Representing and Combining Dynamics in Biologically Plausible Neurons

Sean R. Aubin

January 2, 2017

Abstract

Conceptors, although biologically implausible, admirably capture high dimensional dynamical patterns. This report contains a concise overview of Conceptors and describes how the same dynamical pattern approximation can be achieved, with limitations, in a biologically plausible manner using the Neural Engineering Framework. Two methods are compared to Conceptors with this goal in mind: Rhythmic Dynamic Movement Primitives, both with and without point attractors. In terms of representing a dynamic signal, Dynamic Movement Primitives implemented directly performed well for low-frequency signals, while Conceptors performed well for high-frequency sinusoidal signals. In terms of blending between dynamic signals, Conceptors are distinct from Dynamic Movement Primitives, but this usefulness of this is unclear.

1 Introduction

The intent of this project was to replicate the results of Conceptors in the domain of representing and combining dynamic signals using a neural population [5], but instead doing so in a biologically plausible manner by leveraging the Neural Engineering Framework (NEF) [4]. All code used in this report is available at `https://github.com/Seanny123/nef-conceptors`.

1.1 Conceptors

The Conceptor approach to representing dynamics is inspired by Reservoir Computing (RC), where a randomly connected population of neurons are fed back on themselves to create a dynamic system.

The neurons used in Conceptors, which will be referred to as leaky-tanh neurons, are described by the following activation function act where x_{in} is the input to the neuron, LR is the memory leak rate and t is time-steps of the simulation and t_{max} is the total simulation time:

$$act(x_{in}(t)) = (1 - \text{LR})act(x_{in}(t-1)) + \text{LR}\tanh(x_{in}(t)) \quad \text{where } 1 < t < t_{max} \tag{1}$$

To create the dynamic system illustrated in Figure 1 an array of of neurons is used and recurrently connected to each other. Thus Equation 1 is generalized for multiple neurons in Equation 2.

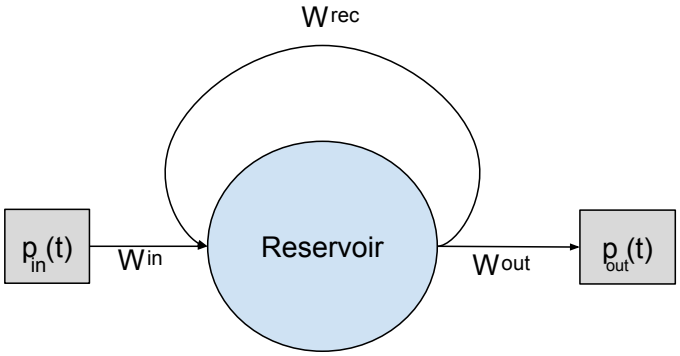


Figure 1: Illustration of Reservoir Computing system.

$$state_i(t+1) = act(W_i^{rec} state_i(t) + W_i^{in} p_{in}(t+1) + b_i) \tag{2}$$

Where i is the neuron index, $state_i$ is the neuron state, W^{in} are the input weights to the neuron population (equivalent to the encoders of the NEF), W^{rec} are the recurrent weights, b_i is the bias term for the neuron and p_{in} is the input pattern. Note that the magnitude W^{rec} is less than their eigenvalues to maintain a stable system.

In RC, the output weights are set to minimize the error between output signal p_{out} and the intended signal which is assumed to be some function of the input signal $f(p_{in})$, as shown in Equation 3.

$$\begin{aligned} p_{out}(t) &= W^{out} state(t) \\ err &= (f(p_{in}) - W^{out} state(t))^2 \end{aligned} \tag{3}$$

This equation can be solved in many different ways, however in the original Conceptors report "Ridge Regression" was used. This is functionally equivalent (same inputs and same outputs) to the Least Squares optimization used typically used in the NEF for solving for decoders. Consequently, W^{out} can be considered equivalent to decoders of the NEF.

The RC approach to dynamic signal representation can only represent one pattern at a time. Conceptors attempt to rectify this. Conceptors also use a recurrently connected group of neurons. However, instead of the recurrent weights being selected randomly and limited in their magnitude, the weights are determined by linear regression such that the system is oscillates without exterior input for all j of K input patterns. This linear regression problem is phrased as follows:

For stable oscillation, the state with input should approximate the state without input, which can be accomplished by finding new recurrent weights W^{opt}

$$act(W^{rec} x(t) + W^{in} p_{in}(t) + b) \approx act(W^{opt} x(t) + b)$$

Thus the recurrent weights can be found from this minimization which is solved

as yet another Least Squares optimization problem.

$$W^{opt} = argmin_{W^{opt}} \sum_{j=1}^{K} \sum_{t=1}^{t^{max}} (W^{rec} x^j(t) + W^{in} p^j_{in}(t+1) + b - W^{opt} x^j(t))^2$$

This dynamic system can then reproduce specific dynamic patterns p^j_{in} by the modification of the recurrent weight matrix by various the Conceptor matrices C^j, where j is the pattern index, such that the update rule becomes:

$$\begin{aligned} state(t+1) &= C^j act(W^{opt} state(t) + b) \\ \text{such that, } p_{out}(t) &= W^{out} state(t) = p^j_{in}(t) \end{aligned} \tag{4}$$

Thus, C^j could also be considered as a "projection matrix" given that the activities of the reservoir are projected onto the correct output pattern by C^j.

To find C^j, let $R^j = X^j (X^j)'/L$ be the reservoir state correlation matrix where X^j is the state of the neurons for pattern j for all time steps L. Let $\alpha \in (0, \text{inf})$ be the "aperture", which the fidelity to the original signal that the output should achieve.

Given that C still projects back onto the reservoir, we want to minimize the deviations of the actual state from the projected state.

$$C = E\left[\|x - Cx\|^2\right]$$

As mentioned before, the degree we want C to influence the state of the reservoir is scaled by α. This influence is taken into account by adding a term

$$C = E\left[\|x - Cx\|^2\right] + \alpha^{-2} \|x - Cx\|^2_{fro}$$

Where $\|x - Cx\|^2_{fro}$ is the Frobenius norm.

Using a derivation contained in Section 5.1 of [5] it is proven that C and R have the same principal component vector orientations. Thus,

$$\begin{aligned} \text{SVD}(R) &= U \Sigma U' \\ \text{SVD}(C) &= U S U' \end{aligned} \tag{5}$$

Additionally, the singular values of C are related to the singular values of R by

$$s_i = \frac{\sigma_i}{\sigma_i + \alpha^{-2}} \tag{6}$$

Equations 5 and 6 provides us a way to derive C, since R is known. This is identical to solving a Least Squares optimization problem, wherein given R and the output signal to reproduce, connection weights are found using SVD. Although ideally this could be shown mathematically, this was demonstrated using a simulation in Figure 2. Additionally, when comparing the the Conceptor weight matrix C with the weight matrix created by

$$W^{\text{SVD}} = \text{encoders} \cdot \text{SVD-decoders}$$

the Root Mean Squared error between W^{SVD} and C was an insignificant 0.058.

An illustration of the formulated Conceptor system is shown in Figure 3.

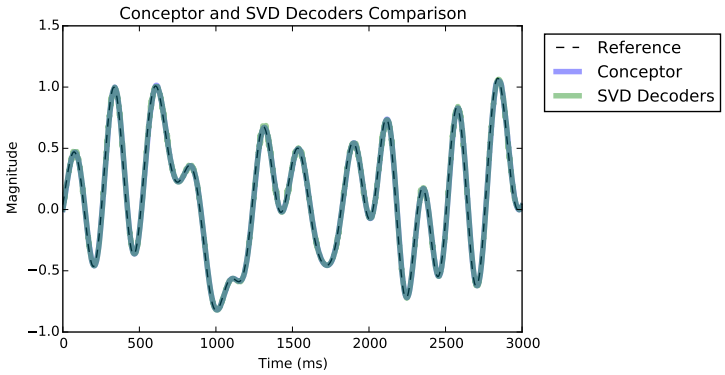


Figure 2: Comparison of decoding result from a population of LIF neurons showing the equivalence of using decoders found with SVD and using a Conceptor matrix (to get rates) followed by decoders found using Least Squares optimization on the original population (as opposed from to the Conceptor-modified rates) to get a scalar value from the Conceptor-modified rates. The results are nearly identical and both reproduce the input signal. Consequently, it can be concluded that Conceptors reproduce the rates of the input signal and are equivalent to weights found with SVD.

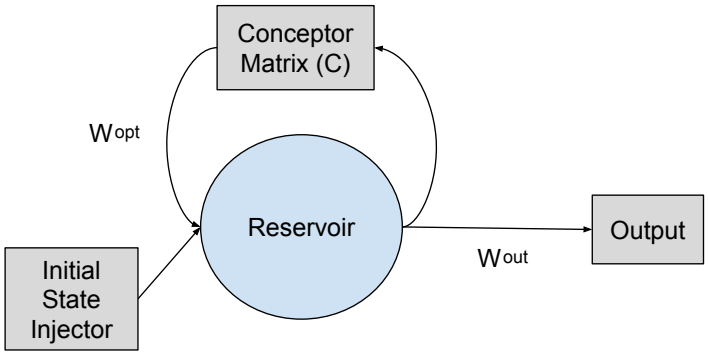


Figure 3: Illustration of Conceptor system. Grey boxes are non-neural components, blue circles are neural components and unlabeled arrows are direct connections to or from neurons. The "Initial State Injector" is required to initialize the neural system so it does not remain inactive, but this value can be any arbitrary value that activates the majority of the neurons.

To switch between signal, different values of C must be swapped out. This constant dramatic modification of the recurrent weights, which are mapped onto synaptic weights, makes this approach biologically implausible, since synaptic weights generally undergo gradual changes.

1.1.1 Similarity to Principle 3 of the NEF

As mentioned previously, W^{in} and W^{out} are respectively equivalent to the encoders and decoders of the NEF. Aside from the aforementioned swapping out of C, the formulation of Conceptors is remarkably similar to Principle 3 of the NEF.

Principle 3 of the NEF defines how recurrent connection weights filtered by synapses can be used to approximate various dynamic systems. Principle 2 defines how to find the weights defined by Principle 3.

Conceptors define recurrent connection weights that can be found using Principle 2 of the NEF, as discussed in the previous section.

Consequently, it is justifiable to say that Conceptors are a reformulation of Principle 3 of the NEF, but without the usual use of axonal low-pass filters to create system dynamics and instead rely on the leaky-tanh neurons described in 1. This dependence has various negative effects, as described in the following sections.

1.1.2 Tuning of Memory Leak Rate

The memory leak rate (LR) of the neuron describe in Equation 1 has a significant effect on the quality of the signal being represented in a purely feed-forward network (Figure 4) and even more drastically in a Conceptor network (Figure 5). Note the results in both of these figures were obtained from a modified reference implementation of Conceptors implemented in Matlab[1]

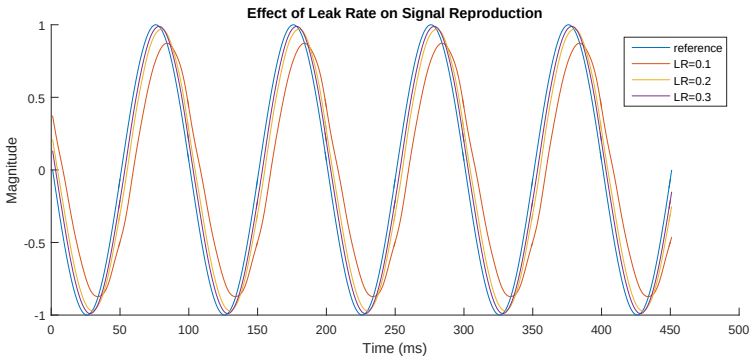
The range of possible LRs is hard to justify, given they have no biological analogue. Additionally, I was not able to mathematically derive the effect of these neurons on the dynamic systems being represented. Given this lack of biological and mathematical grounding, the best justification I can provide is that the sensitivity of both the feed-forward and Conceptor systems to varying LRs stems from the lack of inclusion of the integrating effects of LRs in any of the derivations shown in Section 1.1.

1.1.3 Blending Between Conceptors

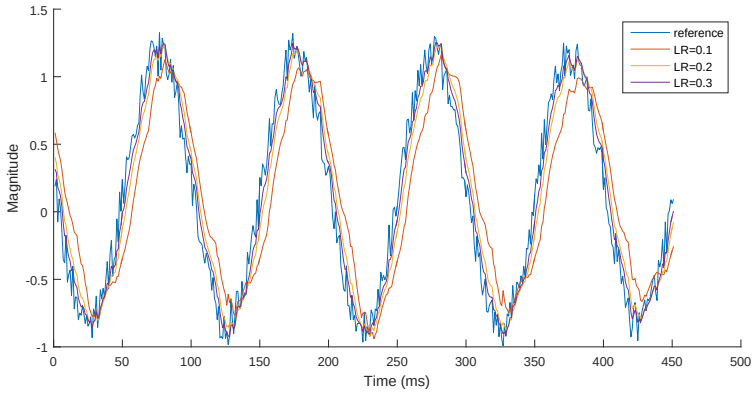
In addition to replicating signals, Conceptors are able to blend between signals by mixing Conceptors using a scaling factor μ. For example, blending between two signals a and b, enabled by the Conceptors C^a and C^b, is performed by modifying μ in the following equation:

$$C = ((1 - \mu)C^a + \mu C^b) \tag{7}$$

[1]Code that was originally used to generate the results seen in `https://youtu.be/DkS_Yw1ldD4` was stripped down to it's basic components and various bugs were corrected. Thus I refer to the code I used as a 'modified' reference implementation.

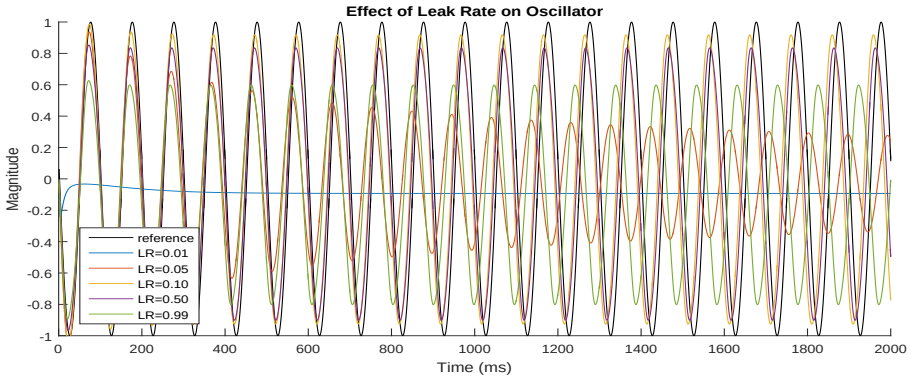


(a) Noiseless Representation

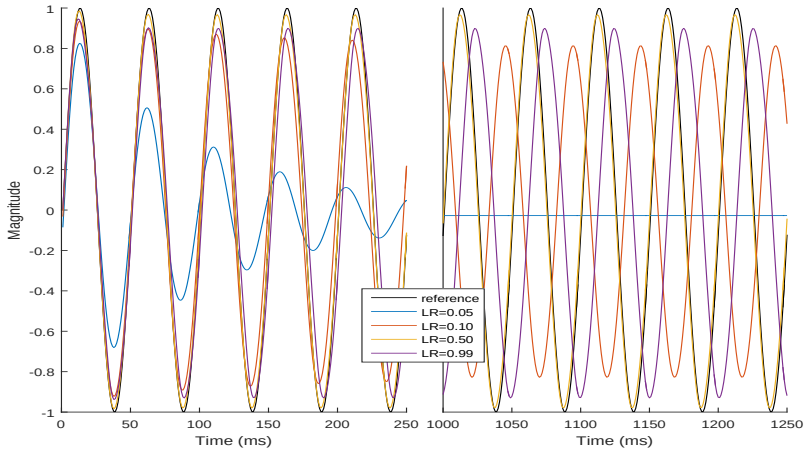


(b) Noisy Representation

Figure 4: The leaky-tanh neuron has the effect of a low-pass filter on attempts to reproduce a sinusoidal signal.

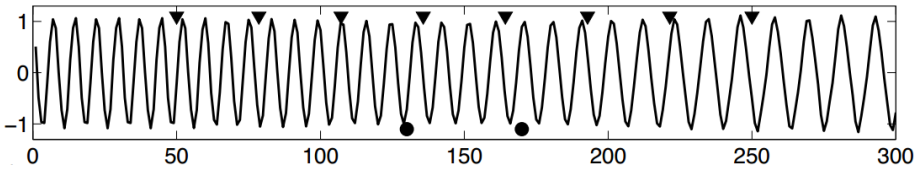


(a) 10 Hz Oscillator

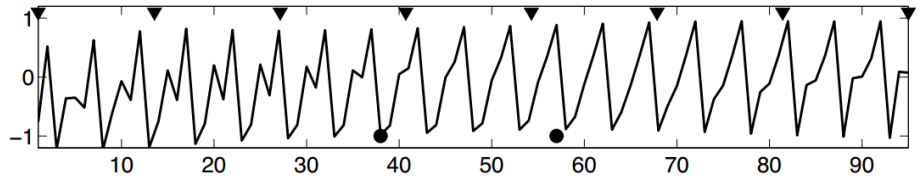


(b) 20 Hz Oscillator

Figure 5: The choice of a good LR becomes more important for higher frequencies, where incorrect choices can cause phase shifts or decaying signals.



(a) Blending between two sinusoids.



(b) Blending between non-sinusoidal signals.

Figure 6: Blending between two signals by modifying μ. Figures reproduced from the original Conceptor publication [5]. The black dots are points were $\mu = 1$ and $\mu = 0$, thus the signals appear in their unmodified form. The x axis is time steps (no units are given in the original publication), while the y-axis is the output.

For sin-waves this gives the effect of blending between frequencies, while more irregular signals have less elegantly describable transitions, as shown in Figure 6.

The ability to reliably recreate and blend dynamic signals in a biologically plausible manner is essential for a complete cognitive system. This report describes how this aforementioned goal was achieved using the Neural Engineering Framework (NEF)[2].

2 Methods

The modified reference implementation of Conceptors, introduced in 1.1.2, was compared to two approaches using the NEF in Nengo[3].

2.1 Biologically Plausible Alternatives to Conceptors

The two approaches investigated using the NEF and Nengo were Rhythmic Dynamic Movement Primitives (rDMPs), either decoded directly from an oscillator or decoded using forcing function taking into account a point attractor.

2.2 Rhythmic Dynamic Movement Primitives

Dynamic Movement Primitives (DMPs) are a way of planning movement along a trajectory using dynamics. By definition, this can be any dynamic system,

[2]The other features of Conceptors, such as their relation to Boolean algebra, online adaptation and classification, are outside of the scope of this report.

[3]A reference implementation of Conceptors in Nengo was attempted, but ultimately failed due to being unable to implement a population that would sustain a repeating pattern via oscillation. The cause of this failure was narrowed down to the difficulty of implementing the leaky-tanh neuron in Nengo, but there was no time to resolve this problem.

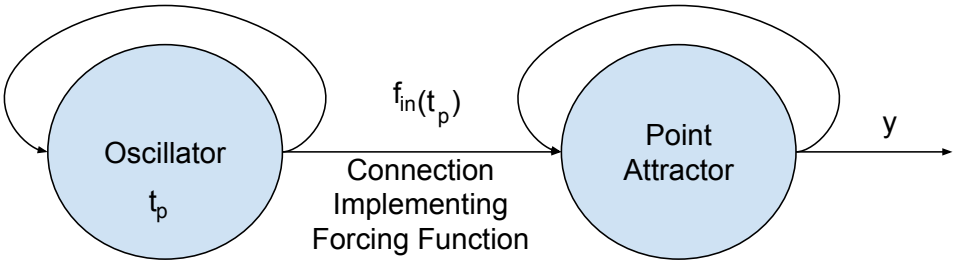


Figure 7: Illustration of point attractor based Dynamic Movement Primitive system. The oscillator provides the repeating timing while the point attractor provides a dynamic system for fine-tuning decoding from the oscillator.

but in [3] both ramps and oscillators are used. To achieve control from the aforementioned dynamic systems, DMPs usually use weighted Gaussian basis functions approximating a path at each time step ($y_{path}(t_p)$ where $0 < t_p < t_{end}$ and t_{end} is the length of the path) to dictate the forces on a moving point as it goes from the starting point to the finishing point, with a point attractor leading the path to the finishing point. This can be described in mathematical terms, as a point attractor modified by a force:

$$\ddot{y} = \alpha_y(\beta_y(g - y) - \dot{y}) + f_{in}(t_p) \tag{8}$$

Where is y is the output of the system and the position of the moving point trying to approximate y_{path} in response to the point attractor forces at each time-step, g is the position of the point attractor, α_y and β_y are (in this report, constant) gain terms of the point attractor, and $f_{in}(t_p)$ is the forcing function that modifies the path that the moving point follows on it's way to the point attractor.

The forcing function is typically implemented by weighted Gaussian basis function that are sequentially activated by a ramp function who's output is t_p. In a neural implementation, the basis functions are leaky-integrate-fire neuron tuning curves and the weights are the decoding weights that correspond to the aforementioned neurons.

To imitate a desired path, which is the goal of this DMP use case, the forcing function needs to be solved in terms of the given path y and the point attractor location and gains by using Equation 8.

Rhythmic Dynamic Movement Primitives (rDMPs) are a variation of DMPs, where instead of having a discrete goal, a repeating path is followed. Consequently, instead of a ramp function activating the basis functions, a repeating ramp is used by taking the *arctan* of an oscillator moving around the circle and spacing the basis functions between $-\pi$ and π. This repeating ramp is the input t_p of $f_{in}(t_p)$. This complete system is shown in Figure 7.

In the case of rDMPs, $f_{in}(t_p)$ is calculated by first defining all variable and constants in Equation 8. The position of the point attractor is heuristically determined to be the center of the input signal y_{path} discretized into an arbitrary amount of sample points:

$$\begin{aligned} &\text{let } \alpha_y = 10,\ \beta_y = 2.5 \\ &g = \frac{\sum y_{path}}{t_{end}} \end{aligned} \tag{9}$$

To define the desired path of the point, y_{path}, in the domain of the oscillator output, x, we use `scipy.interpolate`. This function is represented at $interp()$ in the following equations:

$$\begin{aligned} -\pi < x < \pi \\ y_{ip} = interp(x, y_{path}) \\ \dot{y} = \frac{dy_{ip}}{dx} \\ \ddot{y} = \frac{d^2 y_{ip}}{dx^2} \end{aligned} \tag{10}$$

After the variable and constants are defined, $f_{in}(t_p)$ can be found by solving Equation 8:

$$f_{in}(t_p) = \ddot{y} - \alpha_y(\beta_y(g - y_{ip}) - \dot{y}) \tag{11}$$

For a simpler implementation of rDMPs, it is possible to remove the point attractor from Equation 8. This allows for a path to still be defined via a forcing function which can be calculated by the usual NEF method of decoding a transform given an input signal and a desired output signal. In simpler language, decoders are calculated directly from an oscillator's output and the desired output signal. This simplified, but limited implementation of an rDMP will be referred to as Direct DMPs (dDMPs). Although dDMPs are simpler, this comes at the cost of being unable to change the dynamics on the fly, as there is when using the system defined by changing any of the variables in 11. This aforementioned flexible and adaptable attractor-based implementation of rDMPs will be referred to as aDMPs.

2.3 Switching Between Dynamic Movement Primitives

To switch between the rDMPs, two methods were attempted. The first was simply to switch between two output patterns by inhibiting one output neuron population while releasing the inhibition on the other. The switch was performed by linearly changing the inhibition level from -1 to 0 for one population and vice versa for the other population. The second was to multiply the output of each pattern by a scalar in a similar manner as μ in Equation 7. The switch was performed by linearly changing the scalar from 0 to 1 for one output and vice versa for the other output.

3 Comparison of Results

The representational ability and blending ability of each were the metrics for comparison. For representational ability, the Root Mean Square Error (RMSE) from the original signal, after phase shifts, was the evaluation metric. For blending, the transitions between two signals ($\sin(2\pi 6t)$ and $0.5\cos(2\pi 10t)$),

Signal Frequency (Hz)	2	6	10	15	20
Leak Rate LR	0.05	0.1	0.4	0.6	0.6
Oscillator Frequency (Hz)	1	3	2.5	3.75	5

Table 1: Heuristically set variables for dDMPs and Conceptors that greatly improve their performance.

given a fixed time of 1 second to transition (between the times 0.5s and 1.5s in the figures), were compared qualitatively in terms of smoothness of transition.

All neural networks used a total of 600 rate neurons. In the case of dDMPs, this meant 600 neurons were used in the oscillator. For Conceptors, this meant the reservoir contained 600 neurons. Finally, for aDMPs 300 neurons were used for the oscillator and 300 neurons were used for the attractor network. Nengo networks used LIF rate-based neurons, while Conceptors used the leaky-tanh neurons defined in Section 1.1.

3.1 Representing Signals

For sinusoidal signals, the ideal method for representation depended on the frequency of the signal.

For low frequency signals, dDMPs were superior, given that there are based off an ideal oscillator which is already oscillating at half the desired frequency for signals at 2Hz and 6Hz, and oscillating at one quarter the desired frequency for signals at 10Hz, 15Hz and 20Hz, as shown in Table 1.

Conceptors were much better at representing high frequency sinusoidal signals, as shown in Figure 8. This result is partially because Conceptors fail to reach the desired amplitude at low frequencies, as can be seen in the comparison in Figure 9. The reason for this is unknown, but most likely has to do with setting an appropriate LR, as discussed in Section 1.1.2. This occurred despite efforts to manually choose an ideal value manually, as shown in Table 1.

The success of Conceptors when compared dDMPs is due to how this approach take time into account when solving for their connection weights. Conceptors modify the signal to approximate it's output. Conceptors achieve this by mapping from individual oscillating neuron activities to an output. However, dDMPs instead depend on mapping an oscillator directly onto an output signal, which gives extremely good results as long as the reference oscillator can achieve the frequency of the output signal to a good enough degree, which was not possible with the 15Hz and 20Hz signal.

The abysmal performance of aDMPs is due to the fact that they didn't have the aforementioned benefit possessed by dDMPs and Conceptors which allowed them to compress repeating information. The aDMP attempts to optimize for each bump of a sinusoid separately.

Despite the results shown in Figure 6. No evidence was found for Conceptors being able to approximate jagged signals in any way, regardless of number of neurons, LR, sparsity of recurrent weights, aperture values, frequency of signal, neuron memory or signal magnitude range. This was tested by setting the input signal to a saw-tooth wave. Even when filtered with a low-pass filter, the Conceptor still failed to replicate the oscillation, as shown in Figure 10. This

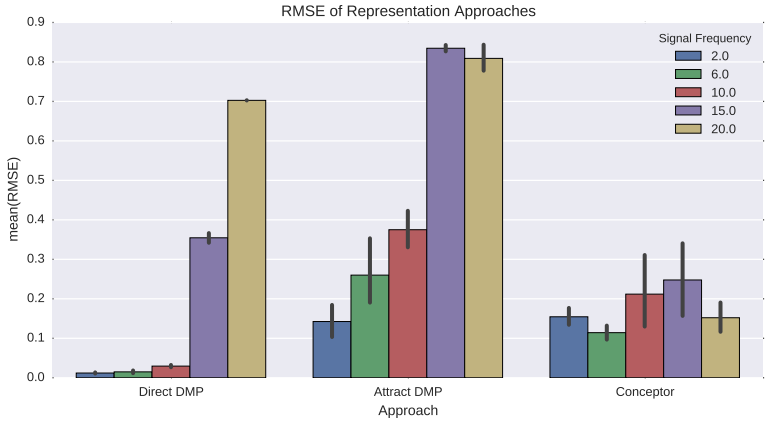


Figure 8: RMSE of various signal representation approaches across different frequencies of a sinusoid signal. Ticks on bars represent bootstrapped 95% confidence intervals.

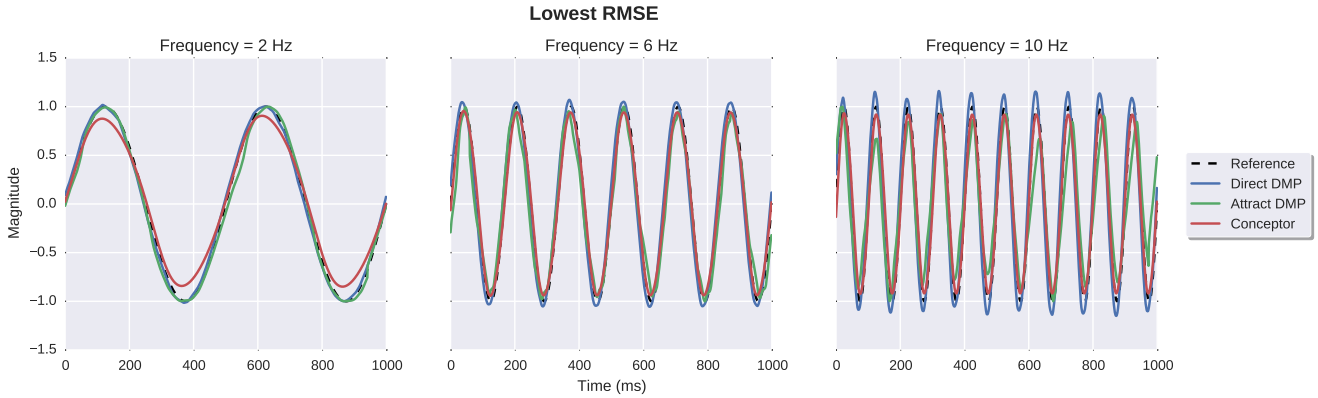
is probably because the activities of the neurons were too limited. Given that the saw-tooth wave can only be expressed as an infinite series of sinusoids, the frequencies required were not found during the Conceptor calculation and the approximation failed.

3.2 Blending Between Signals

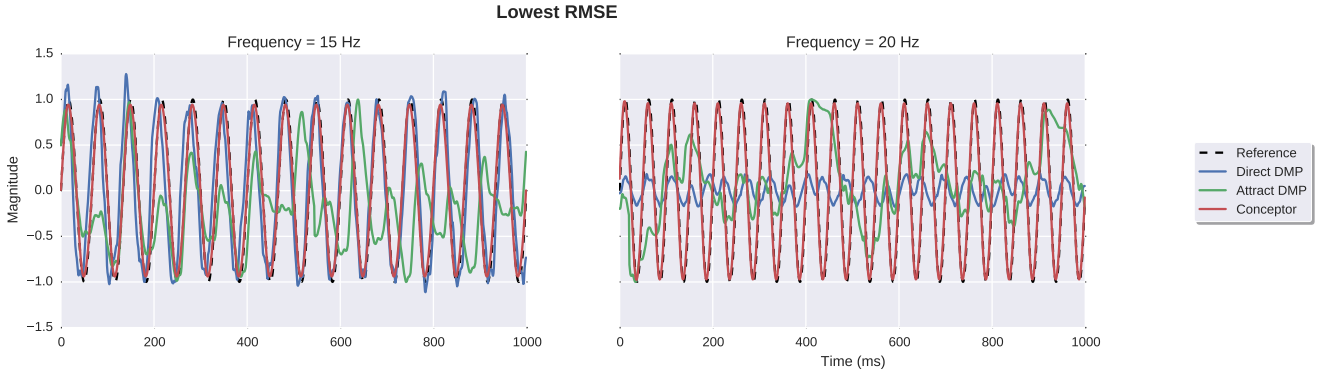
The blending, shown in Figure 11, accomplished by Conceptors by changing μ linearly from 1 to 0, is distinct from the blending accomplished with dDMPs. Additionally, it does seem to replicate the results in Figure 6, other than the various problems covered in the previous section. However, once the frequency of the two target signals was increased, blending also caused a reduction in signal magnitude at the interim frequencies to the point that no signal is output (Figure 12), showing that the description of blending being frequency control used in [5] is too simplistic.

It is unclear whether this blending behaviour is actually useful. When scaled up to 61D to represent a human skeleton [5], where each Conceptor controls a dimension which represents an angle between joints with no external forces acting on the skeleton, the transition between various movements patterns appear jerky and hesitant [4]. This is even after much fine manual tuning of the transitions between different patterns. Consequently, final judgment on what an ideal blending looks like should be withheld until an application is chosen, such as human transitions between movement patterns [1]. The investigation of these various applications and comparison to the various blending methodologies are outside of the scope of this report.

[4]To examine the output, see the video `https://www.youtube.com/watch?v=DkS_Yw1ldD4`



(a) Generating lower frequencies.



(b) Generating higher frequencies.

Figure 9: Generated signals at each frequency with the lowest RMSE for each representation method. Note that dDMPs can occasionally generate excellent signals even in higher frequencies, but the challenge is figuring out a means to do so reliably.

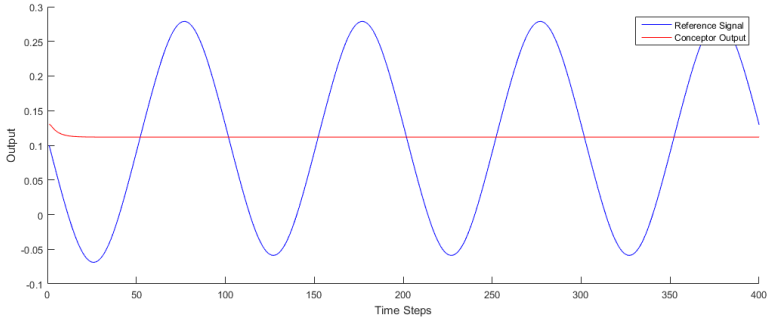


Figure 10: Conceptor failing to replicate the input signal of a low-pass filtered sawtooth wave.

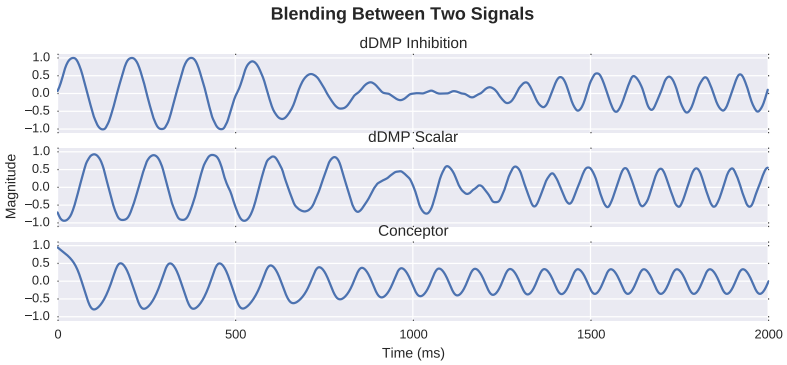


Figure 11: Comparison of approaches to blending between two signals. The transition occurs between 0.5s and 1.5s.

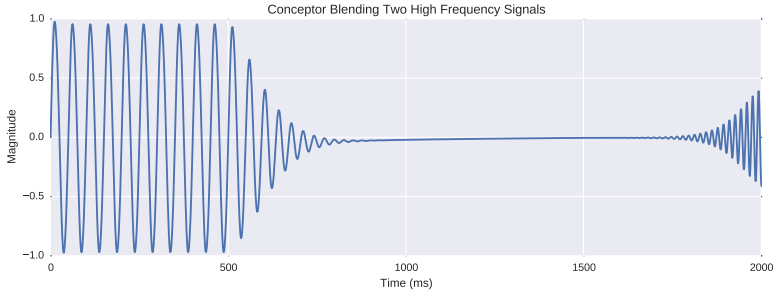


Figure 12: Drastic loss of magnitude when using Conceptors to blend between two high frequency signals. The transition occurs between 0.5s and 1.5s as before.

4 Discussion

This report demonstrated that the dynamic signal representation and combination, previously accomplished Conceptors, could be replicated to a certain degree in a biologically plausible manner using the NEF. It also highlighted the limitations of the various approaches when representing signals in the NEF. Ultimately, Conceptors far surpassed the NEF in representation of high-frequency sinusoidal waves, but were not able to represent non-sinusoidal periodic signals and required significant tuning. Future work should focus on transferring Conceptors to Nengo to see if switching between Conceptor matrices can be accomplished via inhibition and with spiking neurons, as well as exploring Conceptors using Systems Characterization to diagnose the cause of their various deficiencies.

The preliminary biological plausibility allows for many advantages, including comparison with neuroanatomical mapping. For example, the DMP populations of this model could map onto the Central Pattern Generators [2], which are neural circuits in the human spinal chord that assist with the creation of rhythmic movements. Obviously, a sophisticated model of human movement needs to be consulted, to match neurological data exactly, to store the patterns more realistically and to match the dynamics. That being said, at least the approach with the NEF offers this possibility to match this biological data, whereas Conceptors do not.

References

[1] Bruce Abernethy. *Biophysical foundations of human movement.* Human Kinetics, 2013.

[2] Simon M Danner, Ursula S Hofstoetter, Brigitta Freundl, Heinrich Binder, Winfried Mayr, Frank Rattay, and Karen Minassian. Human spinal locomotor control is based on flexibly organized burst generators. *Brain*, 138(3):577–588, 2015.

[3] Travis DeWolf. A neural model of the motor control system. 2015.

[4] Chris Eliasmith and Charles H. Anderson. *Neural engineering: Computation, representation, and dynamics in neurobiological systems.* MIT Press, Cambridge, MA, 2003.

[5] Herbert Jaeger. Controlling recurrent neural networks by conceptors. *arXiv preprint arXiv:1403.3369*, 2014.