Chapter 5

SYMBOLIC REGRESSION OF IMPLICIT EQUATIONS

Michael Schmidt[1] and Hod Lipson[2,3]

[1]*Computational Biology, Cornell University, Ithaca, NY 14853, USA;* [2]*School of Mechanical and Aerospace Engineering, Cornell University, Ithaca NY 14853, USA;* [3]*Computing and Information Science, Cornell University, Ithaca, NY 14853, USA.*

Abstract Traditional Symbolic Regression applications are a form of supervised learning, where a label y is provided for every $\vec{x}$ and an explicit symbolic relationship of the form $y = f(\vec{x})$ is sought. This chapter explores the use of symbolic regression to perform unsupervised learning by searching for implicit relationships of the form $f(\vec{x}, y) = 0$. Implicit relationships are more general and more expressive than explicit equations in that they can also represent closed surfaces, as well as continuous and discontinuous multi-dimensional manifolds. However, searching these types of equations is particularly challenging because an error metric is difficult to define. We studied several direct and indirect techniques, and present a successful method based on implicit derivatives. Our experiments identified implicit relationships found in a variety of datasets, such as equations of circles, elliptic curves, spheres, equations of motion, and energy manifolds.

Keywords: Symbolic Regression, Implicit Equations, Unsupervised Learning

1. Introduction

Symbolic regression (Koza, 1992) is a method for searching the space of mathematical expressions, while minimizing various error metrics. Unlike traditional linear and nonlinear regression methods that fit parameters to an equation of a given form, symbolic regression searches both the parameters and the form of equations simultaneously. This process automatically forms mathematical equations that are amenable to human interpretation and help explicate observed phenomena. This paper focuses on the symbolic regression of functions in implicit form.

R. Riolo et al. (eds.), *Genetic Programming Theory and Practice VII*,
Genetic and Evolutionary Computation, DOI 10.1007/978-1-4419-1626-6_5,

An implicit equation represents a mathematical relationship where the dependent variable is not given explicitly. For example, an implicit function could be given in the form $f(\vec{x}, y) = 0$, whereas an explicit function would be given in the form $y = f(\vec{x})$. Implicit equations can be more expressive and are often used to concisely define complex surfaces or functions with multiple outputs. Consider, for example, the equation of a circle: It could be represented implicitly as $x^2 + y^2 - r^2 = 0$, explicitly using a multi-output square root function as $y = \pm\sqrt{r^2 - x^2}$, or as a parametric equation of the form $x = cos(t)$, $y = sin(t)$, $t = 0..2\pi$. Our goal is to automatically infer implicit equations to model experimental data.

Regressing implicit relationships can be thought of as a form of unsupervised learning. Traditional Symbolic Regression applications are a form of supervised learning, where a label y is provided for every input vector $\vec{x}$ and a symbolic relationship of the form $y = f(\vec{x})$ is sought. When seeking an implicit relationship of the form $f(\vec{x}, y) = 0$, we are looking for any pattern that uniquely identifies the points in the dataset, and excludes all other points in space.

Like clustering methods and other data mining approaches (McConaghy et al., 2008), unsupervised learning has the potential to find unexpected relationships in the data (De Falco et al., 2002; Mackin and Tazaki, 2000; Hetland and Saetrom, 2005). For example, unsupervised learning can create a model from positive examples only, and then use that model to detect outliers that do not belong to the original set. This is important in many practical applications where negative examples are difficult or costly to come by. For example, when training a system to monitor a jet engine, a learning algorithm will typically

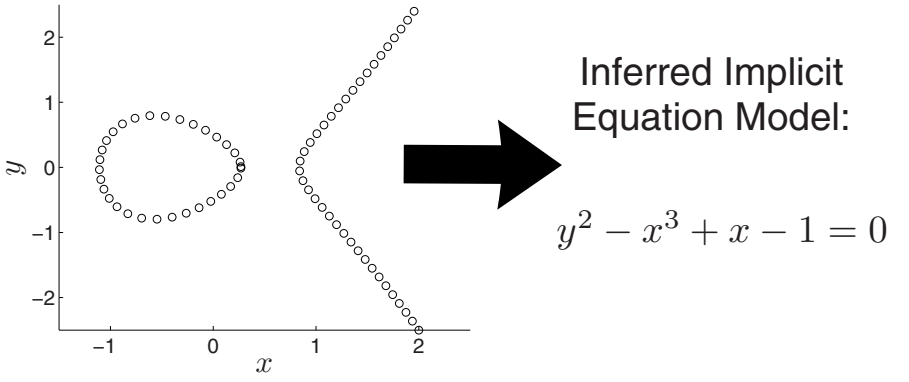


Figure 5-1. Many datasets exist that do not have explicit dependent variables, such as an elliptic curve shown here. Instead, this type of data must be modeled with an implicit equation. We explore using symbolic regression to infer these types of models.

be trained using sensor data from intact operation only, but will be required to alert an operator if abnormal sensor data is detected.

Implicit equations can also provide deeper insight into the mechanism underlying an observed phenomenon by identifying conservations. For example, when observing a pendulum, an explicit equation can be used to fit the data and thus predict the pendulum's future state based on its current and past states. In contrast, searching for implicit relationships can lead to finding equations of invariants, such as conservation of energy or momentum (Schmidt and Lipson, 2009). These conservations can also be used to make predictions, but provide more insight into the underlying principles, beyond prediction.

While symbolic regression has been used to find explicit (Korns, 2006; Duffy and Engle-Warnick, 1999; Bautu et al., 2005) and differential equations (Bongard and Lipson, 2007), it is not immediately obvious how it could be used to search for implicit equations (Figure 5-1). Symbolic regression ordinarily models and predicts a specific signal or value. In implicit equations, the equation always evaluates to zero over the dataset.

A key challenge is that there are an infinite number of valid implicit equations for any given dataset. For example, $sin^2(x)+cos^2(x)-1$ is exactly zero for all points in the dataset, but it is also exactly zero for all points not in the dataset. There are also an infinite number of relationships that are arbitrarily close to zero, such as $1/(1000+x^2)$. In order to utilize symbolic regression, we need to devise a fitness function that avoids these trivial solutions.

We experimented with a number of fitness functions for searching invariant equations. We explored minimizing the variance of the function from zero over the dataset while penalizing trivial equations that are zero everywhere, and numerically solving the implicit equation and minimizing its distance to each data point. Due to the difficulty of trivial solutions and susceptibility to local optima, none of these direct methods worked well.

Based on these results, we looked for a different metric that would relate an implicit equation to the dataset. Rather than attempting to model the data points themselves or the zeros of the target function, we decided to look at the gradients of the data. We found that we could derive implicit derivatives of the data variables using an arbitrary implicit equation, and then compare the two. Instead of fitting data points directly, this approach fits line segments (partial derivatives) derived from the data to the line segments (implicit derivatives) of the implicit function.

To test this approach, we experimented on modeling a number of implicit systems – ranging from equations of circles to equations of motion. We found this to be a reliable method for all these systems, whereas the other methods failed to find even the equation of the circle with similar computational effort.

In the remaining sections, we describe the direct methods in more detail, our proposed fitness for arbitrary implicit equations, the experiments and results on modeling implicit systems, and finally, concluding remarks.

2. The Implicit Equation Problem

The need to search for implicit equations arises when we do not know or do not have an explicit dependent variable in a dataset. Instead, we are given a large vector of data points and our goal is to find an equation that holds true for all of these points. For example, an equation that when solved numerically reproduces the points in the dataset.

An implicit equation has the form:

$$f(x, y, ...) = 0 \tag{5.1}$$

where x, y, etc. are independent variables of the system. Implicit equations in this form may or may not have an explicit equation in general (it may not be possible to solve for any single variable). However, these equations can be solved numerically or graphically when the equation is known.

Our task is to identify expression $f(x, y, ...)$ that satisfies the Equation 5.1 uniquely for all points in the dataset.

3. Direct Methods

Table 5-1. A summary of direct methods and their difficulties

Method	Difficulty
Equations that equal zero at all data points	Trivial solutions such as $0 = 0$, $x - x = 0$, etc
Equations that equal zero near data, but grow with distance	Places too many constraints on the resulting equations
Equations that equal zero but have non-zero derivative	Places too many constraints on the resulting equations
Equations that equal zero but not symbolically zero when simplified	Trivial solutions, just more complex zero identities such as $\cos^2 x^3 + \sin^2 x^3 - 1$
Equations that Equal zero, but nonzero at random point away from data	Trivial solutions such as $f(x) = 1/(100 + x)^2$, which is non-zero near $x = -100$
Numerically solve equation, measure distance from data points to closest zero	Difficult to evolve, many degenerate equations do not have solutions, and computationally expensive

Based on Equation 5.1, it might be tempting to search for equations that evaluate to zero for all data points in the dataset. A simple fitness function for this would be second moment or squared-error from zero. The problem with

this naive method is quickly obvious however: evolution almost immediately converges to a trivial solution such as $x - x = 0$ or $x + 4.56 - yx/y$, etc. These trivial solutions are zero everywhere and are not particularly interesting or useful for analyzing the data.

We tried a slight modification of this method by adding a test for trivial solutions such as $0 = 0$. For each candidate equation, we would perform a quick symbolic simplification to see if the result reduces to zero. Unfortunately, the evolution always converged to more complex identities equal to zero than we could add to our simplification test. For example, $(x-1)-(x^2-2x+1)/(x-1)$ and $-\sin^2(x) - \cos^2(x) + 1$, or more complex elaborations of zero identities.

A third method we tried was rewarding the function for being non-zero away from the points in the dataset. In this circumstance, evolution still converged on trivial solutions that were arbitrarily close to zero over most of the data, but still nonzero away from the data. For example, solutions such as $1/(1 + x^2)$, can become arbitrarily close implicit equations over the data, but are still trivial.

Finally, we decided to try numerically solving the candidate implicit equations and comparing with the data points. This method is extremely slow as the numerical solution requires an iterative procedure. It also has serious evolvability problems. Many candidate equations do not have implicit solutions (for example, $f(x) = 1/x^2$ never crosses zero) which makes finding the numerical solution non-convergent.

We modified this procedure slightly to find the local absolute valued minimum of a candidate equation around each point in the data set, summing the distance from the data points to their minima on the implicit function and the distance of the minima from zero. In the case that there is no local minimum for a data point, we capped the iterated procedure to a maximum distance.

This approach was able to identify implicit versions of simple lines, such as $x+y = 0$, and once found the correct implicit equations in the unit circle dataset (though these solutions were not repeatable). Unfortunately, all runs on more complex datasets, and most runs on the unit circle dataset, became trapped in local optima solutions. A common type of local optima evolved zeros around a part of the dataset (for example $1/(x+a)-b-y$ can model the left and bottom sides of a circle accurately), but rarely jumped to fit remaining data points.

While this final direct method may be a workable approach with more sophistication, it is far from elegant or efficient. Below, we describe a more direct and greatly more reliable and efficient fitness calculation for implicit equations.

4. The Implicit Derivatives Method

The difficulties of the direct methods (Table 5-1) suggest that comparing the zeros of the candidate implicit equation directly is insufficient to reliably find accurate and nontrivial models.

Rather than looking at the individual points, we decided to look at the local derivatives of these points. If the candidate implicit equation is modeling the points in a meaningful way, it should be able to predict relationships between derivatives of each variable. Importantly, we must also be able to measure such a relationship readily from the dataset.

For our method, we propose using the ratio of partial derivatives between pairs of variables (implicit derivatives). The idea is that dividing two partial derivatives of a candidate implicit equation $f(...) = 0$ cancels out the implicit $f(...)$ signal, leaving only the implied derivative between two variables of the system.

For example, in a two-dimensional dataset we could measure variables $x(t)$ and $y(t)$ over time. The system's implicit derivatives estimated from time-series data would then be $\Delta x/\Delta y = x'/y'$ and $\Delta y/\Delta x = y'/x'$, where x' and y' represent the time-derivatives of x and y. Similarly, given a candidate implicit equation $f(x, y)$, we can derive the same values through differentiation: $\delta x/\delta y = (\delta f/\delta y)/(\delta f/\delta x)$ and $\delta y/\delta x = (\delta f/\delta x)/(\delta f/\delta y)$. We can now compare $\Delta x/\Delta y$ values from the experimental data with $\delta x/\delta y$ values from a candidate implicit equation $f(x, y)$ to measure how well it predicts indirect relationships between variables of the system.

Finally, we can use this process in a fitness function for implicit equations. We simply measure the error on all implicit derivatives that we can derive from each candidate equation. In our experiments, we return the mean logarithmic error of these derivatives:

$$-\frac{1}{N}\sum_{i=1}^{N} log\left(1 + |\frac{\Delta x_i}{\Delta y_i} - \frac{\delta x_i}{\delta y_i}|\right) \tag{5.2}$$

where N is the number of data points, $\Delta x/\Delta y$ is a implicit derivative estimated from the data, and $\delta x/\delta y$ is the implicit derivative derived from the candidate implicit equation.

5. Handling Unordered Datasets

The implicit method can also be applied to unordered and non-time series data as there are several ways to estimate implicit derivatives from experimental data. An implicit derivative is simply a local relation of how two variables covary. In 2D, the implicit derivative is the slope of the tangent line. In 3D, the implicit derivatives lie on the tangent plane. In higher dimensions, they lie on the n-dimensional tangent hyperplane.

To generalize this procedure for arbitrary unordered data, one can fit a hyperplane, or higher-order surface such as a conic section (Shpitalni and Lipson, 1995), to local clouds of data points. From each hyperplane, one can then

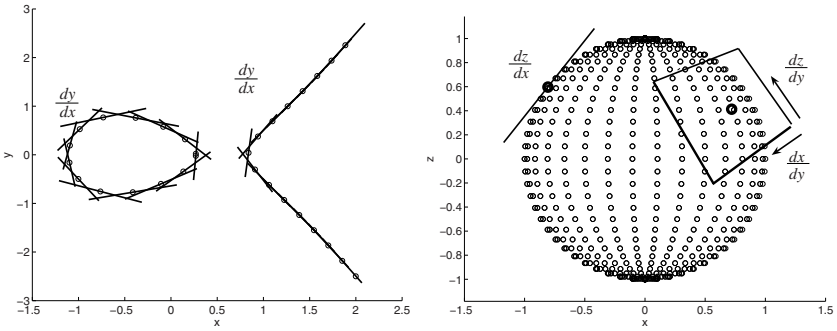


Figure 5-2. Implicit derivatives can be estimated from unordered, or shuffled data, nonparametrically by fitting a hyperplane or higher-order surface to neighboring points. After fitting the neighboring points, simply take any of the implicit derivatives of the locally fit surface.

sample implicit derivatives by taking the implicit derivative of the hyperplane equation (Figure 5-2).

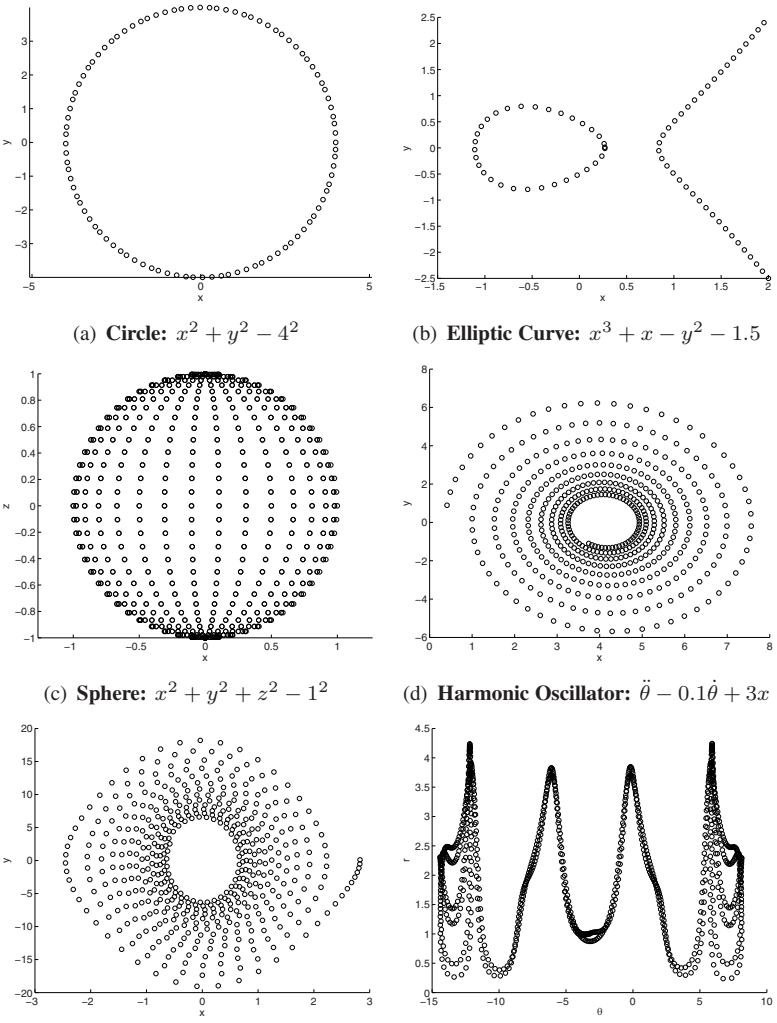
We verified that this procedure works in our experimental datasets by randomly shuffling them and discarding their time ordering. The method regresses the same implicit equations as in our results below using this procedure.

6. Experiments on Implicit Equations

We experimented on six implicit equation problems of varying complexity and difficulty (Figure 5-3). The simplest are the equation of a circle and an elliptic curve. These are well-known two dimensional systems with only two implicit derivatives ($\delta x/\delta y$ and $\delta y/\delta x$) that require implicit equations. A similar but slightly more difficult problem is the 3-dimensional sphere. In each of these systems we can collect data uniformly on their implicit surfaces.

The next three systems are dynamical systems of varying complexity: a simple linear harmonic oscillator, a nonlinear pendulum, and a chaotic spring-pendulum. We simulated single trajectories of each system, recording the positions, velocities, and accelerations for the implicit datasets. In these systems, we are seeking the implicit equation of motion. In the spring-pendulum we are seeking a similar implicit equation, the Hamiltonian, which only uses position and velocity data. The data used for each system is shown in Figure 5-3.

From this data, we estimate the partial derivatives from the data ($\Delta x/\Delta y$) by taking the ratio of the time derivatives. For the circle, elliptic curve, and sphere, we picked an arbitrary time trajectory around their surfaces (two in the case of the elliptic curve). This works because the time component cancels out in the ratio. We could also have fit a local plane to each point to estimate the partial derivatives non-parametrically of unordered data as discussed earlier.



(a) **Circle:** $x^2 + y^2 - 4^2$

(b) **Elliptic Curve:** $x^3 + x - y^2 - 1.5$

(c) **Sphere:** $x^2 + y^2 + z^2 - 1^2$

(d) **Harmonic Oscillator:** $\ddot{\theta} - 0.1\dot{\theta} + 3x$

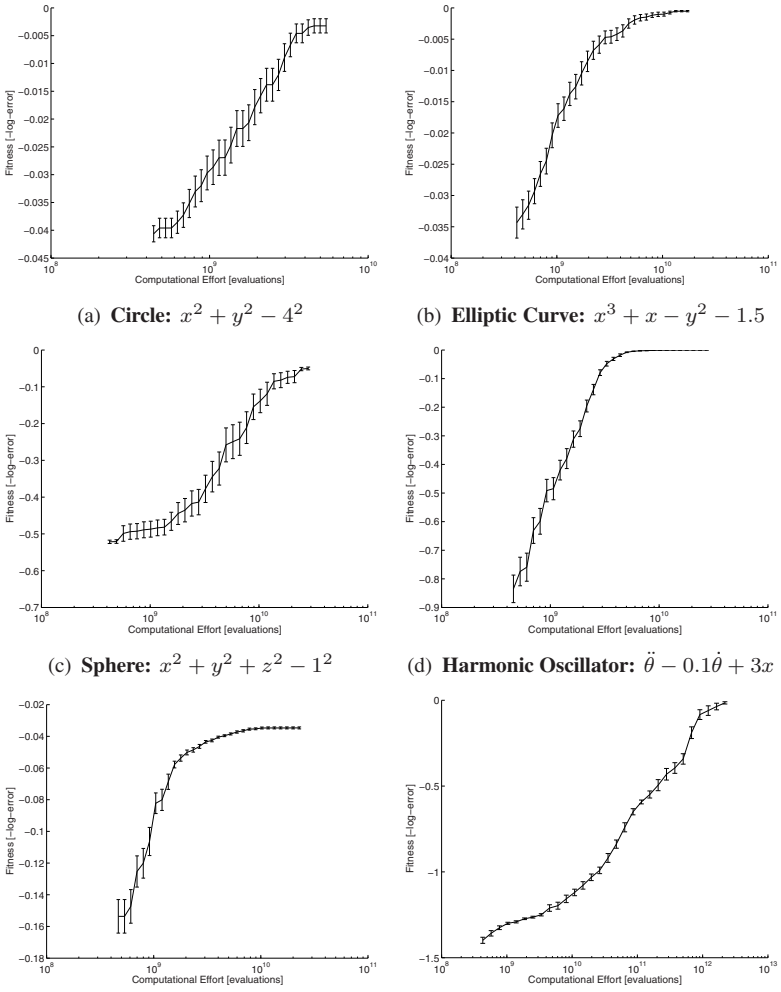
(e) **Pendulum:** $\ddot{\theta} - 0.1\dot{\theta} + 9.8\sin(\theta)$

(f) **Spring Pendulum:** $-10r + 0.5\dot{r}^2 + 5r^2 - 10rcos(\theta) + 0.5r^2\dot{\theta}^2$

Figure 5-3. Data sampled from six target implicit equation systems. Data is collected uniformly for the geometric systems. In the dynamical systems, the data is a single simulated trajectory from a random initial condition.

We used a basic symbolic regression algorithm (Schmidt and Lipson, 2006) to search the space of implicit equations. We use the deterministic crowding selection method (Mahfoud, 1995), with 1% mutation probability and 75% crossover probability. The encoding is an acyclic graph (Schmidt and Lipson, 2007) with a maximum of 128 operations/nodes. The operation set contains ad-

dition, subtraction, multiply, sine, and cosine operations. Fitness was calculated using Equation 5.2.



(a) **Circle:** $x^2 + y^2 - 4^2$

(b) **Elliptic Curve:** $x^3 + x - y^2 - 1.5$

(c) **Sphere:** $x^2 + y^2 + z^2 - 1^2$

(d) **Harmonic Oscillator:** $\ddot{\theta} - 0.1\dot{\theta} + 3x$

(e) **Pendulum:** $\ddot{\theta} - 0.1\dot{\theta} + 9.8\sin(\theta)$

(f) **Spring Pendulum:** $-10r + 0.5\dot{r}^2 + 5r^2 - 10rcos(\theta) + 0.5r^2\dot{\theta}^2$

Figure 5-4. Fitness of the symbolic regression algorithm using the implicit derivatives fitness for each of the six systems. Results are the top ranked solution versus time, averaged over 20 independent trials. Error bars indicate the first standard error.

7. Results on Implicit Equations

We conducted 20 independent trials on each system, recording fitnesses and solutions overtime. Evolution was stopped after a solution converged onto a

near perfect solution. Figure 5-4 shows the mean fitness of the top-ranked solution during the evolutionary runs on a validation dataset.

Each evolutionary run identified the correct implicit equation for these systems, although different systems required more computation than others. The circle took less than a minute to converge on average; the elliptic curve, sphere, and pendulum took five to ten minutes on average; and the spring pendulum took approximately one to two hours.

In comparison, none of the direct methods could find solutions to any of these systems, even with considerably more computational effort. In the case of the circle, the implicit derivatives methods obtained the correct solution 20 out of 20 trials in under one minute per trial. In contrast, the direct methods did not obtain the correct solution even once in 20, one hour trials. The best solution found by the direct method over these runs was $a/(x^2 + b) - y - c = 0$. In the remaining target systems, the direct methods performed even worse.

Over our experiments, we also tracked the Pareto Front of the implicit equation fitness and complexity for each system (Figure 5-5). This front shows the tradeoff between equation complexity and its ability to model the implicit data (Smits and Kotanchek, 2004). Here, we measure the complexity of an equation as the number of nodes in its binary parse tree.

The Pareto fronts tend to contain cliff features where fitness jumps rapidly at some minimum complexity. In the cases where even more complex equations are found on the front, even several times more complex, the improvement in fitness is only marginal.

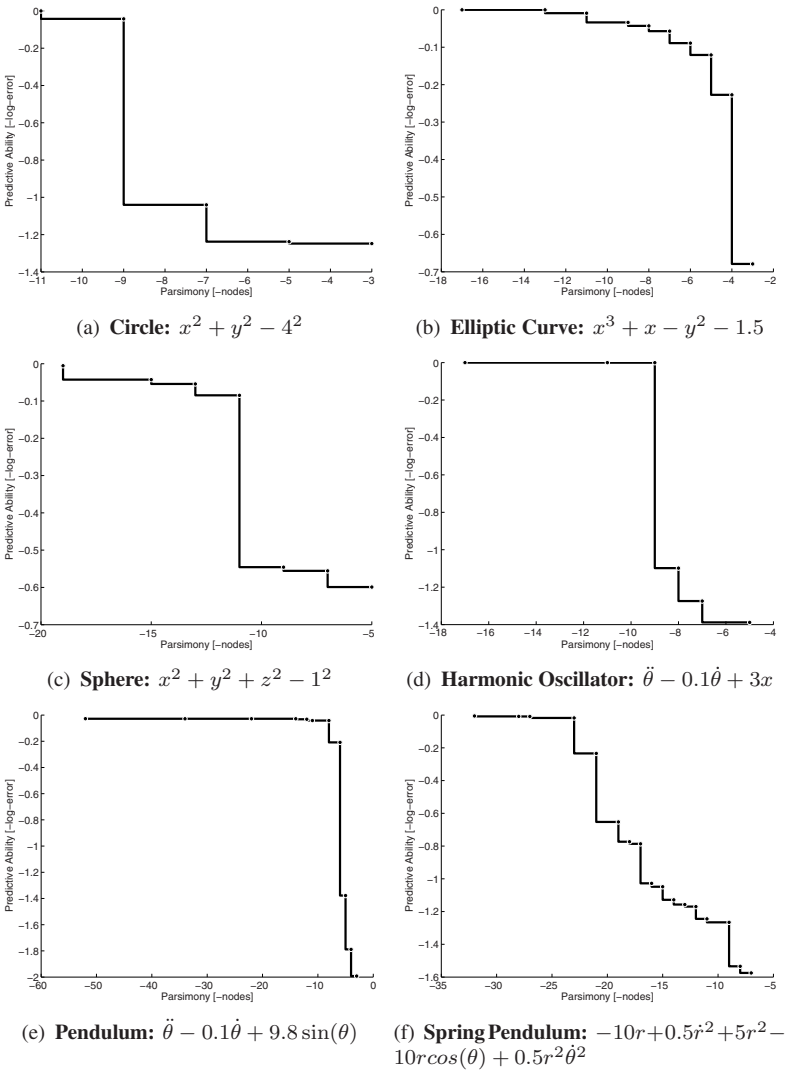
For each system, the simplest implicit equation to reach the highest qualitative fitness on the Pareto front was the exact target equation. Looking more closely at the higher complexity solutions, we found they were elaborations on the exact solution – for example, extraneous terms with very small coefficients, perhaps compensating for small errors in estimating the partial derivatives from the data.

We also noticed that simpler and lower fitness solutions on the fronts contained approximations to the exact solutions – for example, small angle approximations in the pendulum and spring pendulum systems.

8. Conclusion

The ability to search for implicit equations enables searching for multi-dimensional surfaces, equations of motion, and other invariant models in experimental data. However, identifying meaningful and nontrivial implicit equations poses difficult challenges.

We explored several naive fitness methods for rewarding implicit equations to model data. These methods, which considered the individual data points and



(a) **Circle:** $x^2 + y^2 - 4^2$

(b) **Elliptic Curve:** $x^3 + x - y^2 - 1.5$

(c) **Sphere:** $x^2 + y^2 + z^2 - 1^2$

(d) **Harmonic Oscillator:** $\ddot{\theta} - 0.1\dot{\theta} + 3x$

(e) **Pendulum:** $\ddot{\theta} - 0.1\dot{\theta} + 9.8\sin(\theta)$

(f) **Spring Pendulum:** $-10r + 0.5\dot{r}^2 + 5r^2 - 10rcos(\theta) + 0.5r^2\dot{\theta}^2$

Figure 5-5. The fitness and equation complexity Pareto fronts found for each of the six systems. The exact solutions are the simplest equations to reach near perfect fitness. More complex solutions show elaborations on the exact solution, improving fitness only marginally.

the zeros of the implicit equations directly, were unable to solve the simplest implicit equations reliably or consistently.

We showed that looking instead at ratios of partial derivatives of local data points provided a reliable search gradient for a variety of implicit systems. This method identified geometric equations such as elliptic curves and 3-dimensional spheres, as well as equations of motions in nonlinear dynamical systems.

Acknowledgment

This research is supported in part by the U.S. National Science Foundation Graduate Research Fellowship and by U.S. Defense Threat Reduction Agency (DTRA) grant HDTRA1-09-1-0013.

References

Bautu, Elena, Bautu, Andrei, and Luchian, Henri (2005). Symbolic regression on noisy data with genetic and gene expression programming. In *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05)*, pages 321–324.

Bongard, Josh and Lipson, Hod (2007). Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948.

De Falco, Ivanoe, Cioppa, Antonio Della, and Tarantino, Ernesto (2002). Unsupervised spectral pattern recognition for multispectral images by means of a genetic programming approach. In Fogel, David B., El-Sharkawi, Mohamed A., Yao, Xin, Greenwood, Garry, Iba, Hitoshi, Marrow, Paul, and Shackleton, Mark, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 231–236. IEEE Press.

Duffy, John and Engle-Warnick, Jim (1999). Using symbolic regression to infer strategies from experimental data. In Belsley, David A. and Baum, Christopher F., editors, *Fifth International Conference: Computing in Economics and Finance*, page 150, Boston College, MA, USA. Book of Abstracts.

Hetland, Magnus Lie and Saetrom, Pal (2005). Evolutionary rule mining in time series databases. *Machine Learning*, 58(2-3):107–125.

Korns, Michael F. (2006). Large-scale, time-constrained symbolic regression. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic Programming Theory and Practice IV*, volume 5 of *Genetic and Evolutionary Computation*, chapter 16, pages –. Springer, Ann Arbor.

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

Mackin, Kenneth J. and Tazaki, Eiichiro (2000). Unsupervised training of Multiobjective Agent Communication using Genetic Programming. In *Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technology*, volume 2, pages 738–741, Brighton, UK. IEEE.

Mahfoud, Samir W. (1995). *Niching methods for genetic algorithms*. PhD thesis, Champaign, IL, USA.

McConaghy, Trent, Palmers, Pieter, Gielen, Georges, and Steyaert, Michiel (2008). Automated extraction of expert domain knowledge from genetic programming synthesis results. In Riolo, Rick L., Soule, Terence, and Worzel,

Bill, editors, *Genetic Programming Theory and Practice VI*, Genetic and Evolutionary Computation, chapter 8, pages 111–125. Springer, Ann Arbor.

Schmidt, Michael and Lipson, Hod (2007). Comparison of tree and graph encodings as function of problem complexity. In Thierens, Dirk, Beyer, Hans-Georg, Bongard, Josh, Branke, Jurgen, Clark, John Andrew, Cliff, Dave, Congdon, Clare Bates, Deb, Kalyanmoy, Doerr, Benjamin, Kovacs, Tim, Kumar, Sanjeev, Miller, Julian F., Moore, Jason, Neumann, Frank, Pelikan, Martin, Poli, Riccardo, Sastry, Kumara, Stanley, Kenneth Owen, Stutzle, Thomas, Watson, Richard A, and Wegener, Ingo, editors, *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, volume 2, pages 1674–1679, London. ACM Press.

Schmidt, Michael and Lipson, Hod (2009). Distilling Free-Form Natural Laws from Experimental Data. *Science*, 324(5923):81–85.

Schmidt, Michael D. and Lipson, Hod (2006). Co-evolving fitness predictors for accelerating and reducing evaluations. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic Programming Theory and Practice IV*, volume 5 of *Genetic and Evolutionary Computation*, chapter 17, pages –. Springer, Ann Arbor.

Shpitalni, M. and Lipson, H. (1995). Classification of sketch strokes and corner detection using conic sections and adaptive clustering. *ASME Journal of Mechanical Design*, 119:131–135.

Smits, Guido and Kotanchek, Mark (2004). Pareto-front exploitation in symbolic regression. In O'Reilly, Una-May, Yu, Tina, Riolo, Rick L., and Worzel, Bill, editors, *Genetic Programming Theory and Practice II*, chapter 17, pages 283–299. Springer, Ann Arbor.