Computation with mechanically coupled springs for compliant robots

Hidenobu Sumioka, Helmut Hauser, and Rolf Pfeifer

***Abstract*— We introduce a simple model of human's musculoskeletal system to identify the computation that a compliant physical body can achieve. A one-joint system driven by actuation of the springs around the joint is used as a computational device to compute the temporal integration and nonlinear combination of an input signal. Only a linear and static readout unit is needed to extract the output of the computation. The results of computer simulations indicate that the network of mechanically coupled springs can emulate several nonlinear combinations which need temporal integration. The simulation with a two-joint system also shows that, thanks to mechanical connection between the joints, a distant part of a compliant body can serve as a computational device driven by the indirect input. Finally, computational capability of antagonistic muscles and information transfer through mechanical couplings are discussed.**

I. INTRODUCTION

Effectively constructing a computational system is one of the important problems in robotics research because the amount of computation continues to increase for a robot to be engaged in multiple tasks. In the state of the art robotic systems such as [1], a precise joint-angle control architecture has been selected with a rigid body structure and high torque actuators. Although this control architecture allowed a robot to achieve multiple tasks, it has required much computational cost because it has to control every joint angle precisely at all times. However, recent studies inspired from biological systems have reported that a compliant physical body, which often has nonlinear properties, enables achieving several tasks such as locomotion with simpler controllers [2], [3], [4]. This indicates that a part of computation for control of movements can be offloaded to the body, or morphological and material properties, as described by the concept of morphological computation [5], [6].

Despite a great deal of evidence, however, so far there has been no rigorous theoretical basis for this phenomenon. In this context, Hauser *et al.* proposed a theoretical model of morphological computation to understand the capabilities and limitations from a mathematical point of view [7]. They applied the concept of reservoir computing [8] to random, recurrent networks of mass-spring systems. They demonstrated that such networks, which are used in order to mimic real physical bodies, can be used to emulate complex computations, which include the nonlinear combination of temporally integrated information. The networks, although randomly constructed, stay fixed (as a real body does), and only by adding a changeable (i.e., learnable) linear readout, the proposed setup is able to adapt itself in order to emulate such complex computations. For the construction of these networks, Hauser *et al.* used randomly chosen masses and connected them with randomly chosen (regarding their physical properties) springs. The input signal was introduced to the system in form of forces and the linear output was implemented as a weighted sum of all actual spring lengths (i.e, the state of this dynamic system). The corresponding output weights were then adapted in order to emulate a desired computation.

Since they aimed at constructing the theoretical foundation of morphological computation, the structure of springs in the network has not been taken into account. For animals or animal-like robots, however, constraints inherent in physical and mechanical structures are inevitable. For example, muscle fibers of human and other animals, which have nonlinear properties, are subject to constraints in terms of their arrangement: they are basically placed around bones, which are rigid, in parallel and mechanically coupled to each other through a joint between the bones. Although the spring network with such constraints might show lower performance of computation than the general spring network proposed in [7], it should still have computational power to some extent. Furthermore, the mechanical constraints might provide us with a new way to exploit the body as a computational device. Therefore, the identification of computational power in the network will shed light on how much complex signal can be computed through a compliant body. This will also help us design controller embedded into the body.

In the following, we firstly introduce a one-joint system driven by actuation of the springs that are connected around the joint as a simple model of human's musculoskeletal system based on the theoretical model of morphological computation proposed in [7]. In the system, the history of an input signal is memorized as a sequence of the spring movements. The nonlinear combination of temporally integrated input information is computed by a linear and static readout unit. Secondly, the emulation of several different nonlinear combinations is tested to evaluate the performance of the computational capability. We show, through computer simulations, that a same network of mechanically coupled springs can emulate several nonlinear combinations of temporally integrated input information simultaneously even though we use linear and static readouts. After that, in computer simulation on a two-joint system, we also show that, thanks to mechanical connections between the joints, the network can emulate nonlinear and temporal transformation even when it does not receive input information directly.

H. Sumioka, H. Hauser, and R. Pfeifer are with the Department of Informatics University of Zurich, Switzerland {sumioka/hauser/pfeifer}@ifi.uzh.ch

II. Musculoskeletal system as a computational device

Fig. 1 shows a simple model of a musculoskeletal system used in the paper. Two cylindrical rigid objects (links) are connected through a ball joint. The weight of each link is 0.5 [kg]. The upper link is fixed, while the lower link moves based on the activations of N springs, which also connect these links. For the sake of simplicity, the springs are placed with equal spacing around the perimeter of the links: i-th spring is placed at $(r\cos\theta, r\sin\theta)$ (see Fig. 1), where $r = 0.05$ and $\theta = 2\pi\frac{i-1}{N}$. Their dynamics are given by:

$$\dot{x}_1 = x_2 \quad (1)$$

$$\dot{x_2} = -k_1x_1 - k_3x_1^3 - d_1x_2 - d_3x_2^3 + u \quad (2)$$

where, x_1 stands for difference between natural length l_0 and current length l of a spring. We set $l_0 = 0.1$ [m]. x_2 is the derivative of x_1 computed by $(x_1(t) - x_1(t-\delta t))/\delta t$ with $\delta t = 1$ [ms] and u shows an external force to the spring. A positive value of u indicates a contracting force applied to the spring, while a negative one stretches it. To allow diversity of the system, the parameters of each spring (k_1, d_1, k_3, and d_3) are randomly drawn from defined huge ranges: for each spring, k_1 and d_1 are drawn from $[1,0\times10^4, 1.0\times10^6]$ (log-uniform distribution) and k_3 and d_3 are drawn from $[5.0\times10^{12}, 1.0\times10^{13}]$ (uniform distribution). We limit the movement of each spring in a direction where the spring lifts up or down the lower link to avoid twist of the lower link. The simulation of physical dynamics is carried out using ODE (Open Dynamic Engine) (R. Smith, http://www.ode.org/ode.html).

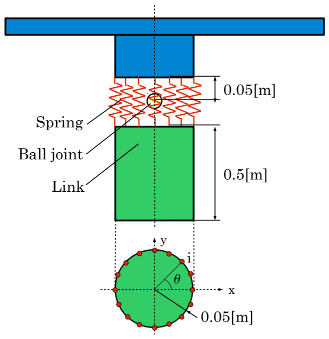

Fig. 1. A simple model of a musculoskeletal system. The lower link is connected with the upper one by a ball joint and multiple springs.

The information processing based on the musculoskeletal model is shown in Fig. 2. A single input signal $I(t)$ at time step t is transformed into external forces $\boldsymbol{u}$ applied to N_{in} springs ($N_{in} \leq N$), which are chosen randomly, by multiplying input weights $\mathbf{w}_{in}$, which are randomly drawn from $[-1.0, 1.0]$. The lower link moves, depending on the actuation of springs caused by their own dynamics and the external forces. The state of the model is measured as the lengths of the springs. A linear and static readout unit computes an output of the model $y(t)$ based on the lengths of all springs with output weights $\mathbf{w}_{out} = (w_{out,1}, w_{out,2}, \cdots, w_{out,N})^T$: $y(t) = \sum_{j=1}^{N} w_{out,j} l_j(t)$, where, $w_{out,j}$ and $l_j(t)$ indicate the output weight for j-th spring and the length of the spring at time t, respectively. While we randomly choose $\mathbf{w}_{in}$ and the coefficients of the springs, the output weights $\mathbf{w}_{out}$ are updated so that the system reproduces a desired signal. For learning we collect the lengths of every single spring $l_j(t)$ at every time step $(t = 1, 2, \cdots, M)$ in a $M \times N$ matrix, $\mathbf{S}$. The desired signal $d(t)$ is also stored as a vector $\mathbf{d} = (d(1), d(2), \cdots, d(M))^T$. Finally, the optimal output weights $\mathbf{w}^*_{out}$ are calculated by $\mathbf{w}^*_{out} = \mathbf{S}^+\mathbf{d}$, where $\mathbf{S}^+$ stands for the (Moore-Penrose) pseudo-inverse of $\mathbf{S}$. Note that the same procedure can be applied in the case of multiple inputs and/or multiple outputs.

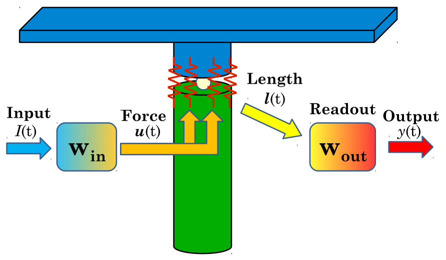

Fig. 2. An overview of information processing in the system. The input signal $I(t)$ multiplied with input weights $\mathbf{w}_{in}$ is applied to several springs as external forces $\mathbf{u}(t)$. The output of the system $y(t)$ is computed as a sum of all length of springs multiplied with output weights $\mathbf{w}_{out}$. While $\mathbf{w}_{in}$ is randomly set and then fixed, $\mathbf{w_{out}}$ is adapted in order to emulate a desired signal.

III. Experiments

A. The performance for nonlinear transformation

First, we tested whether this system has computational capability in terms of the emulation of nonlinear combination that needs temporal integration, following the procedure used in Hauser *et al.* [7]. Hereafter, we refer to a nonlinear combination that needs temporal integration as a nonlinear filter. The tasks for the system were to emulate three different nonlinear filters simultaneously. As the first and second filters, we used 2nd order nonlinear dynamic system and 10th order one, respectively given by:

$$\begin{aligned} y(k+1) &= 0.4y(k) + 0.4y(k)y(k-1) \\ &\quad +0.0048u^3(k) + 0.1, \quad (3) \\ y(k+1) &= 0.3y(k) + 0.05y(k)(\sum_{i=0}^{9} y(k-i)) \\ &\quad +0.06u(k-9)u(k) + 0.1, \quad (4) \end{aligned}$$

where, $u(k)$ and $y(k)$ stand for the input and the output of the system at time step k.

Volterra operator consisting of a quadratic term with a Gaussian kernel was selected as the third filter. For the simulation, the output was computed with discrete expression of Volterra operator given by:

$$y(k) \quad = \quad \sum_{\tau_1=0}^{200}\sum_{\tau_2=0}^{200} 0.04h(\tau_1,\tau_2)u(k-\tau_1)u(k-\tau_2) \quad (5)$$

where, $u(k)$ stands for the input of the system at time step k and h is Gaussian kernel with $\mu_1 = \mu_2 = 100$ [ms] and $\sigma_1 = \sigma_2 = 50$ [ms], i.e., $h(\tau_1,\tau_2) = \exp\left((\tau_1-\mu_1)^2/2\sigma_1^2 + (\tau_2-\mu_2)^2/2\sigma_2^2\right)$. The input signal $u(t)$ at time step t was computed by multiplying outputs of three sine functions that have different frequencies:

$$u(t) = 0.2 \cdot \sin(2\pi f_1 t) \cdot \sin(2\pi f_2 t) \cdot \sin(2\pi f_3 t), \qquad (6)$$

where, we set $f_1 = 2$, $f_2 = 3.1$, and $f_3 = 4.2$ [Hz].

We ran 400 simulations with different networks which are randomly made at the defined range (see in Section II). In each simulation, external forces computed by the input signal were applied to randomly-chosen four springs out of 16 springs. Hauser *et al.* [7] showed that a same morphological structure can be used for different tasks simultaneously (i.e. multiplexing). Therefore, a same network for learning of the three different filters is used to show the multiplexing property of the network. We collected $10,000$ samples of the input signal, the lengths of all springs, and the corresponding desired outputs after initial $5,000$ samples were discarded as transients. After learning, the differences between the desired outputs and the emulated outputs were collected during $5,000$ steps, and then their mean squared errors (MSEs) were computed to evaluate the performance of the acquired readout units. The averages of the MSEs for the three filters, the 2nd order system, the 10th order system, and the Volterra series, were 2.55×10^{-4}, 1.28×10^{-3}, and 7.09×10^{-3}, respectively with standard deviations, 1.98×10^{-4}, 9.17×10^{-4}, and 7.50×10^{-5}. Fig. 3 shows the result of the network with the best performance. To clarify the contribution of the body structure to the computation, we compared the results of our system with the ones of simple linear regression on the raw input data[1]. We can see that our system shows a much higher performance than one of the linear regression. It emulated the 2nd and 10th order systems accurately, while it failed in the emulation of Volterra time series. Since the general model proposed by Hauser *et al.* [7] showed the high performance of the three filters, the mechanical constraints in our system prevent the computational capabilities of the system. Nevertheless, it can still emulate the 2nd and 10th order filters.

Note that the large standard deviations of the performances of the 2nd and 10th order systems are caused by a few outliers. To show this evidence clearly, we sorted 400 networks according to their MSEs for the 2nd and 10th order filters (Fig. 4). As can be seen, most of the networks show high performances in the 2nd and 10th order

[1]We used a linear regression with two weights, w_1 for the actual input $u(t)$ and w_2 to learn a bias. Hence, the resulting output at time step t was $y_{LR}(t) = w_1u(t) + w_2$. The two weights were adapted to emulate the desired integrations.

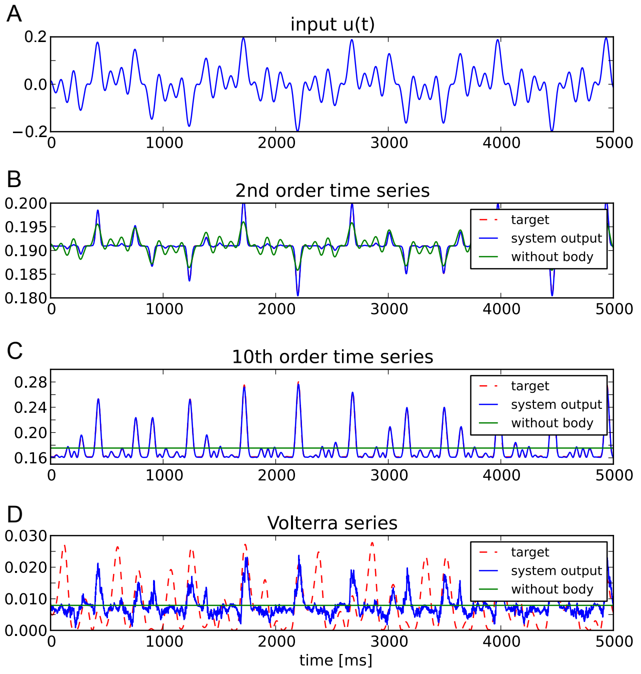

Fig. 3. The performance of the three nonlinear combination and temporal integration tasks. The top figure shows the input $u(t)$. The other three figures show the performances of the 2nd order system, the 10th order system, and the Volterra series. The red line is the target trajectory and the blue line is the output of the system. The green line indicates the output computed by simple linear regression on the raw input.

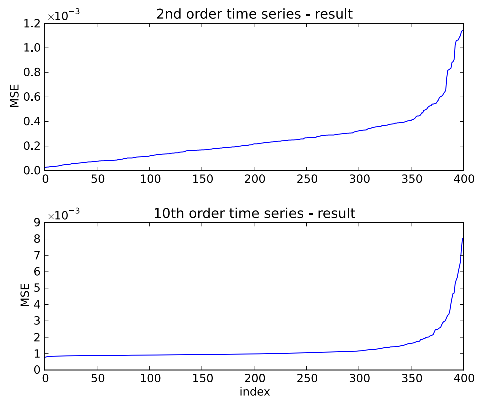

Fig. 4. The mean squared errors for the 2nd order integration task and the 10th order one achieved by 400 random networks. Each index corresponds to a network.

cases except for a few networks that show much lower performance. We compared the networks that show the best performance and the worst one for the 2nd order task in terms of the distribution of input weights $\mathbf{w_{in}}$ to investigate the relationship betweeen the performance and the effect of input signal on the behavior of the network. Fig. 5 shows the value of the input weight at each position in the networks with the best performance and the worst one. The value represents the scaling factor of the input signal. Given positive value of the input signal, a positive weight generates a contracting force to a spring while a negative one provides a force to stretch the spring. As we can see, the directions of the applied forces are clearly separated in the best performance case so that the input signal can drive the system in a manner similar to the way antagonistic muscles are driven. On the other hand, they are conflictive in the worst performance case. This confliction prevents the lower link moving enough to reflect the input signal. As a result, the performance becomes lower. Therefore, the system has robust computational capabilities as long as its movement reflects the input signal.

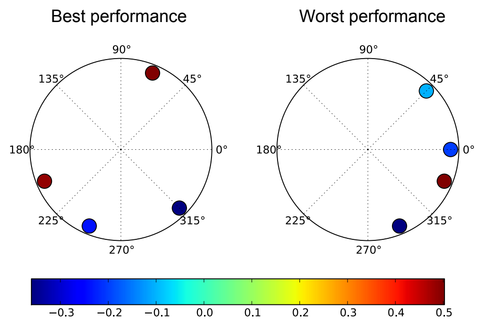

Fig. 5. Distributions of the input weights to compute external force on the networks with the best performance and the worst one. The angle corresponds to θ shown in Fig. 1. Given a positive input, positive values produce forces to contract the corresponding springs while negative ones stretch them. External forces are not applied to other springs.

B. The influence of complexity of the body on the performance

The number of nonlinear elements will affect the performance of the computation although we fixed it above. The fewer elements the system has, the lower the performance will be. Therefore, we examined to which extent the performance depends on the number of springs.

We used the same setting as the one in the previous experiment except that we changed the number of springs from one to 16. For each configuration of springs, we ran 400 random simulations with different spring properties. The number of the springs to which external forces are applied was set to one if $N \leq 3$ and two if $N = 4$ or $N = 5$. In other cases, it was determined by rounding 25% of all springs. The performance was evaluated with MSEs for the 2nd and 10th order filters.

Fig. 6 shows the average and standard deviation of MSEs for the tasks on each configuration. In the cases of fewer springs, the averages of MSEs for both tasks were much higher than ones in the case of 16 springs. The performance improved from the cases of six springs drastically and then leveled off after ten springs. As a result, it turned out that ten springs have already the computational capability to compute the 10th order integration.

C. The influence of mechanical limitation on the performance

We have showed that even spring networks with a mechanical constraint, that is, a ball joint, have computational capabilities to emulate nonlinear filters in the previous experiments. Many joints in animal or animal-like robot are

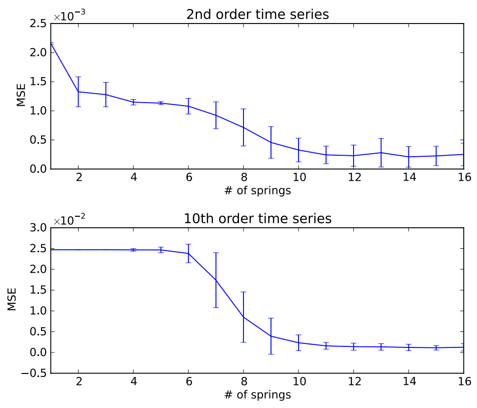

Fig. 6. The performance of the 2nd order and 10th order integrations for different number of springs.

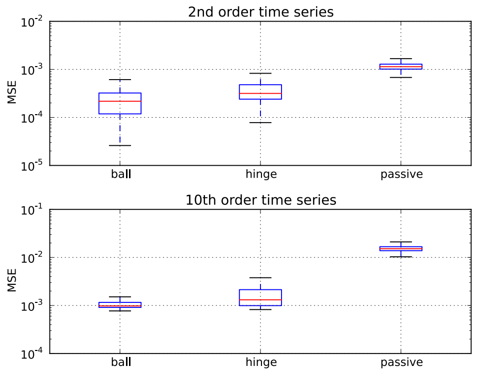

Fig. 7. Boxplots of mean squared errors of 400 different networks with a ball joint, ones with a hinge joint, and ones with a passive joint (see section III-D). The vertical axis in each figure is in logarithmic scale.

modeled not by ball joints but by joints that can rotate only in one axis like a hinge. As a next step, therefore, we investigated to which extent the system with this further constraint can preserve the computational capabilities.

We ran 400 simulations, following the same procedure as the one in the experiment of section III-A except that we replaced a ball joint in the model with a hinge joint. The performance of the system was evaluated for the 2nd and 10th order filters in the same way as the previous experiment: we computed the MSEs between the desired outputs and the outputs by the adapted readout units.

The distributions of the MSEs for each task on the networks with a ball joint and a hinge are displayed by the boxplots in Fig. 7. As one can see, in the case of a hinge joint, the number of the networks that have low performance increases because, due to limited movement of the joint, it is difficult to construct a network that reflects the input signal. Interestingly, however, several networks with a hinge joint still showed smaller MSEs for the 2nd and 10th order filters than the medians of networks with a ball joint. The comparison between distributions of $\mathbf{w_{in}}$ in the networks with the best performance or the worst one (Fig. 8) indicates that, even when the axis of the rotation on a joint is limited, a spring network can have computational capabilities if the movement of the link reflects the input information.

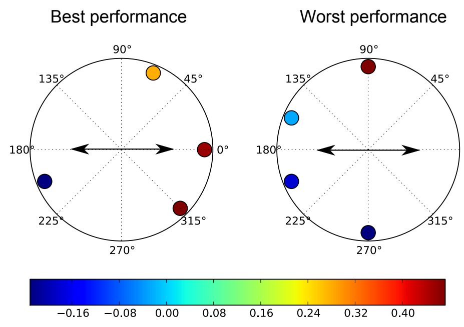

Fig. 8. Distributions of input weights to compute external forces on the networks with the best performance and the worst one. The angle corresponds to θ shown in Fig. 1. The arrows show the direction of the movement of the system. In the best performance case, the positive and negative values work in a manner similar to the way antagonistic muscles work, while only negative valves drive the system in the worst performance.

D. The information transmission through mechanical coupling

The previous experiments have suggested that a network of multiple springs can serve as a computational device if its behavior can reflect input information. This implies that, even when the network cannot receive an input signal directly, it can emulate nonlinear combination of temporally integrated input information if its movement is passively affected by the input signal. In a multi-joint system that includes passive parts and actively-actuated ones, a passive part is affected by the actuation of an active part through a physical link. If the passive part can emulate nonlinear combination and temporal integration of the indirect input signal, it can serve to generate an input-dependent signal to an adjoining part which should be actuated. This might allow us to regard passive elements in a robotic system as a local controller. Therefore, we investigated whether a spring network that receive the indirect input through mechanical connection can emulate nonlinear filters.

We used a two-joint model in which N $(= 16)$ springs are placed with equal spacing around the perimeter of the links connected by each joint (Fig 9). While the parameters of the springs around lower joint are randomly set at the same range as the ones on the previous experiments, the parameters of the springs around upper joint are set so that they are affected by input signals easily: k_1 and d_1 are randomly drawn from $[1,0 \times 10^3, 1.0 \times 10^5]$ and k_3 and d_3 are randomly drawn from $[5.0 \times 10^{11}, 1.0 \times 10^{12}]$. The same input signal shown by Eq. (6) was applied for randomly-chosen four springs around the upper joint with input weights. Since it turns out that the computational performance depends much on the distribution of the input weights, weight $w_{in,i}$ for i-th spring was randomly drawn from:

$$\begin{aligned} w_{in,i} &\in [0.0, 1.0] \qquad if \quad N/2 \geq i \\ w_{in,i} &\in [-1.0, 0.0] \qquad otherwise. \end{aligned}$$

The output of the system was computed as a sum of the weighted lengths of all springs around the lower joint (blue spring in Fig 9). Note that the input signal is not directly applied to springs that are used to compute the outputs of the system. The input information was transferred to the springs around the lower joint through the physical and mechanical connection. The passive movement of the springs was used to emulate the nonlinear filter.

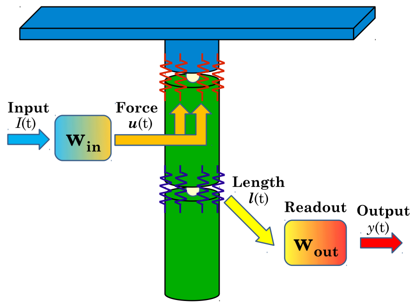

Fig. 9. An overview of the information processing in a multi-joint system with mechanically-coupled springs. The input signals multiplied with input weights $\mathbf{w}_{in}$ are applied to some springs around the upper joint as external forces. The output of the system is computed as a sum of all length of springs around the lower joint multiplied with output weights $\mathbf{w}_{out}$.

We ran 400 simulations, following the procedure described in section III-A. The MSEs between desired outputs and outputs of the adapted readout units were computed for the 2nd and 10th order system tasks. The averages of MSEs for the 2nd and 10th order cases were 1.12×10^{-3} and 1.49×10^{-2}, respectively with standard deviations 2.28×10^{-4} and 2.52×10^{-3}. We shows the distributions of the MSEs for each task and the result of the network with the best performance in Fig. 8 and Fig. 10, respectively. The network show lower performance than ones of one-joint cases. However, we can see that a spring network can emulate nonlinear filters by transmitting the input signal through mechanical connection, compared with the result of simple linear regression shown in Fig. 3.

IV. Discussion and Conclusion

In classical robotic systems, redundant configuration of the actuators were avoided to achieve precise control. However, it has been reported that such properties make the control tasks easier. A bi-articular muscle is a typical example because it is suggested that they play an important role in stiffness control [9] and jumping [4]. In biological systems, each joint is controlled by the actuation of a lot of muscle fibers, which are well-arranged in parallel or obliquely to the long axis of the muscle [10]. The analogy between musculoskeletal system and our system suggests that the redundant configuration of biological muscles has a potential as local computational resource. Furthermore, interestingly, the system showed the best performance when the input signal was applied so that it can drive the system in antagonistic way. This might imply that biological configuration has high computational capability.

However, we found, in the first experiment, that the current structure cannot emulate Volterra series, while, theoretically

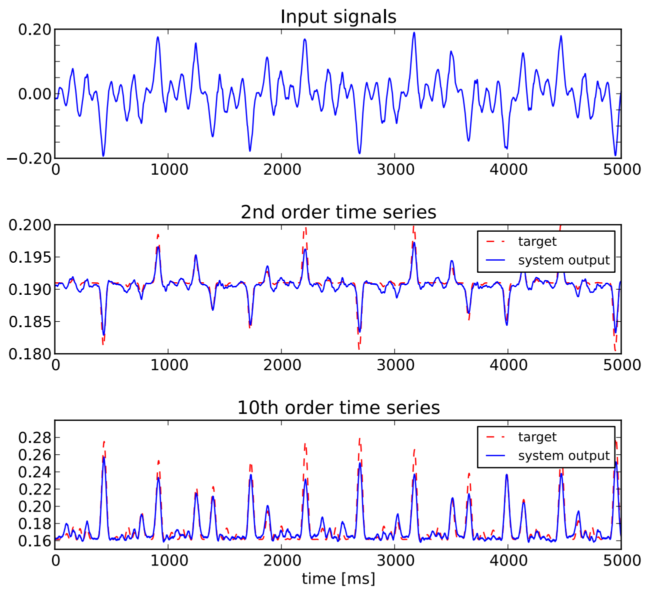

Fig. 10. The performance of nonlinear and temporal integration tasks with the two-joint setup. The top figure shows the input $u(t)$, which consisted of three sinusoidal functions. The other two figures show the performances of the 2nd and 10th order filters, respectively. In each figure, the red line is a target trajectory and the blue line is the output of the system.

speaking, the randomly-connected mass-spring network can emulate with arbitrary precision any operators, which can be approximated by a Volterra series [7]. This difference might come from lack of local interactions between springs in the proposed model because such interactions increase the complexity of a network to be exploited for the emulation. More complex structure inspired from biological systems will help us understand what kind of configuration in a biological system provides higher computational capability.

Another interesting point is that a mechanical link between joints allows us to transfer input information fed into parts of body to other parts. In biological system and robotic one, active elements and passive one are often mixed. For example, during quasi passive dynamic walking, while hip and ankle joints are activated, knee joints would be passive. We showed, in our second experiment, that the movement of a part of the body driven by input signal enables a distant part, which is passive, to compute an input-dependent signal. This might suggest that physical constraints and compliant body allow a central controller to transmit input information to local controllers that are located in more distant parts from the central one.

Although we did not discuss the movement of the system in the paper, we should take it into account to make the system serve as not only a generator of signal but also a generator of movement to achieve a task. If the system driven by a signal shows a meaningful movement, it can be engaged in a movement task, computing control signal by itself. A pattern generator like a central pattern generator (CPG) is a possible application of such system: the system can show periodic movement when we design the readout that can output next state of a periodic input and then feed back its output to the system instead of actual input. In the context of reservoir computing, it has been already reported that this kind of feedback loop enables a network to store persist memory such as periodic pattern [11], [12]. The introduction of feedback loop will allow the system to perform periodic movement with computing its own control signal.

In this paper, we demonstrated that the network of mechanically coupled springs inspired by musculoskeletal system has computational capability to emulate different nonlinear combinations of temporally integrated information. Such network was still able to maintain computational capability even when a further limitation on degrees of freedom was added to the system. We also showed that, in experiment on a two-joint system, thanks to mechanical connection between the joints, a distant part of a compliant body can serve as a computational device driven by the indirect input. These results suggested that each joint in compliant physical body has a potential as computational device. We will introduce a feedback element into our system to design a pattern generator that can perform meaningful movement with computing its own control signal. Experiments with real robots will be also conducted to verify the results.

V. Acknowledgments

This research was funded in part by the European Community's Seventh Framework Programme FP7-ICT-248311 (AMARSi) and FP7-ICT-231688 (Locomorph) and the Swiss National Science Foundation through the National Centre of Competence in Research Robotics.

References

[1] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo: System overview and integration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2002, pp. 2478–2483.

[2] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers," *Science*, vol. 307, no. 5712, p. 1082, 2005.

[3] F. Iida, "Cheap design approach to adaptive behavior: Walking and sensing through body dynamics," in *International symposium on adaptive motion of animals and machines*. Citeseer, 2005.

[4] K. Hosoda, Y. Sakaguchi, H. Takayama, and T. Takuma, "Pneumatic-driven jumping robot with anthropomorphic muscular skeleton structure," *Autonomous Robots*, vol. 28, no. 3, pp. 307–316, 2010.

[5] C. Paul, "Morphological computation:: A basis for the analysis of morphology and control requirements," *Robotics and Autonomous Systems*, vol. 54, no. 8, pp. 619–630, 2006.

[6] R. Pfeifer, M. Lungarella, and F. Iida, "Self-organization, embodiment, and biologically inspired robotics," *Science*, vol. 318, no. 5853, p. 1088, 2007.

[7] H. Hauser, R. Pfeifer, J. A. Ijspeert, and W. Maass, "A theoretical foundation for morphological computation," *Biological Cybernetics*, (submitted).

[8] M. Lukosevicius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[9] M. Kumamoto, T. Oshima, and T. Yamamoto, "Control properties induced by the existence of antagonistic pairs of bi-articular muscles–mechanical engineering model analyses," *Human Movement Science*, vol. 13, no. 5, pp. 611–634, 1994.

[10] A. Freivalds, *Biomechanics of the upper limbs: mechanics, modeling, and musculoskeletal injuries*. CRC, 2004.

[11] W. Maass, P. Joshi, and E. Sontag, "Computational aspects of feedback in neural circuits," *PLOS Comp. Bio.*, vol. 3, no. 1, pp. 1–20, 2007.

[12] J. Li and H. Jaeger, "Minimal energy control of an esn pattern generator," Jacobs University, Tech. Rep. 26, 2011.