Physical Reservoir Computing with Origami – A Feasibility Study

Priyanka Bhovad and Suyi Li

Department of Mechanical Engineering, Clemson University, Clemson, SC, US

ABSTRACT

In the field of soft robotics, harnessing the nonlinear dynamics of soft and compliant bodies as a computational resource to enable embodied intelligence and control is known as *morphological computation.* Physical reservoir computing (PRC) is a true instance of morphological computation wherein; a physical nonlinear dynamic system is used as a fixed reservoir to perform complex computational tasks. These dynamic reservoirs can be used to approximate nonlinear dynamical systems and even perform machine learning tasks. By numerical simulation, this study illustrates that an origami meta-material can also be used as a dynamic reservoir for pattern generation, output modulation, and input sensing. These results could pave the way for intelligently designed origami-based robots that interact with the environment through a distributed network of sensors and actuators. This embodied intelligence will enable the next generations of soft robots to autonomously coordinate and modulate their activities, such as locomotion gait generation and limb manipulation while resisting external disturbances.

Keywords: Physical reservoir computing, origami, morphological computation, soft robotics

1. INTRODUCTION

In traditional robotics, rigid bodied robots are favored over soft or compliant ones, because rigid robots can be easily controlled to a high degree of accuracy and possess a highly predictable deformation pattern. In contrast, soft-bodied robots are difficult to control, as they possess virtually infinite kinematic degrees of freedom and require highly nonlinear dynamic equations to describe their large deformation patterns. To address this challenge, the idea of *morphological computation*[1,2] was conceived. An increasing number of researchers are now embracing the *undesirable* nonlinear dynamics of soft and compliant bodies as computational resource to effectively create an embodied intelligence and control in soft robots.[2–6] Thus, a new computational paradigm has emerged in which the physical body of the robot itself takes part in performing control tasks such as co-ordination and modulation of locomotion gait, simplifying the central controller architecture. For example,Paul investigated the link between morphology and control for biped locomotion and used morphology as a basis to develop tensegrity robots with reduced control requirements.[1,2] Taking this concept even further, Hauser et al.[4,7] proved that a generic recurrent network of a spring-mass-damper system with nonlinear spring and damper properties can be used as a morphological computation device. Caluwaerts et al.[3] further showed that geometric nonlinearity of tensegrity structures can be the basis of a dynamic physical reservoir without the need for nonlinear springs and dampers. Researchers have shown that physical dynamical systems can be used as computational reservoirs to perform complex computation tasks such as approximating nonlinear dynamical systems,[3–6] pattern generation,[3–6] generating and controlling robot locomotion gait,[3,6] and even for speech recognition[8] and machine learning.[6,9,10] This paradigm has become known as *Physical Reservoir Computing.*

Reservoir computing (RC) is a computational framework based on artificial recurrent neural networks (RNNs). RNNs are extensively applied to the problems involving time-series prediction, such as stock market, weather forecasting, robot gait and/or manipulator motion planning, robotic control, text and speech recognition. A recurrent neural network is an artificial neural network where the output of the current time step depends on the output of the previous time step, in addition to the current input. Since RNNs involve forward as well as the back-propagation of input data, training them became a very difficult task. To make the training of RNNs

Further author information: (Send correspondence to P.B.)
P.B.: E-mail: pbhovad@clemson.edu
S.L.: E-mail: suyil@clemson.edu

easier, the concept of a fixed recurrent neural network was independently introduced by Jaeger[11] as Echo State Networks (ESNs) and Maass[12] as Liquid State Machines (LSMs). In this setting, a randomly generated neural network with fixed interconnections and input weights is used, and only the output readout weights are updated using simple techniques such as linear or ridge regression. The network dynamics are governed by nonlinear dynamical equations, thus it transforms the input data stream into a high-dimensional state-space that can capture the nonlinearities and time-dependent information in the input stream. This neural network is called a *reservoir* as it remains fixed throughout the computation and this field of research has become known as Reservoir Computing (RC).

Inspired by the advances in physical reservoir computing; we propose that origami meta-material can function as a dynamic physical reservoir. Origami is an ancient art of paper folding that enables the creation of complex 3-dimensional, compliant, and dynamic mechanisms from a single sheet of material. The scale-free nature of origami means that the same design principles are applicable over large size ranges. We show that the nonlinear geometry and mechanics of origami is ideal for RC applications. We envision that the reservoir computing capability of the origami reservoirs can foster a new family of origami-based soft robots that feature simple control architecture, interact with the environment through distributed sensor and actuator network, and can coordinate and modulate their activities while resisting external disturbances.

In what follows: Section (2) details the lattice framework used to simulate the origami's nonlinear dynamics. Section (3) details the reservoir computing framework applied to origami. Section 4 elucidates the origami reservoir's computing power through various numerical experiments. Finally, Section 5 concludes this paper with a summary and discussion.

2. ORIGAMI DYNAMICS

We adopt the lattice framework approach to simulate the origami meta-material dynamics.[13, 14] This approach represents a crease as pin-jointed stretchable truss element with prescribed stretching spring stiffness K_s and simulates the folding motion in origami by assigning creases with torsional spring stiffness K_b. We triangulate the quadrilateral facets with additional truss elements to approximate the facet stretching and bending, and the bending stiffness of facets is set higher than that of folding creases. Thus, we discretize the continuous origami sheet into pin-jointed truss elements connected at the vertices (or the nodes) of origami. We write the dynamic equations of motion in terms of nodal displacements in Cartesian coordinate system. The external forces can be directly applied at any node. Additionally, we can apply external moment directly at the crease to facilitate folding and unfolding. The corresponding governing equations of motion, in terms of node #p's displacement ($\mathbf{x}_p$) as an example, are:

$$m_p \ddot{\mathbf{x}}_p^{(j)} = \mathbf{F}_p^{(j)}, \tag{1}$$

where the superscript "(j)" represents the j^{th} time step in numerical simulation, and m_p is the equivalent nodal mass. In this study, the mass of the origami sheet is assumed to be equally distributed to all its nodes. $\mathbf{F}_p^{(j)}$ is the summation of internal and external forces acting on this node in that

$$\mathbf{F}_p^{(j)} = \sum \mathbf{F}_{s,p}^{(j)} + \sum \mathbf{F}_{b,p}^{(j)} + \mathbf{F}_{d,p}^{j} + \mathbf{F}_{a,p}^{(j)} + m_p \mathbf{g}, \tag{2}$$

where, the summation includes all the trusses and creases that connect this node #p. The five terms on the right hand side are the forces from truss stretching, crease/facet bending, equivalent damping, external actuation, and gravity, respectively. The formulation of these forces are detailed in Appendix A.

3. RESERVOIR COMPUTING FRAMEWORK

There are two types of actuated creases in the origami reservoir. The first type is "input creases," and they receive external input signal $u(t)$. The second type is "feedback creases," and they receive reference or current output signal $z(t)$. In the case of multiple outputs, different groups of feedback creases are present. The input and feedback creases are randomly selected for the current computation tasks, but they can also be strategically placed depending on the desired soft robotic behaviors. We use the dihedral crease angles $\varphi^{(j)}$ as the *reservoir state* variables to characterize the origami's time responses. We designate these creases as "sensor creases" for

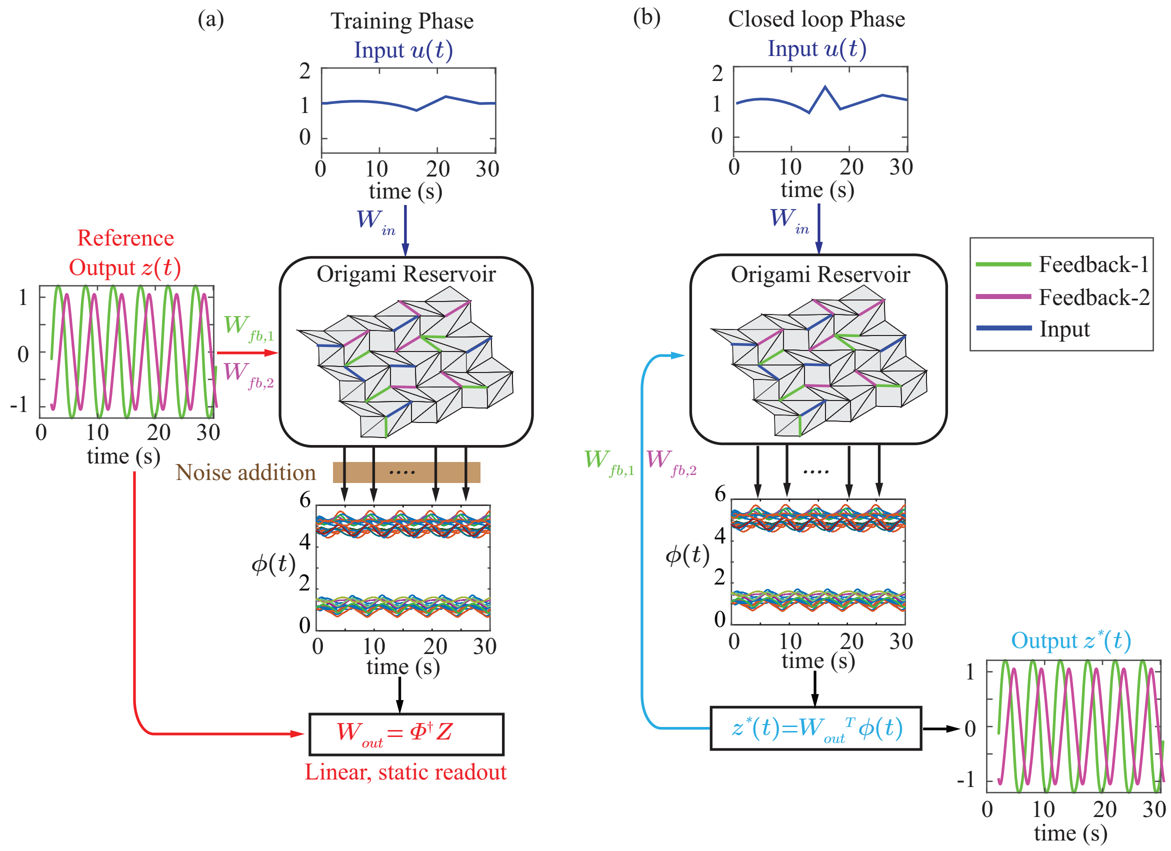

Figure 1. Physical reservoir computing with Miura-Ori meta-material. (a) Training phase for reservoir computing. Feedback creases receive reference signal. White noise is added to the state vector $x(t)$ during the training before calculating output weights W_{out}; (b) Closed-loop phase for reservoir computing. The output weights calculated in training phase are used to calculate current output $z^*(t)$ and it is fed back to the feedback creases. The feedback creases are depicted in magenta and green and the input creases are depicted in blue.

measuring the reservoir states. We denote N_a as the number of actuated creases, and N_s for sensor creases. The actuated creases are typically small subset of all origami creases (i.e., $N_a < N$). The sensor creases, on the other hand, can be all of the origami creases ($N_s = N$) or a small subset as well ($N_s < N$).

Once the selections of input, feedback, and sensor creases are completed, one can proceed to the computing. Physical reservoir computing for tasks that require feedback consists of two phases: The "training phase" and "closed-loop phase." While the emulation and sensing tasks require the training phase only.

Training phase: In this phase, we use the teacher forcing to obtain the readout weights W_{out} corresponding to each reservoir state (aka. the dihedral angles of the sensor creases). Suppose one wants to train the reservoir to generate a nonlinear time series $z(t)$ (aka. the reference output). The feedback creases receive the reference output and it dynamically excites the origami reservoir under an open-loop condition without feedback (Figure 1(a)). The reservoir states $\varphi^{(j)}$ at every time step are measured and then compiled into a matrix $\mathbf{\Phi}$.

Once the numerical simulation is over, we segregate the reservoir state matrix $\mathbf{\Phi}$ into the washout step, training step, and testing (validation) step. The washout step data is discarded to eliminate the initial transient responses. We then calculate the output readout weights W_{out} using the training step data by linear regression:

$$\mathbf{W}_{\text{out}} = [\mathbf{1}\ \mathbf{\Phi}]^{+}\mathbf{Z} = \bar{\mathbf{\Phi}}^{+}\mathbf{Z} \tag{3}$$

where, $[.]^{+}$ refers to the Moore-Penrose pseudo-inverse to accommodate non-square matrix. $\mathbf{1}$ is a column of ones for calculating the bias term $W_{\text{out},0}$ to shift the fitted function when necessary. $\mathbf{Z}$ contains the reference

Table 1. Design of the origami reservoir in this study

Reservoir size and material properties	
Parameter	Value
Size ($x_N \times y_N$)	9×9
Nodal Mass	10 g
EA	100 N
$k_{b,a}$	1 N/(m-rad)
k_b (facet folding)	1 N/(m-rad)
k_b (facet bending)	10 N/(m-rad)
ζ	0.2
Geometric design of Miura-Ori	
Parameter	Value
a	11 mm
b	10 mm
γ	50°
θ	60°
Actuator and sensor creases	
Parameter	Value
No. of sensors (N_s)	N
No. of actuators (N_a)	$0.39N$
No. of Feedback creases	$0.3N$
No. of Input creases	$0.09N$

signals at each time step, and it is a matrix if more than one references present. Lastly, we use testing step data to verify reservoir performance. A white noise of amplitude 10^{-3} is superimposed on the reservoir state matrix during training to ensure the robustness of the readout result against numerical imperfections, external perturbations, and measurement noise in practical applications.[7]

Closed-loop phase: Once the training phase is over and readout weights are obtained, we run the reservoir in the closed-loop condition. That is, instead of using the reference output $z(t)$, the current output $z^*(t)$ is calculated as a weighted linear combination of the reservoir state vector, and is sent to the feedback creases (Figure 1(b))

$$z^*(t) = W_{\text{out},0} + \sum_{i=1}^{N_s} W_{\text{out},i}\varphi_i(t) = \mathbf{W}_{out}^T \bar{\mathbf{\Phi}} \tag{4}$$

where, N_s is the number of sensor creases, and $\bar{\mathbf{\Phi}} = [\mathbf{1}\ \mathbf{\Phi}]$. Thus, the reservoir runs autonomously in the closed-loop phase without any external interventions.

4. PHYSICAL RESERVOIR COMPUTING WITH ORIGAMI

In this section, we show that indeed an origami meta-material can be used as a dynamic reservoir for physical reservoir computing. Our origami reservoir is composed of Miura-Ori, which is perhaps the most famous of all origami patterns. It consists of periodic tessellations of a unit cell composed of four identical parallelograms of sides a,b and internal acute angle γ (Figure 1). The folded geometry can be completely described with a single dihedral folding angle $\theta \in [-\pi/2, \pi/2]$ between two adjacent facets. The number of straight creases in x-direction is y_N and the number of zig-zag creases in the y-direction is x_N.

4.1 Output modulation and input sensing with Origami reservoir

In this section, we demonstrate the computational capability of our origami reservoir. We have two feedback/output signals for the limit cycle generation and one external input to modulate (or switch) the limit

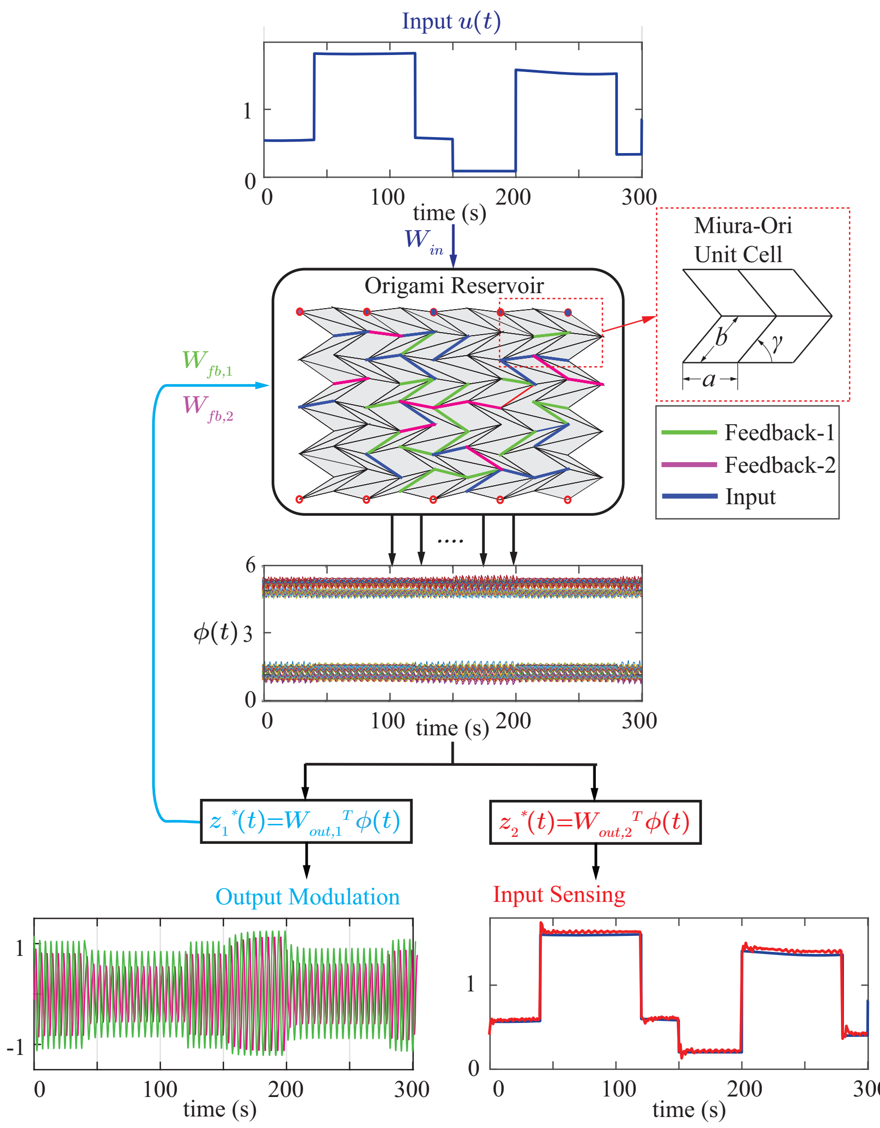

Figure 2. Output modulation and input sensing with origami reservoir. Quadratic limit cycle generation under changing input u(t). $W_{\mathrm{out},1}$ and $W_{\mathrm{out},2}$ are the readout weights for output modulation task and input sensing task, respectively. A sample closed loop trajectory from output modulation experiment is shown in green and magenta. Sensed input (in red) is overlaid with external input u(t) (in blue). Miura-Ori unit cell geometry and the design parameters are shown in inset.

cycles. We randomly choose 15% of the total creases for each feedback signals, and 9% of the total creases as input creases. The selected crease distributions are kept fixed throughout the experiment. In this study, all creases are used as sensor creases ($N_s = N$). Similarly, we randomly initialize input and feedback weights and keep them fixed throughout the experiment. The geometric parameters of the origami reservoir are summarized in table 1. In the following results, we demonstrate the computational power of origami reservoir via generation of the quadratic limit cycle. It is defined as solution to the ODE given by,

$$\dot{x_1} = x_1 + x_2 - u(t)x_1(x_1^2 + x_2^2) \tag{5}$$

$$\dot{x_2} = -2x_1 + x_2 - x_2(x_1^2 + x_2^2) \tag{6}$$

where, parameter $u(t) > 0$ determines the shape of the limit cycle.

In this experiment, we train a single reservoir to find two sets of readout weights $W_{\text{out},1}$ and $W_{\text{out},2}$, for output modulation task and input sensing task, respectively:

Output modulation: First we train the reservoir for 500s and find readout weights $W_{\text{out},1}$. We discard first 30s of data as washout step and then select next 300s data as training set and rest of the 170s of data as test set. Finally, we use readout weights $W_{\text{out},1}$ and response state vector $\varphi(t)$ to find the current output $z^*(t)$. In this phase we feed $z^*(t)$ back to the reservoir feedback creases to generate the quadratic limit cycle with changing input autonomously under closed loop. This capability is important for autonomous soft robots as it allows the robot to switch its behavior in response to the varying external input or environmental conditions, without changing the output readout weights.

Input Sensing: Afterwards, we sense the unknown input applied at input creases through an additional readout $W_{\text{out},2}$. This process takes place under open loop, i.e. we only train the readout using closed loop phase data from output modulation task.

As shown in figure 2, we see that this reservoir can succesfully generate a limit cycle and switch(modulate) it in response to changing external input. Moreover, the reservoir can sense/measure this input via an additional readout, with no internal changes to the configuration.

5. SUMMARY AND DISCUSSION

In this study, we demonstrated physical reservoir computing power of origami. First, we developed nonlinear truss-based framework to simulate origami dynamics. Next, we setup the physical reservoir computing framework for origami and finally, applied it to perform nonlinear computational tasks. We successfully showed that origami reservoir can perform multiple computing tasks simultaneously via an example of 1) quadratic limit cycle generation, 2) output switching in response to external input, and 3) input sensing. We only need to add multiple output readouts to perform these tasks. We use same response state vector to autonomously generate closed loop trajectory and predict the external input.

This implementation opens up an exciting chapter in the design of soft robots with enhanced collaboration between robot body and brain. The robot body itself is used to perform complex nonlinear computations while keeping the central controller configuration simple. Reservoir computing embodied origami can be applied for tasks such as shape morphing, locomotion gait generation, manipulator motion. Reservoir computing makes it possible to control few elements of the soft robot to generate a global target behavior instead of controlling every element precisely. This is in contrast to traditional rigid bodied robotics, where the motion of each element has to be precisely controlled to perform a task. In future, physical reservoir computing embodied origami will lead to mechanically intelligent soft robots that can work autonomously in unstructured dynamic environments via the distributed network of actuators and sensors.

APPENDIX A. ORIGAMI DYNAMICS MODELING

This section details the internal and external forces acting on the origami in truss-frame model.

Truss stretching forces: The truss elements are essentially elastic springs with axial stretching stiffness ($K_s^{(j)} = EA/l^{(j)}$). Here, EA is the material constant, and $l^{(j)}$ is the truss element's length at the current j^{th}

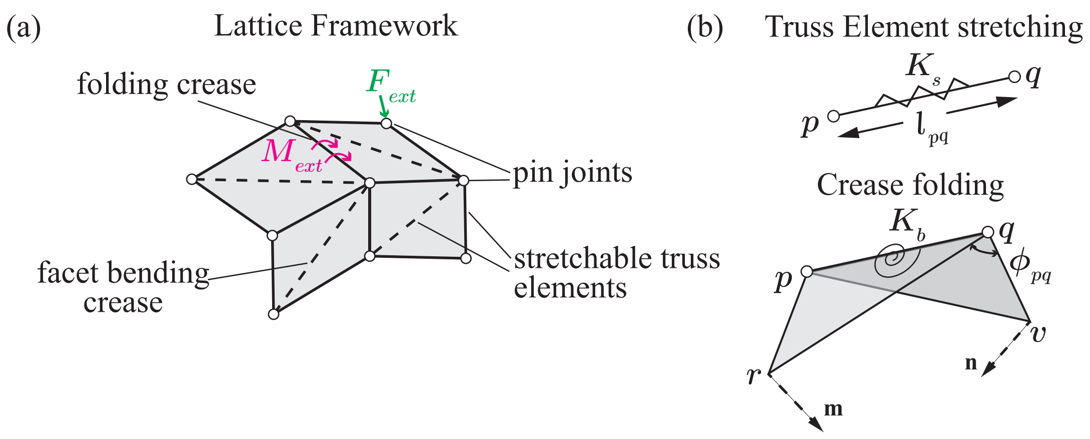

Figure 3. The nonlinear truss-frame model for simulating the origami dynamics. (a) The discretized Miura-Ori unit cell, showing the distribution of truss elements along the creases and across the facets, as well as the nodal masses. (b) Kinematics and mechanics set up to analyze the stretching and bending along the truss $\#pq$. The current surface normal vectors defined by triangles $\#pqr$ and $\#pqv$ are shown as $\mathbf{m}$ and $\mathbf{n}$, respectively.

time step. Thus, the axial stiffness is updated at each time-step, accommodating the truss element's increase in stiffness as it is compressed and vice-a-versa. The stretching forces from a truss connecting node #p and one of its neighboring nodes $\#q$ is,

$$\mathbf{F}_{s,p}^{(j)} = -K_s^{(j)} \left(l_{pq}^{(j)} - l_{pq}^{(0)} \right) \frac{\mathbf{r}_p^{(j)} - \mathbf{r}_q^{(j)}}{|\mathbf{r}_p^{(j)} - \mathbf{r}_q^{(j)}|} \tag{7}$$

where $l_{pq}^{(0)}$ is the truss length at its initial resting state. $\mathbf{r}_p^{(j)}$ and $\mathbf{r}_q^{(j)}$ are the current position vectors of these two nodes, respectively.

Crease/facet bending forces: The crease folding and facet bending are simulated with torsional spring coefficient ($K_b^{(j)} = k_b l^{(j)}$), where k_b is torsional stiffness *per unit length*. Here, we adopt the formulation developed by Liu and Paulino.[14] For example, the force acting on nodes $\#p$ due to the crease folding along the truss between $\#p$ and $\#q$ is:

$$\mathbf{F}_{b,p}^{(j)} = -K_b^{(j)} (\varphi_{pq}^{(j)} - \varphi_{pq}^{(0)}) \frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_p^{(j)}} \tag{8}$$

where $\varphi_{pq}^{(j)}$ is the current dihedral angle along truss pq, and $\varphi_{pq}^{(0)}$ is the corresponding initial value. $\varphi_{pq}^{(j)}$ can be calculated as

$$\varphi_{pq}^{(j)} = \eta \arccos \left(\frac{\mathbf{m}^{(j)} \cdot \mathbf{n}^{(j)}}{|\mathbf{m}^{(j)}||\mathbf{n}^{(j)}|} \right) \text{ modulo } 2\pi \tag{9}$$

$$\eta = \begin{cases} \operatorname{sign}\left(\mathbf{m}^{(j)} \cdot \mathbf{r}_{pv}^{(j)} \right), & \mathbf{m}^{(j)} \cdot \mathbf{r}_{pv}^{(j)} \neq 0 \\ 1. & \mathbf{m}^{(j)} \cdot \mathbf{r}_{pv}^{(j)} = 0 \end{cases} \tag{10}$$

Here, $\mathbf{m}^{(j)}$ and $\mathbf{n}^{(j)}$ are current surface normal vector of the triangles $\#pqr$ and $\#pqv$, respectively, in that $\mathbf{m}^{(j)} = \mathbf{r}_{rq}^{(j)} \times \mathbf{r}_{pq}^{(j)}$ and $\mathbf{n}^{(j)} = \mathbf{r}_{pq}^{(j)} \times \mathbf{r}_{pv}^{(j)}$. In addition, $\mathbf{r}_{pq}^{(j)} = \mathbf{r}_p^{(j)} - \mathbf{r}_q^{(j)}$, $\mathbf{r}_{rq}^{(j)} = \mathbf{r}_r^{(j)} - \mathbf{r}_q^{(j)}$, and $\mathbf{r}_{pv}^{(j)} = \mathbf{r}_p^{(j)} - \mathbf{r}_v^{(j)}$. This definition of $\varphi_{pq}^{(j)}$ ensures that the folding angle for valley crease lies in $(0, \pi]$ and the folding angle for mountain crease lies in $(\pi, 2\pi]$. The derivative between folding angle $\varphi_{pq}^{(j)}$ and the nodal $\#p$'s current position vector is

$$\frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_p^{(j)}} = \left(\frac{\mathbf{r}_{pv}^{(j)} \cdot \mathbf{r}_{pq}^{(j)}}{|\mathbf{r}_{pq}^{(j)}|^2} - 1 \right) \frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_v^{(j)}} - \left(\frac{\mathbf{r}_{rq}^{(j)} \cdot \mathbf{r}_{pq}^{(j)}}{|\mathbf{r}_{pq}^{(j)}|^2} \right) \frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_r^{(j)}} \tag{11}$$

where

$$\frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_r^{(j)}} = \frac{|\mathbf{r}_{pq}^{(j)}|}{|\mathbf{m}^{(j)}|^2}\mathbf{m}^{(j)}, \tag{12}$$

$$\frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_v^{(j)}} = -\frac{|\mathbf{r}_{pq}^{(j)}|}{|\mathbf{n}^{(j)}|^2}\mathbf{n}^{(j)}. \tag{13}$$

Damping forces: Estimating damping ratio and damping force is essential to achieve realistic dynamic responses and reduce numerical simulation error accumulation. In this study, we follow the formulation developed in.[13,15] This formulation first calculates the average velocity of a node with respect to its neighboring nodes ($\mathbf{v}_{\text{avg}}^{(j)}$) to effectively remove the rigid body motion components from the relative velocities and ensure that these components are not damped. Then damping force $\mathbf{F}_{d,p}^{(j)}$ applied on node $\#p$ is given by

$$\mathbf{F}_{d,p}^{(j)} = -c_d^{(j)}(\mathbf{v}_p^{(j)} - \mathbf{v}_{\text{avg}}^{(j)}) \tag{14}$$

$$c_d^{(j)} = 2\zeta\sqrt{K_s^{(j)} m_p} \tag{15}$$

where $c_d^{(j)}$ is the equivalent damping coefficient, and ζ is the damping ratio.

Actuation force: There are many methods to implement actuation to deliver input $u(t)$ and reference/feedback signal $z(t)$ to the reservoir. For example, the actuation can take the form of nodal forces on a mass-spring-damper network,[4,7] motor generated base rotation on octopus-inspired soft arm,[5] or spring resting length changes in a tensegrity structure.[3] For origami, in addition to these modes of actuation, the actuation can take the form of moments that can fold or unfold the selected creases. In this study, We assume that the resting angle $\varphi^{(0)}$ of the input and feedback creases will change — in response to the actuation at every time step — to a new equilibrium $\varphi_{a,0}^{(j)}$ in that[3,16]

$$\varphi_{a,0}^{(j)} = W_{\text{in}} \tanh(u^{(j)}) + \varphi^{(0)} \quad \text{for input creases;} \tag{16}$$

$$\varphi_{a,0}^{(j)} = W_{\text{fb}} \tanh(z^{(j)}) + \varphi^{(0)} \quad \text{for feedback creases.} \tag{17}$$

where W_{in} and W_{fb} are the input and feedback weight associated with these actuated creases. They are assigned before the training and remain unchanged after that. $u^{(j)}$ and $z^{(j)}$ are the input and feedback signal at the j^{th} time step. The magnitude of W_{in} and W_{fb} are selected such that $\varphi_{a,0}^{(j)} \in [0, 2\pi]$ and consistent with the folding angle assignment. This approach of assigning new equilibrium folding angles is similar to traditional neural network studies that use tanh as a nonlinear activation function to transform function $z(t)$ into a new one with magnitudes between $[-1, 1]$. Additionally, it prevents actuator saturation due to spurious extreme values of $z(t)$. Denote the torsional stiffness of actuated creases by $K_{b,a}^{(j)}$, and we can update Equation (8) for the actuated creases (using node $\#p$ as an example)

$$\mathbf{F}_{a,p}^{(j)} = -K_{b,a}^{(j)}\left(\varphi_{pq}^{(j)} - \varphi_{a,0,pq}^{(j)}\right)\frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_p^{(j)}}, \tag{18}$$

The calculation of other terms in this equation are the same as those in the force from crease folding and facet bending. Once the governing equations of motion are formulated, they are solved using MATLAB's `ode45` solver with 10^{-3} second time-steps.

ACKNOWLEDGMENTS

The authors acknowledge the support from the National Science Foundation (CMMI – 1933124), as well as the Clemson University for the generous allotment of computing time on Palmetto cluster.

REFERENCES

[1] Paul, C., *Investigation of Morphology and Control in Biped Locomotion*, PhD thesis, University of Zurich, Switzerland (2004).

[2] Paul, C., "Morphological computation. A basis for the analysis of morphology and control requirements," *Robotics and Autonomous Systems* **54**(8), 619–630 (2006).

[3] Caluwaerts, K. and Schrauwen, B., "The body as a reservoir: locomotion and sensing with linear feedback," *2nd International Conference on Morphological Computation* , 45–47 (2011).

[4] Hauser, H., Ijspeert, A. J., Füchslin, R. M., Pfeifer, R., and Maass, W., "Towards a theoretical foundation for morphological computation with compliant bodies," *Biological Cybernetics* **105**(5-6), 355–370 (2011).

[5] Nakajima, K., Hauser, H., Kang, R., Guglielmino, E., Caldwell, D. G., and Pfeifer, R., "A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm," *Frontiers in Computational Neuroscience* **7**(July), 1–19 (2013).

[6] Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., and Hirose, A., "Recent advances in physical reservoir computing: A review," *Neural Networks* **115**, 100–123 (2019).

[7] Hauser, H., Ijspeert, A. J., Füchslin, R. M., Pfeifer, R., and Maass, W., "The role of feedback in morphological computation with compliant bodies," *Biological Cybernetics* **106**(10), 595–613 (2012).

[8] Fernando, C. and Sojakka, S., "Pattern recognition in a bucket," in [*Advances in Artificial Life*], Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., and Kim, J. T., eds., 588–597, Springer Berlin Heidelberg, Berlin, Heidelberg (2003).

[9] Nakajima, K., Hauser, H., Li, T., and Pfeifer, R., "Exploiting the dynamics of soft materials for machine learning," *Soft Robotics* **5**(3), 339–347 (2018).

[10] Nakajima, K., "Physical reservoir computing-an introductory perspective," *Japanese Journal of Applied Physics* **59**(6) (2020).

[11] Jaeger, H., "The *"echo state"* approach to analysing and training recurrent neural networks," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* (2001).

[12] Maass, W. W., Markram, H., Natschläger, T., Maass, W. W., and Markram, H., "The" liquid computer": A novel strategy for real-time computing on time series," *Special Issue on Foundations of Information Processing of TELEMATIK* **8**(1), 39–43 (2002).

[13] Ghassaei, A., Demaine, E. D., and Gershenfeld, N., "Fast, interactive origami simulation using gpu computation," *Origami-7: Proceedings of the 7th International Meeting on Origami in Science, Mathematics and Education (OSME 2018)* (2018).

[14] Liu, K. and Paulino, G. H., "Nonlinear mechanics of non-rigid origami: an efficient computational approach," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science* **473**(2206), 20170348 (2017).

[15] Hiller, J. and Lipson, H., "Dynamic Simulation of Soft Multimaterial 3D-Printed Objects," *Soft Robotics* **1**(1), 88–101 (2014).

[16] Paul, C., Valero-Cuevas, F. J., and Lipson, H., "Design and control of tensegrity robots for locomotion," *IEEE Transactions on Robotics* **22**(5), 944–957 (2006).