# Symbolic Regression of Multiple-Time-Scale Dynamical Systems

Theodore W. Cornforth
Department of Biological Statistics
and Computational Biology
Cornell University
Ithaca, USA 14853
1-607-254-8940
twc63@cornell.edu

Hod Lipson
Department of Mechanical
and Aerospace Engineering
Cornell University
Ithaca, USA 14853
1-607-254-8940
hl274@cornell.edu

## ABSTRACT
Genetic programming has been successfully used for symbolic regression of time series data in a wide variety of applications. However, previous approaches have not taken into account the presence of multiple-time-scale dynamics despite their prevalence in both natural and artificial dynamical systems. Here, we propose an algorithm that first decomposes data from such systems into components with dynamics at different time scales and then performs symbolic regression separately for each scale. Results show that this divide-and-conquer approach improves the accuracy and efficiency with which genetic programming can be used to reverse-engineer multiple-time-scale dynamical systems.

## Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

## General Terms
Algorithms, Design, Experimentation, Performance, Reliability

## Keywords
Genetic Programming, Symbolic Regression, Dynamical Systems, Evolutionary Algorithms, Time Series, Multiple Time Scales

## 1. INTRODUCTION
One of the most important applications of genetic programming is symbolic regression: the inference of both the form and parameters of an algebraic model for experimental data [13]. Many symbolic regression studies have focused on time series data, with the goal of forecasting [7,15] or the goal of reverse-engineering human-interpretable and parsimonious descriptive models [2,18]. Typically, symbolic regression is used to model time series as explicit linear or nonlinear algebraic equations [10,25], but recent advances have also improved the capabilities of symbolic regression for modeling time series as invariant equations [21] and ordinary differential equations [24].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*GECCO'12*, July 7–11, 2012, Philadelphia, Pennsylvania, USA.
Copyright 2012 ACM 978-1-4503-1177-9/12/07...$10.00.

A notable feature of many time-variant systems that has not been addressed in previous symbolic regression studies is the presence of multiple time scales. Systems with multiple time scales are those in which distinct patterns of behavior emerge as the system is viewed at different scales [11]. For example, the forced Van der Pol oscillator is used to model many natural and synthetic oscillating systems (Figure 1). The fast self-oscillations and slower forced oscillations are readily visible in the time series data for the dynamic variable $x$ as shown in Figure 1A. The frequency spectrum of this data confirms the qualitative observation that the dynamics consist predominantly of two time scales (Figure 1B).

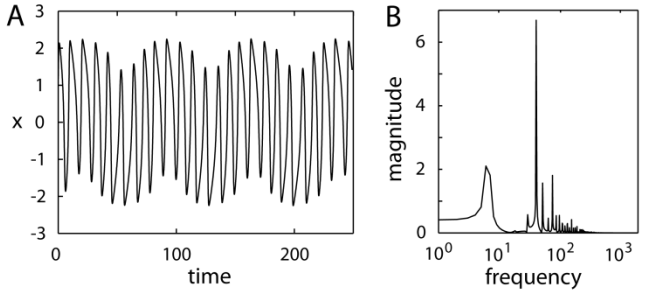



**Figure 1. A forced Van der Pol oscillator time series (A) and its corresponding frequency spectrum (B).**

Dynamical systems with multiple time scales frequently occur in both natural and artificial settings. Some examples include electrical bursting activity in neurons [8], the kinetics of enzyme catalysis [9], the motion of the cytoskeleton within cells [6], many problems in fluid mechanics [4], and spatio-temporal patterns of ecological diversity [17]. Although many previously proposed symbolic regression algorithms may be applicable as-is to the modeling of such systems, these algorithms do not incorporate information about multiple time scales explicitly. This discards an important feature of the system, yet an approach for integrating information about multiple time scales into a symbolic regression framework has not emerged in the literature.

Here, we propose a new method for leveraging information about multiple time scales in symbolic regression algorithms. The proposed algorithm is essentially a divide-and-conquer approach in which dynamics at different scales are separated and independently modeled. In the final portion of the algorithm, the form and parameters of a complete algebraic model describing dynamics at all scales are reconstituted. We demonstrate this approach for several example systems and show that the proposed

algorithm outperforms a state-of-the-art symbolic regression algorithm alone while using the same amount of computational effort.

## 2. METHODS

A summary of the proposed algorithm is shown in pseudocode in Algorithm 1. The algorithm proceeds in three main steps. In Step 1, the observed data is divided into small windows over which slower dynamics are approximately constant. Ordinary symbolic regression is performed on the data separately for each window and the results are stored in an unordered bag of expressions. In Step 2, the best expressions are removed from the bag and any numerical constants replaced with placeholder variables. The optimal value of these placeholder variables are found with traditional nonlinear regression such that the expression models the data within one of several windows selected from across the entire data sample. In Step 3, the optimal values of the placeholder variables found in Step 2 are treated as time series and an additional round of symbolic regression is performed to find the algebraic form of the expressions that model the evolution of the placeholder variables over time. This process is repeated for any number of scales in the data, limited only by the quantity of time series data.

```
Begin Algorithm_1(input DATA)
    Step 1:
        Divide DATA into a set W of (possibly
        overlapping) windows.
        For each window in W, perform symbolic
        regression and add output candidate models
        to bag of models B.
        Rank models in bag B and select a subset M of
        the best models. Denote the number of
        coefficients in model Mi as ki.
    Step 2:
        For each model Mi
            Replace the ki coefficients with ki
            placeholder variables {c1, c2, c3, ..., cki}.
            Divide DATA into a set V of (possibly
            overlapping) windows.
            For each window Vj
                Fit model Mi to the data in the window
                with nonlinear regression.
                Store fitted placeholder variables in
                row j, columns 1 ... ki of a table Ti.
                Also store the time value at the
                midpoint of window Vj in row j, column
                (ki + 1) of table Ti.
            End For
        End For
    Step 3:
        For each table Ti
            For each column q ∈ {1, 2, 3, ..., ki} in
            table Ti
                Perform symbolic regression with column
                q as the dependent variable and the time
                value in column (ki + 1) as the
                independent variable. Rank the
                candidate output models and choose the
                best to replace coefficient cq in model
                Mi.
            End For
        End For
        Return M
End Algorithm_1
```

**Algorithm 1.**

In the following three subsections below, each step of the algorithm is illustrated in greater detail with an extended example involving a synthetic Lotka-Volterra predator-prey system (Figure 2). We assume a partitioned problem setting, that is, both the hare ($h$) and lynx ($l$) data are given and modeling can be performed for either $dh/dt$ or $dl/dt$ separately without considering the entire system at once [2]. In this example, we will consider the task of inferring the correct expression for $dl/dt$ based only on the observed hare and lynx time series data.

A

$$\frac{dh}{dt} = \left\{\frac{\cos(0.05t)}{32.1} + 0.434 + \frac{t^{0.456}}{23.456}\right\}h - \left\{\frac{\sin(0.08t)}{100} + 0.051 + \frac{t^{0.201}}{122.456}\right\}hl$$

$$\frac{dl}{dt} = -(0.0020t + 0.8439)l + \left\{\frac{\sin(0.08t)}{100} + 0.051 + \frac{t^{0.201}}{122.456}\right\}hl$$

B

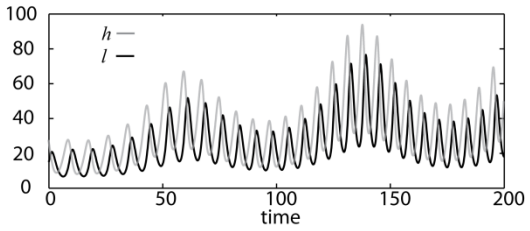



**Figure 2. A Lotka-Volterra predator-prey system with multiple time scales. A) A system of coupled differential equations equivalent to a classic Lotka-Volterra system, except that the coefficients of the hare (*h*), lynx (*l*), and hare-lynx interaction (*hl*) terms are time-dependent functions. B) Numerical integration of the system gives this solution for *h* and *l*.**

### 2.1 Algorithm step 1

The algorithm begins by estimating the time derivative of the observed lynx data and then dividing it into windows (Figure 3A). The size of the windows here were chosen so that the slow dynamics were roughly constant based on inspection, although more sophisticated alternative methods for choosing the window size are possible as discussed below. Over the time span of one window, such as the segment indicated with dashed lines, the slow dynamics contributed by the $l$ and $hl$ coefficients are roughly constant. There exists a far simpler model for the lynx data within this window that is still reasonably accurate (Figure 3B). Existing symbolic regression methods, such as the Eureqa algorithm [21] can be used to find window-specific expressions of this simpler form. Each window in the original data can be processed in this fashion, and the resulting expressions placed in an unordered bag. Most of the expressions in the bag will have similar forms, but all of them will have different constants depending on which window was used for their inference.

### 2.2 Algorithm step 2

The next phase of the algorithm begins by selecting an expression from the bag (Figure 4A) and replacing all constants with placeholder variables {a, b, c, ...}, resulting in a "general form expression" (Figure 4B). Consider again a small window of the original lynx data in which the values of the placeholder variables are approximately constant. Because the lynx and hare data are given, the exact values of the placeholder variables for which the general form expression fits the observed lynx data can be found with ordinary nonlinear regression.

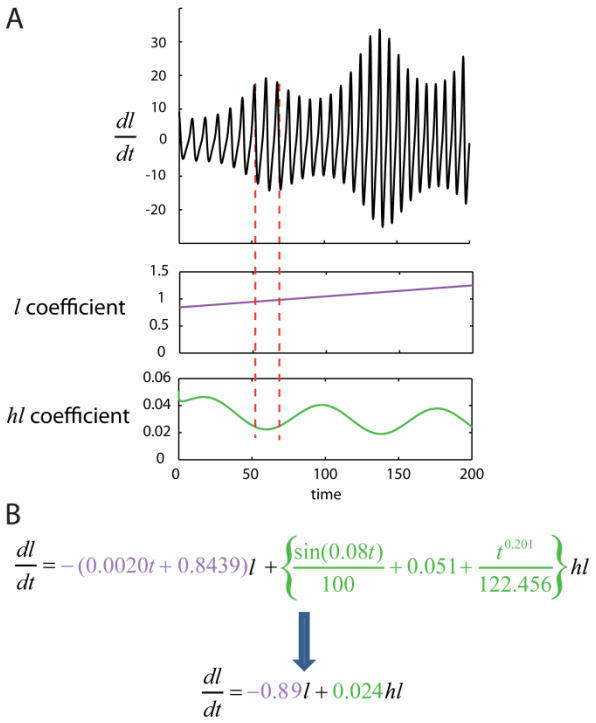



**Figure 3. Step 1: Initial processing of the observed data. A) The time derivative of the lynx data in Figure 2 is estimated and divided into small windows over which the slow dynamics are approximately constant. B) A much simpler model approximates the lynx dynamics within this window.**

Here we used the Levenberg-Marquardt algorithm with random-restarts [16] to perform this nonlinear regression step and to find good fits for the placeholder variables. Once again, these values are window-specific. However, as this nonlinear regression procedure is repeated for many windows that span the lynx time series data (Figure 4C), the initial estimates used at the start of each run can be set to the fitted variable values found in the previous window. This greatly accelerates the rate of achieving good fits and we found the computational cost of this nonlinear regression step to be small on a per-CPU-core basis compared with the computational effort required for symbolic regression.

In this example, window size was set to the same window size used in the first phase of the algorithm above. The number of windows was set to 100 because it was roughly the smallest number that provided adequate resolution for Step 3 of the algorithm. The final result from Step 2 of the algorithm can be expressed as a table in which there are columns for each placeholder variable plus one column for a time point (Figure 4D). Each row has one column for the best fit of each placeholder variable obtained from one window, plus one column for the time point at the center of that window.

## 2.3 Algorithm step 3

The final step of the algorithm processes the tables produced at the end of Step 2, one column at a time. Each column is interpreted as the dependent variable of a time dependent function and is processed once again with symbolic regression (Figure 5A).

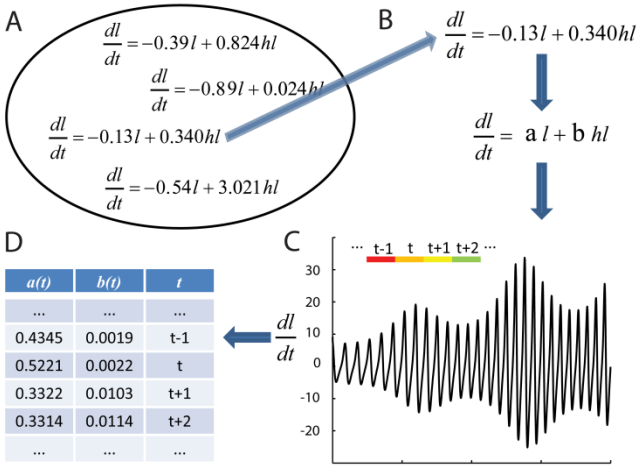



**Figure 4. Step 2: Using window-specific expressions to estimate slow dynamics. Window-specific expressions are taken from the bag (A) and converted into "general form expressions" (B). The placeholder variables (a, b, c, etc.) are fit for several windows across the data using ordinary nonlinear regression (C). The fit for each placeholder variable as a function of window location or time *t*, is stored in a table (D).**

As before, an existing symbolic regression algorithm such as Eureqa can be used to infer the form of this time dependent function, which finally can be substituted for the corresponding placeholder variable from which it was derived (Figure 5B). Repeating this procedure for all columns of the table finally results in the complete inferred expression for the observed lynx data (Figure 5C).

## 2.4 Symbolic regression

An important component of the algorithm proposed here is symbolic regression: the identification of an analytical, mathematical description of a black box system based on empirical data [13]. Symbolic regression is a well-established technique that identifies both the form and parameters of functional algebraic expressions, distinguishing it from methods such as linear and nonlinear regression in which only the parameters of otherwise fixed models are adjusted. In addition, symbolic regression does not rely on the induction or deduction of models based on encoded expert knowledge. Symbolic regression is effectively a search through the space of possible algebraic expressions, guided by the prediction error of candidate models on training data sampled from empirical observations.

Symbolic regression usually relies on population-based stochastic optimization techniques such as genetic programming to search through the space of possible expressions [5,13]. In genetic programming, expressions are typically represented as graphs, with nodes corresponding to primitive mathematical operations such as *add* or *cos*, the input variables of the function, or numerical constants. Mutation and crossover are used to introduce variation into the population throughout the search. Mutation of an expression graph changes a node to a node of a different type, whereas crossover involves the exchange of portions of two expression graphs. For example, mutation to the numerical constant node of a graph representing the expression

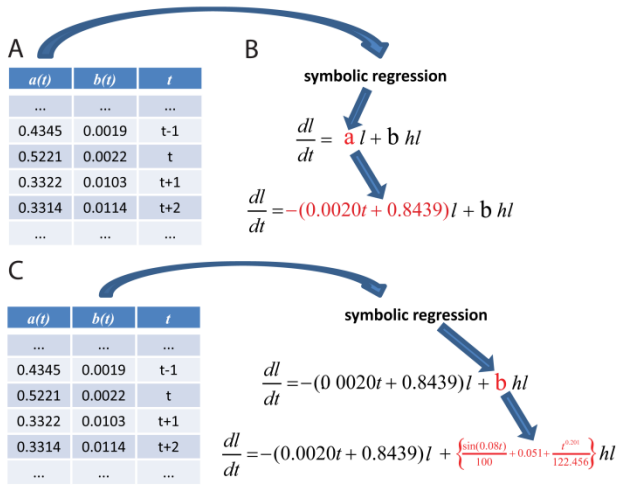



**Figure 5. Step 3: Completing model inference. A) Each column of the table obtained in the previous step is a function of the independent variable *t* and can be modeled with symbolic regression. B) Once the best candidate expression for the function is found, it is substituted for the corresponding placeholder variable. C) This process is repeated for every column in the table until all placeholder variables have been replaced and the final form for the expression *dl/dt* is obtained.**

$f_1(x) = x + 2.3$ might result in the expression $f_2(x) = x + 1.9$, whereas crossover between the expressions $f_3(x) = x^2 - 2x + sin(x)$ and $f_4(x) = x^3 + 5.7$ could produce the hybrid offspring expression $f_5(x) = x^2 - 2x + 5.7$. Search in symbolic regression is guided by a fitness objective designed to minimize error on the empirical training data. Here we define fitness as the mean absolute error (MAE):

$$\text{fitness}(f) = \frac{1}{N}\sum_{i=1}^{N}\left|f(x_i) - y_i\right|$$

where $f$ is a candidate algebraic expression, $x_i$ and $y_i$ are the training data inputs and outputs, respectively, and $N$ is the number of examples in the training data.

For symbolic regression in this work, we use the state-of-the-art Eureqa software [21]. All "symbolic regression alone" experiments used as controls below were also performed with Eureqa. Eureqa implements basic symbolic regression as described above as well as several non-standard techniques. These non-standard techniques include: a directed acyclic graph representation similar to that used in linear genetic programming [3,19], coevolution of rank predictors [20,22], and age-fitness optimization [23]. We refer the reader to these references for additional information on Eureqa and for extensive comparisons of Eureqa with other symbolic regression algorithms.

## 3. EXPERIMENTS

We compared the performance of the proposed algorithm with that of symbolic regression alone for two different modeling problems: a simple sum-of-sines test problem and a more complex simulated oscillating system. The equations used to simulate the training data sets for these two problems are the "target" expressions shown in Table 1A and B, respectively. Those equations were the targets for 50 independent trials of the proposed algorithm and 50 independent trials of symbolic regression alone. The same amount of computational effort, as measured by total number of symbolic regression fitness evaluations, was used in all trials for both of the modeling problems. All performance results below are based on test data generated from the target expressions but not used during model training. Gaussian noise ~$N(0, 0.01)$ was added to the training data for the oscillator problem.

**Table 1. Output models produced from the proposed algorithm and symbolic regression alone.**

**A) sum-of-sines**

target
f(t) = 0.67890 * sin(t) + 0.34560 * sin(100 * t)

best, proposed algorithm
f(t) = 0.67890 * sin(t) + 0.34560 * sin(100 * t)

median, proposed algorithm
f(t) = 0.67879 * sin(0.00011 + t) + 0.34580 * sin(99.98260 * t - 6.14881)

best, symbolic regression alone
f(t) = 0.67890 * sin(t) + 0.34560 * sin(100 * t)

median, symbolic regression alone
f(t) = sin(t) - 0.65618 * sin(t) * sin(-49.55108 * t) * sin(-49.55108 * t)

**B) oscillator**

target
dx/dt = y + x - $x^3$/3
dy/dt = 0.65430 * cos(3.14159 * z) - x
dz/dt = 0.01000

best, proposed algorithm
dx/dt = y + x - $x^3$/3.00049
dy/dt = 0.65432 * cos(-3.14181 * z) - x
dz/dt = 0.00995

median, proposed algorithm
dx/dt = y + x - $x^3$/3.00041
dy/dt = -0.65948 * cos(9.13489 - 3.06381 * z) - 0.99125 * x
dz/dt = 0.00997

best, symbolic regression alone
dx/dt = y + 1.00084 * x - $x^3$/3.00147
dy/dt = 0.65309 * cos(-3.14095 * z) - x
dz/dt = 0.00996

median, symbolic regression alone
dx/dt = y + 0.99894 * x - $x^3$/3.00302
dy/dt = z + 3.73327 * cos(z) - x
dz/dt = 0.01001

Along with the target expressions, representative models from the 50 trials are shown in Table 1. From each individual trial, the best model was selected. These 50 models were then ranked to find the best and median models, which are shown in Table 1. For both problems the proposed algorithm outperformed symbolic regression alone. This can be seen more clearly when looking at the mean absolute error of the best models produced in each of the 50 trials by the proposed algorithm and symbolic regression alone in Figure 6.

Two important factors likely to influence the accuracy and efficiency of the proposed algorithm are the presence of noise and the magnitude of differences in time scales in a system with multiple time scales. For example, a noisy system with dynamics at two very similar time scales is expected to be more difficult to model than a noise-free system with dynamics at two very different time scales. To test the influence of these factors, we

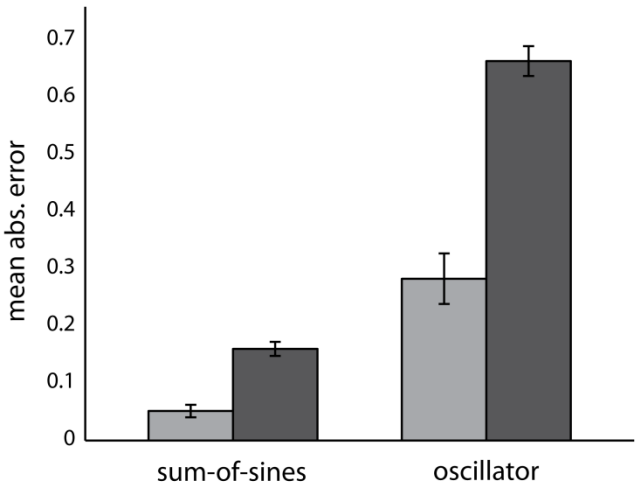



**Figure 6. Performance of the proposed algorithm compared with symbolic regression alone on two example problems. Light gray is the proposed algorithm, dark gray is symbolic regression alone, and error bars are ± standard error of the mean. In both cases, the difference between algorithms is significant by an unpaired t-test, $p < 0.01$.**

used the same sum-of-sines test problem shown above but added Gaussian noise and decreased the frequency difference between the two sinusoidal components of the signal. As before, 50 trials were run for both the proposed algorithm and symbolic regression alone with the same amount of computational effort used for both. A comparison of the performance of the two algorithms is shown in Figure 7. The proposed algorithm outperforms symbolic regression alone for low noise levels and a large degree of separation between the two time scales, but this advantage decreases as the noise level increases and the separation between time scales decreases.

## 4. DISCUSSION

We proposed a new divide-and-conquer approach to the modeling of dynamical systems with multiple time scales. The core of the algorithm is symbolic regression, yet we showed that the proposed algorithm outperforms symbolic regression alone for several example problems. Although we use only simulated problems here, we expect that similar results will hold for modeling with empirical data obtained from physical systems. Our efforts toward applying the algorithm to empirical systems are focused on systems seen in physics such as cantilever beams and physical oscillators [14], as well as systems arising in chemical kinetics [1] and ecology [17].

Perhaps the main assumption made by the algorithm relates to the windowing procedure. The correct choice of window size in Step 1 is important for the success of the algorithm and a priori information for making this choice would generally not be available. In the example above, the size of the window was chosen by inspection. However, hand tuning of the window is not an ideal solution and we are investigating methods for identifying optimal window sizes automatically. One possibility for systems with periodic dynamics is to simply take the frequency spectrum of the signal as in Figure 1B and estimate the optimal window length such that the fastest dynamics are preserved but the slower dynamics would appear approximately constant. This could be

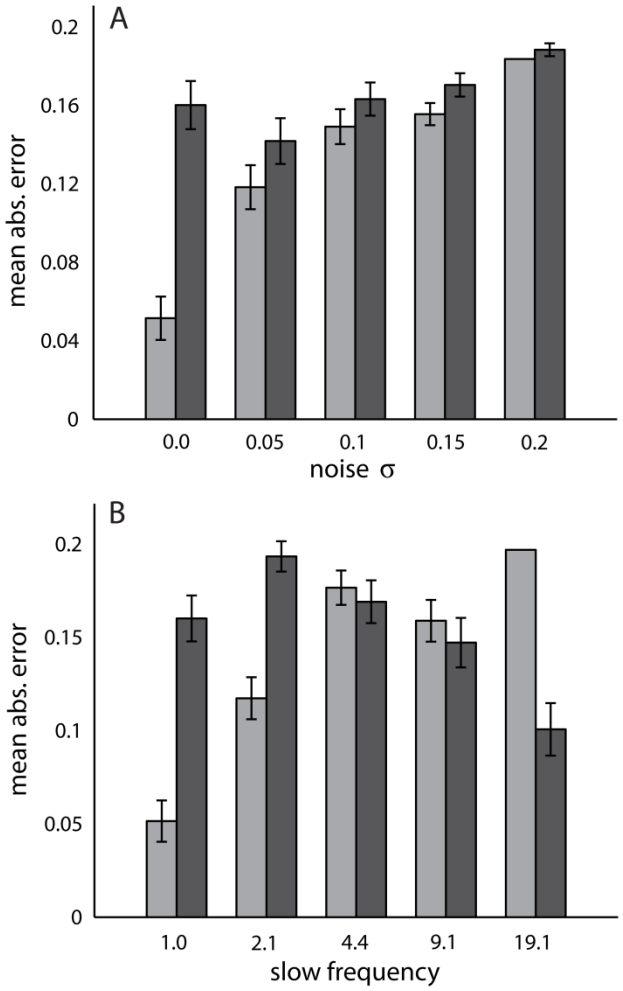



**Figure 7. Effects of noise and time scale separation on algorithm performance. To further study the performance of the proposed algorithm, the target for the sum-of-sines problem above was changed to:**

$$f(t) = 0.6789 * \sin(sf * t) + 0.3456 * \sin(100 * t) + N(0, \sigma)$$

**where *sf* is the frequency of the slow component and $\sigma$ is the standard deviation of Gaussian noise added to the training data. The effects on mean absolute error of altering *sf* are shown in A and the effects of altering $\sigma$ are shown in B. Light gray is the proposed algorithm, dark gray is symbolic regression alone, and error bars are ± standard error of the mean.**

repeated for the next fastest dynamics, etc. until window sizes appropriate for dynamics at each time scale are identified.

In the examples here, we consider only systems with two distinct time scales, but nothing about the proposed algorithm limits its applicability to systems with an arbitrary number of time scales. Instead, the limiting factors would be noise, the resolution of data, and the degree of separation between time scales. Typically the resolution of experimental data and the degree of separation between time scales would be determined before the application of the proposed algorithm, but denoising techniques such as unscented Kalman filtering [12] could reduce many of the

difficulties associated with corruption of the experimental data by noise.

For future work, we plan to extend the preliminary research discussed here in three ways. First, empirical data from physical systems will be used as discussed above. Second, we plan to implement an optimized and fully parallelized version of the hill climbing algorithm used in Step 2 of the proposed algorithm. Finally, the problems here were chosen to be relatively small so that several replicate trials could be performed with both the proposed algorithm and symbolic regression alone. We expect that the relative benefits of the proposed algorithm over symbolic regression alone will increase as the difficulty of the problem increases. We are currently testing this hypothesis to gain insight into how the performance of both algorithms scale with problem difficulty.

## 5. ACKNOWLEDGMENTS

Support was provided by the Tri-Institutional Training Program in Computational Biology and Medicine, NSF Grant ECCS 0941561 on Cyber-enabled Discovery and Innovation (CDI), and the U.S. Defense Threat Reduction Agency (DTRA) grant HDTRA 1-09-1-0013.

## 6. REFERENCES

[1] Argemí, A., Saurina, J. 2007. Study of the degradation of 5-azacytidine as a model of unstable drugs using a stopped-flow method and further data analysis with multivariate curve resolution. *Talanta* 74, 176-182.

[2] Bongard, J.C. and Lipson, H. 2007. Automated reverse engineering of nonlinear dynamical systems. *PNAS* 104, 9943-9948.

[3] Brameier, M. and Banzhaf, W. 2007. *Linear Genetic Programming.* Springer, New York.

[4] Chen, W. and Fish, J. 2001. A dispersive model for wave propagation in periodic heterogeneous media based on homogenization with multiple spatial and temporal scales. *J. Appl. Mech.* 68, 153-161.

[5] DeJong, K.A. 2006. *Evolutionary Computation: a Unified Approach.* MIT Press, Cambridge, MA.

[6] Deng, L., Trepat, X., Butler, J.P., Millet, E., Morgan, K.G., Weitz, D.A., Fredberg J.J. 2006. Fast and slow dynamics of the cytoskeleton. *Nat. Materials* 5, 636-640.

[7] Guven, A. 2009. Linear genetic programming for time-series modeling of daily flow rate. *J. Earth Syst. Sci.* 118, 137–146.

[8] Harris-Warrick, R.M. and Flamm, R.E. 1987. Multiple mechanisms of bursting in a conditional bursting neuron. *J. Neurosci* 7, 2113-28.

[9] Henzler-Wildman K.A., Lei M., Thai V., Kerns S.J., Karplus M., Kern D. 2007. A hierarchy of timescales in protein dynamics is linked to enzyme catalysis. *Nature* 450, 913-916.

[10] Jara, E.C. 2011. Long memory time series forecasting by using genetic programming. *Genet. Prog. Evol. Mach.* 12, 429–456

[11] Jones, C.K.R.T. (ed.) 2001. *Multiple-Time-Scale Dynamical Systems.* Springer, New York.

[12] Julier, S., Uhlmann, J., Durrant-Whyte, H.F. 2000. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Trans. Automat. Contr.* 45, 477-482.

[13] Koza, J.R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA.

[14] Le Pourhiet, A., Corrège, M., Caruana, D. 2003. Control of self-oscillating systems. *IEEE Proc. Control Theory Appl.* 150, 599-610.

[15] Mulloy, B.S., Riolo, R.L., Savit, R.S. 1996. Dynamics of genetic programming and chaotic time series prediction. In: Genetic Programming 1996: Proceedings of the First Annual Conference, Stanford University, CA, pp. 166–174.

[16] Nocedal, J. and Wright, S.J. 2006. *Numerical Optimization.* (2nd ed.) Springer, New York.

[17] Pandolfi, J.M. 2002. Coral community dynamics at multiple scales. *Coral Reefs* 21,13-23.

[18] Sakamoto, E. and Iba, H. 2001. Inferring a system of differential equations for a gene regulatory network by using genetic programming. In: Proceedings of the 2001 Congress on Evolutionary Computation, pp. 720-726.

[19] Schmidt, M.D. and Lipson, H. 2007. Comparison of tree and graph encodings as a function of problem complexity. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07), pp. 1674-1679.

[20] Schmidt, M.D. and Lipson, H. 2008. Coevolution of fitness predictors. *IEEE Trans. Evol. Comp.* 12, 736-749.

[21] Schmidt, M.D. and Lipson, H. 2009. Distilling free-form natural laws from experimental data. *Science* 324, 81-85.

[22] Schmidt, M. D. and Lipson, H. 2010a Predicting solution rank to improve performance. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO '10), pp. 949-956.

[23] Schmidt, M.D. and Lipson, H. 2010b. Age-fitness Pareto optimization. *Genetic Programming Theory and Practice* 8, 129-146.

[24] Schmidt, M. D., Vallabhajosyula, R. R., Jenkins, J. W., Hood, J. E., Soni, A. S., Wikswo, J. P., Lipson, H. 2011. Automated refinement and inference of analytical models for metabolic networks. *Physical Biology* 8, 055011.

[25] Wagner, N., Michalewicz, Z., Khouja, M., McGregor, R.R. 2007. Time series forecasting for dynamic environments: The DyFor genetic program model. *IEEE Trans. Evol. Comp.* 11, 433–452.