Naval Research Laboratory

Washington, DC 20375-5320

NRL/FR/5750--18-10,317

Approximate Morphism via Machine Learning for an Electronic Warfare Simulation Component

Donald E. Jarvis

Advanced Techniques Branch
Tactical Electronic Warfare Division

August 14, 2018

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
14-08-2018	NRL Formal Report	4/2017 - 7/2017

4. TITLE AND SUBTITLE

Approximate Morphism via Machine Learning for an Electronic Warfare Simulation Component

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Donald E. Jarvis

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/FR/5750--18-10,317

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375-5320

10. SPONSOR / MONITOR'S ACRONYM(S)

NRL

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Electromagnetic waveforms are an essential component of high-fidelity radar and electronic warfare digital computer simulations. Sampled representations of radar waveforms are widely used for their physical realism and suitability for algorithimic processing. However, this fidelity comes at a price because operations on radar waveforms are often a computationally costly simulation bottleneck. In this report, we propose a method for constructing a reduced, feature-based alternative radar waveform model component derived from a given high-fidelity component. The resulting model will be related to the original through an approximate morphism. The proposed method is illustrated with a highly simplified waveform model. Both linear and nonlinear approaches are considered; in particular, a role for machine learning techniques is identified.

15. SUBJECT TERMS

Electronic warfare; Machine learning; Manifold learning; Morphism; Radar waveforms; Signal space; Simulation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Donald E. Jarvis
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	SAR	20	**19b. TELEPHONE NUMBER** *(include area code)*
Unclassified/A	Unclassifed/A	Unclassified/A			202-404-7682

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

This page
intentionally
left blank

CONTENTS

This page
intentionally
left blank

APPROXIMATE MORPHISM VIA MACHINE LEARNING FOR AN ELECTRONIC WARFARE SIMULATION COMPONENT

1. INTRODUCTION

The idea of a collection of related models, all describing the same entity but at different levels of fidelity, has been a theme in modeling and simulation for decades. The underlying motivation can range from pragmatic considerations such as reducing computational cost, to scientific considerations such as eliciting the essential features of a model from an overwhelming body of details. Different approaches to this general concept include approximate morphism [1, 2] and multi-resolution modeling [3]. While Davis and Bigelow [3] warned against the "insidious" errors that can result from assuming that low-resolution models must contain only a subset of the information in high-resolution models, they also presciently pointed out the role that data mining (which we take as a contextual synonym for machine learning) may play in the discovery of useful aggregations.

In this report we propose a framework for the semi-automated derivation of a *reduced* model from a higher-fidelity *reference* model. The reference and reduced models are related by an approximate morphism in a sense defined in the report. The morphism and consequent reduced model is arrived at through machine learning methods. The framework is demonstrated with a simple example adapted from electronic warfare modeling and simulation.

1.1 Conceptual Discussion

The ability to model a phenomenon or entity at multiple levels of fidelity is a core concept in the theory of modeling and simulation. However, the derivation of a simplified model from a more complex one is a challenging problem that requires careful analysis. Here we describe the key concepts of our approach, illustrated with the concrete example of a radar waveform.

Radar waveforms play a critical role in electronic warfare (EW) modeling and simulation. In traditional high fidelity simulation, this waveform is represented with a sampled video representation that respects the Nyquist criterion.[1] This representation can readily have different digital signal processing (DSP) algorithms applied to it that correspond either to physical effects while it is propagating, or to actual processing algorithms applied after the signal has been captured by a receiver. For example, the $1/R^2$ signal strength falloff with distance traveled can be modeled by simply scaling all the waveform samples by the spreading loss factor; the effect of multipath or of scattering from a complex object can be modeled with convolution; the range to a return can be estimated by way of a matched filter; etc. In the problem of interest in this report, such DSP operations represent the bulk of the computational cost in a simulation.

The goal of this research is a general method for discovering an alternative, *reduced* realization of a model component such as the waveform described above. This realization is reduced in the sense that it

Manuscript approved July 26, 2018.

[1] In fact, a sampled representation of the complex envelope would be used in practice; this distinction is immaterial for the purposes of this report.

has lower computational cost: the algorithms that operate on it run faster, and it may also enjoy a smaller memory footprint. The reduced waveform representation and algorithms are not required to have a straightforward interpretation, and, indeed, they are expected not to; the only requirement is a good behavioral match to the reference.

It is reasonable to question whether such a reduced realization exists. The DSP operations such as those outlined above, while expensive, have a close connection to the underlying physics, which by its nature captures necessarily relevant phenomena. Therefore it appears that these expensive algorithms are unavoidable.

On the other hand, it is a priori plausible that reduced realizations might exist. For example, while a sampled video waveform can be represented in a physics-based model with hundreds of floating point quantities, it was originally generated in terms of only a handful of waveform descriptor parameters, and it is processed down by a receiver to a few descriptive parameters. This is a strong basis for questioning whether the "fat middle" of the calculation is absolutely necessary—the model may actually have many fewer degrees of freedom than its rich sampled representation suggests.

As a thought experiment, consider a square pulse described not with a sampled representation but in terms of its amplitude, frequency, and duration parameters (Fig. 1). This is still a complete description of the pulse in the sense that the high-fidelity sampled representation can be constructed from this information (assuming we are also provided with the formula these parameters appear in). Observe that signal strength falloff can be implemented with a simple adjustment of the amplitude parameter, instead of laborious scaling of all the samples. Similarly, for many applications, Doppler shift can be modeled by an adjustment to the frequency parameter, again instead of a DSP operation on a sampled representation. However, if other operations on the waveform are required, this simple parameter adjustment approach breaks down. For example, the reflection of a square pulse from a spatially extended object is not a square pulse anymore, and so has no adequate description in terms of the three square pulse parameters above. Therefore, another way to describe the goal of this research is a general method to discover a small collection of parameters rich enough to adequately describe *all* waveforms that may arise in a particular EW scenario (e.g., one rich enough to include both transmitted and reflected pulses), along with processing algorithms that operate on this reduced representation. In addition, the algorithms are expected to be of greatly reduced computational complexity, i.e., they will be more in the spirit of parameter adjustments than DSP algorithms. The research hypothesis is that a smaller amount of data in the representation can enable computationally smaller algorithmic operations.

The framework proposed here provides automation support for the discovery of such reduced representations and their concomitant algorithms by posing the problem in a form amenable to powerful new techniques emerging in machine learning research.

1.2 Related Work

Algorithms for model reduction have a long history. Here we describe a few examples of special relevance to the work at hand.

A classic reference on simplification of discrete-event systems is Sevinc and Foo (1990) [4]. It relies critically on a distance metric between pairs of states, even noting that the state space often has a natural topology that should be respected in the simplification. It achieves state space size reduction via state

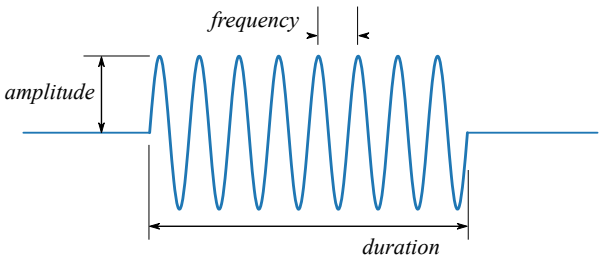

Fig. 1 — A square pulse described in terms of amplitude, frequency, and duration parameters. Operations that produce another square pulse as output can often be described simply in terms of an adjustment to these parameters. However, when the operation produces a qualitatively different pulse type, the parameters here are not adequate to describe it.

merging. The algorithm is in two passes. First, states are grouped together into clusters or abstract states according to the distance metric. This step may introduce minor inconsistencies. In the second pass, any inconsistencies (detected by their violation of the conditions for exact homomorphism) are resolved by relocating states to different clusters.

Saadawi et al. (2016) [5] propose a method for incrementally learning a reduced model from a continuous-system reference model in the context of the Discrete Event System Specification (DEVS) framework. Working in DEVS [1] has the advantage of providing all the data needed for the learning task, without affecting the structure or behavior of the reference model. The method applies to atomic models, but the greatest advantage is expected for complex coupled models (which are formally a kind of atomic model). The authors show how linear regression may be used to predict output and future state from input and current state, and how this learning algorithm may be upgraded to more powerful methods such as artificial neural networks as warranted by the problem at hand.

A sophisticated body of techniques has been developed for the discretization of continuous systems for the purpose of control system design; representative examples include Tabuada (2009) [6], Reissig (2011) [7], and Tarraf (2012) [8]. In particular, a specific notion of approximate homomorphism has appeared for model reduction in approaches based on the optimality principle [9, 10], the original source of the "curse of dimensionality" [11].

Besides the model reduction problem considered here, dimensionality reduction techniques applied to radar waveforms have other applications, such as rapid waveform classification [12]. More broadly, data compression can aid the movement of signal representations over a network in a distributed simulation.

Unless the behavior of a model is exactly preserved in the reduction step, the critical question of the validity of the resulting model must be acknowledged [13, 14]. Indeed, it is often (and perhaps always ought to be) a validation criterion that guides the reduction process. Verification and validation is application-specific and so will not be directly addressed in this report; however, its importance is too great to leave it entirely unmentioned.

2. THE REFERENCE WAVEFORM MODEL

Simulation of radar and electronic warfare scenarios relies heavily on the interchange of electromagnetic (EM) waveforms. Here we develop a simplified EM waveform model component for EW simulation. This example will serve as the high-fidelity reference model from which a reduced model will be derived through the approximate morphism framework. It was selected to favor clarity of exposition over concrete realism.

The waveform model consists of a waveform *representation* as well as *algorithms* that operate on that representation. A sampled representation of these waveforms for simulation is a well-established practice (e.g., see Mitchell (1976) [15]). The algorithms correspond to operations on the waveform as it propagates through the environment. A waveform in an EW simulation follows a life cycle that typically consists of the following stages:

1. Synthesis of a sampled waveform from a collection of descriptive parameters. For example, a linear frequency modulated "chirp" pulse [16] can be described in terms of a starting frequency and a chirp rate. Samples of the waveform can be computed using a mathematical formula that includes the descriptive parameters.
2. Propagation of the waveform through an environment. This includes effects such as spreading loss, multipath, atmospheric attenuation, and other effects.
3. The waveform may be scattered or reflected zero or more times. In some EW simulation applications, the ability to model the reflection of an EM waveform from a scattering body is essential.
4. Scattered and propagated waveforms are eventually incident on a receive antenna.
5. The receiving system applies signal processing algorithms to the received waveform to elicit meaningful information from it.

We observe that these algorithms fall into three broad types: (1) synthesis, which has a few descriptive parameters as input and a sampled waveform as output; (2) processing, where a sampled waveform input is transformed into a sampled waveform output; and (3) analysis, which takes a sampled waveform as input and produces a few descriptive parameters as output. Consequently, our notional reference model will represent the wide range of algorithmic possibilities with three exemplars: (1) a synthesis algorithm, (2) a convolution processing algorithm, and (3) a classifier analysis algorithm.

2.1 Synthesis

In our reference model, the synthesis algorithm is simply the application of a formula to compute samples of a chirp waveform given its descriptive parameters of starting frequency f_0 and chirp rate γ (additional constants include the sample period T, and the waveform duration implicit in the number of samples) (Fig. 2).

$$x_k = \sin 2\pi \left(f_0 + \tfrac{1}{2}\gamma kT \right) kT \tag{1}$$

2.2 Processing

In general, a waveform may undergo a variety of processing effects (spreading loss, scattering, etc.) on zero or more occasions as it propagates through an EW simulation. In the notional reference model, we consider only a single processing algorithm, namely, convolution of the signal x with a fixed two-tap kernel (Fig. 3) for a result as shown in Figure 4. This processing algorithm is representative of the effects of scattering or multipath.

$$x * (2\delta(n-n_1) - 3\delta(n-n_2)) \tag{2}$$

2.3 Analysis

In our reference model, the analysis algorithm is a classifier. A classifier attempts to determine key features of a waveform. In this model, we consider the interesting difference to be between transmitted and scattered waveforms. We selected a Fisher linear discriminant [17], which attempts to distinguish between a waveform that has never been through the convolution from one that has (Fig. 5).

2.4 Data as System, System as Object

The waveform element described above would normally be thought of as data that is operated on by a *system*, which produces an output dependent on input and internal memory. In a block diagram depiction, a waveform would be data on an input wire that is operated on by a block—representing the system—to produce processed waveform data on the output wire.

However, we observe that the waveform element itself can also be understood abstractly as a system (similar to the state machine of computer science [18] or the discrete-time dynamic system of optimal control theory [19]). The sampled waveform data is a vector representing its persistent internal state. Processing algorithms are invoked by inputs that advance the state from one configuration to the next (for example, in our reference model, the convolution operation advances waveform state from a pure chirp vector as in Figure 2 to a reflected chirp as in Figure 4), and analysis algorithms compute outputs as a function of state. Furthermore, it is clear that this viewpoint of waveform-as-system does not conflict with the waveform-as-data viewpoint above, but complements it.

We also observe that, while the representation-plus-algorithms description of the waveform model has much in common with the definition of an object in the usual software engineering sense, they are not quite equivalent. For example, since the behavior of a system (as classically defined) should depend only on its inputs and internal state, the implementation of a system in software should not maintain references or pointers to mutable structures that lie outside of it, which software objects commonly do. Therefore, we can think of the representation-plus-algorithms description as a software object with special restrictions, the details of which we will not belabor here. (For a discussion of some of the issues involved see chapter 3 of Abelson and Sussman (1996) [20].)

3. MODEL REDUCTION FRAMEWORK

Viewing the waveform model above as a dynamical system, it can be represented symbolically as follows. The state is a vector x of waveform samples. In general, the system has a state advance function

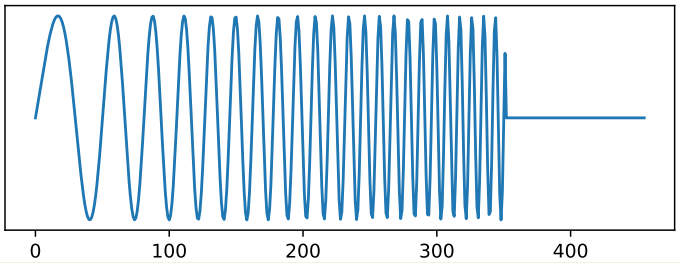

Fig. 2 — An example of a chirp waveform as it would exit from a transmitter, prior to being modified by multipath or scattering. These waveforms can be modified in their starting frequency and chirp rate parameters.

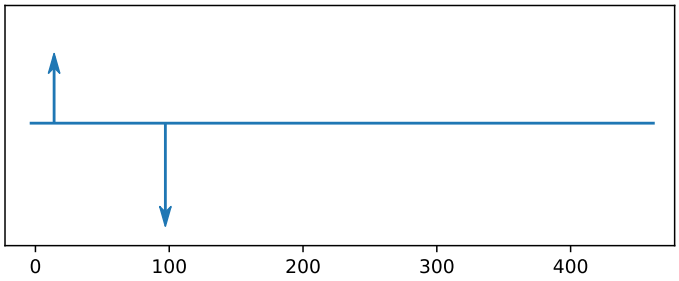

Fig. 3 — The simple two-tap impulse response kernel used in the processing step

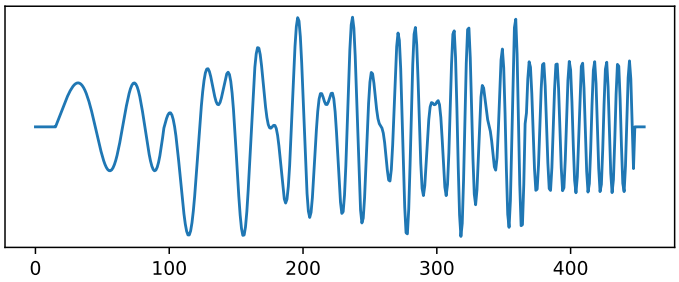

Fig. 4 — An example of a reflected waveform, the result of convolving the chirp in Figure 2 with the impulse response in Figure 3

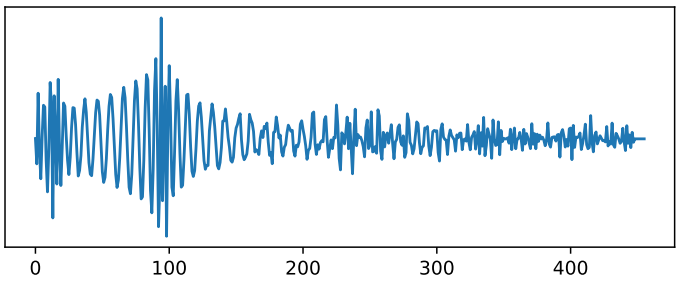

Fig. 5 — This Fisher linear discriminant is designed such that its dot product with a reflected chirp such as Figure 4 is greater than zero, and its dot product with a pure chirp such as Figure 2 is less than zero

mapping the current state x to the next state $f(x,u)$, where the input u specifies a processing operation. (In the case of our example model, there is only one processing operation, and it takes no arguments, so u is not strictly necessary, but we will preserve it as a placeholder for the more general case.)

We also allow our system to have an output function $y = g(x)$ which returns results of the analysis algorithms. Again, while the output can provide several descriptive parameters in general (i.e., y in general is a vector containing several parameters), our example model has only one analysis operation; therefore the output y contains a single quantity only, namely an indication of whether this waveform has been scattered or not.

Finally, our system also has a synthesis function $x_0 = s(p)$ which, from the system viewpoint, is merely a utility for expressing an initial state with desired properties. For our example, the initial parameters are starting frequency and chirp rate, so $p = (f_0, \gamma)$.

On these functions s, f, g a morphism μ [21] can be defined as follows.

$$\begin{aligned} \mu(s(p)) &= \tilde{s}(p) \\ \mu(f(x,u)) &= \tilde{f}(\mu(x), u) \\ g(x) &= \tilde{g}(\mu(x)) \end{aligned} \tag{3}$$

These equations can be illustrated as commutative diagrams (Fig. 6).

The problem at hand can be understood as follows. Given a reference system described with functions s, f, g, construct a corresponding reduced system $\tilde{s}, \tilde{f}, \tilde{g}$ related to the reference via a state morphism μ. Furthermore, we do not expect to satisfy the morphism equations exactly in practice; a good engineering approximation (one that preserves model validity) is all that is required.

In the definition above, the morphism leaves certain quantities alone. It does not transform the inputs u of the state-advance function nor p of the state synthesizer or the output of g. In fact, the morphism μ operates solely on state. This has implications for the software object viewpoint of the system: the inputs p and u and the output y are the same types in the reference and reduced systems. The internal states are different types, X in the reference system and $\tilde{X} = \mu(X)$ in the reduced, but internal state is private to a software object and

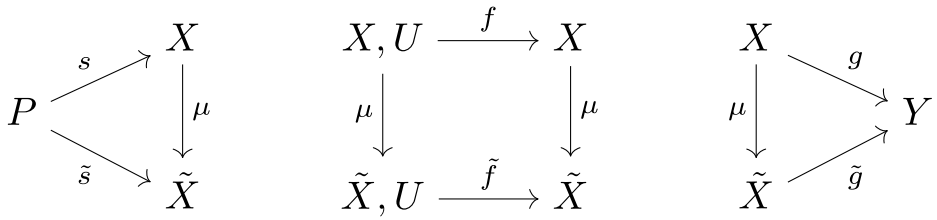

Fig. 6 — This diagram, an informal depiction of the relationships in Eq. (3), illustrates the synthesis s, processing f, and analysis g of high-resolution waveforms $x \in X$ in the reference model; the corresponding synthesis $\tilde{s}$, processing $\tilde{f}$, and analysis $\tilde{g}$ of feature-based waveform representations $\tilde{x} \in \tilde{X}$ in the reduced model; and the morphism $\tilde{x} = \mu(x)$ relating them. Synthesis is from descriptive parameters $p \in P$, processing depends on input $u \in U$ in general, and analysis yields descriptive output parameters $y \in Y$.

not observable to clients of that object. Therefore, in principle, the reference and reduced software objects share a common interface and differ only in details of internal implementation; the reduced object can be substituted for the reference object in a simulation code without impacting the results.

Another implication is that examples of reference system behavior can be mapped via the state morphism to corresponding examples of reduced system behavior. This fact is key to the approach proposed in this report, namely to learn the reduced system functions $\tilde{s}, \tilde{f}, \tilde{g}$ from examples of their behavior, where these examples are mapped down from high-fidelity examples via the state morphism.

An outline of the procedure for the approximate realization of $\tilde{s}, \tilde{f}, \tilde{g}$ is as follows.

1. Collect examples of the different system operations using the high-dimensional representation in the reference model. (For example, this can be done by scripted exercise of the reference model, or sampling data from high-fidelity runs. The exact procedure for collecting this data is problem-dependent and beyond the scope of this report.) This collection of high-dimensional waveforms and operations on them, is called here the exemplar database.
2. Construct a mapping of the high-dimensional reference representation to a low-dimensional reduced representation. In practice this step is accomplished with an unsupervised learning algorithm. This mapping is effectively the state morphism.
3. Apply the state morphism mapping to the database of high-dimensional examples to yield a corresponding collection of system operations on the low-dimensional representation.
4. In a second machine learning step, apply appropriate supervised learning algorithms to generalize from these low-dimensional examples to construct general low-dimensional operations.

4. MODEL REDUCTION PROCEDURE

Now we describe our experiment in which the steps above were carried out in practice.

4.1 Collect Examples

First, we build up the exemplar database, which consists of several collections: an indexed collection of sampled signals, a synthesis collection associating chirp parameters with the resulting waveform, a processing collection associating an input waveform with its corresponding output waveform as regards the convolution operation, and an analysis collection associating a waveform with an indication of whether it is assessed to be a pure or convolved chirp.

For our example, the procedure for populating the exemplar database was as follows:

1. Choose values for the starting frequency and chirp rate parameters using a random number generator.
2. Synthesize a pure chirp signal using these parameters (in our experiment these signals are all 456 samples in length).
3. Add the pure chirp signal to the signals collection in the exemplar database; say this signal has index k in the collection of signals.
4. To the synthesis collection in the database, add an association between the chirp parameters and the signal of index k, to indicate that a synthesis operation with these parameters results in signal k.
5. Apply the convolution operation to the pure chirp, yielding a convolved chirp (also of length 456 samples).
6. Add this convolved chirp to the signals collection; say it has index $k+1$ in that collection.
7. To the processing collection, add an association between signals k and $k+1$ to indicate that applying the convolution operation to input signal k results in output signal $k+1$.
8. Apply the Fisher linear discriminant to the pure and convolved chirp signals, quantizing its output to $+1$ for reflected or -1 for pure.
9. It turned out that the Fisher linear discriminant had perfect classification performance on the examples considered; therefore in effect the information added to the analysis collection in the database is an association between k and -1 (indicating a pure signal) and an association between $k+1$ and $+1$ (indicating a reflected signal).

The numbered steps above were repeated 1800 times. This resulted in 3600 signals in the signal collection, 1800 entries each in the synthesis and processing collections, and 3600 entries in the analysis collection.

4.2 Unsupervised Learning of the Approximate Morphism

The next step is to apply an unsupervised machine learning algorithm for construction of an approximate morphism. An appropriate dimensionality reduction or manifold learning algorithm, such as Principal Components Analysis (PCA) [22], Isomap [23], or t-Distributed Stochastic Neighbor Embedding (better known as t-SNE) [24] are plausible candidates.

In this framework, selecting an appropriate state morphism is a substantial decision to be made by the analyst. The goal is to identify a mapping to a state representation which is inexpensive to store and to work with algorithmically. There is some degree of heuristic judgment in its selection.

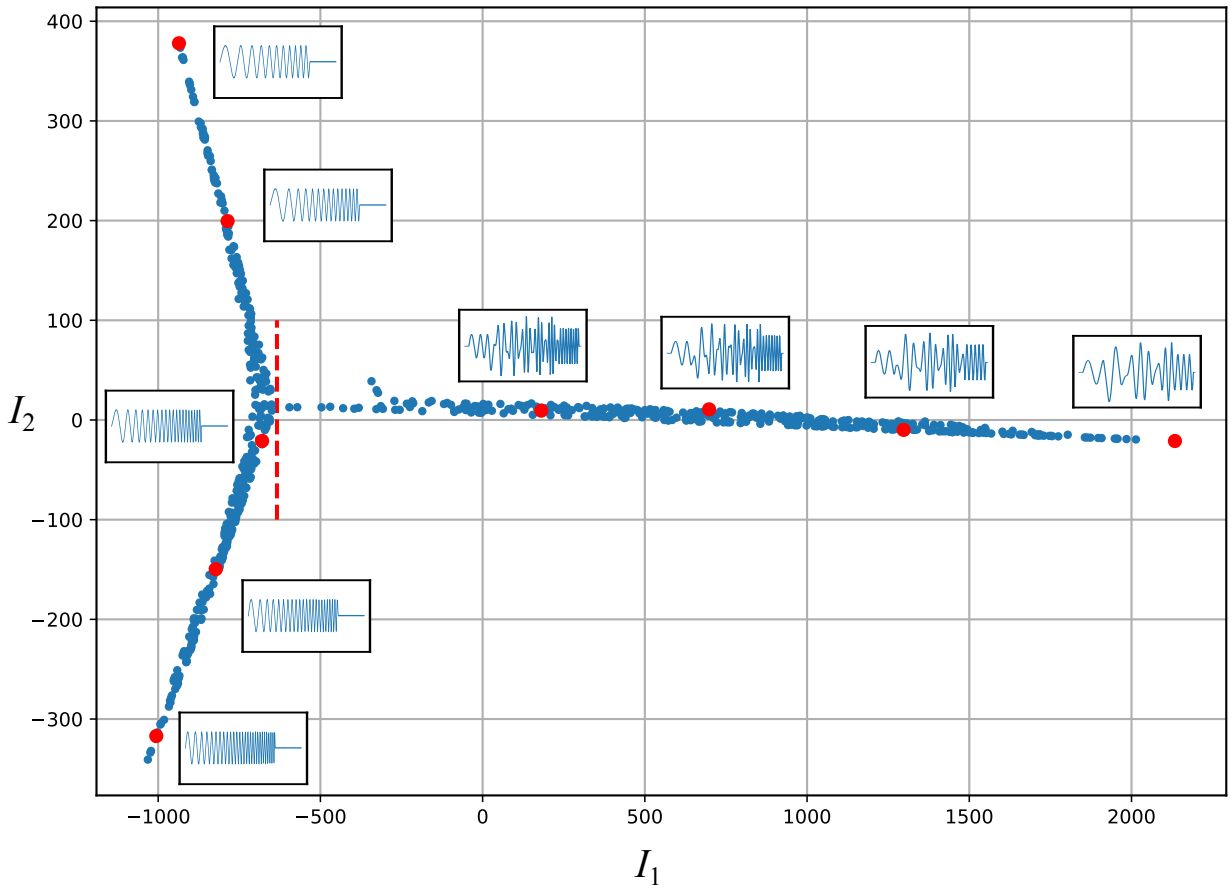

Fig. 7 — Result of using the Isomap algorithm to map 456-dimensional vectors (waveforms) to 2-dimensional vectors (points). Similar waveforms are mapped to nearby points. A few selected points are annotated with the waveforms they correspond to. (The dashed line illustrates the threshold described in section 4.3.3 of this report.)

In this example we will use the classic manifold learning algorithm Isomap to apply the morphism μ. Isomap seeks to preserve the intrinsic geometry of a data set in a continuous state space, especially as encoded in the local metric information. (This is similar in spirit to the approach taken in Sevinc and Foo (1990) [4], where the state space was discrete.) Our conjecture is that signals with similar underlying parameters, either of synthesis or analysis, will be geometrically nearby to each other; that the qualitative "smoothness" of an operation on the reference representation will be preserved in the reduced representation by the metric resemblance; and that "smooth" operations are easier for machine learning algorithms to approximate than highly convoluted ones.

In this work we applied the Isomap algorithm as implemented in scikit-learn [25] to the collection of 3600 waveforms. The high-dimensional waveforms each consisted of 456 samples, and so each could be interpreted as a point in 456-dimensional space. An output from Isomap of dimension 2 was accepted, because it appeared to capture the relevant information—and is also a convenient dimension for plotting results

in this report. Therefore, for the original collection of 3600 456-dimensional reference representations, we constructed a corresponding collection of 3600 2-dimensional (2D) reduced representations (Fig. 7).

An interesting characteristic of this framework is that the morphism itself never need be generalized from examples to a general state mapping rule. Isomap, for example, computes a low-dimensional vector corresponding to each high-dimensional vector, but (as originally described in Tenenbaum et al. (2000) [23]) does not provide a general algorithm for transforming *any* high-dimensional vector to the low-dimensional space. However, for the framework proposed here, such a general rule is simply never called for.

Another interesting characteristic is that the mapping of function examples in the exemplar database from high-dimensional reference representations to low-dimensional reduced representations involves no computation at all. Rather, it is accomplished simply by reinterpreting the indices in the synthesis, processing, and analysis collections as referring not to the collection of high-dimensional signals, but rather to the collection of corresponding low-dimensional signals.

4.3 Supervised Learning of Reduced Operations

The final major step in the framework is the learning of reduced operation algorithms $\tilde{s}, \tilde{f}, \tilde{g}$ from the low-dimensional training exemplars.

In this step, we found ourselves tinkering with different machine learning algorithms for different operations. At the current state of technology, machine learning is not general-purpose—different machine learning algorithms are appropriate to different domains—so it is reasonable to leave this selection to the discretion of the analyst.

Because the emphasis of this report is on the framework rather than any particular machine learning algorithms (not to mention that machine learning research is advancing at a phenomenal pace), we restricted ourselves to simple classical methods for function learning, and will not dwell unduly on the details of this step.

4.3.1 Learning a synthesis algorithm

The learned synthesis algorithm attempts to map the two chirp parameters (f_0, γ) (starting frequency and chirp rate, respectively) to the reduced, 2D representation computed by Isomap. The first Isomap coordinate I_1 was computed with a simple neural network (2-element input layer, 4-element hidden layer, 1-element output layer) (Fig. 8). The second coordinate I_2 was even simpler: it was found that it could be computed with a linear regression on the two input parameters (Fig. 9).

For an example of what the learned synthesis algorithm does, consider inputs of starting frequency 1.8 and chirp rate 5.7. Looking up the value at $(1.8, 5.7)$ on the learned synthesis functions depicted in Figures 8 and 9, we can read out approximate Isomap coordinate values of -780 for I_1 and 200 for I_2. There is a selected point near $(-780, 200)$ on Figure 7, and the waveform associated with that point is a chirp with parameter values of approximately $(1.8, 5.7)$. This is the sense in which the synthesis functions, learned from and generalizing from the output of Isomap, constitute a quick algorithm for calculating the reduced 2D representation of a chirp waveform directly from parameters.

The performance of the learned synthesis algorithm against the reference is illustrated in Fig. 10. In that figure, each line segment connects the coordinate specified by Isomap with its approximation as computed

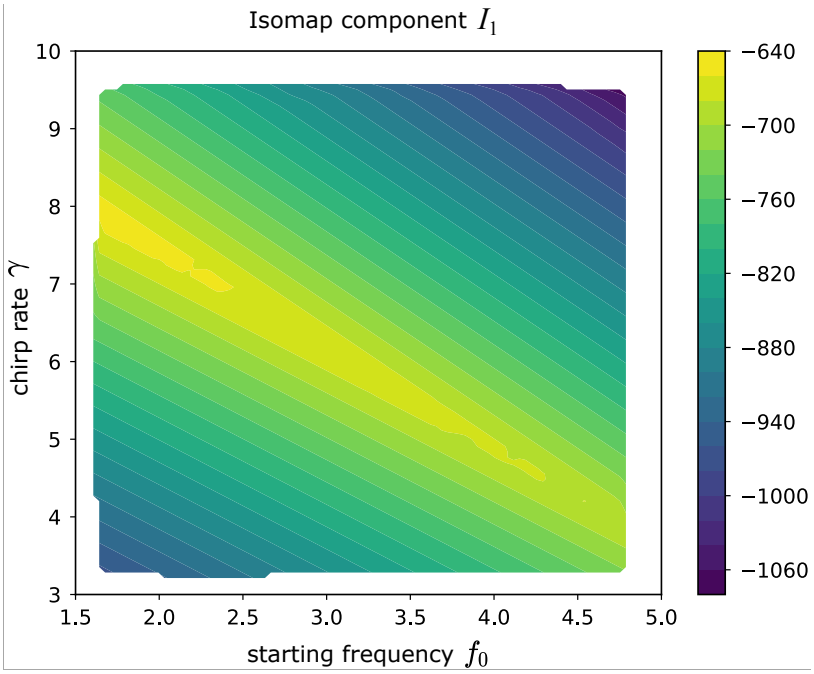

Fig. 8 — A plot of the mapping from synthesis inputs of starting frequency and chirp rate, (f_0, γ), to the first component I_1 of the 2D reduced representation, implemented with a neural network

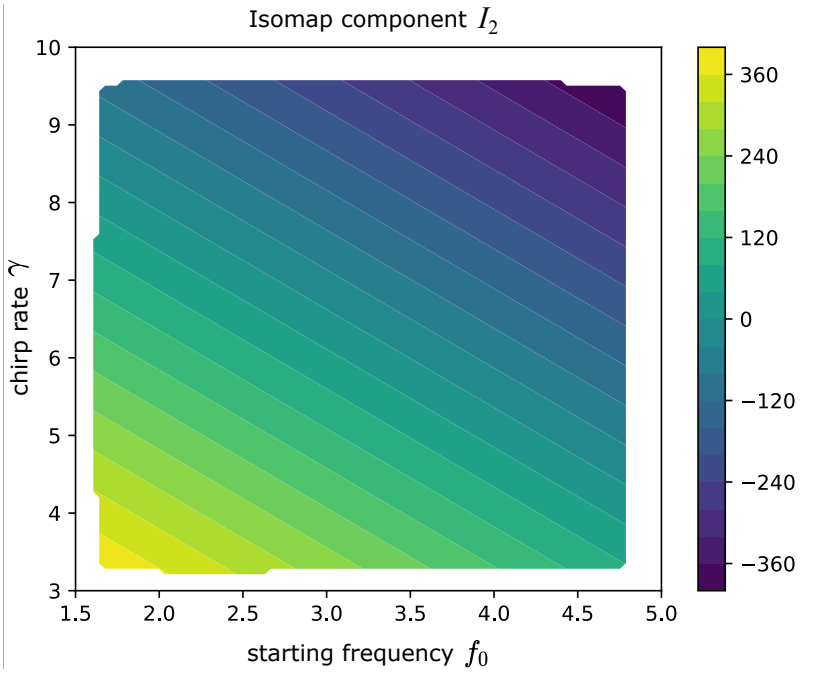

Fig. 9 — A plot of the mapping from synthesis inputs to the second component I_2 of the 2D reduced representation, implemented with linear regression

by the learned synthesis algorithm, for the same chirp parameter values. (In the ideal case, the Isomap mapping would be learned perfectly by the synthesis algorithm, and the line segments would all have length zero.)

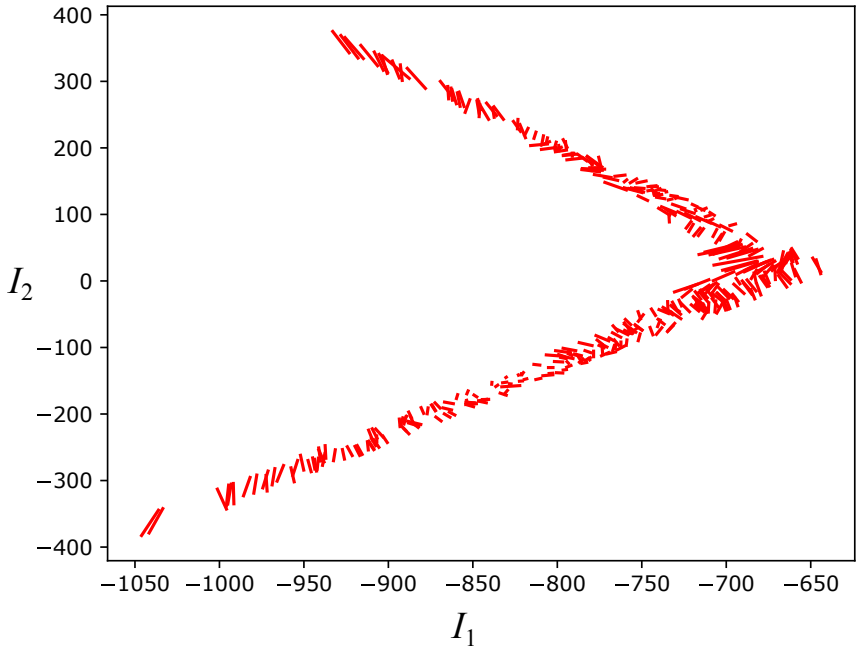

Fig. 10 — A plot illustrating the discrepancies of reduced 2D coordinate signal representations as arrived at by Isomap and their approximation by the synthesis algorithm learned from exemplars

4.3.2 Learning a processing algorithm

In the reference model, the convolution processing algorithm takes a sampled pure-chirp waveform such as Figure 2 and outputs a convolved waveform such as Figure 4. Here the task is to learn a function that takes the 2D coordinate corresponding to a pure chirp to the 2D coordinate corresponding to a convolved version of that chirp.

While a convolution itself is linear, we lose the advantage of this linearity when the signals are passed through the nonlinear Isomap state morphism. Yet, remarkably, it turned out that a linear regression-type function was well suited to learning this operation. In scikit-learn, it was implemented using the multi-layer perceptron because of that estimator's support for multiple outputs. The network architecture consisted of a 2-element input layer, a 1-element hidden layer, and a 2-element output layer. The activation function was set to the identity. The results clearly illustrate the mapping from pure to convolved chirps (Fig. 11).

4.3.3 Learning an analysis algorithm

A glance at the Isomap (Fig. 7) shows the pure and convolved waveforms clearly separated, which suggests that deciding between a pure and convolved chirp should be readily solvable by any of a number

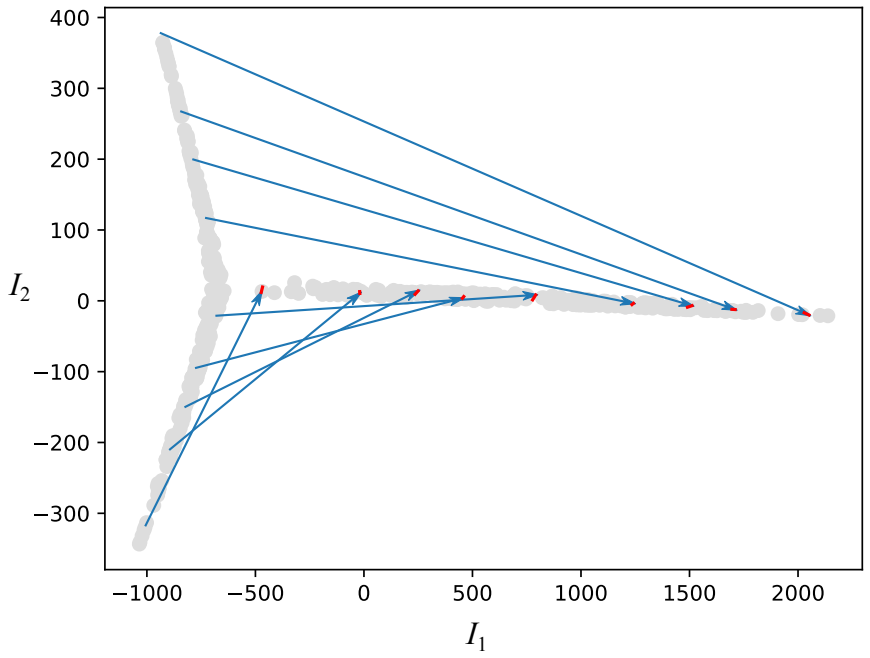

Fig. 11 — The arrows (blue) illustrate the convolution operation learned on 2D coordinates from exemplars. The line segments at the arrow endpoints (red) depict the discrepancy between the learned approximation and the original solution dictated by Isomap; ideal performance would correspond to all of these line segments having length zero. The general layout of the original data are indicated in gray.

of simple methods. For this algorithm, a DecisionTreeClassifier was applied. A depth of 1 (i.e., a single conditional) was enough to give decent performance. The decision tree compares the first reduced coordinate with a threshold of -633.854; signals that exceed this threshold are declared convolved; otherwise they are declared pure chirps.

5. RESULTS—PERFORMANCE OF REDUCED SYSTEM

There are two clear sources of error in the proposed approach, each associated with the approximations introduced by our use of machine learning.

First is the loss of information due to the morphism. This source of error can be understood by considering Figure 7 and asking, does the 2D point really capture everything relevant about the 456-D waveform?

While any unsupervised learning algorithm selected to carry out the morphism step is designed to preserve salient information and relationships, it is entirely possible that the algorithm discards impactful state characteristics. In part, this is because the unsupervised learning algorithm, as used in this framework, has no formal dependence on, or knowledge of, the reference implementation; it cannot "look inside" the s, f, g algorithms in an attempt to determine what must be preserved. Rather, any information about what is important in the reference model is brought back only indirectly and informally by analysts in their selection of an unsupervised learning algorithm.

The second source of error is in the approximations made by the supervised learning algorithms to the low-dimensional exemplars. In this report, an example of this error is illustrated in Figure 10. This error represents a common tradeoff: an estimator with more degrees of freedom can do a better job in matching the low-dimensional exemplar training data, but requires more data to train and is more computationally expensive to run.

Having acknowledged these sources of error, and that the waveform model used in this report is a simplified one, we evaluated the performance of the proposed framework as follows.

We randomly sampled 2000 starting frequency and chirp rate pairs (f_0, γ). So that both pure and convolved examples are represented, the even-numbered samples had the convolution processing applied and the odd ones did not—or, equivalently, f and $\tilde{f}$ were identity operations for this case. Then, for each input pair we checked whether the equation

$$g\left(f\left(s\left(f_0,\gamma\right),u\right)\right) = \tilde{g}\left(\tilde{f}\left(\tilde{s}\left(f_0,\gamma\right),u\right)\right) \tag{4}$$

holds (recall that the input u serves only as a placeholder in this example). This equation can be understood as sending the input through a high-dimensional pathway through the reference model on the left-hand side, and through a corresponding reduced low-dimensional pathway on the right-hand side. Both pathways consisted of applying the synthesis algorithm to the input parameters, convolution processing of the chirp (in the odd-numbered cases only), and analysis of the signal to determine whether it is pure or reflected.

The reduced model produced an error, i.e., Eq. (4) failed to hold, in 6 out of these 2000 trials, for an error rate of 0.3%.

For a comparative excursion, a similar experiment was performed using PCA to compute the morphism instead of Isomap, and is described here in summary form. Experimenting with PCA, a reduced representation of 60 dimensions was decided on (vs. 2 dimensions for Isomap). From exploratory plots of the functions to be learned for the synthesis algorithm, it was judged that approximating them with machine learning algorithms would provide no computational benefit, and so this step was skipped (no reduced algorithm for synthesis was computed; the reference algorithm was used for synthesis in both the reference and reduced models). The reduced processing algorithm was learned with a multi-layer perceptron, as in the Isomap case. Also as in the Isomap case, the analysis algorithm was a decision tree classifier, of depth 4 (vs. depth 1 for Isomap). In an evaluation like the one described above, this PCA-based reduced model produced an error in 30 out of 2000 trials, for an error rate of 1.5%.

So even with the advantages of (1) a "pass" on learning a synthesis algorithm, (2) the fact that the processing algorithm is a linear transformation, (3) a more powerful classifier, and (4) a considerably higher dimensional representation, PCA still produced a higher error rate than Isomap. In this experiment, PCA was outperformed by Isomap in constructing a useful morphism.

6. CONCLUSIONS

This report presented the problem of discovering a description of a waveform in terms of a small number of parameters, and algorithms that operate on it via computationally lightweight adjustments of those

parameters. The problem was illustrated with the square wave example in Figure 1. It was noted that finding a small parameter set that is rich enough to describe all the waveforms that arise in a simulation is a challenging problem.

This report proposed an innovative framework that uses machine learning to solve this problem. The reduced parameter set is arrived at by an unsupervised learning algorithm that plays the role of a morphism—in the reported example, these parameters are the coordinates computed by the Isomap manifold learning algorithm. The parameter-adjustment-style algorithms that operate on this reduced parameter set are learned from examples using supervised learning algorithms; our examples included formulas based on linear regression and small neural networks.

This research provides an approach to the goal of a reduced representation of EM waveform elements that may substantially improve the computational efficiency of radar and EW simulations. This general framework may also apply to other categories of models.

The framework involves a novel application for machine learning. Unlike other applications where computationally intensive methods are used to learn algorithms that are otherwise unknown (such as an optimal strategy for playing Go [26]), here the goal is to discover computationally lightweight (reduced) alternatives to known (reference) algorithms. Another novel aspect is the coupling between representation, and the algorithms that operate on this representation, as orchestrated by the state morphism.

Underlying sources of error in this framework were identified. Future research can take these limitations as a starting point: how can the learning problem be formulated in a unified way to increase the efficiency of the result? It is possible that such a formulation would require completely different machine learning methods.

The use of dimensionality reduction techniques such as Isomap are instrumental to the method described here. The experimental results suggest that these techniques can be of considerable interest in themselves for gaining insight into the EW signal space.

The role of the software interface to a simulation component was highlighted in its relation to the reduced representation. These observations apply regardless of how a reduced representation is arrived at (e.g., by machine learning methods or manual analysis). The problem of arriving at a common interface that applies at multiple levels of fidelity is a significant modeling and simulation challenge in its own right.

7. ACKNOWLEDGMENTS

We thank Jeff Byers, Jerry Gansman, Justin Manning, and Brian Rybicki for many helpful technical discussions.

REFERENCES

1. B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems* (Academic Press, second edition, 2000).

2. B. P. Zeigler, "The role of approximate morphisms in multiresolution modeling: Can we relax the strict lumpability requirements?," *The Journal of Defense Modeling and Simulation* (2016), doi: 10.1177/1548512916659826.

3. P. K. Davis and J. H. Bigelow, "Experiments in multiresolution modeling (MRM)," rept. MR-1004-DARPA, RAND Corporation (1998).

4. S. Sevinc and N. Foo, "Discrete event model simplification via state classification," in *AI and Simulation: Theory and Applications: Proceedings of the SCS Eastern Multiconference*, pp. 211–216 (April 1990).

5. H. Saadawi, G. A. Wainer, and G. J. Pliego, "DEVS execution acceleration with machine learning," in *Proceedings of the 2016 Spring Simulation Multiconference – TMS/DEVS Symposium on Theory of Modeling and Simulation, TMS/DEVS 2016* (April 2016).

6. P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach* (Springer, 2009).

7. G. Reissig, "Computing abstractions of nonlinear systems," *IEEE Transactions on Automatic Control* **56**(11), 2583–2598 (November 2011).

8. D. C. Tarraf, "A control-oriented notion of finite state approximation," *IEEE Transactions on Automatic Control* **57**(12), 3197–3202 (December 2012).

9. N. Jiang, S. Singh, and R. Lewis, "Improving UCT planning via approximate homomorphisms," in *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pp. 1289–1296, Richland, SC, 2014 (International Foundation for Autonomous Agents and Multiagent Systems).

10. B. Ravindran and A. G. Barto, "Approximate homomorphisms: A framework for non-exact minimization in markov decision processes," in *Proceedings of the Fifth International Conference on Knowledge Based Computer Systems (KBCS '04)* (2004).

11. W. B. Powell, *Approximate Dynamic Programming* (Wiley Inter-Science, 2007).

12. J. B. Collins, M. McCarrick, and W. Smith, "Investigating the application of machine learning techniques for the succinct representation of radar signals," NRL report in preparation.

13. J. J. Nutaro and B. P. Zeigler, "Towards a probabilistic interpretation of validity for simulation models," in *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, DEVS '15, pp. 197–204 (Society for Computer Simulation International, 2015).

14. D. Fraedrich and A. Goldberg, "A methodological framework for the validation of predictive simulations," *European Journal of Operational Research* **124**(1), 55–62 (2000).

15. R. L. Mitchell, *Radar Signal Simulation* (Artech House, 1976).

16. M. A. Richards, *Fundamentals of Radar Signal Processing* (McGraw-Hill, 2005).

17. R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification* (Wiley-Interscience, second edition, 2001).

18. M. Shields, *An Introduction to Automata Theory* (Blackwell Scientific Publications, 1987).

19. D. P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models* (Prentice-Hall, Englewood Cliffs, NJ, 1987).

20. H. Abelson and G. J. Sussman, *Structure and Interpretation of Computer Programs* (MIT Press, Cambridge, MA, second edition, 1996).

21. W. J. Gilbert and W. K. Nicholson, *Modern Algebra with Applications* (John Wiley & Sons, Hoboken, New Jersey, second edition, 2004).

22. I. Jolliffe, *Principal Component Analysis* (Springer, second edition, 2002).

23. J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science* **290**(5500), 2319–2323 (2000).

24. L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research* **9**, 2579–2605 (2008).

25. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research* **12**, 2825–2830 (2011).

26. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature* **529**(7587), 484–489 (Jan. 2016).