Global Solutions for Nonlinear Systems Using Qualitative Reasoning[*]

E. Bradley[†]
University of Colorado, Department of Computer Science, Boulder, CO 80309-0430
lizb@cs.colorado.edu

A. O'Gallagher and J. Rogers
National Institute of Standards and Technology, Boulder CO 80303-3328
[abbie,jrogers]@boulder.nist.gov

Annals of Mathematics and Artificial Intelligence, **23**:211-228 (1998)

Abstract

This paper explores how qualitative information can be used to improve the performance of global optimization procedures. Specifically, we have constructed a nonlinear parameter estimation reasoner (NPER) for finding parameter values that match an ordinary differential equation (ODE) model to observed data. Qualitative reasoning (QR) is used within the NPER, for instance, to intelligently choose starting values for the unknown parameters and to empirically determine when the system appears to be chaotic. This enables odrpack, the nonlinear least-squares solver that lies at the heart of this NPER, to avoid terminating at local extrema in the regression landscape. odrpack is uniquely suited to this task because of its efficiency and stability. The NPER's robustness is demonstrated via a Monte Carlo analysis of simulated examples drawn from across the domain of dynamics, including systems that are nonlinear, chaotic, and noisy. It is shown to locate solutions for noisy, incomplete real-world sensor data from radio-controlled cars used in the University of British Columbia's soccer-playing robot project. The parameter estimation scheme described in this paper is a component of pret, an implemented computer program that uses a variety of artificial intelligence techniques to automate *system identification* — the process of inferring an internal ODE model from

[*] Contribution of the National Institute of Standards and Technology and not subject to copyright in the United States.
[†] Supported by NSF NYI #CCR-9357740, NSF #MIP-9403223, ONR #N00014-96-1-0720, and a Packard Fellowship in Science and Engineering from the David and Lucile Packard Foundation.

external observations of a system — a routine and difficult problem faced by engineers from various disciplines.

1 Introduction

System identification (SID), the process of inferring an internal ordinary differential equation (ODE) model from external observations of a system, is a routine but difficult problem faced by engineers; consider, for example, building a controller for a radio-controlled (R/C) car with unknown dynamics. In general, SID proceeds in two interleaved phases: first, *structural identification*, in which the form of the equation is determined, and then *parameter estimation*, in which values for the coefficients are obtained. If structural identification produces an incorrect model, no coefficient values can make its solutions match the sensor data. In this event, the structural identification process must be repeated—often using information about why the previous attempt failed—until the process converges to a solution, as shown diagrammatically in Figure 1.

In linear physical systems, parameter estimation is well-understood. One textbook approach [1] is to choose a generic ODE system $\dot{\vec{x}} = A\vec{x}$, fast-Fourier-transform (FFT) the sensor data, and use the characteristics of the resulting impulse response[1] to determine the coefficients of A. If a solution exists, this process is relatively straightforward. The difficulties—and the subtleties employed by practitioners—arise where noisy or incomplete data are involved, or where efficiency is an issue. See [1, 2] for some examples.

In *nonlinear* systems, SID is vastly more difficult. Because linear signal processing methods like FFTs do not apply, we must fall back on regression, and nonlinear regression landscapes typically exhibit local extrema that can trap numerical methods. Finding the optimal solution in such landscapes is the difficult problem addressed by global optimization research [3, 4]. This paper contributes to this body of work by presenting a new, highly effective global optimization method—constructed using a combination of qualitative reasoning (QR) and local optimization techniques—for the nonlinear problems encountered in SID.

The context within which we apply these ideas is the computer program pret [5], which automates the SID process diagrammed in Figure 1 by building an artificial intelligence (AI) layer on top of a set of traditional SID techniques. This AI layer automates the high-level stages of the identification process that are normally performed by a human expert. In particular, several forms of QR are combined in a custom logic system [6, 7] to perform structural identification, assembling a combination of user-specified and

[1]The natural frequencies, which appear as spikes on the impulse response, yield its eigenvalues; the off-diagonal elements can be determined via a residual analysis of the *mode shapes* between those spikes.

automatically generated model fragments into nonlinear ODE models that fit both the domain physics and the observations.

pret's nonlinear parameter estimation reasoner (NPER), the specific topic of this paper, uses QR techniques to automate the selection of coefficient values for these ODE models—the functions represented by the lower box in Figure 1. The method that lies at the core of the NPER is odrpack [8, 9], a robust nonlinear least-squares solver. Around this core is built a layer of QR techniques that allow pret to automatically interact with and exploit odrpack's unique and powerful features. This layer can, for instance, intelligently choose starting values for the unknown coefficients, helping odrpack avoid local extrema. QR can be used to determine cutoff frequencies for filtering algorithms, so noise can be removed without disturbing the data's structure. Given qualitative observations, a mainstay of pret, we can also use QR to interpret ODRPACK's results on an abstract level—quickly and yet correctly. QR-guided parameter estimation techniques like these, a collection of which are the topic of the remainder of this paper, are elements of the type of analysis that an expert human user would perform during an interaction with odrpack.

The next section introduces pret by way of the real-world example previously mentioned: the R/C cars used in the University of British Columbia's soccer-playing robot project. These devices cannot be controlled without an accurate ODE model of their dynamics—something that is not part of the manufacturer's specifications sheet. Following this example—which shows *both* SID phases—we focus on the parameter estimation phase and describe the inner workings of the NPER. To demonstrate its capabilities, we present a suite of examples, drawn from across the domain of dynamics, including simulated systems that are nonlinear, chaotic, and noisy, as well as the real-world R/C car example mentioned above.

2 An Example

pret constructs ODE models of target systems, linear or nonlinear, in one variable or many. It is written in scheme [10] and maple [11]; its implementation is a hybridization of traditional numerical analysis methods, such as nonlinear regression, with QR and logic programming.

Figure 2 shows how a user instructs pret to build a model of the dynamics of an R/C car. The details of the syntax are covered elsewhere [12, 5]; briefly, the bulk of the user's input consists of three types of information about the target system: `hypotheses`, `observations`, and `specifications`. The first are ODE fragments from which pret constructs the model and the third prescribe resolutions to which that model must adhere. `Observations`, which play a more-important role in this paper, range from the purely quantitative to the purely qualitative. The source of the `numeric` observation in the `find-model` call of Figure 2, for example, is a camera above the car, while the qualitative observations were extracted from e-mail messages from the project analysts and engineers.

To construct an ODE model from this information, pret employs a special logic engine [6] to combine powerful mathematical formalisms, such as the link between the divergence of an ODE ($\nabla \cdot f$) and the friction of the system that it describes, with domain-specific notions, such as force balances in mechanical systems, in order to:

1. build ODE models from the user's hypotheses;
2. check those models against the observations;
3. manipulate actuators and interpret the resulting sensor data to verify or augment observations of the system structure [13];
4. and, if the user's input is inadequate, to
   - synthesize hypotheses from power-series expansions [14], and
   - infer unobserved internal state variables using time-series embedding theory [13].

Effective QR-guided parameter estimation, the focus of this paper and the goal of the methods described in the next section, is one element in step 2. In order to show how pret produces the type of qualitative information used therein, the next few paragraphs give a narrative description of both steps 1 and 2, the top and bottom boxes in Figure 1, respectively.

pret takes a generate-and-test approach. In the `mechanics` domain, it uses force balances to assemble hypotheses into models; in all domains, it examines hypothesis combinations in order of increasing complexity. In Figure 2, the first candidate model is $ay = 0$. A scheme function called on this ODE establishes the fact `(order <y> 0)` which expresses that the order of the highest derivative of y in this model is zero. This fact conflicts with facts inferred from simple geometric reasoning on the numeric observation—specifically, that the `<y>`-value of the time series in the `numeric` observation is not constant at the specified resolution—so this model is ruled out. This demonstrates pret's abstract-reasoning-first approach; only a few steps of inexpensive qualitative reasoning suffice to quickly discard the model.

pret's structural identification module then continues through a series of hypothesis combinations, checking each against the observations and discarding most in a similar fashion: for instance, all models involving the time-dependent term in the third hypothesis conflict with the `autonomous` observation; all whose divergences are non-negative are ruled out by conflict with the `damped` observation.

Eventually, pret produces the model:

$$\begin{aligned} \dot{x} &= v \, \cos\theta \\ \dot{y} &= v \, \sin\theta \\ \dot{\theta} &= \rho \, v \\ \dot{v} &= \alpha + \gamma \, v \ . \end{aligned} \tag{1}$$

None of the implemented rules disqualifies this model by purely qualitative means, so pret invokes the NPER. The resulting parameter estimates enable the ODE solutions to match the data to within the prescribed resolution, so this candidate model is returned as the answer. The details of this computation are the topic of the next section of this paper.

Space requirements force us to omit most of the known hypotheses and observations and much of the interesting reasoning that proceeds in this example. The point of this presentation is to provide context for the parameter estimation sections that follow. The actual R/C car `find-model` call contains many more hypotheses and qualitative observations, and most of the models that pret constructs from the former are discarded quickly and easily using QR on the latter.

3 Guiding Nonlinear Parameter Estimation with QR

The next two subsections describe our NPER, which takes as input a model with unknown parameters and some qualitative information derived by the outer layers of pret, and returns parameter values for the model.

3.1 Adapting the NLS Solver with QR

The method of nonlinear least squares (NLS) is commonly applied to the problem of estimating unknown parameters, say ξ, of a nonlinear function constructed to model observed values. These methods find parameter values $\hat{\xi}$ that minimize the sum of the squared differences between the fitted function values and the observed data. In our analysis, we fit the observed data to a series generated by a numerical ODE integration procedure that takes the definition of an ODE and produces a time-series approximation to its solution. We use the NLS procedure provided by odrpack [8, 9], an efficient and stable trust region Levenberg-Marquardt code, to estimate the coefficients β and initial conditions Y_0 that complete the definition of the ODE. The ODE integration routine we use, ddebdf from the package depac [15], is robust and suitable for the ODEs we encounter.

Most NLS procedures, including odrpack, solve a sequence of Taylor-series approximations to a nonlinear model. Starting values ξ^0 for the unknowns of the model—in this case, $\xi^0 \equiv [\beta^0, {Y_0}^0]$—are needed to form the initial NLS search direction. If ξ^0 are too far from the global solution ξ^*, the initial search direction is likely to be poor, and the estimation procedure will be unable to find a good solution. Because pret is designed for both linear and nonlinear ODE models, we cannot set ξ^0 by employing the techniques in [16]. However, we have been able to use QR to construct a robust strategy for obtaining the best possible solution for a given data set and a specific ODE model. Our strategy exploits the qualitative information derived by the outer layers of pret as well as the sophisticated features of odrpack. It also makes use of the facts that each variable is a relatively smooth function of time and that the data are homoscedastic, that is, observed with a constant-variance noise component. We use the first property in finding the solution, and the second in determining its adequacy.

Our strategy mimics human reasoning: it attempts to discriminate between the true structure in the data and the overlaid noise. The procedure is based on the following:

- If data are observed without errors (i.e., without noise), then the solution $\hat{\xi} \equiv [\hat{\beta}, \hat{Y}_0] \in \Re^p$ can be obtained by fitting the first p observations. This small system is less sensitive to ξ^0 and thus easier to solve.

- When the data are noisy, the solution $\hat{\xi}$ obtained using only the first p observations may not provide good results for the whole series.

An engineer would therefore begin the optimization process by fitting a segment whose length n_1 is just long enough to reflect the data's structure. Once a solution $\hat{\xi}_{n_1}$ is found for this smaller problem, the analyst can set ${\xi^0}_{n_2} \leftarrow \hat{\xi}_{n_1}$ and solve for $\hat{\xi}_{n_2}$, the solution given the data segment of length n_2. In this way, the engineer steps through the data until finally $\hat{\xi}_n \equiv \hat{\xi}$ is found using the full data set.

Our NPER automates the reasoning described in the preceding paragraph. The algorithm consists of five basic steps.

1. **Extract smoothed variables**—
   The data's structure is ascertained by estimating the total number of oscillations, using either a simple moving average or an FFT. This estimate is used to find the appropriate cut-off frequency for a low-pass filter that is applied to the observed data to remove the higher frequency noise. The smoothed data that result from this filtering operation are effectively noise-free.

2. **Estimate error variances**—
   If the magnitudes of the noise in the different state variables are not the same, the solution will be influenced more by the variables with larger errors than by the variables with smaller errors, even if their relative accuracy is the same. This distortion can be corrected by weighting the residuals for each variable by the reciprocal of the variance of the errors in this variable. Although these variances are not known *a priori*, they can be approximated by ${\tilde{\sigma}^2}_i$, the average of the squared differences between the values of the smoothed and raw data for the ith variable. In addition, the values of ${\tilde{\sigma}^2}_i$ can be used to construct a threshold value for testing when the estimated parameters $\hat{\xi}$ adequately capture the characteristics of the data. The test statistic is $\nu \equiv \hat{\sigma}^2/\tilde{\sigma}^2$ and ν-threshold $\equiv 2$, where $\hat{\sigma}^2$ denotes the residual variance of the least squares solution and $\tilde{\sigma}^2$ denotes the variance computed using the weighted differences between the observed and smoothed data: the solution is considered adequate if $\hat{\sigma}^2$ is less than twice $\tilde{\sigma}^2$.

3. **Fit smoothed variables**—
   If the data have no error component, it is relatively easy to estimate $\hat{\xi}$ using the first p points, even when good starting values ξ^0 are not available. Similarly, it is relatively easy to find a good solution for a small segment of length n_1 of the smoothed data, and these results will be approximately the same as those obtained using all of the raw data. Having solved for $\hat{\xi^s}_{n_1}$, the solution based on the first n_1 observations of the smoothed state variables, we compare ν to ν-threshold to determine whether the solution is acceptable. If it is, then we set $\xi^0{}_{n_2} \leftarrow \hat{\xi^s}_{n_1}$, and fit the data segment of length n_2. As long as $\nu \leq \nu$-threshold we double the length of the next data segment. If $\nu > \nu$-threshold, we reset $\xi^0 \leftarrow \hat{\xi^s} + \delta$, where δ is a small perturbation, and then re-fit a slightly shorter data sequence to try to obtain a better solution. In this case, we use smaller increments in the data segment length to step through the remainder of the series.

4. **Fit raw (un-smoothed) data**—
   Given good starting values, $\xi^0 = \hat{\xi^s}$, where $\hat{\xi^s}$ denotes the solution obtained using the smoothed data as described in step 3, we proceed to fit the raw data. However, even with good starting values we have found it advantageous to step through the data as described above. A serendipitous benefit of this step-through procedure is that it enables us to recognize chaotic models and modify the parameter adjustment procedure accordingly.

5. **Evaluate the final solution**—
   If $\nu \leq \nu$-threshold, the solution is deemed to be satisfactory; otherwise the model is deemed to be incompatible with the data. In either case, the value of ν, along with $\hat{\xi}$ and various other statistics, are returned to pret.

In the algorithm above, the number of low-frequency oscillations is the qualitative information about the data's structure that allows us to form the low-pass filter. The result of the filtering operation (step 1) enables the accurate construction of $\tilde{\sigma}^2{}_i$and provides effectively noise-free data, thereby reducing the difficulties caused by the lack of good starting values (step 3). Knowledge that the data is homoscedastic permits us to use $\tilde{\sigma}^2{}_i$ to form weights that equalize the effect of the different variables in the formation of the least-squares solution. It is also used to form ν, which is then used to determine when, and by how much, the length of the data segment can be increased (steps 3 and 4), and to indicate when there is evidence of chaotic behavior so that the procedure can be adjusted accordingly. In addition, ν is used to assess the quality of the final solution (step 5). Our algorithm combines these QR features with those embedded in the design of odrpack and depac producing a robust and versatile NPER. For example, the ODE routines in depac can detect if a system is stiff, thereby allowing appropriate adaptation.

3.2 Automating the NLS Solver Call with QR

The QR-adapted NLS procedure described in the previous subsection has a variety of facilities and inputs that an expert can use to tailor and focus the solution process, and pret's knowledge bases and reasoning systems concern exactly the type of higher-level information that is needed to exploit these hooks.

As shown in the R/C car example, pret performs a variety of QR tasks during its structural identification phase, involving both the `observations` and the candidate model. The knowledge derived in this process can be used to intelligently (a) set up the call to the QR-adapted NLS solver and (b) interpret the results that it returns. The inputs used in step (a) are the starting values β^0 and ${Y_0}^0$, and bounds on the coefficients and state variables; in step (b), the NPER reasons about the output statistic ν.

A parameter estimation call necessarily involves numerical data for system state variables, so at least *some* of the Y_0 are always observed. If *all* of the state variables have been observed, the starting value problem is easy; pret simply sets ${Y_0}^0$ to the first tuple in the data file. Any noise in those numbers is dealt with by the smoothing facilities described in the previous subsection.

A fully *observable* system, however, is extremely rare in engineering practice; as a rule, many—often, most—of the state variables either are physically inaccessible or cannot be measured with available sensors. This is control theory's *observer problem*: inferring the internal state of a system from observations of its outputs. In linear systems, this is difficult; in nonlinear systems, it is an open problem. pret does not attempt a general solution; it simply automates some of the simpler methods in the controls literature. Expanding pret's repertoire of *qualitative observer theory* techniques is a primary thrust in our current research efforts [13, 14].

Two forms of reasoning help the NPER solve parts of the observer problem, allowing it to compute values for the QR-adapted NLS solver's Y_0 arguments: divided differencing and symbolic algebra. As a first step in the odrpack call setup, divided differences are used recursively to fill in unknown starting values ${Y_0}^0$ for variables that are related by derivative chains to variables with known Y_0. For instance, if velocity were unobserved in the R/C car example, the starting value could be filled in using first-order forward differences on the (known) positions. A less-simplistic differencing method would generate better estimates, but because of the power of the methods described in the previous subsection, we have not found this to be necessary. Once the difference loop has filled in all possible Y_0, pret uses symbolic algebra, together with qualitative information inferred during the structural identification phase, to solve every equation in the model for each variable whose Y_0 remains unknown. This solve-and-substitute process is iterated over the set of equations until no further Y_0 values can be inferred.

The coefficient and state variable bounds arguments passed to the QR-adapted NLS solver can also be leveraged to guide odrpack's solution process. For example, consider a trial model, a second-order linear ODE, being matched against a system whose sensor observations contain an oscillation. The outer layers of pret recognize this oscillation using geometric reasoning [13]; the logic system [6, 7] then infers that the two roots of the model's characteristic polynomial must be complex. Symbolic algebra on the model coefficients produces the corresponding algebraic constraints, which, together with constraint propagation and whatever numerical information can be inferred from the other observations, are used to set up the parameter estimator's coefficient bounds. State variable bounds are also constructed in the obvious ways implied by the discretization of qualitative knowledge—for instance, if we know that a system's state is between two landmarks.

An expert user of odrpack can reason heuristically on the output statistics described in the previous section in order to decide whether a fit is good or bad. pret automates this process simplistically, using landmark thresholds whose values are determined by observing human experts. Currently, a model is deemed adequate if $\nu < 2$.

There are many other ways to use available qualitative information to improve the function of the NLS solver. For instance, filtering destroys various important features of chaotic data; if a system is observed to be chaotic, pret instructs the parameter estimator to use higher cutoff frequencies on its filtering schemes[2]. Simple geometric frequency-domain reasoning can identify potentially stiff systems[3] and guide the choice of numerical ODE integrator. These and related ideas are ongoing research topics in our group.

4 Monte Carlo Study

The task of the NPER described above is not simply to find the coefficients $\hat{\beta}$ and initial conditions $\hat{Y_0}$ that best fit the given data, but also to distinguish between good models and bad models. To be useful, the procedure must be able to

1. report "success" when the solution represents the data well, and
2. report "failure" otherwise.

It must do this reliably and as quickly as possible in the presence of noise, since sensor measurements are invariably noisy. In this section we present the results of a series of Monte Carlo experiments that verify that our NPER accomplishes these tasks.

The problems described in this section differ from the R/C car problem in that the target of the modeling task is not a sensor-equipped physical system, but rather a simulated time series from a known

[2]As discussed in the previous section, the QR-adapted NLS solver has ways of recognizing chaos and reacting accordingly; if this knowledge exists in the outer layers of pret, however, it is much more efficient to pass it along explicitly and avoid duplication of effort.

[3]One need only identify the characteristic widely spread peaks on the spectrum.

ODE. These 10 problems, shown in Figure 3, include systems that are linear (e.g., `Springs&Masses`), nonlinear (e.g., `Spring&Pendulum`), and chaotic (e.g., `Lorenz`). Each of these models is numerically integrated using the "true" parameter values—$\xi^*{=}[\beta^*,{Y_0}^*]$, listed in Figure 3—to produce the "true" values of the data, Y^*. "Observed" sets of data $Y_r, r = 1, 2, \ldots$, are then constructed using $Y_r \equiv Y^* + \epsilon_r$, where ϵ_r is the rth array of Gaussian noise generated using a model-specific, diagonal covariance matrix.

Figures 4, 5 and 6 show one such realization of observed data for `Springs&Masses`, `Spring&Pendulum`, and `Lorenz`, respectively. Observed data are shown as gray dots; true values are shown as a black dotted line; and the solution is shown as a black solid line. The fitted results match the data so well that, at this level of resolution, the dotted and solid lines are indistinguishable.

We test our NPER by applying it to each set of observed data (each of which is similar to the observed data in Figures 4, 5 and 6) using every model in Figure 3 of appropriate order. Starting values β^0 are set to a vector of ones, and ${Y_0}^0$ set to the first "-tuple" of the smoothed data. We analyze 100 realizations of the data when supplying the NPER the "correct" model, that is, the model used to generate the data. Ten realizations are examined when the model supplied to the NPER is not the one used to generate the data. There are a total of 1280 individual problems.

Figure 7 presents the results from our study when the model used to generate the data is the model supplied to the NPER. This figure shows that, when the NPER is given the correct model it can fit the data precisely 99% of the time. Furthermore, the NPER was able to correctly distinguish between adequate and inadequate models every time when given a model other than the one used to generate the data. In summary, the NPER correctly distinguishes between good and bad models 1270 times out of 1280—a better than 99% success rate.

Our parameter estimation procedure as well as the highly successful results we describe in this section depend critically upon the QR we apply. The significance of being able to solve such complex nonlinear systems without user-supplied *good* starting values cannot be overemphasized.

5 Parameter Estimation for the R/C Car

Figure 8 shows a noisy, real-world data set from sensors on an R/C car used in the University of British Columbia's soccer-playing robot project. As described earlier, pret uses QR and logic programming to derive the fourth-order model given by eq. (1) and set up the NPER call. The estimated parameters and standard deviations[4] are

$$\hat{\beta} = \begin{bmatrix} \rho \\ \alpha \\ \gamma \end{bmatrix} = \begin{bmatrix} 0.0661\pm0.0021 \\ 91.6\pm2.3 \\ -0.649\pm0.037 \end{bmatrix}$$

and

$$\hat{Y_0} = \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \\ v_0 \end{bmatrix} = \begin{bmatrix} 10.82\pm0.24 \\ 63.214\pm0.036 \\ 2.84\pm0.11 \\ -8.1\pm1.1 \end{bmatrix};$$

this solution is also shown graphically in Figure 8.

This model, although relatively easy compared to some of those in our Monte Carlo study, benefited significantly from the reasoning we employ to balance noise effects between variables. The report returned to pret is shown in Figure 9; since $\nu = 1.323$ is less than the heuristic threshold of 2, these numbers represent a successful fit.

This solution surprised the University of British Columbia analysts. They knew that the car had started from rest, and thus that the initial condition v_0 should be zero, not -8.1 as estimated. Further reflection upon this discrepancy, however, led them to realize that the system dynamics might include a delay, which was confirmed with a senior engineer on the project.[5] Thus, the NPER not only solved the system, but actually enabled the experts to identify what was wrong with the model fragments they had suggested.

These results underscore an important point: pret is an engineer's tool, not a scientific discovery system. Its goal is to construct the simplest ODE that accounts for the observations and specifications that appear *explicitly* in the `find-model` call, not to infer physics that the user left implicit. In this example, an incorrect model is supplied to the NPER because the `find-model` call omits two pieces of knowledge—the car starts from rest and the system has a delay. The ODE formed by pret thus meets the explicit requirements, but does *not* match the expert's intuition. By abusing the (implicit) boundary conditions, however, the NPER is able to fit the observed data within the required tolerances—and therefore it reports success. Similarly, pret does not attempt to exceed the `resolution` prescribed by its user; the lack of a `specification` for `<v>` in the last line of Figure 2, for instance, implies that an exact fit of that state variable is not important to the user. Because of this, pret does not add terms to the ODE in order to model the "bump" in the velocity data in Figure 8. This is not an unwelcome side

[4]The standard errors are derived under the assumption of a diagonal noise covariance. The validity of this assumption is questionable given the apparent correlation structure in the residuals for variable θ.

[5]The correspondence was impressive: the -8.1 value that the NPER estimates for v_0 is within one standard deviation of the hypothesized seven-cycle delay between command and response.

effect of the finite resolution; it is an intentional and useful by-product of the abstraction level of the modeling process.

6 Related Work

This paper discusses work in three broad areas: modeling, parameter estimation, and global optimization.

Modeling research spans many fields, from the cognitive science-related branch of AI [17] through design engineering [18], nonlinear dynamics [19], and control theory [20] to the qualitative reasoning community [21]. The spectrum ranges from models and tools that use a language that is very close to the physics of the system (e.g., QPT/QPE [22, 23]) to models that use a language that is well suited to describe the system mathematically (e.g., ODEs). Like many other projects, for example [24, 25], pret aims to integrate quantitative and qualitative information; unlike other QR modeling tools, it takes a practical engineering approach: it works with noisy, incomplete sensor data from partially observable real-world systems, and its aim is not to "discover" the underlying physics, but rather to find the simplest ODE that can account for the observed behavior.

The parameter estimation phase of the modeling process has also received attention in a variety of fields. This includes control theory's work on Kalman filtering [26], an interesting AI tool [27] that uses dynamics to improve the estimation process, and lately, research by the QR community [16].

pret's NPER is most closely related to the work of the QR community. The differences, however, are significant.

- pret and its NPER are designed for nonlinear (as well as linear) systems.
- pret uses only general mathematics in its analysis, and does not rely on knowledge specific to the domain in question.
- pret has a very general structural identification procedure that is not restricted to any specific form of equation.

Finally, nonlinear optimization is addressed in a variety of books, including [28], [29], and [30]. Quality software for solving the unconstrained NLS problem is available from packages such as odrpack [8, 9], minpack [31] and nl2sol [32]. The harder problem of finding global solutions to nonlinear optimization problems is being addressed by a number of researchers, including [33]. Two books on this topic are [4] and [3], the first of which is primarily about general continuous nonlinear problems, and the second of which addresses special cases such as concave problems and network problems.

7 Conclusion

This paper describes the construction of an NPER (nonlinear parameter estimation reasoner) that solves a *global* optimization problem by augmenting a *local* nonlinear least-squares solver with qualitative information and qualitative reasoning. This reasoner is part of pret, an implemented computer program that uses a variety of artificial intelligence techniques to automate *system identification*—the process of inferring an internal ODE model from external observations of a system—a routine and difficult problem faced by engineers.

The combination of qualitative reasoning and numerical methods is very powerful; it allows the NPER to find the optimal parameter choices to match an ordinary differential equation (ODE) model to a set of observed data. In particular,

- QR can effectively compensate for a lack of good starting values, thereby enabling the nonlinear least-squares solver to avoid local extrema in regression landscapes;
- QR enables the NPER to automatically determine and adapt to the structure of the data; and finally
- QR allows pret to quickly and correctly assess when a model is good and when it is bad.

We demonstrate the robustness of our NPER through a simulation study that uses problems drawn from across the domain of dynamics—including systems that are nonlinear, chaotic, and noisy; in these, our NPER is correct more than 99% of the time. Moreover, in a real-world application that involves modeling a commercial radio-controlled car, we show how pret and the NPER construct a model that not only adequately matches the experimental data, but actually enables the project analysts to identify what is wrong with their initial ideas about the system. Specifically, pret and the NPER built a model that matched the observations *but not their intuition*, and these disparities led them to understand the system dynamics better.

We use this anecdote to emphasize that pret is an engineer's tool, not a scientific discovery system. Its goal is not to infer physics that the user left implicit, but rather to construct the simplest model that accounts for the observed behavior of a high-dimensional black-box system. The power of the qualitative-reasoning based NPER described in this paper is a significant step in this direction.

Acknowledgments: The authors thank Reinhard Stolle, Matt Easley, and Brian LaMacchia for the code and/or ideas they contributed to this project, and Ray Spiteri for providing the R/C data.

References

[1] J.-N. Juang. *Applied system identification.* Prentice Hall, Englewood Cliffs, NJ, 1994.

[2] L. Ljung, editor. *System Identification: Theory for the User*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[3] R. Horst, P. Pardalos, and N. Thoai. *Introduction to Global Optimization*, volume 3 of *Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, Dordrecht, Germany, 1995.

[4] A. Torn and A. Zilinskas. *Global Optimization*. Number 350 in Lecture Notes in Computer Science. Springer Verlag, Berlin, 1989.

[5] E. Bradley and R. Stolle. Automatic construction of accurate models of physical systems. *Annals of Mathematics and Artificial Intelligence*, 17:1–28, 1996.

[6] R. M. Stolle and E. Bradley. Multimodal reasoning for automatic model construction. In *Proceedings AAAI-98*, 1998.

[7] R. Stolle and E. Bradley. Multimodal reasoning for automatic model construction. In *Proceedings AAAI-98*, 1998.

[8] P. T. Boggs, R. H. Byrd, J. E. Rogers, and R. B. Schnabel. User's reference guide for ODRPACK—software for weighted orthogonal distance regression. Internal Report 4103, National Institute of Standards and Technology, Gaithersburg, MD 20899, 1991.

[9] P. T. Boggs, R. H. Byrd, and R. B. Schnabel. A stable and efficient algorithm for nonlinear orthogonal distance regression. *SIAM Journal of Scientific and Statistical Computing*, 8(6):1052–1078, 1987.

[10] J. Rees and W. Clinger. The revised^3 report on the algorithmic language Scheme. *ACM SIGPLAN Notices*, 21:37, 1986.

[11] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt. *Maple V Language Reference Manual*. Springer-Verlag, New York, 1991.

[12] E. Bradley. Automatic construction of accurate models of physical systems. In *Proceedings of the 8th International Workshop on Qualitative Reasoning about Physical Systems*, Nara, Japan, 1994.

[13] E. Bradley and M. Easley. Reasoning about sensor data for automated system identification. In *Proceedings of the 2nd International Symposium on Intelligent Data Analysis (IDA-97)*, 1997. Also in press, *Intelligent Data Analysis*.

[14] E. Bradley, M. Easley, and A. Hogan. Qualitative observer theory. In process.

[15] L. F. Shampine and H. A. Watts. DEPAC—design of a user oriented package of ODE solvers. Technical Report SAND79-2374, Sandia National Laboratories, 1979.

[16] A. C. Capelo, L. Ironi, and S. Tentoni. The need for qualitative reasoning in automated modeling: A case study. In Y. I. and Adam Farquhar, editors, *Proceedings of the 10th International Workshop on Qualitative Reasoning*, pages 32–39, Stanford Sierra Camp, CA, 1996. Also published as AAAI Technical Report WS-96-01.

[17] P. Langley, H. A. Simon, G. L. Bradshaw, and J. M. Zytkow, editors. *Scientific Discovery: Computational Explorations of the Creative Process.* MIT Press, Cambridge, MA, 1987.

[18] D. L. Smith. *Introduction to Dynamic System Modeling for Design.* Prentice-Hall, 1994.

[19] H. D. I. Abarbanel. *Analysis of Observed Chaotic Data.* Springer, 1995.

[20] K. J. Astrom and P. Eykhoff. System identification—a survey. *Automatica*, 7:123–167, 1971.

[21] D. G. Bobrow, editor. *Qualitative Reasoning about Physical Systems.* MIT Press, Cambridge MA, 1985.

[22] K. D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[23] K. D. Forbus. The qualitative process engine. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning About Physical Systems.* Morgan Kaufmann, San Mateo CA, 1990.

[24] A. Farquhar and G. Brajnik. A semi-quantitative physics compiler. In *Proceedings of the 8th International Workshop on Qualitative Reasoning about Physical Systems*, Nara, Japan, 1994.

[25] B. C. Williams. A theory of interactions: Unifying qualitative and quantitative algebraic reasoning. *Artificial Intelligence*, 51, 1991.

[26] R. E. Kalman. A new approach to filtering and prediction problems. *J. Basic Eng.*, 82D:35–45, 1960.

[27] E. Hung. Parameter estimation in chaotic systems. Technical Report AIM 1541, MIT Artificial Intelligence Lab, May 1995.

[28] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[29] R. Fletcher. *Practical Methods of Optimization.* John Wiley and Sons, second edition, 1987.

[30] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, Inc., Boston, 1981.

[31] J. J. Moré, D. C. Sorensen, K. E. Hillstrom, and B. S. Garbow. *Sources and Development of Mathematical Software, W. J. Cowell, ed.*, chapter The MINPACK project. Prentice-Hall, 1984.

[32] J. E. Dennis, D. M. Gay, and R. E. Welsch. An adaptive nonlinear least-squares algorithm. *ACM Transactions on Mathematical Software*, 7(3):348–368, 1981.

[33] P. T. Boggs, A. J. Kearsley, and J. W. Tolle. A practical algorithm for general large scale nonlinear optimization problems. *SIAM Journal on Optimization*, in press.

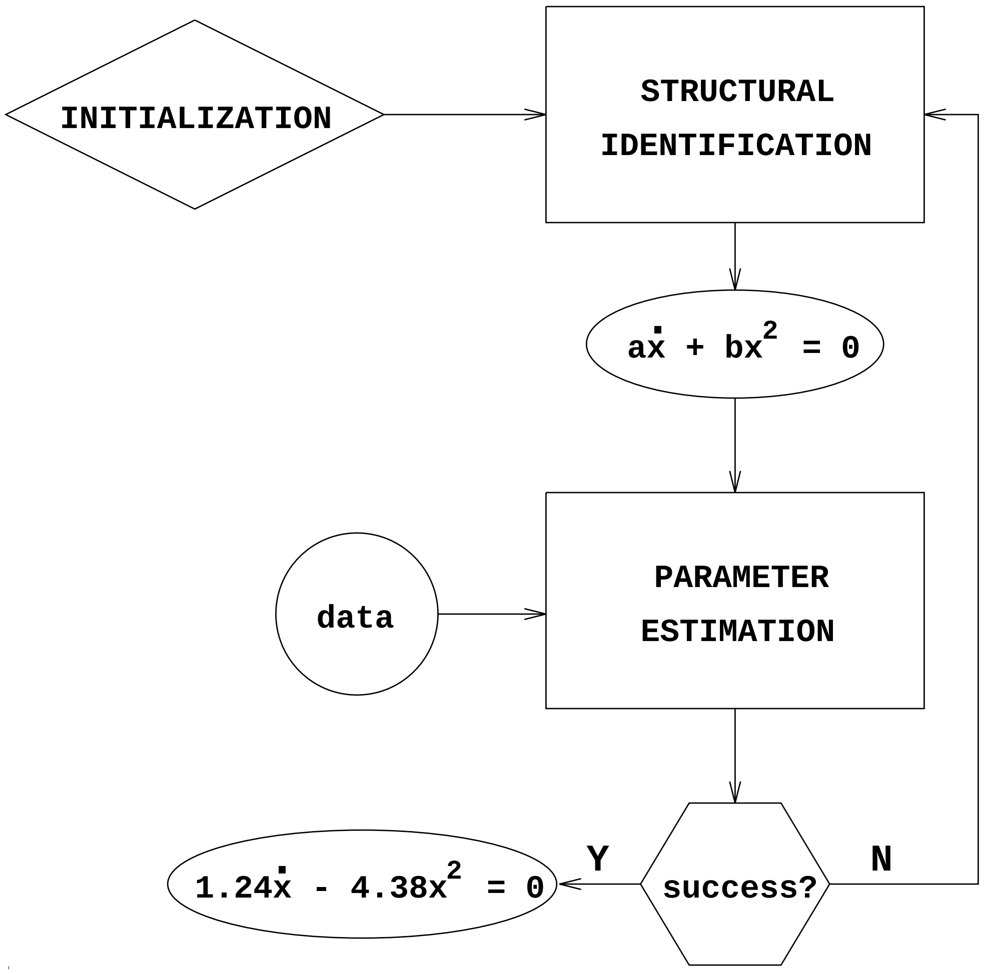

Figure 1: The System Identification Process. Structural identification yields the general form of the model; in parameter estimation, values for the unknown coefficients in that model are determined. The program pret automates this process; the topic of this paper is the parameter estimation box.

```
(find-model
  (domain mechanics)
  (state-variables <r> <theta>)
  (point-coordinates <r> <theta>)
  (coordinate-transformations ((<x> (* <r> (cos <theta>)))
          (<y> (* <r> (sin <theta>)))
          (<v> (expt (+ (expt <x> 2) (expt <y> 2)) .5))))
  (hypotheses
    (<force> (* a <y>))
    (<force> (* b (deriv (deriv <y>))))
     ...
    (<force> (* c <time>))
    (<force> (* d (deriv <theta>))
    (<force> (* e <v>))
    (<force> f))
  (observations
    (constant <theta>)
    (autonomous <x>)
     ...
    (damped <x>)
    (numeric (<time> <x> <y> <theta> <v>)
             (0.017 10.4 63.1 2.4 2) (0.033 10.4 63.3 ...) ...)
  (specifications
    (resolution <y> absolute 0.1)))
```

Figure 2: Instructing pret to find an ODE model of an R/C car. Angle brackets identify special keywords, such as state variables or domain properties; `a`...`f` are unknown constants. The `coordinate-transformation` information is necessary because the hypotheses and observations are expressed in different coordinate systems.

Second Order Models

`CircularPendulum`

$$\begin{aligned}
\dot{y}_1 &= y_2 \\
\dot{y}_2 &= -\beta_1(\beta_2 y_2 + \beta_1 \sin[y_1]) - \beta_3 \sin[y_1 - t/\beta_4] \\
\beta^* &= [0.9091,\ 0.1,\ 0.5,\ 5.4978]^T \\
Y_0{}^* &= [-0.09317,\ 0.2897]^T
\end{aligned}$$

`DrivenPendulum`

$$\begin{aligned}
\dot{y}_1 &= y_2 \\
\dot{y}_2 &= \beta_1 \beta_4 \cos[\beta_5 t] - \beta_2 y_2 - \beta_3 \sin[y_1] \\
\beta^* &= [1.0,\ 2.5,\ 98.0,\ 80.0,\ 4.90.0]^T \\
Y_0{}^* &= [0.1,\ 0.1]^T
\end{aligned}$$

`Glycolytic`

$$\begin{aligned}
\dot{y}_1 &= \beta_2 + 2\beta_4 \cos[\beta_5 t] - \beta_1 y_1 y_2^2 \\
\dot{y}_2 &= \beta_1 y_1 y_2^2 - \beta_3 y_2 \\
\beta^* &= [1.0,\ 0.999,\ 1.0,\ 0.21,\ 1.69]^T \\
Y_0{}^* &= [1.2,\ 0.8]^T
\end{aligned}$$

`Pendulum`

$$\begin{aligned}
\dot{y}_1 &= y_2 \\
\dot{y}_2 &= -\beta_1 \sin[y_1] - \beta_2 y_2 \\
\beta^* &= [1.0,\ 0.1]^T \\
Y_0{}^* &= [0.7,\ 0.0]^T
\end{aligned}$$

`PredatorPrey`

$$\begin{aligned}
\dot{y}_1 &= \beta_2 y_1\,(1 - y_1 - y_2/(y_1 + \beta_1)) - \beta_4 y_1(\sin[\beta_5 t] + 1) \\
\dot{y}_2 &= y_2\,(y_1(\beta_3 + 1)/(y_1 + \beta_1) - 1) \\
\beta^* &= [0.5,\ 5.0,\ 3.8,\ 2.08234,\ 1.5]^T \\
Y_0{}^* &= [0.2\ 1.1]^T
\end{aligned}$$

Third Order Models

`Chua`

$$\begin{aligned}
\dot{y}_1 &= \beta_1(y_2 - (\beta_5 y_1 + \tfrac{1}{2}(\beta_4 - \beta_5)(|1 + y_1| - |y_1 - 1|))) \\
\dot{y}_2 &= y_1 - y_2 + y_3 \\
\dot{y}_3 &= -(\beta_2 y_2 + \beta_3 y_3) \\
\beta^* &= [9.0,\ 14.286,\ 0.0,\ -0.142857,\ 0.285714]^T \\
Y_0{}^* &= [-0.676,\ 0.369,\ 0.792]^T
\end{aligned}$$

`Lorenz`

$$\begin{aligned}
\dot{y}_1 &= \beta_1(y_2 - y_1) \\
\dot{y}_2 &= \beta_2 y_1 - y_2 - y_1 y_3 \\
\dot{y}_3 &= y_1 y_2 - \beta_3 y_3 \\
\beta^* &= [16.0,\ 45.0,\ 4.0]^T \\
Y_0{}^* &= [-0.279,\ 1.353,\ 33.164]^T
\end{aligned}$$

`Pentagonal`

$$\begin{aligned}
\dot{y}_1 &= \sin[\beta_1 y_2] - y_3 \cos[\beta_2 y_1] \\
\dot{y}_2 &= y_3 \sin[\beta_3 y_1] - \cos[\beta_4 y_2] \\
\dot{y}_3 &= \beta_5 \sin[y_1] \\
\beta^* &= [2.4,\ 0.43,\ -0.65,\ -2.43,\ 0.15]^T \\
Y_0{}^* &= [0.0,\ 0.0,\ 0.0]^T
\end{aligned}$$

Fourth Order Models

`Springs&Masses`

$$\begin{aligned}
\dot{y}_1 &= y_2 \\
\dot{y}_2 &= -\beta_1 y_1 - \beta_2(y_1 - y_3) \\
\dot{y}_3 &= y_4 \\
\dot{y}_4 &= \beta_2(y_1 - y_3) - \beta_3 y_3 \\
\beta^* &= [0.1,\ 0.2,\ 0.3]^T \\
Y_0{}^* &= [0.1,\ 0.1,\ 0.1,\ 0.1]^T
\end{aligned}$$

`Spring&Pendulum`

$$\begin{aligned}
\dot{y}_1 &= y_2 \\
\dot{y}_2 &= y_4^2(\beta_1 + y_1) - \beta_2 y_1 + \beta_3 \cos[y_3] \\
\dot{y}_3 &= y_4 \\
\dot{y}_4 &= (\beta_3 \sin[y_3] + 2 y_2 y_4)/(-\beta_1 - y_1) \\
\beta^* &= [0.1,\ 2.0,\ 9.8]^T \\
Y_0{}^* &= [0.1,\ 0.1,\ 0.1,\ 0.1]^T
\end{aligned}$$

Figure 3: ODE Models for Monte Carlo Study

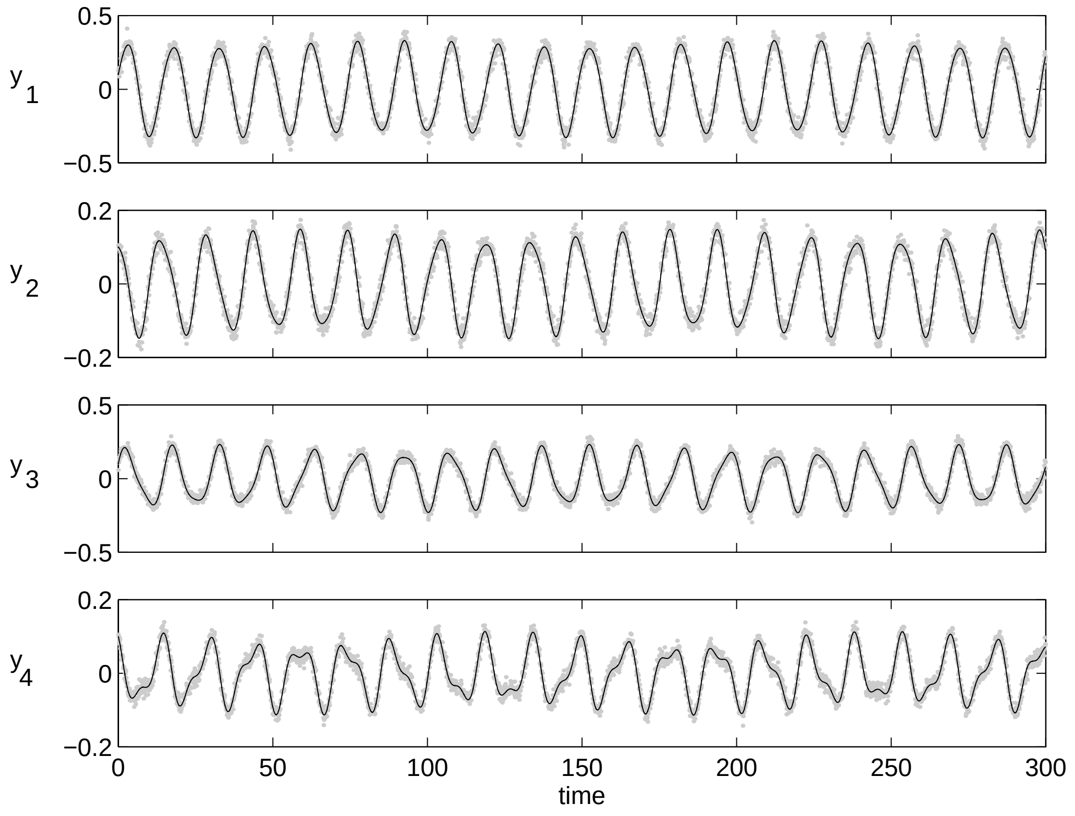

Figure 4: `Springs&Masses`

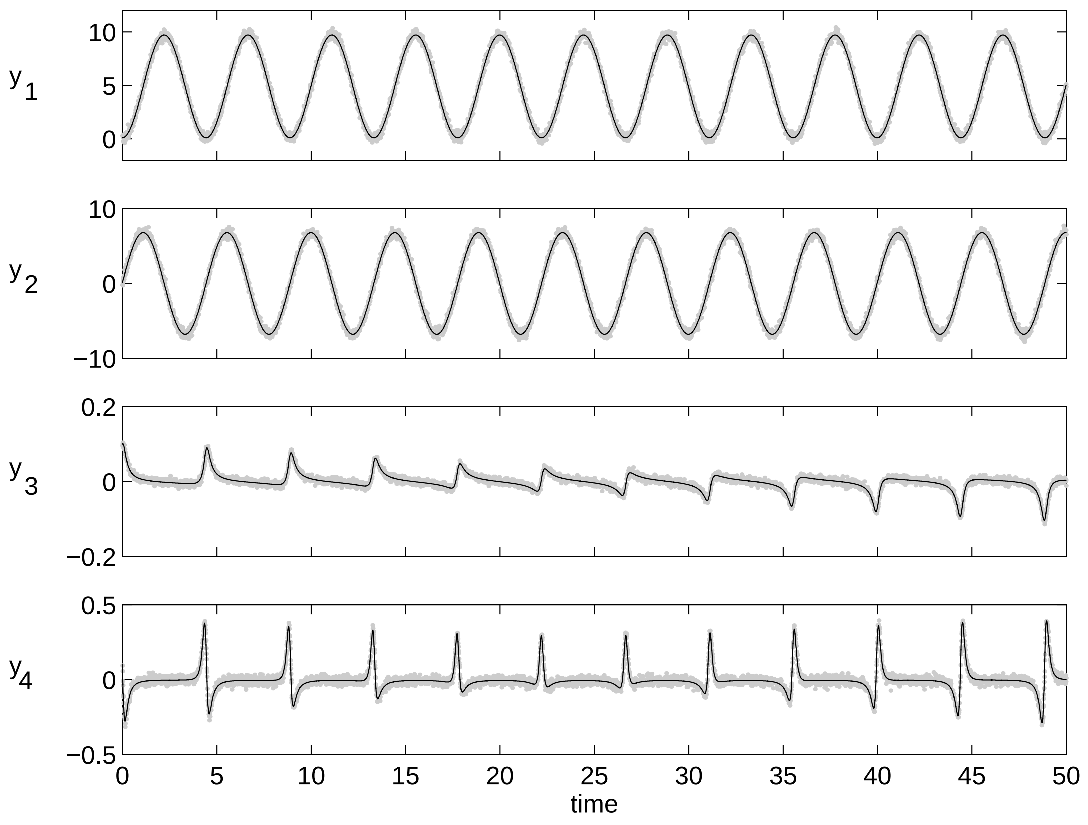

Figure 5: `Spring&Pendulum`

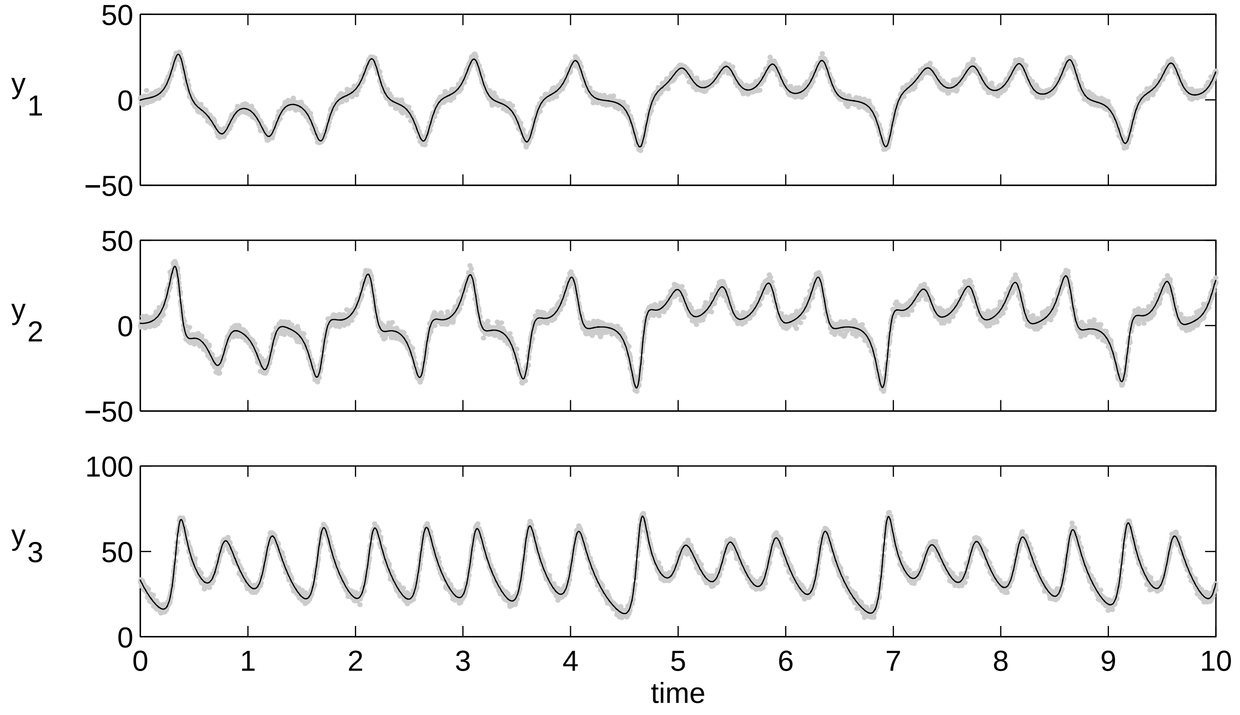

Figure 6: `Lorenz`

second-order models:	`Pendulum`	100%
	`DrivenPendulum`	100%
	`CircularPendulum`	97%
	`Glycolytic`	100%
	`PredatorPrey`	100%
third-order models:	`Lorenz`	100%
	`Chua`	99%
	`Pentagonal`	95%
fourth-order models:	`Springs&Masses`	100%
	`Spring&Pendulum`	99%

Figure 7: Monte Carlo Results. Table entries show what percentage of the time the NPER recognized the correct model.

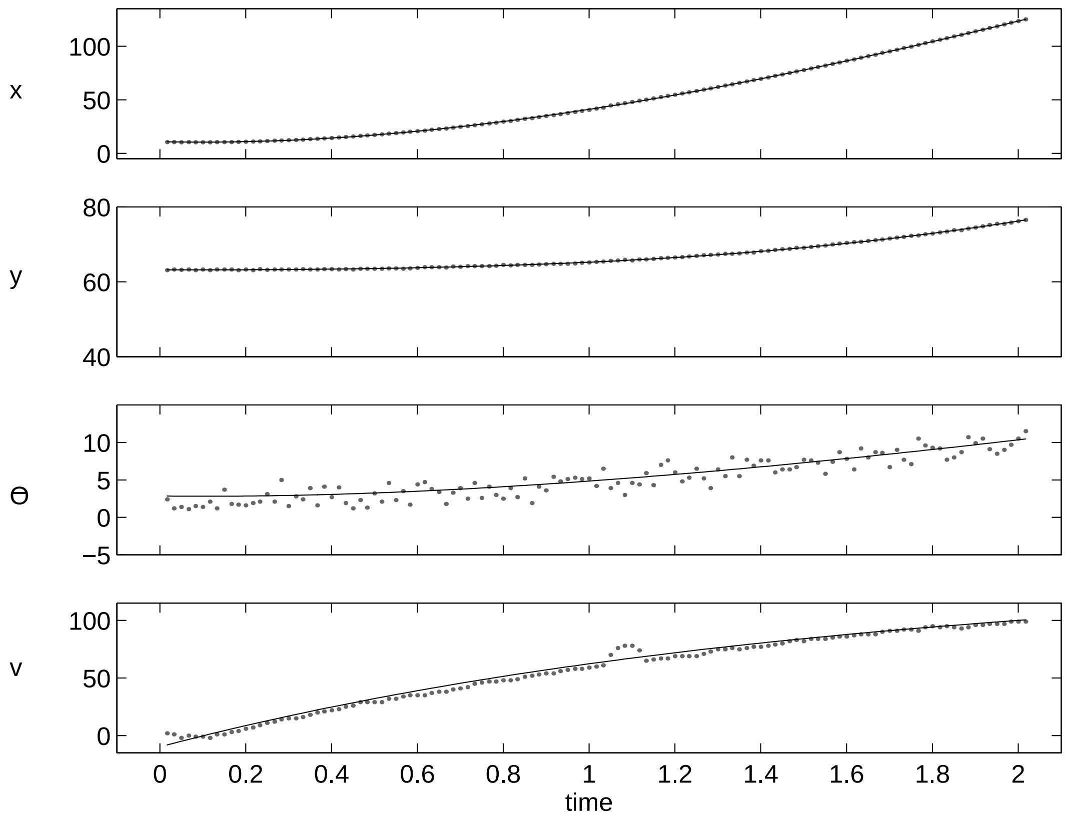

Figure 8: R/C car data and pret's fitted model. Observed sensor data are denoted by gray dots; estimated NPER solution is solid line.

```
(dataset "rc_cars_00")
(n                             121)
(degree                          4)
(initial-sum-of-squares  0.00000E+00)
(final-sum-of-squares    8.61396E+02)
(final-variance          7.55610E+00)
(var-flt test                  1.323)
(condition-number        2.06503E-03)
(info                            1)
(degrees-of-freedom            114)
(parameters
((  6.6115494E-02  2.0812335E-03  3.1767457E+01  1.9809924E+00)
(   9.1634748E+01  2.2764590E+00  4.0253194E+01  1.9809924E+00)
(  -6.4935280E-01  3.7373714E-02  1.7374586E+01  1.9809924E+00)
(   1.0819114E+01  2.3610613E-01  4.5823098E+01  1.9809924E+00)
(   6.3214041E+01  3.6177659E-02  1.7473226E+03  1.9809924E+00)
(   2.8384913E+00  1.0845794E-01  2.6171357E+01  1.9809924E+00)
(  -8.0776653E+00  1.0738973E+00  7.5218230E+00  1.9809924E+00)))
```

Figure 9: NPER output file. `var-flt test` denotes $\nu \equiv \hat{\sigma}^2/\tilde{\sigma}^2$, the ratio of the residual variance at the least squares solution to the residual variance estimated from the smoothed data.