Discovery of Equations: Experimental Evaluation of Convergence *

Robert Zembowicz **Jan M. Żytkow**
Department of Computer Science, Wichita State University, Wichita, KS 67208
robert@wise.cs.twsu.edu zytkow@wise.cs.twsu.edu

Abstract

Systems that discover empirical equations from data require large scale testing to become a reliable research tool. In the central part of this paper we discuss two convergence tests for large scale evaluation of equation finders and we demonstrate that our system, which we introduce earlier, has the desired convergence properties. Our system can detect a broad range of equations useful in different sciences, and can be easily expanded by addition of new variable transformations. Previous systems, such as BACON or ABACUS, disregarded or oversimplified the problems of error analysis and error propagation, leading to paradoxical results and impeding the true world applications. Our system treats experimental error in a systematic and statistically sound manner. It propagates error to the transformed variables and assigns error to parameters in equations. It uses errors in weighted least squares fitting, in the evaluation of equations, including their acceptance, rejection and ranking, and uses parameter error to eliminate spurious parameters. The system detects equivalent terms (variables) and equations, and it removes the repetitions. This is important for convergence tests and system efficiency. Thanks to the modular structure, our system can be easily expanded, modified, and used to simulate other equation finders.

Introduction

All discovery systems that discover numerical regularities (BACON: Langley, Simon, Bradshaw, & Żytkow 1987; ABACUS: Falkenhainer & Michalski 1986; FAHRENHEIT: Żytkow, 1987, Żytkow, Zhu, & Hussam 1990; IDS: Nordhausen & Langley 1990; KEPLER: Wu & Wang 1989; E*: Schaffer 1990) use modules that find mathematical equations in two variables from a sequence of data. In this paper we describe an empirical Equation Finder (called EF) implemented with a particular concern towards a broad scope of equations it can detect, modularity of design, flexibility of control, and sound treatment of error. All these features are important for a high quality equation finder useful in modern science. Because empirical equations are discovered for many purposes, and are used to acquire more knowledge, a faulty mechanism for equation discovery may have a major detrimental effect on other discoveries.

Scientists recognize that a measurement error is associated with each experimental data point, and that all knowledge derived from experimental data carries corresponding errors. EF implements these tenets and systematically applies the theory of error propagation. EF computes the error for each transformed variable and for each parameter in each equation. The parameter error is needed to generalize the equations to more than two variables, and FAHRENHEIT uses these values in the generalization process (Żytkow, Zhu & Hussam 1990). BACON, ABACUS, and IDS use a constant system error but that causes undesired and paradoxical results and impedes their use in practical applications. E* ignores the error altogether.

Error plays several other roles in EF. It is used in the weighted least square fit for each type of equation and in statistical tests that evaluate and rank the equations. Parameter error is used in EF to eliminate spurious terms in the equations.

In the central part of this paper we analyze the convergence properties of EF. When a known equation is used to generate data, the evaluation of EF's results is unproblematic, because the source equation should be among the results. We describe two convergence tests, based on this idea, which must be satisfied by a scientific quality equation finder. We demonstrate the performance of our system on those tests. Previous equation finders do not perform as well because they fail the errors.

EF: model fitting and evaluation

Input to EF consists of N numeric data points (x_i, y_i, σ_i), $i = 1, \ldots, N$, where x_i are values of the independent variable x, y_i are values of the dependent variable y, σ_i represents the uncertainty of y_i (scientists call it *error*, for statisticians it is *deviation*, while the term *noise* is often used in AI). Output is a list of

*Work supported by the Office of Naval research under the grants No. N00014-88-K-0226 and N00014-90-J-1603.

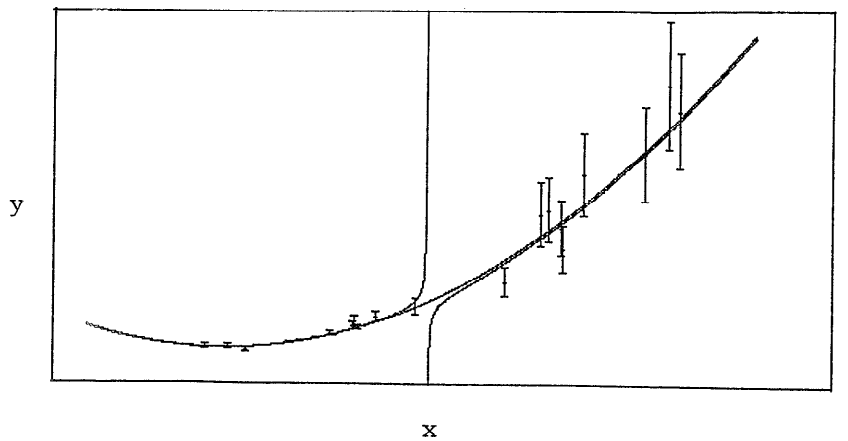

Figure 1. Sample data and 3 equations with similar goodness of fit

acceptable models.

Model Fitting. Model Fitter, a component of EF, uses *chi-square fitting*, known also as *weighted least-squares* (Eadie *et al.* 1971, Press *et al.* 1989) to fit given numeric data points (x_i, y_i, σ_i) to a finite number of models. Model is a function template $y = f(x, a_1, \ldots, a_q)$ (for example, $y = a_1 + a_2x + a_3x^2$) whose parameters' values $a_1, \ldots, a_q$ are determined by the requirement that the value of χ^2,

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - f(x_i, a_1, \ldots, a_q)}{\sigma_i} \right)^2 \qquad (1)$$

is minimal. The value of χ^2 is the sum of squares of deviations of data points (x_i, y_i) from the model, weighted by errors σ_i, so that measurements which are more precise carry greater weight. At theminimum of χ^2, its derivatives with respect to the a_j all vanish,

$$\sum_{i=1}^{N} \frac{y_i - f(x_i, a_1, \ldots, a_q)}{\sigma_i^2} \cdot \frac{\partial f(x_i, a_1, \ldots, a_j, \ldots, a_q)}{\partial a_j} = 0, \qquad (2)$$

for $j = 1, \ldots, q$. In general, the set (2) of equations is non-linear and not easy to solve. For a polynomial model, however, the set (2) can be solved by algebraic or matrix operations, producing efficiently a unique solution. Our Model Fitter can consider polynomials of any degree. In addition to the original variables x and y Model Fitter can work on data expressed in any transformed variables: $x' = x'(x)$ and $y' = y'(x, y)$.

Parameter Error. Standard deviations of parameters $a_1, \ldots, a_q$ at the values that minimize χ^2 are calculated according to the formula for error propagation (Eadie *et al.* 1971):

$$\sigma_{a_j}^2 = \sum_{i=1}^{N} \left(\frac{\partial a_j}{\partial y_i} \right)^2 \cdot \sigma_i^2, \qquad j = 1, \ldots, q \qquad (3)$$

(parameter values $a_j, j = 1, \ldots, q$ are solutions of equations (2), therefore they are functions of x_i, y_i, σ_i).

Removing vanishing parameters. If the absolute value of a fitted parameters a_j is smaller than the corresponding error σ_{a_j}, then zero could be also a good value for a_j. EF sets this parameter to zero. The new simplified equation has a similar goodness of fit (Zembowicz & Żytkow 1991).

Figure 1 demonstrates the need for the removal of vanishing parameters. For the plotted data, EF finds 3 models: $y = a_1 + a_2x + a_3x^2$, $y = b_1 + b_2x + b_3x^2 + b_4x^3$, and $y = c_1 + c_2x + c_3x^2 + c_4/x$; b_4 and c_4 are "zero-valued": $|b_4| << \sigma_{b_4}$, $|c_4| << \sigma_{c_4}$. Note that (i) there is no visible difference between the first two models; (ii) the data does not prefer any of the models, all have almost the same goodness of fit defined by (1), but $y = a_1 + a_2x + a_3x^2$ is the simplest. Higher degree polynomials will always be created, it is important to eliminate those which are unnecessary.

Fit Evaluation. For the best fit to each model, EF calculates the value of χ^2 according to Equation (1), and assigns the probability $Q = Q(\chi^2, N - q)$ ($N - q$ is the number of degrees of freedom in the data left after the fit) that this χ^2 value has not been reached by chance. A small value of Q means that the discrepancy between the data (x_i, y_i, σ_i) and the model $y = f(x, a_1, \ldots, a_q)$ is unlikely to be a chance fluctuation. The *threshold of acceptance* is defined using the probability Q: all models for which Q is smaller than some minimum value Q_{min}, are rejected. The best models are those with the largest values of Q.

EF: the search space

The class of polynomial models is too narrow for many applications. To extend the scope of detectable functions, EF uses data transformations, combined with application of polynomial model fitters to the transformed data (Figure 2). For each pair of variables x',

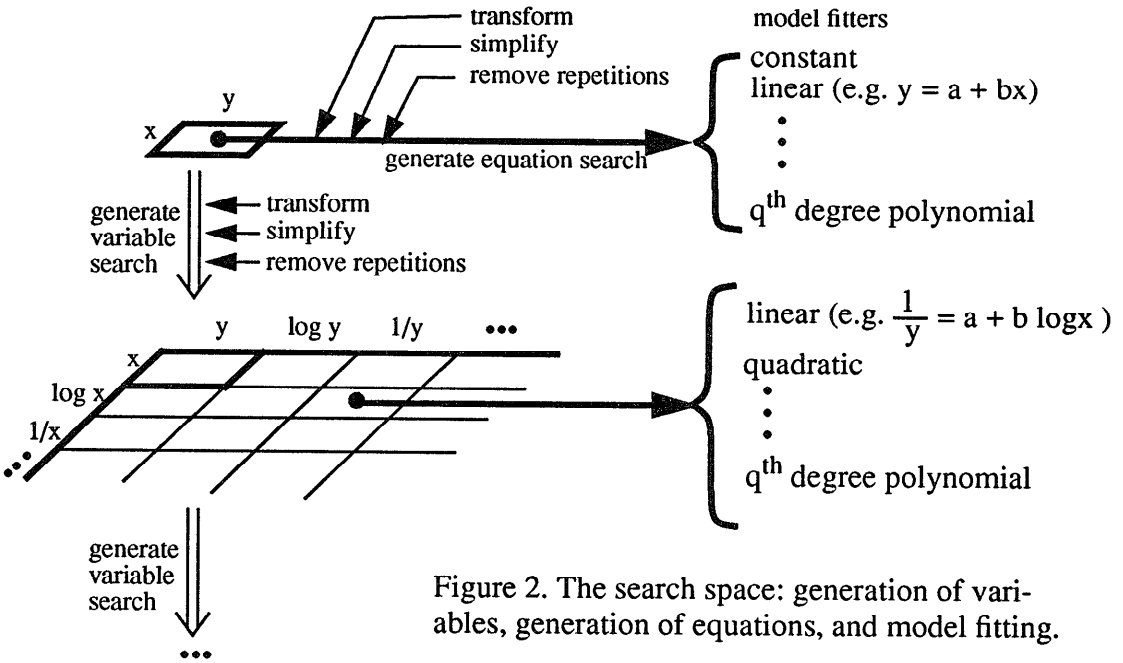

Figure 2. The search space: generation of variables, generation of equations, and model fitting.

y', and the error σ' of y', Equation Generation Search tries to find acceptable models.

Generation of new variables. The variable Generation Search (left hand side of Figure 2) creates new variables by applying a number of transformation rules, initially to the original variables x and y,

$$x \rightarrow x' = t_1(x), \tag{4}$$

$$(x, y, \sigma) \rightarrow (y', \sigma') = \left(y'(x, y), \left| \frac{\partial y'}{\partial y} \right| \cdot \sigma \right), \tag{5}$$

and then to the transformed variables. Note, that each time when EF uses y in a transformation, it also transforms the associated σ according to (5). Each new term is simplified and then compared against all the previous terms to ensure its uniqueness.

The default set of transformations used by EF includes logarithm, exponent, inverse, square root, multiplication, and division. Adding new transformation rules is simple. Only the transformation formula must be provided; the formula for error transformation is automatically developed by the module that calculates analytical derivatives. For example, the inverse and multiplication transformations are defined by

name:	INVERSE	PRODUCT
parameters:	(p)	$(p\ q)$
formula:	$(/\ 1\ p)$	$(*\ p\ q)$
application condition:	$p \neq 0$	–

Equation Generation Search. For each possible combination of variables x' and y', and for each polynomial up to the maximum user-specified degree, the operator Generate New Equations proposes a polynomial equation $y' = f(x', a_1, \ldots, a_q)$, which is then solved for y, if possible. If this seems a new equation, the Model Fitter finds the best values for the parameters $a_1, \ldots, a_q$, and then evaluates the model, as described in the previous section.

The search driver coordinates the search in the spaces of variables and equations (Figure 2). The system applies the Generate New Variable operator to extend the space of variables, and the Generate New Equation operator to generate, fit, and evaluate equations.

The search starts from the two original variables: x and y (upper left part of Figure 2). The application of the Generate New Equation operator leads to a constant model $y = a$, linear model $y = a + bx$, and other polynomials up to the predefined maximum polynomial degree (upper right part of Figure 2). If none of the models passes the evaluation, the Generate New Variables operator is applied repeatedly (see lower left part of Figure 2). If we consider only three transformations: logarithm, inverse, and multiplication, then the new variables would be:

$$\log x, \quad \frac{1}{x}, \quad xy, \quad \log y, \quad \frac{1}{y}$$

Now, using the original and new variables, the Generate New Equation search applies to each combination of x-type and y-type variables would form a large number of models (lower right part of Figure 2). For example, if maximum polynomial degree is 1, then the

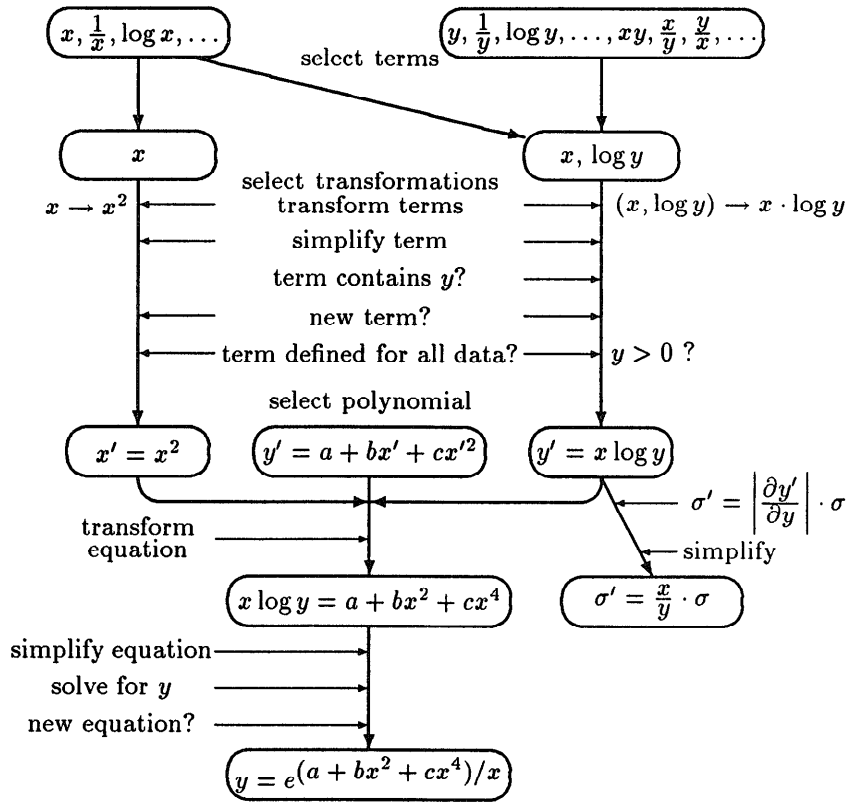

Figure 3. Generation of new variables and equations: the backtrace of all actions responsible for the generation of equation $y = e^{(a+bx^2+cx^4)/x}$

following new equations would be generated:

$$y = a + b/x \quad y = ae^{bx} \quad y = a + b\log x$$
$$y = \frac{1}{a+bx} \quad y = ax^b \quad y = \frac{1}{a+b\log x} \quad y = \frac{a}{x} + \frac{b}{x^2}$$
$$y = \frac{x}{a+bx} \quad y = ae^{b/x} \quad y = \frac{a+b\log x}{x}$$

If still none of the models is acceptable, the Generate New Variable search applies again. This operator, for the same three transformations, would generate the following new variables:

$$\log(\log x), \frac{1}{\log x}, x\log x, \frac{\log x}{x}, xy\log x, y\log x, \frac{y}{x},$$
$$\log(\log y), \frac{1}{\log y}, y\log y, \frac{\log y}{y}, xy\log y, x\log y, \frac{x}{y},$$
$$x^2y, xy^2, \log(xy), \frac{1}{xy}, \log x \cdot \log y, \frac{\log x}{y}, \frac{\log y}{x}$$

Note that the number of variables grows rapidly.

Figure 3 demonstrates the creation of new variables and equations from another perspective. It depicts the backtrace of actions needed to generate the equation $y = e^{(a+bx^2+cx^4)/x}$; this equation can be generated at depth two in the Generate New Variables search (the original variables x and y are at depth zero).

The search terminates when an acceptable model is found or at the predefined maximum search depth.

User control over search The user can change the maximum depth of both searches, that is, the maximum polynomial degree and the maximum transformation search depth. The user can also specify which transformation rules are to be considered. Another parameter controls the minimum search depth. It can force EF to continue the search even if an acceptable model was found. This option is particularly useful when the user is not satisfied with any of the models already found and wants EF to deepen the search. The minimum search depth is also needed for the two convergence tests described in the next section.

This simple search driver is very useful for testing and for studying the properties of EF, but it can be easily rearranged in various ways.

EF: experimental evaluation of convergence

Various tests can be applied to equation finders. One important application of EF is in a scientific laboratory, as a part of a machine discovery system (Żytkow, Zhu, & Hussam 1990). That application, however, does not allow us for an easy systematic testing of EF on a broad range of equations. In addition, it does not provide us with an undisputed "right answer", against

which we can compare the outcome of EF. Even if we collect data about a phenomenon which is described by a particular equation E, it is always possible that an interference of another unexpected phenomenon will result in data that does not satisfy E. Schaffer (1990) employed another schema of testing. He uses the actual equations proposed by scientists as the standard of validity. Although the result proposed by the scientist should be among those proposed by EF within the measurement error, this approach is not appropriate for testing EF alone, because scientific standards of acceptance combine several criteria, some of which use background knowledge not available to EF, such as interpretation of parameters, and derivability from other equations.

In this paper we propose two convergence tests that apply to any empirical equation finding (A-EF) system to verify its performance on all equations it can considered. Both tests use data generated from known equations. For each equation $y = f(x)$, we use f to generate data, adding Gaussian error of controlled size to all data points. The source equation $y = f(x)$ should be among the acceptable solutions found by A-EF. However, one cannot require that the source equation is the unique solution, even if the system removes zero-valued parameters described previously.

First, the larger are experimental errors, the more models fit the data with similar goodness of fit. Second, the number of acceptable models increases also when the range of values of the independent variable x represented in data is narrowed down and does not characterize the overall behavior of $f(x)$. For most real-world data our EF reports more than one acceptable model.

Both these multiple solution problems can be solved or alleviated if the user can make additional experiments in the expanded range of x and/or improve the accuracy of experiments. If the source function $f(x)$ is in the scope of A-EF's search, then a gradual reduction of error should eventually leave the source function as the only acceptable model. When the scope of the data is expanded, the situation is not so simple, although many initially accepted models can be ruled out. Consider, for example, the function $y = a\sin(bx)$. If the experimental errors σ are larger than $|a|$, then $y = 0$ is an acceptable model, no matter how far the range of x is extended. Only if the error is reduced, this procedure may lead to a unique solution.

The following two tests can verify the convergence of any equation finding system (A-EF) to the source equation. In both tests, we automated the gradual expansion of the scope and reduction of error.

Test #1. Given an empirical equation finding system,

1. Take a function $y = f(x)$ recognizable by A-EF.
2. Choose an error level E and a range R_x of x.
3. Generate random data points x_i, $i = 1, \ldots, N$, $x_i \in R_x$, generate y_i according to f, and add random Gaussian errors to y_i according to E.
4. Run the curve fitter to the depth sufficient to discover f.
5. If f is not among acceptable models, then report the failure of A-EF.
6. If there is only one acceptable model,
   then go to step 1 and take another function,
   else extend the range R_x and go to step 3.

A-EF passes test #1 for a class F of functions within its closure, if for each function f in F the test #1 converges to f, except when the difference between the maximum and minimum values of f is significantly smaller than the error E.

Test #2. This test differs from test #1 only in step 6. Instead of increasing the range, the error is decreased:

6'. If there is only one acceptable model,
   then go to step 1 and take another function,
   else reduce the error level E and go to step 3.

A-EF is passes the second test on a class of functions F, if the loop converges to f for each function f in F.

We have implemented a convergence tester which uses both tests. It generates data points and calls an equation finding system to be evaluated.

We have run both tests on our EF for all types of functions $y = f(x, a_1, \ldots, a_q)$ that can be discovered at the first level of transformations for the maximum polynomial degree of 3. We have used the logarithm, exponent, inverse, square, square root, division, and multiplication as transformations. Under these conditions, our EF tries 193 possible models.

For each function f, we have run both tests several times, varying the number of generated data points from 20 to 100. All parameter values a_i for f were generated randomly for each test run. The tables show ranges of the independent variable x. The error level of 10% means that for each data point the error was drawn from the normal (Gaussian) distribution of the deviation set to 10% of the value of the dependent variable y. The tables show counts of acceptable models ($Q \geq 0.001$) with the zeroing parameters removed (with the exception of Table 1).

Tables 1–4 present selected results. Table 1 presents combined results of both tests for the function $y = \sqrt{a + b\sqrt{x} + cx}$ when vanishing parameters $|a_i| < \sigma_{a_i}$ were not removed. The number of acceptable models does not converge to 1. However, if the vanishing parameters are removed, the resulting equations are simplified and removed if they were already discovered, then the number of acceptable models goes down to one, (Table 2). This demonstrates the importance of parameter zeroing.

Table 3 shows the result of the first test (for the fixed relative error $E = 0.1$), while table 4 shows the result of the second test (for fixed range $R = [1, 10]$) respectively, for several functions. The number of expected

Table 1: The number of acceptable models for $y = \sqrt{a + b\sqrt{x} + cx}$

Error level	Range of independent variable [1,5]	[1,7.5]	[1,10]	[1,12.5]	[1,15]
10%	110	77	54	38	30
1%	44	25	17	13	9
0.1%	15	10	5	4	3
0.01%	6	3	2	2	2

Table 2: The number of acceptable models for $y = \sqrt{a + b\sqrt{x} + cx}$; zero-valued parameters removed

Error level	Range of independent variable [1,5]	[1,7.5]	[1,10]	[1,12.5]	[1,15]
10%	76	62	41	29	21
1%	37	23	15	12	8
0.1%	14	9	4	3	2
0.01%	5	2	1	1	1

Table 3: Test #1: number of acceptable models for several equations; relative error is fixed at 10%

Source model	Range of independent variable [1,5]	[1,7.5]	[1,10]	[1,15]
$y = a + b\sqrt{x} + cx$	9	3	2	1
$y = (a + bx + cx^2)^2$	8	3	2	1
$y = a + be^x + ce^{2x}$	5	2	2	1
$y = \log(a + bx + cx^2)$	10	3	1	1
$y = (a + bx^2 + cx^4)^2$	11	2	2	1
$y = \sqrt{a + be^x + ce^{2x}}$	9	4	3	1
$y = \log(a + b\sqrt{x} + cx)$	7	4	1	1

Table 4: Test#2: number of acceptable models for several equations; the range is fixed at x in [1, 10]

Source model	Error level 10%	1%	0.1%	0.01%
$y = a + b\sqrt{x} + cx$	38	15	2	1
$y = (a + bx + cx^2)^2$	46	12	2	1
$y = a + be^x + ce^{2x}$	28	7	2	1
$y = \log(a + bx + cx^2)$	59	14	1	1
$y = (a + bx^2 + cx^4)^2$	39	10	2	1
$y = \sqrt{a + be^x + ce^{2x}}$	39	14	3	1
$y = \log(a + b\sqrt{x} + cx)$	40	10	1	1

models converges to 1 when the range of x increases or the error level decreases, as expected.

Conclusions

In contradistinction to its predecessors, EF treats experimental error and evaluation of equations in a systematic and statistically sound manner. We have introduced two convergence tests that should be satisfied by any equation finding system. We have run those simulation tests on our EF and we have found that our system has the desired convergence properties. The convergence tests require the elimination of equivalent variables and equations, which is important also for the efficiency of search.

Thanks to the modular structure of EF, it can simulate various equation finding systems, such as different versions of BACON1, or the equation finders of ABACUS and IDS. For instance, BACON1 (Langley et al. 1987) can be simulated by selecting transformations xy and x/y (conditional upon monotonicity of data), by applying constant and linear models, and by setting $\sigma_{y'} = \sigma_y$ for each transformed variable y'.

Our EF has been successfully applied to scientific laboratory data and was incorporated into a larger machine discovery system FAHRENHEIT used in real world applications (Żytkow, Zhu, and Hussam 1990) and into Forty-Niner (Żytkow and Baker 1991), which mines databases in search for useful regularities. Because equations are seldom a useful description of data in databases, the EF search in databases is triggered by a simple functionality test which uses a general definition of a function.

References

Eadie, W.T., Drijard, D., James, F.E., Roos, M., Sadoulet, B. 1971. *Statistical Methods in Experimental Physics*, North-Holland Publ.

Falkenhainer, B.C., & Michalski, R.S. 1986. Integrating Quantitative and Qualitative Discovery: The ABACUS System, *Machine Learning, 1*: 367-401.

Langley, P., Simon, H.A., Bradshaw, G., & Żytkow J.M. 1987. *Scientific Discovery; Computational Exploration of the Creative Processes*, Boston, MA: MIT Press.

Nordhausen, B., & Langley, P. 1990. An Integrated Approach to Empirical Discovery. in: J.Shrager & P. Langley eds. *Computational Models of Scientific Discovery and Theory Formation*, 97-128, San Mateo, CA: Morgan Kaufmann Publ..

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T. 1989. *Numerical Recipes in Pascal*, Cambridge: Cambridge University Press.

Schaffer, C. 1990. A Proven Domain-Independent Scientific Function-Finding Algorithm, *Proceedings of AAAI-90*, 828-833, Menlo Park, CA: AAAI Press.

Wu, Y., & Wang, S. 1989. Discovering Knowledge from Observational Data, in: Piatetsky-Shapiro, G. (ed), *Knowledge Discovery in Databases, IJCAI-89 Workshop Proceedings*, Detroit, MI, 369-377

Zembowicz, R., and Żytkow, J.M. 1991. Automated Discovery of Empirical Equations from Data, *Proceedings of ISMIS-91*, 429-440, Springer-Verlag.

Żytkow, J.M. 1987. Combining Many Searches in the FAHRENHEIT discovery system, *Proceedings of Fourth International Workshop on Machine Learning*, 281-287, Los Altos, CA: Morgan Kaufmann Publ..

Żytkow, J.M., Zhu, J. & Hussam, A. 1990. Automated Discovery in a Chemistry Laboratory, *Proceedings of the AAAI-90*, 889-894, Menlo Park, CA: AAAI Press.