Supporting Online Material for

Distilling Free-Form Natural Laws from Experimental Data

Michael Schmidt and Hod Lipson*

*To whom correspondence should be addressed. E-mail: hod.lipson@cornell.edu

Published 3 April 2009, *Science* **324**, 81 (2009)
DOI: 10.1126/science.1165893

This PDF file includes:

Other Supporting Online Material for this manuscript includes the following: (available at www.sciencemag.org/cgi/content/full/324/5923/81/DC1)

Movie S1
Data Sets S1 to S15 as a zipped archive: invar_datasets.zip

Supporting Online Text

Distilling Freeform Natural Laws from Experimental Data

Michael Schmidt and Hod Lipson*

Materials and Methods

S1. The Predictive Ability Criterion

To search for potential *law equations*, we need a method that discriminates trivial equations, such as coincidental invariants, from equations that represent intrinsic relationships, such as energy conservation. We define a potential law equation to be *nontrivial* if it can predict differential relationships between two or more variables.

One such relationship that is readily quantifiable from both the law equation and experimental data is the partial derivative between pairs of variables. If our experiments collect time-series data, we can estimate the partial derivative between any pair of variables by taking the ratio of their numerical derivatives over time. For example, in a system with two state-variables x and y:

$$\frac{\Delta x}{\Delta y} \approx \frac{dx}{dt} \Big/ \frac{dy}{dt} \tag{S1}$$

We use nonparametric fitting – local polynomial fits (*S1*) – to estimate the time-derivatives of each state-variable. In the case where we do not have time-series data, but instead random point samples, we could alternatively estimate the partial derivatives directly using two-dimensional non-parametric fitting.

A candidate law equation – an equation we wish to test for triviality – can also derive the same partial derivatives between variable pairs using basic calculus. We do this by taking the ratio between partial derivatives of the equation. For example, for an equation $f(x,y)$ over variables x and y:

$$\frac{\delta x}{\delta y} = \frac{\delta f}{\delta y} \Big/ \frac{\delta f}{\delta x} \tag{S2}$$

We now have two estimates of the partial derivative: one estimated from the data, and one predicted by the candidate law equation f. To measure how well the equation predicted this relationship, we take the difference of Eqns. (S1) and (S2) over the dataset.

$$-\frac{1}{N}\sum_{i=1}^{N}\log\left(1+abs\left(\frac{\Delta x_i}{\Delta y_i}-\frac{\delta x_i}{\delta y_i}\right)\right) \tag{S3}$$

There are many metrics for combining the residuals – such as squared-error, mean error, correlation, etc. Here, we chose to use the *mean-log-error* for numerical reasons. The magnitude of the partial derivatives can grow large when the denominator approaches or crosses zero. The mean log-error squashes these high-magnitude residuals, while not discarding them entirely. In cases where the denominator is precisely zero, we discard the data sample. By convention, we measure the negative mean-log-error to define a maximization criterion.

S2. Calculating the Predictive Ability

Here we detail the predictive ability calculation in greater generality. While Eqns. (S2) and (S3) work for 2-dimensional systems using only numerical approximations, we need to consider symbolic relationships for higher order systems.

Specifically, we need to handle the case where one variable is dependent on another in order to calculate partial derivatives in Eq. (S3) correctly. Consider calculating $\delta x/\delta y$ in a 3-dimensional system with variables x, y, and z. When taking the partial derivative of $f(x,y,z)$, we can't assume variable independence in general. Therefore, we need to perform a symbolic derivative.

For example, consider the equation of a sphere: $f(x,y,z) = x^2 + y^2 + z^2$. When calculating $\delta f/\delta x$, we must consider y and z being dependent on x or vice-versa. Using the chain-rule, the symbolic derivative is thus:

$$\frac{\delta}{\delta x}\left[x^2+y^2+z^2\right]=2x+2y\frac{\delta y}{\delta x}+2z\frac{\delta z}{\delta x} \tag{S4}$$

In order to evaluate $\delta f/\delta x$ we need to fill in the partial derivatives on the right-hand-side of Eq. (S4). We have already approximated these values from the data in Eq. (S1). Therefore, we can re-write Eq. (S4) as:

$$\frac{\delta}{\delta x}\left[x^2+y^2+z^2\right]\approx 2x+2y\frac{\Delta y}{\Delta x}+2z\frac{\Delta z}{\Delta x} \tag{S5}$$

In general however, we should *not* assume that *every* variable is interdependent on all others – only a subset. For example in a 3-dimensional system, we only need to assume one pair of dependent variables; and in a 4-dimensional system, two pairs. So, continuing this example of the sphere equation, we have either:

$$\frac{\delta}{\delta x}\left[x^2+y^2+z^2\right]\approx 2x+2y\frac{\Delta y}{\Delta x} \tag{S6}$$

or

$$\frac{\delta}{\delta x}\left[x^2+y^2+z^2\right]\approx 2x+2z\frac{\Delta z}{\Delta x} \tag{S7}$$

For the general case, we can pick either case Eq. (S6) or Eq. (S7) for our calculation of Eq. (S2). We call this choice the *variable pairing* – which variables we assume are interdependent. We now refine Eq. (S3) – the measure of predictive ability – to incorporate the variable pairing:

$$\min_{pairing}\left\{-\frac{1}{N}\sum_{i=1}^{N}\log\left(1+abs\left(\frac{\Delta x_i}{\Delta y_i}-\frac{\delta x_i}{\delta y_i}\bigg|_{pairing}\right)\right)\right\} \tag{S8}$$

We could optionally measure error using all possible pairings. However, we have found empirically that taking the worst-case pairing, as in Eq. (S8), provides the best results for our computational law equation search.

One final adjustment we can make to the partial derivative pair metric is the sign of the of the $\Delta x/\Delta y$ and $\delta x/\delta y$ terms in Eq. (S8). The partial derivative pairs define a cloud of line segments in phase space, therefore we are only interested in matching the line but not necessarily the direction of the line. Negating the $\Delta x/\Delta y$ term or taking the absolute value of both can affect the signs of terms in the optimal law equation (for example, sign differences between Lagrangian and Hamiltonian equations).

S3. Example Partial Derivative Pairs

Here we provide an example calculation of a partial derivative pair for the double-pendulum Hamiltonian. This example is for a single pairing, namely assuming θ_1 and θ_2 are interdependent. The Hamiltonian for the double pendulum (with coefficients removed) is:

$$f=\omega_1^2+\omega_2^2+\omega_1\omega_2\cos\left(\theta_1-\theta_2\right)-\cos\theta_1-\cos\theta_2$$

We first take partial derivatives of f with respect to two variables (θ_1 and θ_2) as in Eq. (S5), substituting estimated partial derivatives on the right-hand-side with those from the data as in Eq. (S1):

$$\frac{\delta f}{\delta\theta_1}=-\omega_1\omega_2\sin\left(\theta_1-\theta_2\right)\cdot\left(1-\frac{\Delta\theta_2}{\Delta\theta_1}\right)+\sin\theta_1+\frac{\Delta\theta_2}{\Delta\theta_1}\sin\theta_2$$

$$\frac{\delta f}{\delta\theta_2}=-\omega_1\omega_2\sin\left(\theta_1-\theta_2\right)\cdot\left(\frac{\Delta\theta_1}{\Delta\theta_2}-1\right)+\frac{\Delta\theta_1}{\Delta\theta_2}\sin\theta_1+\sin\theta_2$$

We calculate the partial derivative pair by taking the quotient as in Eq. (S2):

$$\frac{\delta\theta_1}{\delta\theta_2}=\frac{\frac{\delta f}{\delta\theta_2}}{\frac{\delta f}{\delta\theta_1}}\approx\frac{-\omega_1\omega_2\sin\left(\theta_1-\theta_2\right)\cdot\left(\frac{\Delta\theta_1}{\Delta\theta_2}-1\right)+\frac{\Delta\theta_1}{\Delta\theta_2}\sin\theta_1+\sin\theta_2}{-\omega_1\omega_2\sin\left(\theta_1-\theta_2\right)\cdot\left(1-\frac{\Delta\theta_2}{\Delta\theta_1}\right)+\sin\theta_1+\frac{\Delta\theta_2}{\Delta\theta_1}\sin\theta_2}$$

We can now estimate the partial derivative pair numerically using this expression by filling in the symbolic variables with values from the experimental data. To see why this

expression is indeed the correct partial derivative, we can simplify it further symbolically by factoring out $1/\delta\theta_2$ from the numerator and $1/\delta\theta_2$ from the denominator:

$$\frac{\delta\theta_1}{\delta\theta_2} \approx \frac{1/\Delta\theta_2}{1/\Delta\theta_1} \cdot \frac{-\omega_1\omega_2 \sin(\theta_1 - \theta_2)\cdot(\Delta\theta_1 - \Delta\theta_2) + \Delta\theta_1 \sin\theta_1 + \Delta\theta_2 \sin\theta_2}{-\omega_1\omega_2 \sin(\theta_1 - \theta_2)\cdot(\Delta\theta_1 - \Delta\theta_2) + \Delta\theta_1 \sin\theta_1 + \Delta\theta_2 \sin\theta_2}$$

Cancelling the right most factor, this simplifies further to:

$$\frac{\delta\theta_1}{\delta\theta_2} \approx \frac{\Delta\theta_1}{\Delta\theta_2}$$

So, the partial derivative ratio resolves numerically to our estimated partial derivative pair from the experimental data, relating Eqns. (S1) and (S2).

We can perform similar derivations for other pairs of variables and different choices of dependent pairs. Some partial derivative pairs produce more interesting symbolic relationships that are more difficult to simplify. However, we have confirmed them to be numerically exact.

S4. Searching a Space of Equations

The partial derivative pairs metric, Eqns. (S3) and (S8), effectively defines a landscape over the space of equations. While the landscape is difficult to visualize due to its dimensionality and size, it is smoother and more well-defined than one might expect. Our method uses genetic programming to explore this landscape. In fact, most of the time, starting from a small number of random initial points in the landscape, this method can descend to the global optimal equation. We call the paths the algorithm takes to the final solution its trajectory in equation space.

Symbolic regression (*S2*) is an established method for searching the space of expressions computationally by minimizing various error metrics. Both the *parameters* and the *form* of the equation are subject to search. In symbolic regression, many initially random symbolic equations compete to model experimental data in the most parsimonious way. It forms new equations by recombining previous equations and probabilistically varying their sub-expressions. The algorithm retains equations that model the experimental data well while abandoning unpromising solutions. After an equation reaches a desired level of accuracy, the algorithm terminates, returning its most parsimonious equation that is most likely to correspond to the intrinsic mechanisms of the observed system.

In symbolic regression, the *genotype* or *encoding* represents symbolic expressions in computer memory. Often, the genotype is a binary tree of algebraic operations with numerical constants and symbolic variables at its leaves (*S3, S4*). Other encodings include acyclic graphs (*S5*) and tree-adjunct grammars (*S6*). The *fitness* of a particular genotype (a candidate equation) is a numerical measure of how well it fits the data, such as the equation's correlation or squared-error with respect to the experimental data.

One way to visualize the evolution of the equation genome is to track the ancestors of the final equation over the running time of the algorithm. Fig. S1 shows the ancestry trees for the equation of the ellipse. Several initially random equations evolve independently before coalescing. Predictive ability is initially low and some ancestors parent less accurate equations that eventually lead toward the exact solution (Fig. S1A). Equation

complexity is also initially high on average (Fig. S1B). After several generations however, the ancestry converges to simple and predictive equations, eventually finding an equation whose parameters can be tuned to find the exact solution (Fig. S1C).

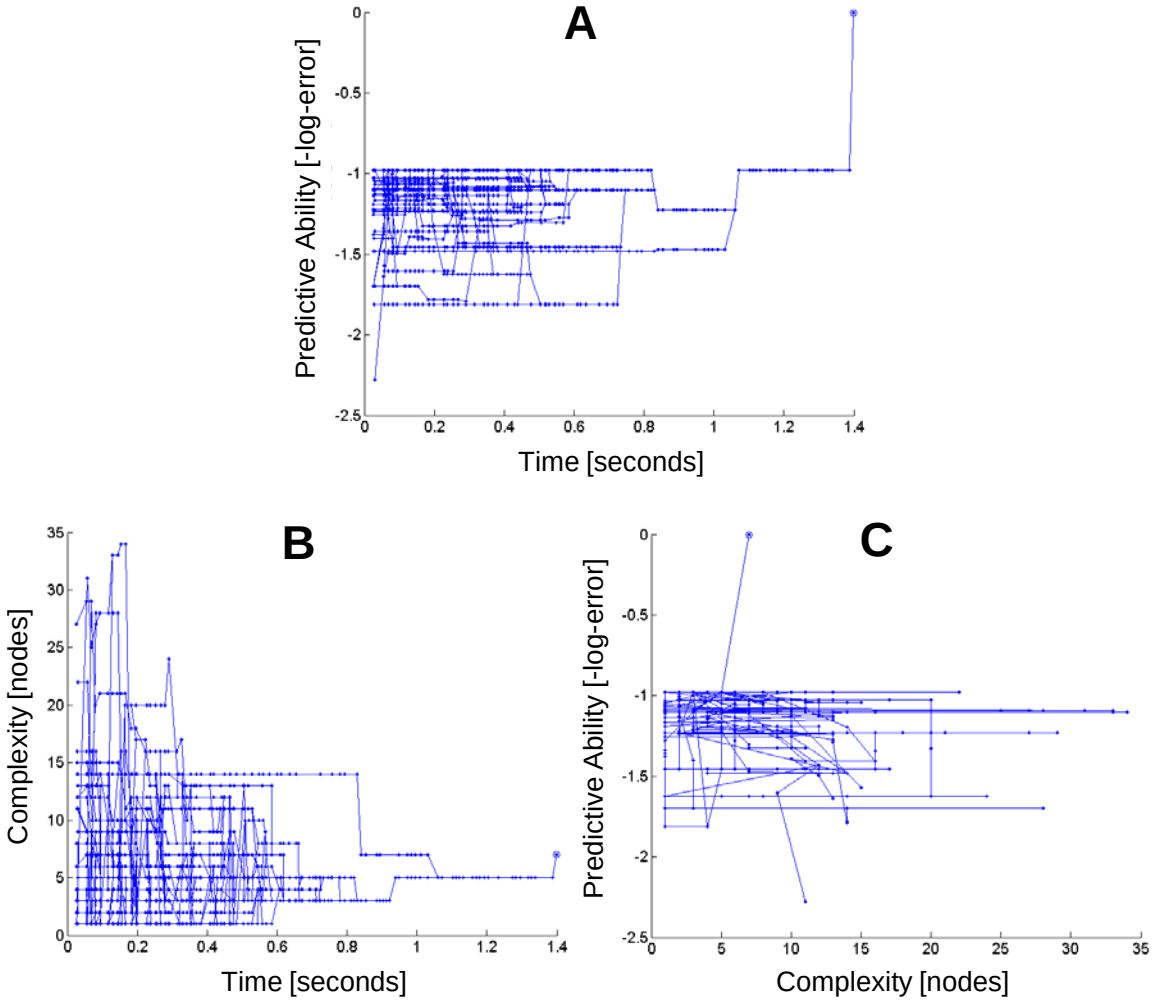


Fig. S1. Ancestor trajectories in equation space while searching for the equation of an ellipse. Dots indicate crossover and mutation events while lines represent parameter tuning over time. **(A)** Several initially random equations with varying predictive ability evolve independently before coalescing toward the exact solution over the running time of the algorithm. **(B)** The ancestors also vary in equation complexity – measured as the number of nodes in their expression trees. Initial equations tend to have higher complexity, but simplify over time toward the exact solution. **(C)** The same trajectories plotted over predictive ability and complexity shows the ancestor trajectories converge toward a simple and high predictive ability neighborhood before finding the correct equation structure whose parameters can be tuned to the exact solution.

We can also look at an individual trajectory (Fig. S2) to see how the equations vary during the evolutionary search. The first equation is randomly initialized and has poor accuracy. Gradually, point mutations vary individual terms in the equation. Crossovers introduce larger changes, such as adding or replacing terms evolved in other ancestry

sequences. In each step, the accuracy improves, until convergence onto the exact ellipse equation.

Accuracy	Equations in Sequence	Event
-1.4197	$x + x - c_3 - y$	*random*
-1.41347	$x + x + x - c_4 - y$	*mutation*
-1.41339	$x + x + x - \sin(c_3) - y$	*mutation*
-1.13805	$x + x + x - \sin(y) - (x - x)$	*crossover*
-1.08904	$(x + x)\cdot x - \sin(y) - (x - x)$	*mutation*
-1.08574	$(x + x)\cdot x - \sin(y) - c_1$	*mutation*
-1.01841	$(x + x)\cdot x - y - c_1$	*mutation*
-0.978484	$(x + x + x)\cdot x - y - c_{13}$	*mutation*
-0.914336	$(x + y - c_3)\cdot y + x\cdot x\cdot c_{15}$	*mutation*
-0.303559	$(x + y - c_4)\cdot y + x\cdot x\cdot c_{15}$	*mutation*
-0.0692607	$(x + y - \sin(x))\cdot y + x\cdot x\cdot c_{15}$	*crossover*
-0.0140815	$(x + y - x)\cdot y + x\cdot x\cdot c_{15}$	*mutation*
-0.0050732	$(x + y - x)\cdot y + x\cdot x\cdot c_{16}$	*mutation*
-0.0050732	$y\cdot y + c_3\cdot x\cdot x$	*mutation*

Fig. S2. Sequence of solutions as they evolve to model the equation of an ellipse. This sequence represents a single trajectory in Fig. S1. Small mutations and crossover events during the evolutionary search slowly converge this sequence onto the exact equation.

The algorithm's search over a space of equations for a natural law and building the Pareto front is a computationally intensive task, possibly requiring several hours or days of computation. However, the search is readily parallelizable as many candidate functions need to be evaluated simultaneously. We distributed our computations using the island-population model (*S7, 8*) and used a fitness-prediction model (*S9*) to reduce overall computational cost and to improve the local search gradient.

In a 32-core implementation, 10 minutes for the pendulum to a day for the double pendulum. The time for two-dimensional geometric invariants to be found on the Pareto front during the algorithm's search was approximately 5 minutes. The single-mass air-track laws took approximately 10 minutes to appear. The double-mass air-track laws took approximately one to two hours to appear. The pendulum laws took approximately 15 minutes to appear. And the most challenging, the double-pendulum system, took approximately one to two days of computation.

S5. Finding Symbolic Parameters

The search over equation space produces equations with bulk parameters; however, we can use a second equation search to identify the fully parameterized equation with symbolic parameters such as lengths, masses, etc. For example, our method found the following equation for the double pendulum with bulk parameters:

$$k_1\omega_1^2 + k_2\omega_2^2 + k_3\omega_1\omega_2\cos\left(\theta_1 - \theta_2\right) - k_4\cos\theta_1 - k_5\cos\theta_2$$

The question is what are the symbolic representations for the k_i coefficients? To find the fully parameterized equation, we simply need data from similar systems but with different physical configurations and hence varying bulk parameters – for example, collecting data from several double pendula that have different arm lengths and masses.

One way to help identify the units in a potential law equation is to require the evolved expressions to be consistent in physical units, and to provide the algorithm with physically-meaningful building blocks such as the masses and lengths of the system's components, while requiring all other constants to remain unit-less. This approach still does not eliminate completely some fundamental ambiguities.

Alternatively, once we have found the law equation with bulk coefficients, we can refit it very easily to data from another system that has different parameters. If we do this on several different system configurations, we can obtain bulk coefficients for each configuration of the system versus the physical parameters (eg. k_i values versus *length* and *mass* values of the collection of systems).

With bulk coefficient values from several systems, we can now find an equation for each individual coefficient using explicit symbolic regression (eg. find the equation of k_i as a function of the system masses and lengths).

We have done this in silico using 100 simulated double pendula with random masses and arm lengths. We first collected data from these double pendula by simulating them numerically and then refitting the coefficients of the double-pendulum law equation for each. Since the partial derivative pairs metric is scale invariant, we divide out the first coefficient to put all equation in a normal form. This allows us to compare coefficients across multiple double pendulum equations. Finally, we use explicit symbolic regression to find the equation for each coefficient:

$$k_1/k_1 = 1$$

$$k_2/k_1 = m_2L_2^2/(m_1L_1^2 + m_2L_1^2)$$

$$k_3/k_1 = 2.00055m_2L_2/(m_1L_1 + m_2L_1)$$

$$k_4/k_1 = 19.6/L_1$$

$$k_5/k_1 = 19.6\cdot m_2L_2/(m_2L_1^2 + m_1L_1^2)$$

where m_1, L_1, m_2, and L_2 are the masses and lengths of the first and second arms respectively. The remaining coefficient 19.6 is a multiple of the gravitational acceleration 9.8 m/s (which we do not vary).

By setting $k_1 = m_1L_1^2 + m_2L_1^2$, we can finally write out the fully parameterized law equation for arbitrary double pendula:

$$L_1^2(m_1+m_2)\omega_1^2 + m_2L_2^2\omega_2^2 + 2\cdot m_2L_1L_2\omega_1\omega_2\cos(\theta_1-\theta_2)$$
$$-19.6\cdot L_1(m_1+m_2)\cos\theta_1 - 19.6m_2L_2\cos\theta_2$$

Finding explicit equations for the parameters is much simpler than finding law equations from scratch. Symbolic regression found each coefficient expression in less than 30 seconds, compared with the tens of hours required to find the original bulk coefficient equation.

S6. Representing Invariant Equations

The acyclic graph (Fig. S3B) represents symbolic equations and is encoded internally as floating-point assembly code – a list of floating-point operations and parameter values. Operations can load an input variable or a parameter value, or perform a floating-point operation on any previous operation outputs (eg. *add*, *subtract, multiply, sine,* or *cosine*

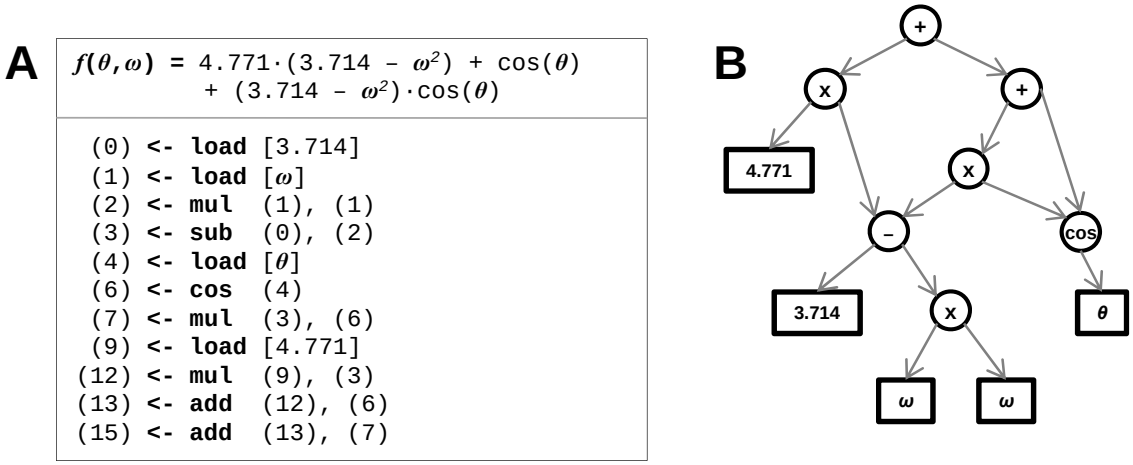


Fig. S3. Two equivalent representations of an example equation $f(\theta,\omega) = 17.719 - 4.771\cdot\omega^2 + 4.714\cdot\cos\theta - \omega^2\cdot\cos\theta$. **(A)** The algorithm stores and evolves equations represented by a list of floating point operators over a system's variables. Each operation can load a variable, load a parameter, or perform an mathematical operation on any previous operation. Unused lines have been omitted for clarity. **(B)** The raw list can be interpreted more intuitively by an acyclic graph where several sub-trees are reused by multiple terms. Both **(A)** and **(B)** represent the same equation.

commands). Each operation represents a leaf or parent node in the acyclic graph. The graph is rooted by the final operation in the list. Fig. S3A shows a raw encoding of an example equation.

We can construct the graph of a list encoding by tracing backward from the last operation recursively. One notable consequence of this encoding is that some operations are unconnected in the graph – no operations branching from the output node may reference certain nodes. In effect, these vestigial sections are free to drift during regression since they have no impact on the equation (phenotype). These sections are omitted in Fig. S3A.

We initialize the algorithm with random equations by generating a random list of floating-point operations, limited to 128 operations. We introduce variation using point mutation and crossover. A point mutation can randomly change the type of the floating-point operation (for example, flipping an *add* operation to a *multiply* or an *add* to an

system *variable*), or randomly change the parameter constant associated with that operation (if it is used). The crossover operation recombines two existing equations to form a new equation. To perform crossover, we select a random location in the list, and copy all operation and parameter values to the left of this point from the first parent and remaining operations and parameters to the right from the second parent.

We limit the size of the equation graph to narrow our search to human-interpretable equations (equations we could fit on a piece of paper). We allowed a maximum of 128 nodes, each possibly representing five types of mathematical operations, two to four variables, or a parameter constant. Ignoring the infinite parameter space, we are effectively searching a space of roughly 10^{108} parameterized equations.

Results and Analysis

S7. Detecting Laws in Synthetic Systems

In addition to physical laws such as Hamiltonians, Lagrangians, and equations of motion, the partial derivative pair criterion can also decipher implicit equations and geometric constraints. Table S1 summarizes the algorithm's search over time and the Pareto fronts for several synthetic manifolds and simulated dynamical systems.

Systems with parameter constants tend to exhibit gradual convergence whereas parameter-less equations converge rapidly at differing times. There is a similar inflection trend among all the Pareto fronts – an equation with some minimum complexity achieves very high predictive ability. The inflection of the double linear oscillator is more subtle, which we suspect is due to the large number of terms and polynomial approximations in its Hamiltonian equation.

Table S1. The predictive ability and Pareto fronts of several synthetic manifolds and simulated dynamical systems. Error bars denote the standard error of predictive ability

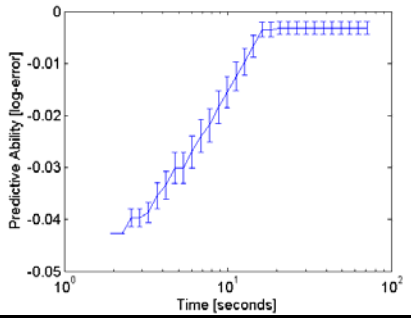
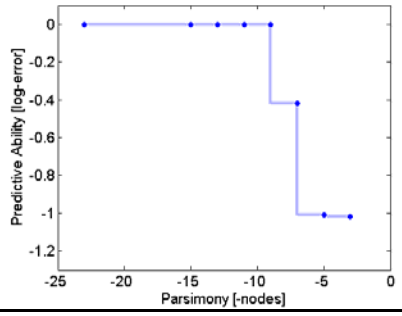
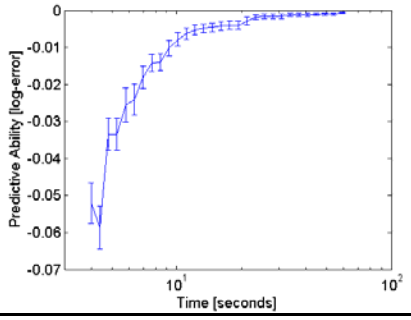
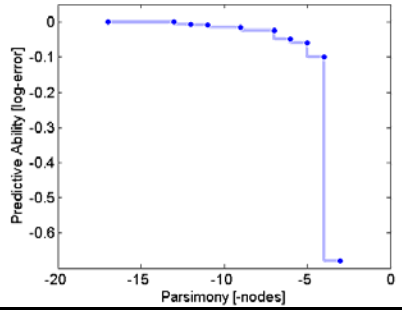
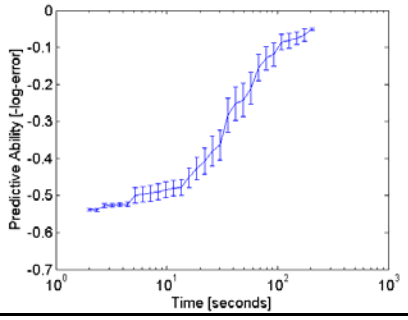
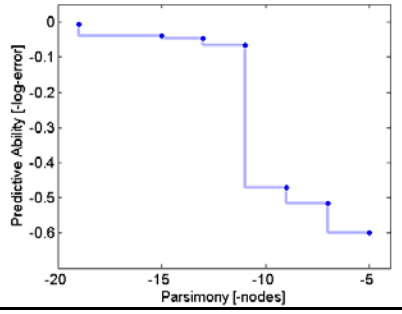
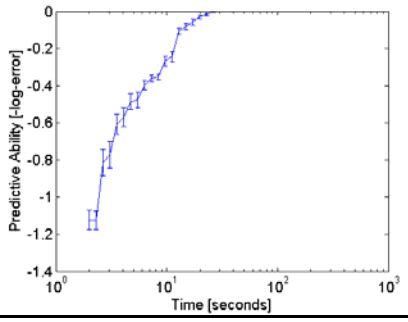
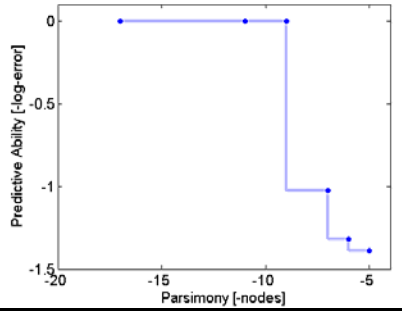
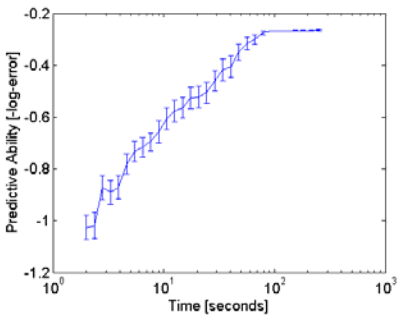
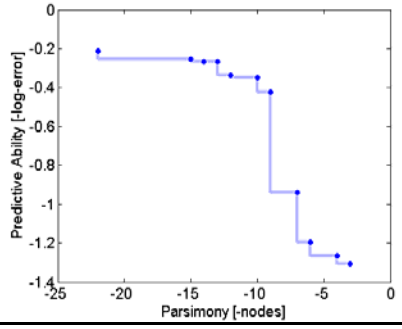
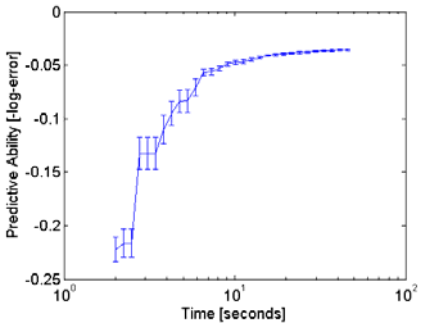
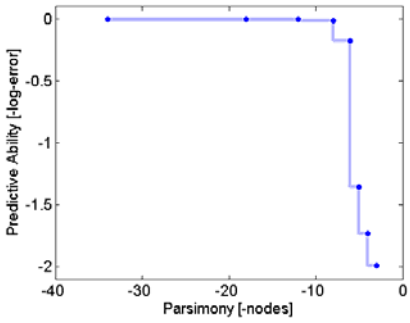
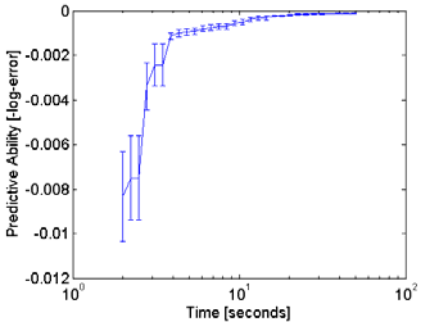
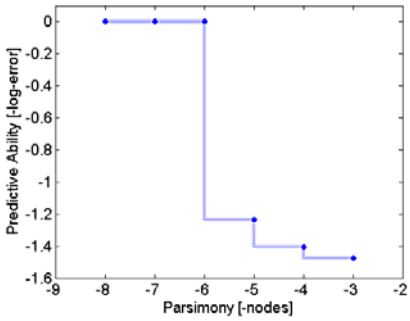
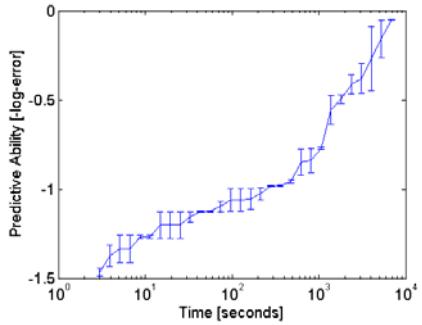
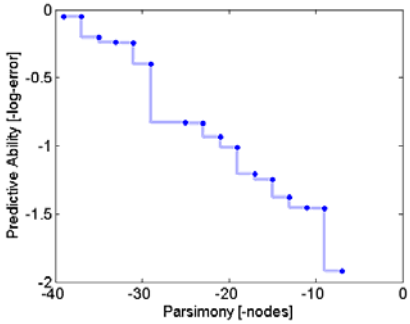
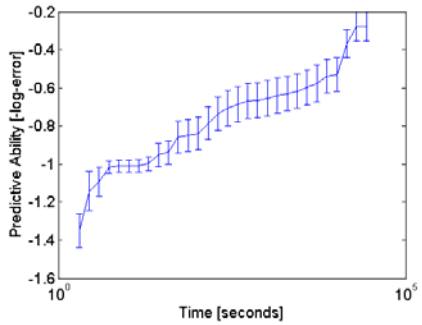
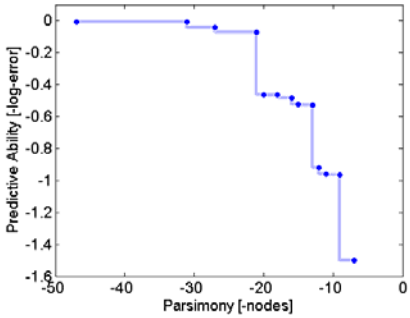
System	Predictive Ability Over Time	Accuracy/Complexity Pareto Front
Circle: $x^2 + y^2$		
Elliptic Curve: $x^3 + x - y^2$		
Sphere: $x^2 + y^2 + z^2$		
Linear Oscillator: $a - 0.1 \cdot v + 3 \cdot x$		
Linear Oscillator: $x^2 + 0.3 \cdot v^2$		

Table S1 (cont.) The predictive ability and Pareto fronts of several synthetic manifolds and simulated dynamical systems. Error bars denote the standard error of predictive ability.

System	Predictive Ability Over Time	Accuracy/Complexity Pareto Front
Pendulum: $\alpha - 9.8 \cdot \sin(\theta)$		
Pendulum: $\omega^2 - 9.8 \cdot \cos(\theta)$		
Double Linear Oscillator $x_1^2 + (x_1 - x_2)^2 + (1 - x_2)^2 + 2 \cdot v_1^2 + v_2^2$		
Double Pendulum $\omega_1^2 + 0.5 \cdot \omega_2^2 + \omega_1 \omega_2 \cos(\theta_1 - \theta_2) - 19.6\cos(\theta_1) - 9.8\cos(\theta_2)$		

S8. Equation-Space and Accuracy/Complexity Tradeoff

For any finite set of experimental data, there is potentially an infinite set of equations that maximize any type of error metric. For example, a 1000^{th} order polynomial can perfectly fit any dataset of 1000 or fewer unique data points. While it is immensely more difficult to find arbitrarily accurate equations using the partial derivative predictive ability criterion, it is still important to have some qualitative understanding of what the domain of equations looks like.

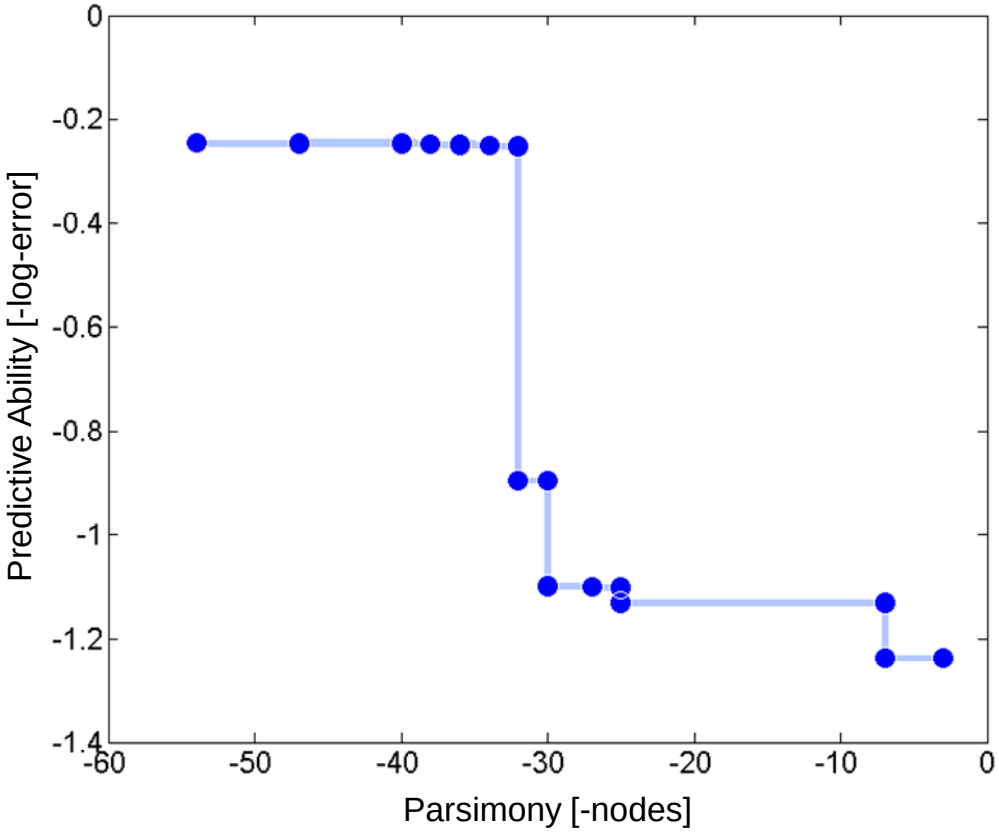


Fig. S4. The accuracy/complexity pareto front of the double pendulum. The pareto front shows the tradeoff between equation complexity and its ability to derive accurate partial derivative. At some minimum complexity (32 nodes), predictive accuracy jumps rapidly. Equations almost twice as complex improve the accuracy only marginally. These high complexity equations tend to contain the simpler exact equation, but add many smaller terms to compensate noise. The parsimonious and accurate equation at the inflection is the Hamiltonian and Lagrangian of the double pendulum.

Consider the relationship between equation complexity and accuracy of fitting the experimental data. Qualitatively there two extremes: extremely complex equations (eg. Taylor series, neural networks, and Fourier series) with near perfect accuracy and simple, single-parameter models with baseline accuracy. The equations in-between these two extremes are the most challenging to find and identify, and their behavior is more interesting. Fig. S4 shows the Pareto front of equation accuracy versus equation complexity for the double-pendulum.

The algorithm may also fail to find interesting relationships, due to either lack of convergence, inappropriate building blocks, or absence of any governing law. In this case, the front may be poorly formed with only exceedingly complex solutions reaching high predictive ability.

At certain minimum complexities, the equation's predictive ability jumps dramatically and then plateaus. In other words, there is a relatively simple equation that captures some intrinsic relationships of the system (but perhaps not perfectly). By parsimony arguments, we can reason this equation to be a likely governing law candidate. The equation at the inflection in this example is indeed the conservation of energy equation (Hamiltonian), supporting this assumption.

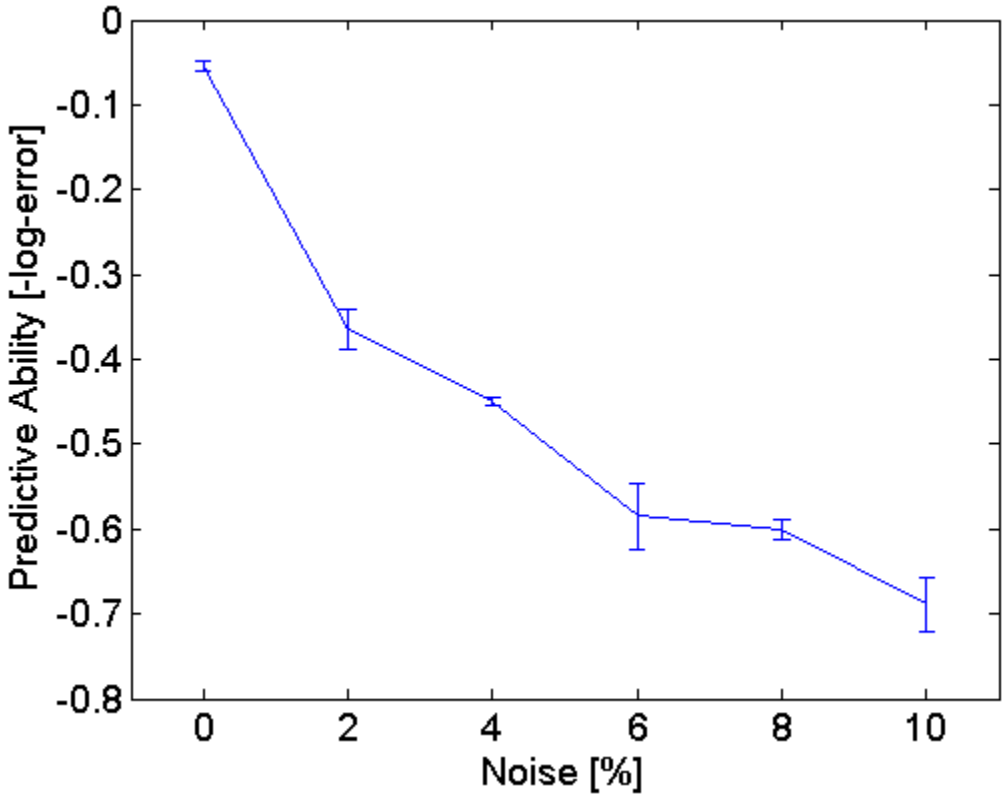


Fig. S5. The mean predictive ability on a withheld test set of the best law equations detected versus the amount of normally distributed noise in the data set for the simulated double linear oscillator. Error bars show the standard error. The percent noise is the ratio of the standard deviation of the noise and the standard deviation of the original signal.

S9. Impact of Noise

Noise can make inference tasks significantly more difficult. In particular, noise makes approximating the gradient (numerical derivatives) more difficult because derivatives can be highly sensitive to noise. We use Loess smoothing (*S1*) – a non-parametric fitting method – to remove high frequency noise from the motion tracking system. Loess smoothing updates each sample in the dataset by fitting a small order polynomial to the sample and its nearest neighbors.

Other methods, such as filtering and convolution, also reduce high-frequency noise, but do not readily produce estimates of the signal derivative. Using Loess smoothing, we obtain the numerical derivatives directly from the smoothing procedure by evaluating the symbolic derivatives of the local polynomial fits at each data sample.

We have examined the impact of noise on the predictive ability for the double linear oscillator (Fig. S5). Noise reduces the ability to find accurate law equations substantially, either simply requiring more time to compute or obscuring the law equation entirely depending on the noise strength. We measure the noise strength (percent noise) as the ratio of the standard deviation of the random noise to the standard deviation of the exact signal.

S10. Building a Physical Alphabet

By searching for physical laws from observational data, the algorithm is learning the language and rules of physics. One method to compartmentalize knowledge acquired from different systems is to identify common terms and calculations that pervade different phenomena. Here we quantify commonalities found among the linear oscillator, double linear oscillator, pendulum, and double pendulum.

The Pareto front of these systems summarizes several equations that maximize parsimony and accuracy for deriving dynamical relationships between variables. The terms in these equations are in this sense useful, and may comprise a common physical language.

We have decomposed the equations on the Pareto fronts of these systems to look for common terms by extracting all sub-trees (sub-expressions, and terms) of each equation. We then count the number of repeated terms. In order to compare terms between different systems, we only require positions to match other position variables (eg. angles versus Euclidean positions), velocities with angular velocities, etc. When comparing the double pendulum terms with double linear oscillator terms, we additionally require the variable number to match.

Fig. S6 shows the 24 most frequently occurring sub-trees. We can see that single variable terms dominate in frequency. More interestingly, we next see trigonometric terms for potential energies and kinetic energy and velocity terms. Following terms include trignometic terms for gravitational forces, acceleration forces, and spring potentials.

With this information on useful physical terms, the algorithm could reuse them for future systems, bootstrapping its knowledge into higher complexity systems. We plan to explore this possibility further in future work.

Term	Complexity	Frequency	Systems
$k\,x_2$	1	23.578	4
$k\,x_1$	1	21.6514	4
$k\,v_2$	1	21.5596	4
$k\,v_1$	1	19.1743	4
$k\cos(\theta_2)$	2	6.51376	2
$k\cos(\theta_1)$	2	6.14679	2
$k\,v_2^2$	3	5.50459	4
$k\,v_1^2$	3	4.49541	4
$(x_1 - x_2)$	3	4.22018	2
$k\,v_1\,v_2$	3	4.0367	2
$k\,a_2$	1	2.75229	2
$k\,a_1$	1	2.75229	2
$k\,x_2^2$	3	1.37615	2
$k\sin(\theta_2)$	2	1.19266	2
$k\sin(\theta_1)$	2	1.19266	2
$k\cos(\theta_1)$	2	0.917431	2
$k\cos(\theta_2)$	2	0.917431	2
$k\,x_1^2$	3	0.825688	2
$k\,x_1^2$	3	0.825688	2
$k\,x_1 + k\,v_1$	3	0.733945	3
$k\,x_2 + k\,v_2$	3	0.733945	3
$k\,x_1 + k\,x_2$	3	0.642202	2
$k\,x_1 + k\,v_1 - k\,a_1$	5	0.458716	2
$k\,x_2 + k\,v_2 - k\,a_2$	5	0.458716	2

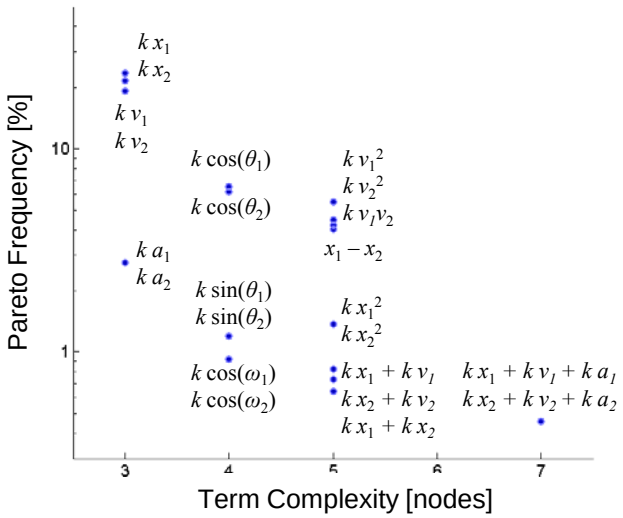


Fig. S6. The occurrence of common terms among the pareto fronts of the linear oscillator, double linear oscillator, pendulum, and double pendulum sorted by frequency of appearance. Several terms re-emerge between these systems revealing a common physical language for kinetic and potential energy, trigonometry, sum of forces.

S11. Data Collection and Preprocessing

We used motion tracking cameras and software (Vicon MX) to collect data on physical systems such as the double-pendulum. We place several infrared markers on the experimental device, place it into an arbitrary initial condition, and observe its dynamics.

The motion tracking produces time-series data of 3-dimensional Euclidean position coordinates for each infrared marker. We use many infrared markers in order to minimize noise and occlusions effects during the tracking. Afterward, we then combine the time-series of each marker to calculate the essential state-variables of the system – 2-dimensional coordinates, angles, etc. For example, in the double-pendulum, we project all 3-dimensional tracking points to its principle plane, and then calculate the angle of the two pendulum arms by taking the arctangent between segments of the infrared markers.

While motion tracking systems have become quite accurate and automated (*S10*), we must still handle noise and occlusion in the time-series data. Noise amplifies when the system experiences high velocities or when the number of cameras that can see a particular infrared marker changes.

In the double-pendulum, the infrared markers on the second arm become occluded from nearly all cameras when it passes behind the upper arm. In this case, the motion tracking produces null position coordinates, which we strip out before processing. Therefore, some of our time-series data contains gaps.

S12. Evolutionary Parameters

We use the fitness prediction symbolic regression algorithm described in (*S9, S11, S12*) to search the space of symbolic equations. We use the deterministic crowding selection method (*S13*), with 1% mutation probability and 75% crossover probability. The encoding is an operation list acyclic graph with a maximum of 128 operations/nodes (*S5*). Single-point crossover exchanges operations in the operation list at a random split. The operation set contains addition, subtraction, multiply, sine, and cosine operations.

Genetic programs are readily parallelizable to several computers and server clusters where available. We distributed the symbolic regression evolution over 8 quad core computers (32 total cores) using the island distributed computation method (*S7, S8*). The island model partitions the population of solutions into separated smaller populations residing on each computer (or core). We spread a population of 2048 equations over 32 CPU cores; therefore each island population has 64 equations.

The island model populations are faster to evolve because there are fewer individuals and less work to calculate fitness values per population. However, smaller individual populations are more prone to drift and saturation of similar solutions. To maintain higher diversity, we migrated solutions between populations at regular intervals. Every 1,000 generations (averaged over all populations), we randomly shuffle all equations among random pairs of populations.

The fitness predictor population contains 512 predictors, distributed over 32 cores. The fitness predictor subset size is 128 indices to the full training data set. Predictors are also evolved using deterministic crowding, but with 10% mutation and 50% crossover.

We calculate fitness using variations of Eq. (S8), where we modify the signs of partial derivative pairs using negation or absolute value to vary the types of law equations we search for. For predicted fitness values, we only calculate Eq. (S8) over the smaller subset of the fitness predictor rather than the entire data set.

S13. Results with Missing Building Blocks

It is interesting to note that in the absence of appropriate building blocks, the algorithm develops approximations. For example, eliminating sine and cosine as building blocks causes the pendulum invariant to be expressed as $\omega^2 + k_1\theta^2 - k_1\theta^4$, thereby exploiting the Taylor series expansion of the cosine function. Eliminating cosine but not sine drives the algorithm to discover and exploit the equality $\cos(\theta) = \sin(\theta + \pi/2)$ or more complex equivalences (Table S2).

Table S2. Summary of Detected Approximations with Missing Building Blocks

Building Blocks	Detected Pendulum Law	Approximation Discovered
*, +, –, cos(), sin()	$\omega^2 - 19.6\cdot\cos(\theta)$	*Exact Solution*
*, +, –, sin()	$\omega^2 - 19.5999\cdot\sin(-1.57079 + \theta)$	*Trigonometric identity*
*, +, –	$\omega^2 + 9.7108\cdot\theta^2 - 0.7042\cdot\theta^4$	*Taylor series expansion (4^{th} order)*

S14. Video

We have created a video that demonstrates the dynamics of the physical systems we experimented on and visualizes the search over equation-space for detecting physical laws. Several screenshots of the video (S1) are shown in Fig. S7.

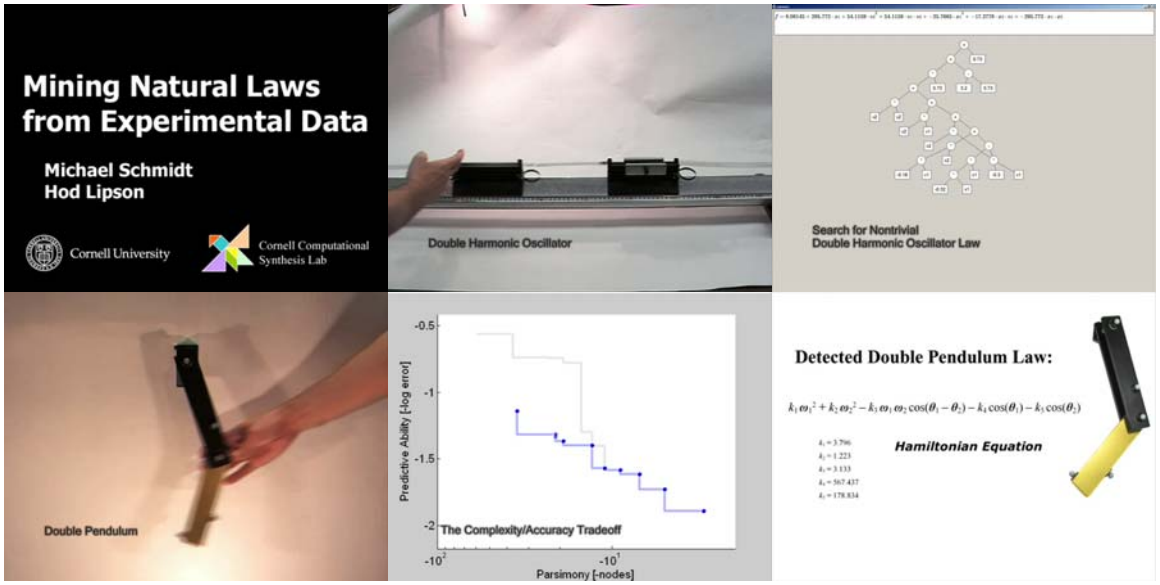


Fig. S7. The supplemental movie shows our experimentation with the double linear oscillator and double pendulum, visualization equation search, and the exploration of the pareto front.

The single-car air-track is a harmonic oscillator with slight damping from the air and its two springs. With only minimal noise and damping, it was the simplest physical system that we examined. Given velocity and position recorded from the pendulum over 30 seconds, the algorithm detected the system's energy conservation and Lagrangian equations. Given acceleration data also, it detected the system's differential equation of motion corresponding to Newton's second law.

The double-mass air-track consisted of two coupled harmonic oscillators of different masses. There was significant noise in this dataset as a result of compression of the middle spring. The algorithm still detected the Lagrangian and Hamiltonain equations.

The pendulum is a nonlinear oscillator. Given only position data, the algorithm detected that the device is confined to a circle. Given angular positions, velocities, and accelerations, it detected energy conservation, the Lagrangian, and the Newtonian equation of motion. The algorithm also detected several inexact expressions through small angle approximations – for example using x in place of $sin(x)$ and $1–x^2$ in place of $cos(x)$. To detect the complete nonlinear trigonometric terms, the algorithm required data spanning larger angles (roughly ±40˚).

The double-pendulum is the most complex system we studied. It is a coupled nonlinear oscillator system that exhibits rich dynamics (*S14*) and chaos at certain energies (*S15*) making it challenging to model (*S16, S17*). We focused only on detecting its energy laws by only providing angular positions and velocities. Similar to the single-pendulum, there are several approximate law equations that mask the identification of its exact laws.

Additionally, there is higher measurement noise and dampening errors due to higher velocities of the second arm. However, these challenges were overcome by balancing data measured from the double pendulum while operating at its two different regimes – namely, in-phase and chaotic regimes.

S15. Data Files

All datasets are available in the online supporting material; here we describe their format and content. A list of these files is shown in Table S3. In the online supporting material, all datasets are compressed into a single file named "invar_datasets.zip", for download.

The dataset files are plain-text files. The first line, lists the variable names delimited by white-space. Each subsequent line is a data sample with a different number of columns based on the number of variables. The first column is always a *trial number* (in the case that the dataset contains multiple independent trajectories or groups of data points). The second column is always *time* (or some other number for ordering in the case of non-time series). The remaining columns correspond to the variable values of the variables listed on the first line. Optionally, further columns specify the numerical time derivatives of each variable.

Table S3. List of Datasets Available in the Online Supporting Material

Filename	Points	Description
circle_1.txt	360	The unit circle over x, y
sphere_1.txt	2000	The unit sphere over x, y, z
pendulum_a_1.txt	493	Simulated pendulum over θ, ω, α
pendulum_h_1.txt	493	Simulated pendulum over θ, ω
linear_h_1.txt	512	Simulated linear oscillator over x, v, a
linear_a_1.txt	512	Simulated linear oscillator over x, v
double_linear_h_1.txt	821	Simulated double linear oscillator over x_1, x_2, v_1, v_2
double_pend_h_1.txt	2516	Simulated double pendulum over θ_1, θ_2, ω_1, ω_2
real_linear_a_1.txt	870	Motion-tracked linear oscillator over x, v, a
real_linear_h_1.txt	870	Motion-tracked linear oscillator over x, v
real_pend_circle_1.txt	600	Motion-tracked pendulum over x, y
real_pend_a_1.txt	556	Motion-tracked pendulum over θ, ω, α
real_pend_h_1.txt	556	Motion-tracked pendulum over θ, ω
real_double_linaer_h_1.txt	495	Motion-tracked double linear oscillator over x_1, x_2, v_1, v_2
real_double_pend_h_1.txt	1520	Motion-track double pendulum over θ_1, θ_2, ω_1, ω_2

References

S1. W. S. Cleveland and S. J. Devlin, *Journal of the American Statistical Association* **83**, 596 (1988).

S2. J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. (MIT Press, Cambridge, MA, USA, 1992).

S3. M. Ben, J. W. Mark and W. B. Geoffrey, in *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA* A. M. S. Zalzala, Ed. (IEE London, UK, Sheffield, UK, 1995), vol. 414, pp. 487--492.

S4. D. Edwin and B. P. Jordan, in *Genetic Programming and Evolvable Machines*. (2003), vol. 4, pp. 211--233.

S5. M. Schmidt and H. Lipson, paper presented at the Proceedings of the Genetic and Evolutionary Computation Conference, London, 2007.

S6. X. H. Nguyen, R. I. McKay and D. L. Essam, in *The Australian Journal of Intelligent Information Processing Systems*. (2001), vol. 7, pp. 114--121.

S7. F. Francisco, S. Giandomenico, T. Marco and V. Leonardo, in *Parallel Metaheuristics* A. Enrique, Ed. (Wiley-Interscience, Hoboken, New Jersey, USA, 2005), pp. 127--153.

S8. G. Christian, P. Marc and D. Marc, in *Genetic and Evolutionary Computation Conference Late Breaking Papers* R. Bart, Ed. (Chicago, USA, 2003), pp. 80--87.

S9. M. D. Schmidt and H. Lipson, *IEEE Transactions on Evolutionary Computation* **12**, 736 (Dec, 2008).

S10. W. Greg and F. Eric. (2002), vol. 22, pp. 24-38.

S11. M. D. Schmidt and H. Lipson, paper presented at the Proceedings of the Genetic and Evolutionary Computation Conference, Late Breaking Paper, 2005.

S12. M. D. Schmidt and H. Lipson, in *Genetic Programming Theory and Practice IV* L. R. Rick, S. Terence and W. Bill, Eds. (Springer, Ann Arbor, 2006), vol. 5, pp. -.

S13. S. W. Mahfoud, University of Illinois at Urbana-Champaign (1995).

S14. P. M. Jaeckel, T. , *Royal Society of London Proceedings Series A* **454**, 3257 (1998).

S15. T. Shinbrot, C. Grebogi, J. Wisdom and J. A. Yorke, *American Journal of Physics* **60**, 491 (1992).

S16. Y. Liang and B. Feeny, *Nonlinear Dynamics* **52**, 181 (2008).

S17. W. A. Mor M, Gottlieb O, in *Proceedings of The 21st ASME Biennial Conference on Mechanical Vibration and Noise*. (Las Vegas, Nevada, USA, 2007).