + \left(1 - \alpha^{\mathrm{rls}}\right) \boldsymbol{w}_1(k) \boldsymbol{P}$
  $\tilde{x}_2(k) \leftarrow \alpha^{\mathrm{rls}} x_2(k) + \left(1 - \alpha^{\mathrm{rls}}\right) \boldsymbol{w}_2(k) \boldsymbol{P}$
  Update $\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{S}_1^{\mathrm{rls}}, \boldsymbol{S}_2^{\mathrm{rls}}$ using RLS.
end for
%% Evaluation %%
for $k = T_{\mathrm{test}}, T_{\mathrm{test}} + 1, \cdots, T_{\mathrm{test}} + \tau$ **do**
  $P_s \leftarrow f(\tilde{x}_1(k-1))$
  $P_f \leftarrow g(\tilde{x}_2(k-1))$
  for $i = 1, 2, \cdots, m$ **do**
    Wait for $\frac{0.125}{m}$ seconds
    $\boldsymbol{P}((i-1)N + 1 : iN) \leftarrow [P_1, P_2, \cdots, P_N]$
  end for
  $\tilde{x}_1(k) \leftarrow \boldsymbol{w}_1(k) \boldsymbol{P}$
  $\tilde{x}_2(k) \leftarrow \boldsymbol{w}_2(k) \boldsymbol{P}$
end for

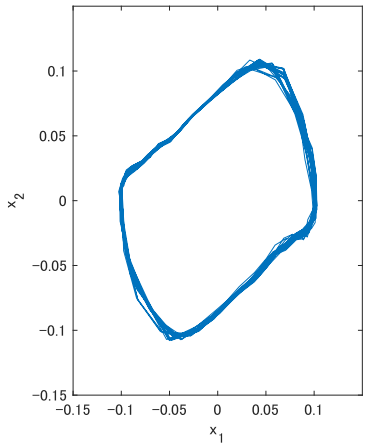

Fig. 7. Generated limit cycle after learning.

C. Generation from different initial states

In the previous section, we discussed the case where, after learning by the recursive least squares method, we stopped learning and started evaluation as is. This means that the evaluation was started from a point (x_1, x_2) near the target limit cycle. In this study, we also examined the trajectory generated when the evaluation was started from a point far from the target limit cycle. A total of nine cases were validated, starting with initial conditions of $x_1 = -0.1, 0$ or 0.1 and $x_2 = -0.1, 0$ or 0.1.

V. Results

First, the 25 cycles generated by feeding back only the prediction data, which started from a point near the target limit cycle at the time of evaluation, are shown in Fig 7. Although there was some variation in the trajectory due to prediction errors, it can be seen that the overall motion generally generated the target limit cycle.

Then, the cycles generated starting from states outside the limit cycle are shown in Fig. 8. Except for the case starting from $(x_1, x_2) = (0.1, 0.1)$, the limit cycles were inaccurate right at the start, but began to produce highly accurate limit cycles within approximately 10 seconds. When starting from $(x_1, x_2) = (0.1, 0.1)$, the state diverged.

VI. Discussion

From Fig. 7, we confirmed that the limit cycle can be generated without deviating significantly from the target limit cycle. Furthermore, Fig. 8 show that even when starting from a point far from the target limit cycle, in many cases the state approached the target limit cycle and generated limit cycles with high accuracy. In the process, the output of the pneumatic physical reservoir returned to the target trajectory even though it passes through a point further away from the target limit cycle, suggesting that it can return to its original trajectory even when a large disturbance is introduced. This is an important property for applying this method to actual equipment that is constantly subjected to disturbances. One

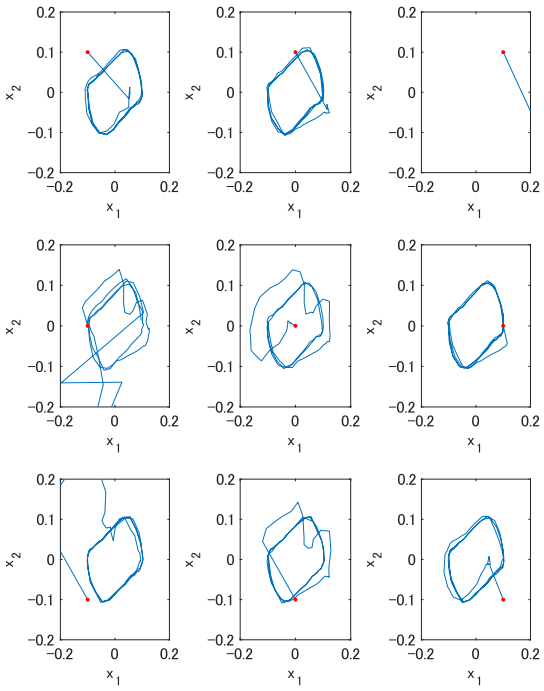

Fig. 8. Output trajectories generated from states outside of the limit cycle. The red dots indicate the starting states.

way to reduce cases of divergence is to add noise to the training data to obtain more robust parameters.

The limit cycle generation in this study has the following four significance.

- A limit cycle was generated using only the pressure information in a relatively simple pneumatic tube.
- Predictions of two different output x_1, x_2 were used. This can be said to take advantage of the multitasking nature of the physical reservoir [8].
- The time-dependent input-output relationship was learned with static parameters that do not change with time, as expressed in (22).
- Limit cycles robust to large disturbances were generated.

The generated limit cycle generally follows the target limit cycle, but there are some areas where the trajectory is disrupted, such as in the region where x_2 is near its maximum value. The complexity of the pneumatic pipeline system is expected to be able to generate these parts with higher accuracy. For example, the number of inlets on the feedback side can be increased and the distance between the inlets can be increased as a method of increasing the complexity of the pipeline system. As a future prospect, it is hoped that a reservoir incorporating feedback loops will be applied in soft exoskeleton control using pneumatic reservoir computing to provide stable assistance that is resistant to external disturbances.

REFERENCES

[1] Kohei Nakajima and Ingo Fischer (eds.) Reservoir Computing: Theory, Physical Implementations, and Applications. Springer, 2021.

[2] Herbert Jaeger and Harald Haas, Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication, Science, vol. 304, no. 5667, pp. 78-80, 2004.

[3] Wolfgang Maass, Thomas Natschläger and Henry 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, 2002.

[4] Helmut Hauser, Auke J. Ijspeert, Rudolf M. Füchslin, Rolf Pfeifer and Wolfgang Maass, Towards a Theoretical Foundation for Morphological Computation with Compliant Bodies, Biological Cybernetics, vol. 105, pp. 355-370, 2011.

[5] Helmut Hauser, Auke J. Ijspeert, Rudolf M. Füchslin, Rolf Pfeifer and Wolfgang Maass, The Role of Feedback in Morphological Computation with Compliant Bodies, Biological Cybernetics, vol. 106, pp. 595-613, 2012.

[6] Laurent Larger, Antonio Baylón-Fuentes, Romain Martinenghi, Vladimir S. Udaltsov, Yanne K. Chembo and Maxime Jacquot, High-Speed Photonic Reservoir Computing Using a Time-Delay-Based Architecture: Million Words per Second Classification, PHYSICAL REVIEW X, vol. 7, pp. 011015, 2017.

[7] Lennert Appeltant, Miguel Soriano, Guy Van der Sande, Jan Danckaert, Serge Massar, J. Dambre, Benjamin Schrauwen, Claudio Mirasso, Ingo Fischer, Information Processing Using a Single Dynamical Node as Complex System, Nature Communications, vol. 2, pp. 468, 2011.

[8] Kohei Nakajima, Helmut Hauser, Tao Li, and Rolf Pfeifer, Exploiting the Dynamics of Soft Materials for Machine Learning, Soft Robotics, vol. 5, no. 3, pp. 339-347, 2018.

[9] Jacob Torrejon, Mathieu Riou, Flavio Abreu Araujo, Sumito Tsunegi, Guru Khalsa, Damien Querlioz, Paolo Bortolotti, Vincent Cros, Kay Yakushiji, Akio Fukushima, Hitoshi Kubota, Shinji Yuasa, Mark D. Stiles and Julie Grollier, Neuromorphic Computing with Nanoscale Spintronic Oscillators, Nature, vol. 547, pp. 428-431, 2017.

[10] Toshihiro Kawase, Tetsuro Miyazaki, Takahiro Kanno, Kotaro Tadano, Yoshikazu Nakajima and Kenji Kawashima, Pneumatic Reservoir Computing for Sensing Soft Body: Computational Ability of Air in Tube and Its Application to Posture Estimation of Soft Exoskeleton, Sensors and Materials, vol. 33, no. 8, pp. 2803-2824, 2021.

[11] Hiroyuki Hayashi, Toshihiro Kawase, Tetsuro Miyazaki, Maina Sogabe, Yoshikazu Nakajima and Kenji Kawashima, Online Assistance Control of a Pneumatic Gait Assistive Suit Using Physical Reservoir Computing Exploiting Air Dynamics. In Proceedings of 2022 IEEE International Conference on Robotics and Automation (ICRA), 2022, pp. 3245-3251.

[12] Jun Li, Kenji Kawashima, Toshinori Fujita and Toshiharu Kagawa, Control Design of a Pneumatic Cylinder with Distributed Model of Pipelines, Precision Engineering, vol. 37, no. 4, 880-887, 2013.

[13] Wolfgang Maass, Prashant Joshi, Eduardo D Sontag, Computational Aspects of Feedback in Neural Circuits, PLOS Computational Biology, vol. 3, no. 1, Article e165, 2007.

[14] Daisuke Sakamoto, Chongho Youn and Toshiharu Kagawa, Pressure Change in Tee Branch Pipe in Oscillatory Flow, vol. 5, Article 257283, 2013.

[15] Ken Caluwaerts, Michiel D'Haene, David Verstraeten and Benjamin Schrauwen, Locomotion Without a Brain: Physical Reservoir Computing in Tensegrity Structures, Artificial life, vol. 19, no. 1, pp. 35-66, 2013.