# Co-evolutionary Variance Can Guide Physical Testing in Evolutionary System Identification

Viktor Zykov Josh Bongard Hod Lipson
*Computational Synthesis Laboratory*
*Sibley School of Mechanical and Aerospace Engineering*
*Cornell University, Ithaca, NY 14853*
[viktor.zykov | josh.bongard | hod.lipson]@cornell.edu

## Abstract

*Co-evolution of system models and system tests can be used for exploratory system identification of physical platforms. Here we demonstrate how the amount of physical testing can be reduced by managing the difficulty that a population of tests poses to a population of candidate models. If test difficulty is not managed, then disengagement between the two populations occurs: The difficulty of the evolved test data supplied to the model population may grow faster than the ability of the models to explain them. Here we use variance of model outputs for a given test as a predictor of the tests' difficulty. Proper engagement of the co-evolving populations is achieved by evolving tests that induce a particular amount of variance. We demonstrate this claim by identifying nonlinear dynamical systems using nonlinear models and linear approximation models.*

## 1. Introduction

A wide range of control engineering applications require good knowledge of the controlled plant's internal structure and parameters. Traditionally such models are created manually using domain specific knowledge, but the use of automated system identification is of growing interest [1]. Manual modeling is becoming increasingly difficult as the complexity of modern plants is increasing. Automated system identification is particularly desirable in variable or remote systems, where the systems or its environment may undergo some unanticipated or rapid change, and therefore its internal model must also be updated autonomously *in situ*. A particular assumption we make here is that the system identification process needs to occur with minimal experimentation on the target system, as experimentation may be costly, slow, or detrimental to the target system.

Koza [2] used Genetic Programming (GP) to infer the internal structure of a target object in symbolic form by approximating the impulse response function for a linear time-invariant system. Gray reported the application of GP for nonlinear model structure identification based on experimental data from a coupled water tank and helicopter engine [3, 4]. Short-term chaotic time series prediction using GP was reported by Koza [5] for the logistic map in 1992, by Mulloy [6] in 1996.

In our previous work [7, 8], we presented the Estimation-Exploration Algorithm (EEA), a general co-evolutionary algorithm for the automated analysis of and synthesis for nonlinear physical systems. When the EEA is applied to a class of complex nonlinear physical systems – such as the mechanical double pendulum reported in Section 3 – the method was found to fail because the evolution of the test population occurred at a higher rate than that of the model population. This typically resulted in dominance of one population over the other: the fitness gradient in the model population is lost because all models fail equally badly against a set of difficult to explain results produced by the target system when provided with these difficult tests. Disengagement has been identified as one of several pathologies that can plague artificial co-evolutionary systems [9, 10].

We demonstrate the existence of correlation between the amount of model disagreement induced by a test and its effectiveness for the further evolution of the candidate models. Using this correlation, we present two approaches that allowed us to reduce the rate of development of the test population in a controlled manner so that the co-evolutionary engagement between the two populations is maintained.

## 2. Managing Test Complexity in EEA for System Identification

In its original implementation, the exploration phase of the EEA always supplies the most difficult tests to the estimation phase [7, 8], which may result in the failure of the estimation phase to find a model explaining the resulting complex behavior of the target system.

In Figure 1, we demonstrate the results of an original EEA run aimed at finding a system of linear differential equations approximating the output time series of a complex unknown target system. 10 EEA runs shown in Figure 1 used the same parameters as specified in section 4.1, but the population of tests was co-evolved with the unchanging goal of inducing maximum disagreement among the most successful evolved models. This caused very difficult tests to emerge early in the EEA run, and the models evolved by that time could not explain them well (they obtained a high subjective error) – they did not evolve an inner structure that could satisfactorily explain the behavior of the target system recorded after testing it with the difficult tests.

Alternatively, in order to allow the models to evolve gradually and be able to explain the tests in order of increasing complexity, we can monitor the process of model evolution and look for the signs of co-evolutionary disengagement. In our case, disengagement is signaled by an increase in the model populations' error as new tests are added to the set used to measure the accuracy of the models.

If this undesirable growth in error is detected during a particular EEA cycle, the algorithm should return to the point before disengagement occurred. For this, the algorithm has to withdraw the difficult test and the corresponding outputs obtained from the target system from the pool of inputs and outputs used by the EEA to evolve the new models, because the EEA has just failed to properly "digest" the most recent test: inclusion of this difficult test in the test pool used to train models caused the models' errors to increase too much. Once this model is withdrawn and the state of the algorithm rolled back, a new test of a lower complexity should be found by the exploration phase.

An easier test can either be taken from the bank of tests that have already been set aside as too difficult during earlier EEA cycles, or the algorithm can try to find a new, less difficult test.

Here we introduce the EEA Roll-Back procedure. It allows us to avoid disengagement between models and tests by undoing the effects of any model evolution that inflates error too much. The roll-back procedure provides us with means of engineering the tests of the required reduced difficulty that can be used to replace difficult tests that have been temporarily set aside by the algorithm.

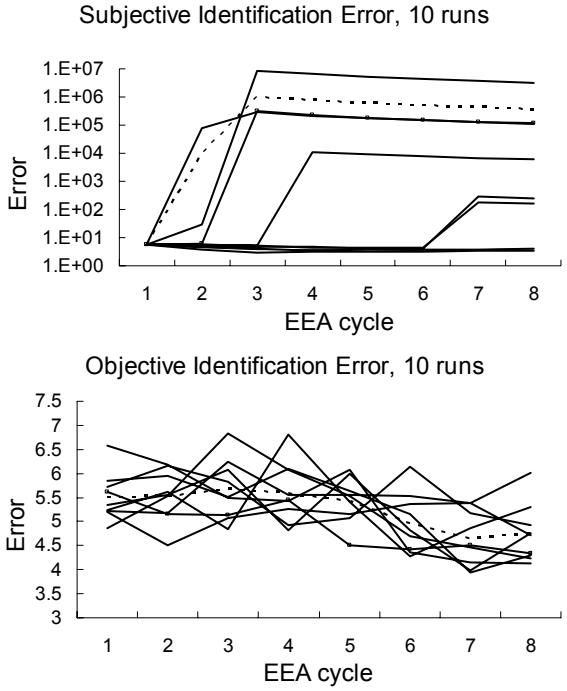



**Figure 1. Disengagement during the EEA run: the subjective error in identifying a complex plant explodes while objectively EEA performance does not change. Dotted line shows error averages.**

**Roll-Back Outline**

In each EEA cycle, a pass through the estimation phase is considered successful if:

- It results in a better fitness than the previous cycle;
- Its fitness falls into the target fitness margin; or
- It is the first EEA cycle – this allows the consecutive EEA cycles to improve on the initial results.

If the preceding estimation phase was not successful:

- The resulting model and test populations are discarded and replaced by the populations saved after the last successful pass through the estimation phase,
- The difficult test that resulted in this unsuccessful EEA cycle is saved in the bank of the complex tests; and
- Half the variance of this difficult test is used as a target variance for selecting the next test from the bank or evolving one.

If the preceding estimation phase pass was successful:

- The resulting models and tests populations are saved as successful for possible future roll-back;
- If the bank of difficult tests is empty, a new test is evolved with no variance limitation; or
- If the bank is not empty, the least difficult test is withdrawn for use in the next pass through the estimation phase.

## 3. Approach I: Symbolic Identification

In this section we describe the identification of a nonlinear target system in which it is assumed that neither the parameters nor the structure of the target system are known, but that the number of state variables is known, which for this target system is two. Each candidate model is therefore represented by a pair of differential equations, and a test is represented as a pair of real values, indicating the initial values of the two state variables.

The target system identified in this approach is a nonlinear dynamical system—the two-eyed monster system— described in [11], and given as

$$x' = y + y^2 \tag{1}$$

$$y' = -x + \frac{1}{6}y - xy + \frac{5}{6}y^2 . \tag{2}$$

Given some initial conditions $IC = [x_0; y_0]$, the target system is integrated using the Runge-Kutta method, using a step size of $h = 0.01$, from $t_0 = 0$ *s* to $t_{end} = 2$ *s*.

To initiate the EEA, a random initial condition for each variable is chosen from [–10; 10], and supplied to the target system. The resulting time series of *x* and *y* are returned to the estimation phase.

### 3.1. Estimation Phase

In the estimation phase, each genome encodes a set of differential equations that, when labeled with the five known parameters, produces a candidate model. Each genome is encoded as a forest in which there is one parse tree for each state variable, and each parse tree encodes the differential equation for that variable. Nonterminal nodes are labeled from the set [*sin*; *cos*; +; –; *; *%*; *pow*], where *%* is protected division, and terminal nodes are labeled from the set [*v*; *p*; *t*], where *v* indicates a state variable, *p* indicates a parameter, and *t* represents the current time in seconds. Terminal nodes also have an associated real value that is rounded to an integer in [0; 1] when the terminal node has label *v* or *p*, and is discarded for label *t*. The integer is treated as an index when paired with the terminal label: for example $v(0) = x$, $v(1) = y$; and $p(0) = -1$ and $p(1) = 1$, the two constants provided to the algorithm for identifying the parameters of the target system. The maximum depth allowed for any tree was set to 9.

During the first pass through the estimation phase, 300 random forests are generated. For each node that is created in each tree, a label is chosen from the combined set [*sin*; *cos*; +; –; *; *%*; *pow*; *v*; *p*; *t*] with equiprobability if the depth is less than five, and from the terminal set [*v*; *p*; *t*] with equiprobability otherwise. A random real number is also selected for the node. If the arity of the labeled node is one or more, a left subtree is created; if the arity is two, a right subtree is also created.

Each forest is evaluated as follows. For each test the current state variable values, the parameter values and *t* are supplied to the forest, and *x*' and *y*' are calculated. These values are used by Runge-Kutta integration to produce new *x* and *y* values. The model is evaluated for the same time period as the target system (2 *s*). The subjective error of the current genome is then calculated using

$$s_e = \frac{\sum_{i=1}^{v}\sum_{j=1}^{1000} max\left(\left|t_{ij}^{(1)} - m_{ij}^{(1)}\right| + \left|t_{ij}^{(n)} - m_{ij}^{(n)}\right|\right)}{vt}, \tag{3}$$

where $v = 2$ is the number of state variables in the system; *t* is the number of tests currently in the test suite; $t_{ij}^{(k)}$ is the value of variable *i*, at time interval *k*, produced by the target system using test *j*; and $m_{ij}^{(k)}$ is the value of variable *i*, at time interval *k*, produced by the candidate model using test *j*. The *max* term ensures that the system will first generate models that mimic major transients in the target system, such as narrow and tall spikes in the time series, because such transients produce large differences over short time periods.

Once all of the genomes have been evaluated, deterministic crowding is used to produce a new generation of genomes [13]. This method provides sustained diversity maintenance throughout the generations of the evolutionary process. All genomes are grouped into pairs at random, and are then crossed using one-point crossover, and then mutated. Mutation is as follows. For each tree, a node is selected at random. If the node is a terminal, then the associated real value may be replaced with a new random value in [0; 1] (p = 0.5), or the node may be mutated (p= 0.5). If the selected node is non-terminal, it is always mutated. Node mutation involves the selection of a new label from the combined set if the depth is less than five or from the terminal set if the depth is equal to five. If the new label has an arity different from the original node label, then new subtrees may have to be created (using the process of node creation described above) or deleted. Crossover is also used, in which one node from each of the parent forests may be selected, and subtree crossover carried out (node selection is not restricted to the same state variable tree in both parent forests). The child forests are then evaluated, and those with lower subjective errors than their associated parents replace them.

Evolution continues until a forest is discovered with $s_e \leq \in (= 0.1)$, or until the best model has not been

replaced for 40 generations. If at the termination of the pass the best model has $s_e > 0.1$, then the most recent test is stored in the test bank; otherwise the test is added to the test suite. If there is a test in the bank that, when added to the test suite induces an error of $s_e < 0.2$, it is withdrawn and added to the test suite, and the next pass in the estimation phase commences. Otherwise, the exploration phase commences using the 10 best models output by the current pass through the estimation phase. On the second and subsequent passes through the estimation phase, the 10 best models from the previous pass seed the initial random population.

### 3.2. Exploration Phase

If a test is not withdrawn from the test bank, then a new one must be created. This is accomplished in the exploration phase. Each pass through the exploration phase begins with an initial random population of 300 genomes (tests from previous passes are not used to seed the initial random population). Each genome is composed of $v$ real values (in this case $v = 2$), where each value indicates the initial value of the corresponding variable. Each genome is evaluated as follows. The encoded initial condition is supplied to each of the 10 models output by the estimation phase, and the resulting behaviors are recorded. The fitness of a test is then given as

$$t_f = \frac{\sum_{i=1}^{v} \sigma\left(t_{i1}^{(n)} - m_{i10}^{(n)}\right)}{v}, \quad (4)$$

where $m_{ij}^{(n)}$ is the final value of the $i^{\text{th}}$ variable from model $j$, and $\sigma\left(t_{i1}^{(n)} - m_{i10}^{(n)}\right)$ gives the variance of these final signals across all 10 models.

When all of the tests have been evaluated, a new generation of genomes is produced using deterministic crowding as explained in the previous sub-section. In this case however, child genomes that induce higher model variance than their assigned parent genome replace it. The exploration phase is then continued for a set number of generations; when the phase terminates, the test that induces the most model variance is output to the target system.

Termination occurs when the algorithm has passed through the estimation phase 40 times.

After each pass through the estimation phase, the best model was saved and its objective error was calculated using

$$o_e = \frac{\sum_{i=1}^{v} \sum_{j=1}^{1000} max\left(\left|t_{ij}^{(1)} - m_{ij}^{(1)}\right| + \left|t_{ij}^{(n)} - m_{ij}^{(n)}\right|\right)}{vt}, \quad (5)$$

where 1000 random, unseen sets of initial conditions are generated as test data.

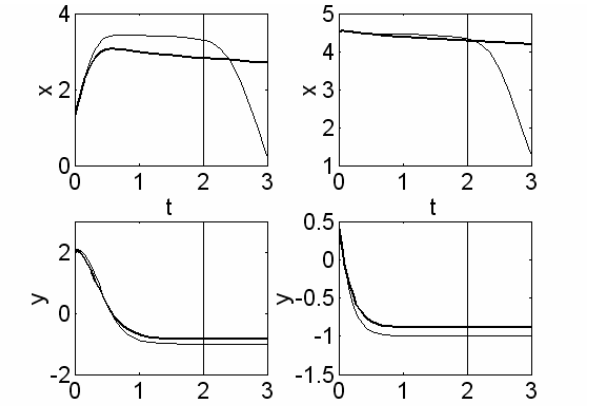
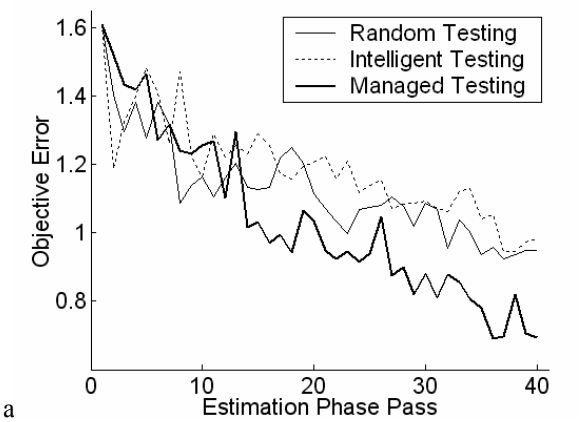



**Figure 2. The behavior of the best evolved model against two different tests. The left column shows the response of the two state variables to the first test, and the right column shows the response to the second test. The thick line shows the behavior of the target system (the two-eyed monster) to the same two tests. The vertical lines show the time period for which the models were originally evolved.**

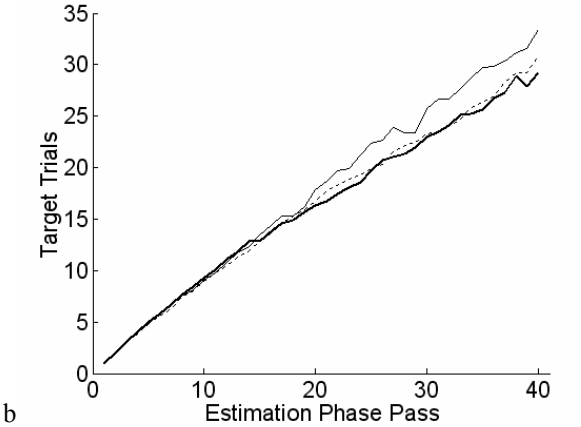
### 3.3. Results

Three variations of the above algorithm were performed. In the first variation which serves as the control, the exploration phase was not used: rather than evolving a test that induces model variation, a random test was generated and output to the target system. In the second variation, the exploration phase was conducted for 30 generations during each pass through it. In the third variation, the amount of effort dedicated to finding a difficult test was dynamically changed. During the first pass through the exploration phase, the best test from a population of 300 random tests is output to the target system; i.e. only one generation is evaluated. In subsequent passes through the exploration phase, the search for difficult tests is intensified if the previous test was successfully explained, and relaxed if the previous test was not successfully explained: if the best model in the most recent pass through the estimation phase achieved $s_e > 0.1$, then the number of generations performed in the exploration phase is doubled; otherwise, the number of generations performed in the exploration phase is halved. If the new number of generations to be performed is less than one then it is set to one; if the number is greater than 32 it is set to 32; if the new number of generations is not an integer, it is rounded down to the nearest integer.

For each of these three algorithm variants, 30 independent runs were performed. Figure 2 reports the behavior of the best evolved model found using the second algorithm variant. This model, when simplified,




a




b

**Figure 3. Mean Performance of the Three Algorithm Variants. a: Mean objective errors of the best models. b: Mean number of target trials performed after each pass through the estimation phase.**

gives:

$$x' = -0.1585y + ty + \mathbf{y^2} + t\sin(\sin(t \angle \cos(\cos(t))))$$
$$y' = -cos(sin(cos(sin(x)))) + 0.8415y + \sin(t^2/0.5402) - \mathbf{x} - t - y\cos(\sin(\cos(\sin(x)))) + \mathbf{0.8415y^2} + y\,sin(t^2/0.5402) - \boldsymbol{xy} - ty$$

As can be seen, the algorithm has discovered four of the governing terms, including a close approximation to the $^6/_5\,y^2$ term, which required the evolution of a real-valued constant.

Figure 3b shows that, although the differences between the three variants in this case are slight, the fact that both variant two and three consistently require fewer target trials than variant one (random testing) suggests that intelligent testing does indeed reduce the amount of required target trials.

Finally, we have gathered direct evidence that there is a positive correlation between the ability of a test to induce model disagreement and its difficulty. This

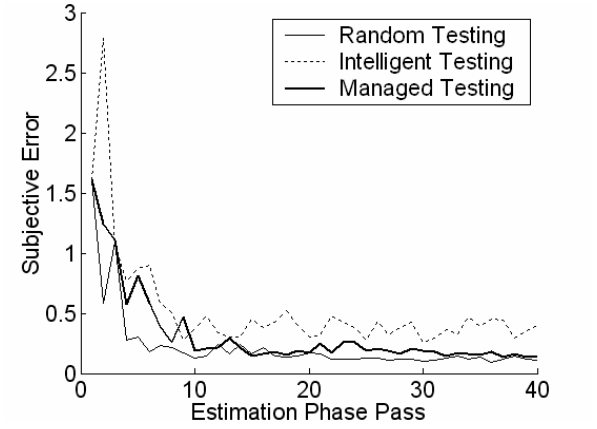



**Figure 4. Mean subjective error of the best model output by the estimation phase for each of the three algorithm variants.**

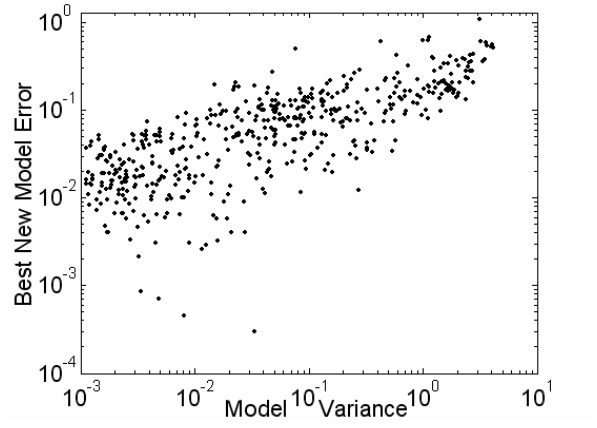



**Figure 5. Correlation between the ability of a test to induce model variance and its difficulty.**

correlation is reported in Figure 5. First, a population of 300 models was trained using a single test, as explained in section 3.1. Then, a set of 500 random tests were generated. The amount of disagreement induced in the 10 best models for each of these tests was then calculated using Eqn. 4 and recorded. Each test was then used to further evolve the models: the original evolved model population was evolved further using the original test and the one of the 500 random tests as described in section 3.1. When the population stagnated, the objective error of the new best model was recorded. The model population was then rolled back to its original state, and the next random test was supplied to the model population.

Figure 5 reports the original model variance versus the objective error of the new best model for each test. As can be seen, model variance is a good indicator of test difficulty.

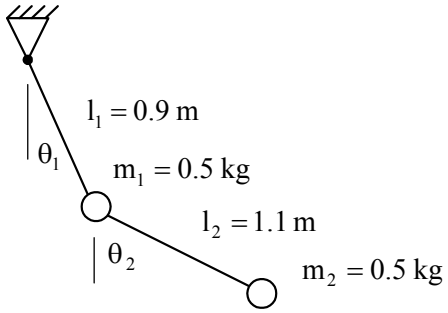



**Figure 6. Mechanical double pendulum.**

## 4. Approach II: Linear Approximation

In this section, we evolve a linear system that can approximate the output time series of a complex nonlinear system over a limited period of time. In our search, we assume that neither the parameters, nor the structure, nor the number of state variables of the target system are known. Each candidate model here is represented by a system of linear differential equations for which only the maximum number of state variables is limited. An intelligent test is represented as a pair of real values, indicating the initial values of the two state variables.

A simulated mechanical double pendulum was used as the hidden target system. This target system for both its chaotic behavior and well known mathematical description [11,12], thereby allowing for easy numeric simulation. Physical parameters of the simulated double pendulum are specified according to Figure 6.

At the initialization stage, a set of random initial models of the system are generated. Models are represented as fully-connected systems of linear ordinary differential equations:

$$\begin{cases} \dfrac{dy_1}{dt} = \dfrac{1}{\tau_1}\left(w_{11}y_1 + w_{12}y_2 + \ldots + w_{1n}y_n\right) \\ \dfrac{dy_2}{dt} = \dfrac{1}{\tau_2}\left(w_{21}y_1 + w_{22}y_2 + \ldots + w_{2n}y_n\right), \\ \vdots \\ \dfrac{dy_n}{dt} = \dfrac{1}{\tau_n}\left(w_{n1}y_1 + w_{n2}y_2 + \ldots + w_{nn}y_n\right) \end{cases} \quad (6)$$

where $y_1$ .. $y_n$ are linear system variables, $\tau_1$ .. $\tau_n$ are the time constants, $w_{11}$ .. $w_{nn}$ are the weights determining the extent to which internal linear system variables influence one another, and $n$ is the maximum order of complexity selected for the evolving linear systems.

Any $i$-th differential equation of the linear system

$$\tau_i \frac{dy_i}{dt} = -y_i + \sum_{j=1}^{N} w_{ji} y_j \quad (7)$$

can be thought of as an $i$-th node of a linear continuous-time network connected to all inputs and outputs of all other nodes with the synaptic weights $w_{i1}$, $w_{i2}$ .. $w_{in}$.

For the estimation phase of the EEA, each candidate linear system was described by a genome listing all its respective parameters:

$$\left[\tau_1\ w_{11}\ w_{12}\ w_{13}\ldots w_{1n}\ \tau_2\ w_{21}\ w_{22}\ w_{23}\ldots w_{2n}\ldots \right.$$
$$\left.\ldots\tau_n\ w_{n1}\ w_{n2}\ w_{n3}\ldots w_{nn}\right] \quad (8)$$

Each genome is represented by a string of double precision floating point values of length $[n \cdot (n + 1)]$. All time constants are initialized randomly with the values in the range $[4h, 1 + 4h]$ – where $h$ is the time constant of the Runge-Kutta method used for the linear model simulation. Weights are initialized randomly with the values in the range [–1, 1].

### 4.1. Estimation Phase

During the first pass through the estimation phase, 40 random models based on the template given in Eqn. (6) are generated. The GA of the estimation phase is aimed at minimizing the amount of deviation of the time-series $y_1(t)$ and $y_2(t)$ generated by a candidate model from the original time series of the double pendulum angles $\Theta_1(t)$ and $\Theta_2(t)$ over a finite period of time $T$ given some initial conditions $y_1(0) = \Theta_1(0)$ and $y_2(0) = \Theta_2(0)$ with all other variables of the candidate linear model initialized with zero values $y_3(0) = 0$, $y_4(0) = 0$ … $y_n(0) = 0$.

The subjective error $s_e$ (visible to the EEA) is measured as the average of $K$ maximum absolute disagreements between the original $\Theta_i(t)$ and $j$th approximated output time series $y_{ij}(t)$ and $\Theta_i(t)$ over the period $T$:

$$s_e = \frac{\sum_{i=1}^{v}\sum_{j=1}^{K} \max\left|\Theta_i - y_{ij}\right|}{K}. \quad (9)$$

Deterministic crowding (DC) was used to select parent chromosomes of candidate linear systems. The estimation phase uses the operators of crossover and mutation. Due to the use of DC, crossover is implemented for all pairs of parents by randomly choosing a locus in the two parent linear system chromosomes and swapping the terms to the right of the locus between the copied parents. Mutation is implemented with the probability of 0.04 per gene in one of the following equiprobable ways: by either multiplying the gene value by $e^{rand[-0.5\,..\,0.5]}$, or by $0.98 + rand[0\,..\,0.04]$. If the gene represents a weight $w_{11}$ .. $w_{nn}$, then its sign can also be inverted with probability 0.04. If the gene represents a time constant $\tau_1$ .. $\tau_n$, then it is protected from being reduced by mutation below 4 integration time steps. Evolution continues until the best model has not been replaced for 50 generations.

Roll-Back was implemented according to the outline in section 2. The desired error margin has been set to 0.02 rad. This equals approximately 2% of the overall maximum travel of the simulated double pendulum during 0.2 *s* given various initial conditions, which is conventionally considered acceptable precision in control engineering. The target variance is set to 0.01 of the target variance specified for the previous cycle in case of its failure.

### 4.2. Exploration Phase

At this phase, the intelligent tests are evolved to induce the desired variance among the best evolved models so far. Each test genome has to fully specify $v$ inputs to the target system (in this case $v = 2$) – both initial angles $\Theta_1(0)$ and $\Theta_2(0)$ of the double pendulum. Therefore, every test chromosome consists of $v$ genes, each gene being represented by a real value in the range $[0, 2\pi]$.

Variance $V$ of every IC genome is evaluated by evaluating the $N = 5$ best evolved models with this test and measuring the resulting disagreement among all resulting $[N \cdot (N-1) / 2]$ pairs of the time-series output by the models

$$V = \frac{\sum_{i=1}^{v} \sum_{j=1}^{N} \sum_{k=j+1}^{N} max \left| m_{ij} - m_{ik} \right|}{N(N-1)/2}, \qquad (10)$$

where $\left| m_{ij} - m_{ik} \right|$ is the maximum absolute difference between the output time series generated by models $j$ and $k$ for variable $i$ over the simulation time $T$.

After variance estimation, the difference of the resulting variance from the target variance of the current exploration run was estimated for each $i$th test as

$$\Delta_i = \left| V_i - V_T \right|. \qquad (11)$$

The smaller is $\Delta_i$, the higher the fitness of the $i$th test.

Deterministic Crowding (DC) was used for selecting parent chromosomes as described in section 3.1. Crossover was used analogously to section 4.1. Per-gene mutation probability was of set to 0.1, mutation was implemented equiprobably by either multiplying the gene value by $0.98 + rand[0 \,..\, 0.04]$, or by replacing the gene value by a new random value in the range $[0, 2\pi]$. Every population consists of 40 individual models. Evolution continues until the best test has not been replaced for 50 generations.

The algorithm iterates through the estimation and exploration phases until either the desired subjective error level has been achieved, or the maximum allowed number of cycles has been reached.

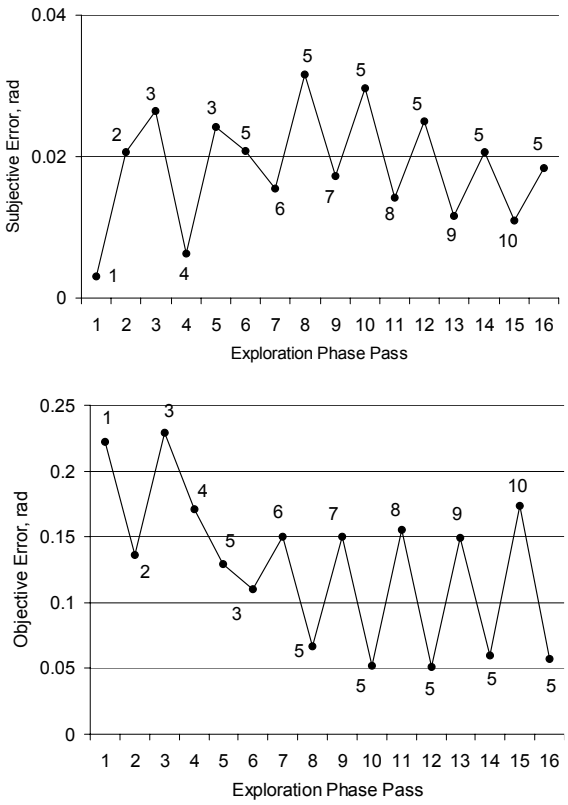



**Figure 9. Subjective and objective system identification error of the best evolved linear models approximating the behavior of a double pendulum over 0.2 *s*. The number of the intelligent test used for the estimation phase at every cycle is denoted above the data points.**

In order to obtain unbiased algorithm performance assessment, the error of the current best model was also measured objectively at each EEA cycle. Objective error measurement was implemented by defining a fixed set of tests. To create a fixed set of initial conditions, the 2-dimensional space ($\Theta_1$, $\Theta_2$) was meshed into a 2D rectangular grid: [25] points equally spaced within the square defined by [0, 0] to [2π, 2π] were used to calculate objective error.

### 4.3. Results

Figure 9 reports the results of running the EEA for 16 cycles. Subjective error increase accompanied by the objective error drop reflects the shift in the goal of the algorithm from finding a solution to one specific problem to generalizing over the two sets of input-output pairs: the first and the current test.

The subjective error dynamics follow a convex pattern which is common during a successful run of the

EEA. Initially, the models only have to generalize over a small amount of the input-output data obtained by experimentation on a target system, which is relatively easy. In the middle of the run, with the increase in the amount of data that has to be explained by the models, the subjective error grows, reflecting that the algorithm is struggling to achieve good results. Eventually, when the general structure of the target system has been captured by the candidate models, the new pairs of input and output signals only serve to confirm the knowledge of the model about the hidden system; subjective error goes down.

## 5. Discussion and Future Research

System identification is particularly important for evolvable hardware applications such as remote robotics, in which the mechanical as well as the electronic state of the system may change unpredictably, and new models of the system must be generated automatically. Here we have demonstrated two methods for managing test difficulty during co-evolutionary system identification. We have shown that both methods improve the ability of the algorithm to infer the structure of nonlinear systems.

In future work, we plan to improve the inference process to better predict the behavior of a target system over longer periods of time and with higher precision. Another important research objective is finding more new ways of reducing the amount of physical experimentation on a target system while maintaining and improving the identification capabilities.

We also plan to apply the algorithm to several physical systems, including a quadrupedal robot and the lac operon gene system in *E. coli* bacteria.

## Acknowledgments

This research was supported in part by the NASA program for Research in Intelligent Systems, Grant number NNA04CL10A.

## References

[1] Ljung L., System Identification: Theory for the User. Prentice-Hall, Inc. Englewood Cliffs, NJ, 1999.

[2] Koza, John R., Keane, Martin A., and Rice, James P. 1993. Performance improvement of machine learning via automatic discovery of facilitating functions as applied to a problem of symbolic system identification. 1993 IEEE International Conference on Neural Networks, San Francisco. Piscataway, NJ: IEEE Press. Volume I. Pages 191-198. 1993

[3] Gray, G. J., Murray-Smith, D. J., Li, Y. and Sharman, K.C. Nonlinear Model Structure Identification using Genetic Programming. Late Breaking Papers at the Genetic Programming 1996 Conference Stanford University July 28-31, pp. 32-37, 1996

[4] Gray, Gary J. and Murray-Smith, David J. and Li, Yun and Sharman, Ken C. and Weinbrenner, Thomas. Nonlinear model structure identification using genetic programming. Control Engineering Practice, Volume 6, pp. 1341-1352, 1998

[5] Koza, John R. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, Massachusetts, pp. 507 – 513, 1992.

[6] Mulloy, Brian S., Riolo, Rick L., and Savit, Robert S. Dynamics of Genetic Programming and Chaotic Time Series Prediction. in Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University. Cambridge, MA: The MIT Press. Pages 166-174, 1996

[7] Bongard, J. C. and H. Lipson. Automated robot function recovery after unanticipated failure or environmental change using a minimum of hardware trials. In Proceedings of The 2004 NASA/DoD Conference on Evolvable Hardware, pp. 169–176, Seattle, WA, 2004.

[8] Lipson, Hod and Josh Bongard, An Exploration-Estimation Algorithm for Synthesis and Analysis of Engineering Systems using Minimal Physical Testing, Proceedings of the ASME Design Automation Conference (DAC04), Salt Lake City, UT, Sept. 2004.

[9] Bongard, J. C. and H. Lipson Automating Genetic Network Inference with Minimal Physical Experimentation Using Coevolution, in Proceedings of the 2004 Genetic and Evolutionary Computation Conference (GECCO), Springer, pp. 333-345, 2004.

[10] Borrelli, R. and C. Coleman. Differential Equations: A Modeling Approach, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[11] Arnold, V. I. Problem in Mathematical Methods of Classical Mechanics, 2nd ed. New York: Springer-Verlag, p. 109, 1989.

[12] Wells, D. A. Theory and Problems of Lagrangian Dynamics. New York: McGraw-Hill, pp. 13-14, 24, and 320-321, 1967.

[13] Mahfoud S. W., Niching methods for Genetic Algorithms, Ph.D. dissertation from the University of Illinois, Urbana Champaign, IL, USA, 1995.