Reasoning about sensor data for automated system identification

Elizabeth Bradley and Matthew Easley*

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

Intelligent Data Analysis: An International Journal, 2(2):123-138,
Elsevier Science (1998); www.elsevier.com/locate/ida

Abstract. The computer program PRET automatically constructs mathematical models of physical systems. A critical part of this task is automating the processing of sensor data. PRET's intelligent data analyzer uses geometric reasoning to infer qualitative information from quantitative data; if critical variables are either unknown or cannot be measured, it uses delay-coordinate embedding to reconstruct the internal dynamics from the external sensor measurements. Successful modeling results for two sensor-equipped systems, a driven pendulum and a radio-controlled car, demonstrate the effectiveness of these techniques.

1 Introduction

Constructing a model that a scientist or engineer can use to better understand or control a physical system typically requires analysis of the input/output behavior of that system. Different applications require different types of models; each type calls upon different measurement and reasoning techniques. In particular, formulating an internal ordinary differential equation (ODE) model from external observations of a system is known as *system identification* (SID). SID has two phases, as shown in Figure 1: *structural identification*, in which the general form of the equations that govern the unknown dynamics is determined, and *parameter estimation*, in which coefficient values that match that model to the actual sensor data are found.

* Supported by NSF NYI #CCR-9357740, ONR #N00014-96-1-0720, and a Packard Fellowship in Science and Engineering from the David and Lucile Packard Foundation.

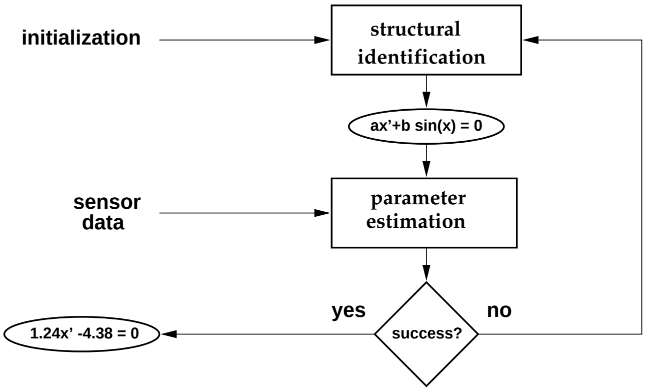


Fig. 1. System identification phases. Structural identification yields the general form of the model; in parameter estimation, values for the unknown coefficients in that model are determined. The PRET modeling tool automates this process using artificial intelligence techniques.

One of the aims of qualitative reasoning (QR)[10, 26], a branch of artificial intelligence (AI), is to automate the modeling process by abstracting knowledge, information, and reasoning to a qualitative level. The computer program PRET[4] is an example of a QR modeling tool. It automates the SID process that is diagrammed in Figure 1 by building an AI layer around a set of traditional SID techniques. This layer automates the high-level stages of the modeling process that are normally performed by a human expert. PRET combines several forms of QR via a special first-order logic inference system[23, 24] to intelligently assess the task at hand; it then reasons from that information to automatically choose, invoke, and interpret the results of appropriate lower-level techniques.

Differential equations are perhaps the most broadly applicable, well formalized, and powerful class of models in current use. Most physical systems that are of interest to scientists and engineers are *dynamic*: the values of their important properties — the rotation rate of a pulsar or the heat flow in a power plant — change with time. ODEs are perfectly suited to capture this kind of behavior. If the physics depends on *multiple* variables (i.e., *spatiotemporal* dynamics), partial differential equations (PDEs) — which are mathematically much more difficult and far less well-understood than ODEs — are the model of choice. Though PDEs are more general, PRET works with ODE models because they are both broadly applicable and also supported by a well-developed, highly formalized body of mathematical knowledge that applies *in any domain*, whether it be celestial mechanics, chemical plant design, or population biology. These reasoning rules allow additional knowledge to be inferred about an ODE and/or the system it models. For example, if a system is known to be autonomous, its model cannot explicitly contain the variable time; if a system oscillates, the imaginary parts of at least one pair of the model's roots must be nonzero. PRET exploits these types of rules in order to generate and test ODE models.

PRET (Figure 2) models linear and nonlinear systems by assembling combi-

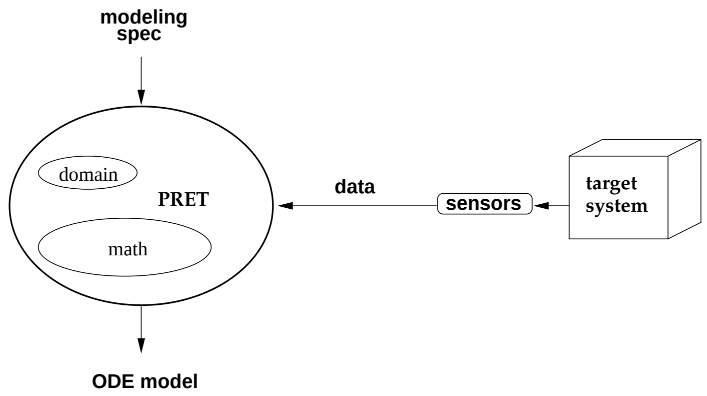


Fig. 2. The PRET modeling tool automates the system identification process, using domain theory to build ODE models from user specifications and ODE theory to check those models against observations like sensor data. The topic of this paper is PRET's intelligent sensor data analysis module.

nations of user-specified and automatically generated model fragments into an ODE system that both fits the domain physics and matches a given set of qualitative and quantitative observations. When human experts perform this task, they make use of a variety of well-established modeling techniques; their reasoning about a given physical system and possible candidate models takes place at an abstract level first and resorts to more detailed level later in the modeling process. PRET's goal is to mimic this strategy and attempt to find a model quickly — with no more detailed reasoning than necessary. Its basic strategy is a two-phase generate-and-test procedure. The generate phase — wherein models are constructed from hypotheses — necessarily depends on the domain involved; in mechanics, for instance, one might synthesize a model by using Newton's laws to combine force terms; in electronics, one might use Kirchhoff's laws to sum voltages in a loop or currents in a cutset. PRET's *domain theory reasoning* emulates this process; model generation in each domain is governed by one or two simple, powerful, domain-specific hypothesis combination rules, such as $\Sigma < \texttt{force} > = 0$ in the domain `mechanics`. The test phase, wherein a candidate model is checked against the known observations, is guided by the (much larger and domain independent) ODE theory described at the end of the previous paragraph. The intelligent data analysis strategies described in this paper, which automatically process the sensor information from the data acquisition link shown in Figure 2, are part of the test phase.

During the test phase, PRET checks the candidate model against a set of user-specified observations of the target system. PRET's special first-order logic-based inference engine[24] uses these observations — which take on various abstraction levels, ranging from low-level numeric data to high-level qualitative descriptions of the system's behavior — to rule out incorrect models as quickly and cheaply as possible. Properly processed (in tandem with the ODE theory), those observations let the engine eliminate large classes of models in a purely qualitative manner, guiding PRET efficiently through the exponentially complex search space of hypothesis combinations. Two special mechanisms allow this inference engine to achieve this form of high-level-first reasoning: abstraction levels and meta-level control. The former are static annotations to ODE rules that are used to direct the search for an inconsistency toward a quick, abstract proof. For example, qualitative reasoning rules are assigned a more-abstract level than rules that encode numerical reasoning. Meta-level control dynamically guides the search toward a cheap and quick proof of contradiction. Rules that are likely to lead to such a contradiction are chosen before other rules, and subgoals that will fail quickly are evaluated before other subgoals. Whereas abstraction levels are static, meta-level control guides the search relative to the current state of the inference engine. See [13, 24] for more details on the issues and implementation of the inference engine.

The abstraction level of an observation necessarily dictates both how it is processed and the breadth of its implications; *Qualitative* observations, such as "the system is linear," play a more wide-ranging role in the model test process than a highly situation-specific sensor data set (e.g., one that was gathered at a particular drive frequency). For instance, if the target system is known to be damped, PRET can use algebraic reasoning about the divergence of ODEs to quickly and cheaply discard *any* candidate model that is conservative. When observations involve numeric data, however, the conclusions drawn from the inference process are typically less general. In order to leverage these kinds of observations as much as possible, PRET incorporates a variety of intelligent data analysis techniques, which are the focus of this paper. PRET's sensor data analyzer can, for instance, use geometric reasoning to distill qualitative features from a data set and recognize that a sensor time series is converging to a fixed point; from this, the inference engine can conclude that the system is damped and thus that the divergence of any successful model must be negative. QR techniques like algebraic and geometric reasoning are far less computationally expensive than techniques that involve the processing of numbers; also, because of their inherent abstraction, they apply to wider problem classes. This application breadth and expense reduction are particularly critical here because of the complexity of PRET's search spaces.

PRET's capabilities have been demonstrated in a variety of simulated and real systems. For example, control of a commercially acquired radio-controlled (R/C) car used at the University of British Columbia could not be attained without an adequate model of the device. PRET has been successfully applied to this problem;

its inputs included a time series of the car's position and heading, the UBC analyst's set of known model fragments, and mathematical formalizations of a few of his qualitative observations (e.g. "it pulls to the left" or "we assume there is a simple form of frictional damping"). PRET examined different combinations of the model fragments during its structural identification stage, settled on one that matched the qualitative observations, and estimated parameters for this model using its nonlinear parameter estimation reasoner[3]. The results were interesting: PRET's model was correct — i.e., it met the specification — but it did not match the analyst's intuition. The problem was that he had omitted a few crucial observations (e.g., "it started from rest"), and PRET, of course, only models features that are made explicit in the specification. The discrepancy was useful in a very powerful way: it allowed a human analyst to identify what was missing from his description of the problem and to add the appropriate piece of knowledge to the specification. The R/C car modeling example is covered in much more depth in [3].

This paper describes a collection of automatic data analysis methods, such as the geometric and algebraic reasoning techniques mentioned two paragraphs above, that are specifically tailored to process sensor data and generate the kinds of information that PRET's inference engine can exploit to build effective models of systems like the R/C car. PRET's intelligent data analysis module, which instantiates these methods, combines ideas from a variety of very different fields — AI, control theory, nonlinear dynamics, and numerical analysis. An important and difficult part of its task is to effectively automate decisions about when to use which techniques and in what sequence, how to set up the invocations of the appropriate code modules, and how to interpret the results. The core of this intelligent data analyzer is based on Hsu's cell-to-cell mapping paradigm for dynamics analysis[14]. This method requires a full state-space trajectory, however, and fully *observable* systems[2] are rare in engineering practice. Sensor data are almost always incomplete and/or noisy; worse yet, the true state variables may

[2] those whose state variables are all known and measurable

not even be known to the user. The partial solutions that we propose to this *observer problem* are based on delay-coordinate embedding[1]. Acting together, geometric reasoning and delay-coordinate embedding allow this intelligent data analyzer to infer, from a quantitative data set, exactly the kinds of qualitative information that PRET can exploit to quickly verify or discard models.

The next two sections describe this intelligent data analysis module, first covering the geometric reasoning that is used to distill qualitative information out of a quantitative data set, and then describing how delay-coordinate embedding techniques can be used to infer knowledge about unobserved state variables. Following these background sections, we present an example PRET run on a parametrically driven pendulum, highlighting the role that the intelligent data analysis techniques play in the successful modeling of the system. We then discuss the results and their implications, describe PRET's relationship to other work, and summarize.

2 Distilling Qualitative Information from Quantitative Data

The intelligent data analyzer's geometric reasoner distills qualitative properties from a numeric data set using phase-portrait analysis, asymptote recognition, and other simple computer-vision techniques. Dynamical systems practitioners typically reason about phase portraits, rather than time series, because the phase-space representation — which suppresses time — brings out the qualitative properties of the system under examination. For example, recognizing a damped oscillation in a time series from a linear system requires detailed examination of the amplitude decay rate of and the phase shift between two decaying sinusoidal time-domain signals. The same behavior manifests in a much more obvious form — a single symmetric spiral — on a phase portrait. The types of automated phase-portrait analysis[2, 27, 28] techniques used here are designed to capture this kind of information and generate the corresponding qualitative descriptions (e.g., `damped-oscillation`). This kind of information is perfectly suited for use by PRET's inference engine; its qualitative nature allows the engine to verify or discard large classes of candidate models. For example, if sensor measurements of a state variable indicate that it is undergoing a damped oscillation, the inference engine can immediately rule out all ODE models that are unstable, critically damped, or overdamped[3].

PRET's geometric reasoner is based on the cell-to-cell mapping formalism of Hsu[14, 15], which discretizes a set of n-dimensional state vectors onto an n-dimensional mesh of uniform boxes or *cells*. In Figure 3, for example, the circular trajectory — a sequence of two-vectors of floating-point numbers measured by a finite-precision sensor — can be represented as the *cell sequence*

$$[...(0,0)(1,0)(2,0)(3,0)(4,0)(4,1)(4,2)(4,3)(4,4)(3,4)...]$$

Because multiple trajectory points are mapped into each cell, this discretized

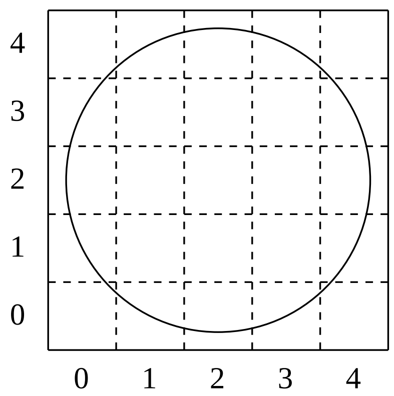


Fig. 3. Identifying a limit cycle using simple cell mapping

representation of the dynamics is significantly more compact than the original series of floating-point numbers and therefore much easier to work with. This

[3] Stability analysis of a linear ODE involves finding the roots of its *characteristic polynomial* (e.g., $as^2 + bs + c = 0$ for the ODE $a\ddot{x} + b\dot{x} + c = 0$). Only if those roots are imaginary can the system oscillate; only if their real parts are negative is the oscillation damped.

is particularly important when complex systems are involved, as the number of cells in the grid grows exponentially with the number of dimensions[4]. Though the approximate nature of this representation does abstract away much detailed information about the dynamics, it preserves many of its important invariant properties; see, e.g., Hao[12] or Lind & Marcus[20] for more details. This point is crucial to the phase-portrait analysis methods used here; it means that conclusions drawn from the discretized trajectory are also true of the real trajectory — that is, a repeating sequence of cells in the former, as in Figure 3, implies that the true dynamics are on a limit cycle. In this manner, low-level, finite-precision numerical data can be converted into a high-level qualitative classification (here, a `limit-cycle` fact) that triggers the use of associated ODE rules in PRET's inference engine. Its coarse-grained nature gives this scheme some important limitations, both subtle and obvious, many of which are described below. Interestingly enough, some of these apparent limitations can actually be turned to PRET's advantage.

Given the cell-dynamics formalism described in the previous paragraph, the dynamics of a discretized trajectory can be quickly and qualitatively classified using simple geometric heuristics. Some of these classification heuristics are trivial (e.g. determining if the trajectory exits the mesh), but detecting limit cycles or oscillations requires subtler pattern recognition techniques. Below are several of the geometric reasoner's dynamics classifications, the corresponding heuristics, and some associated implications for the ODE model:

- `fixed-cell`: when a trajectory relaxes to a single cell and remains within that cell for a fixed percentage of its total lifetime. This can, for instance, be used to recognize when a second-order linear system is overdamped. Appropriate mesh geometry choices can extend this method to asymptote recognition.
- `limit-cycle`: when the trajectory contains a finite, repeating sequence of

[4] The example of Figure 3 is two-dimensional, but the cell dynamics formalism generalizes smoothly to arbitrary dimension.

cells. These patterns are identified by discarding any transients and searching for periodic mapping sequences; they indicate that the system is either conservative or externally driven (nonautonomous)[5].

- `damped-oscillation`: when a trajectory enters a fixed cell via a decaying oscillation. This pattern is detected by recognition of an inward spiral; such dynamics can indicate, for instance, that a system is underdamped and thus that at least one pair of the model's natural frequencies must be complex.
- `constant`: when a state variable does not change over the duration of the trajectory. This computation involves a simple serial scan on each mesh axis; its results are particularly useful to PRET because they have wide-ranging implications about the order of the system.
- `sink-cell`: when a trajectory exits the mesh. This information is used to identify unstable trajectories.

Many other classifications are possible (e.g., `chaotic`); some are less useful to PRET than others — because their implications either are more limited in range or require processing at a less-abstract reasoning level.

The cell size, mesh boundary, and trajectory length affect the validity and efficiency of the cell-dynamics classification. Among other things, a small limit cycle may be classified as a fixed point, and behavior outside the mesh will not be classified at all. All of these discretization and boundary effects are not, in fact, problems; rather, they actually allow PRET to represent and work with the abstraction levels implied by the finite range and resolution that are such fundamental features of a modeling hierarchy — e.g., to avoid including saturation and crossover distortion effects when asked to "model the *small-signal* behavior of the op amp *to within 10mV*." Specifically, we use the range and resolution information from user instructions to set up the mesh boundary and cell size, assuring that behavior outside the range or below the resolution is not modeled.

[5] Some authors define the terms "limit cycle" and "periodic orbit" differently; we consider them to be equivalent.

3 Delay-Coordinate Methods for Observer Theory

If all of a system's state variables are identified and measured, the geometric reasoning techniques described in the previous section can be applied directly to the sensor data. A fully *observable* system like this, however, is 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. Worse yet, the true state variables may not be known to the user; temperature, for instance, can play an important and often unanticipated role in the modeling of an electronic circuit. This is part of control theory's *observer problem*: how to (1) identify the internal state variables of a system and (2) infer their values from the signals that *can* be observed. The arsenal of time-series analysis methods developed by the nonlinear dynamics community in the past decade[1] provides powerful solutions to both parts of this problem. This section describes the two methods, Pineda-Sommerer (P-S)[22] and false near neighbor (FNN)[18], that PRET uses to infer the dimension of the internal system dynamics from a time series measured by a single output sensor[6].

Both P-S and FNN are based on *delay-coordinate embedding*, wherein one constructs m-dimensional *reconstruction-space* vectors from m time-delayed samples of the sensor data. For example, if the time series in Figure 4 is embedded in three dimensions ($m = 3$) with a delay of 0.2, the first two points in the reconstruction-space trajectory are (32.0 22.0 19.0) and (28.0 16.0 23.0). Sampling a single system state variable is equivalent to projecting a d-dimensional state-space dynamics down onto one axis; embedding is akin to "unfolding" such a projection, albeit on different axes. The central theorem relating such embeddings to the underlying dynamics was suggested in [25] and proved in [21]; informally, it states that given enough dimensions (m) and the right delay (τ), the reconstruction-space dynamics and the true (unobserved) state-space dynam-

[6] Techniques like divided differences can, in theory, be used to derive velocities from position data; in practice, however, these methods often fail because the associated arithmetic magnifies sensor error.

t	x
0.1	32.0
0.2	28.0
0.3	22.0
0.4	16.0
0.5	19.0
0.6	23.0

Fig. 4. An example delay-coordinate embedding with an embedding dimension of three and a delay of 0.2.

ics are topologically identical[7]. This is an extremely powerful theorem: it lets us analyze the underlying dynamics using only the output of a single sensor. In particular, many properties of the dynamics, such as dimension (i.e., fixed point, limit cycle, chaotic attractor, etc.), are preserved by diffeomorphisms; if they are present in the embedding, they exist in the underlying dynamics as well. There are, of course, some important caveats, and the difficulties that they pose are the source of most of the effort and subtlety in these types of methods. Specifically, in order to embed a data set, one needs m and τ, and neither of these parameters can be measured or derived from the data set, either directly or indirectly, so algorithms rely on numeric and geometric heuristics to estimate them.

The Pineda-Sommerer algorithm creates such estimates; it takes a time series and returns the delay τ and a variety of different estimates of the dimension m. The procedure has three major steps: it estimates τ using the mutual information function, uses that estimated value τ_0 to compute a temporary embedding dimension E, and uses E and τ_0 to compute the *generalized dimensions* D_q, also known as "fractal dimensions." The standard algorithm for computing the fractal dimension of a trajectory, loosely described, is to discretize state space into ϵ-boxes, count the number of boxes occupied by the trajectory, and let $\epsilon \to 0$.

[7] More formally, the reconstruction-space and state-space trajectories are diffeomorphic iff $m \geq 2d + 1$, where d is the true dimension of the system.

Generalized dimensions are defined as

$$D_q = \frac{1}{q-1} \limsup_{\epsilon \to 0} \frac{\log \sum_i p_i^q}{\log \epsilon} \tag{1}$$

where p_i is some measure of the trajectory on box i. D_0, D_1, and D_2 are known, respectively, as the capacity, information, and correlation dimensions; all three are useful to PRET as estimates of the number of state variables in the system. The actual details of the P-S algorithm are quite involved; we will only present a qualitative description:

- Construct 1- and 2-embeddings of the data for a range of τs and compute the saturation dimension D_* of each; the first minimum in this function is τ_0. The D_* computation entails:
  - Computing the information dimension D_1 for a range of embedding dimensions E and identifying the saturation point of this curve, which occurs at D_*. The D_1 computation entails:
    - Embedding the data in E-dimensional space, dividing that space into E-cubes that are ϵ on a side, and computing D_1 using equation (1) with $q = 1$.

Ideally, of course, one lets $\epsilon \to 0$ in the third step, but floating-point arithmetic and computational complexity place obvious limits on this; instead, one repeats the calculation for a range of ϵs and finds the power-law asymptote in the middle of the log-log plot of dimension versus ϵ. P-S incorporates an ingenious complexity-reduction technique: the ϵs are chosen to be of the form 2^{-k} for integers k and the data are integerized; this allows most of the mathematical operations to proceed at the bit level and vastly accelerates the algorithm. To increase the precision of this computation, we have also implemented an arbitrary-length virtual integer package that facilitates the integerization.

The false near neighbor algorithm is far simpler than P-S. It takes a τ and a time series[8] and returns m. FNN is based on the observation that neighboring

[8] We run P-S first and then use its τ in FNN. Other methods, such as autocorrelation[1] can also be used to estimate τ.

points may in reality be projections of points that are very far apart, as shown in Figure 5. The algorithm starts with $m = 1$, finds each point's nearest neigh-

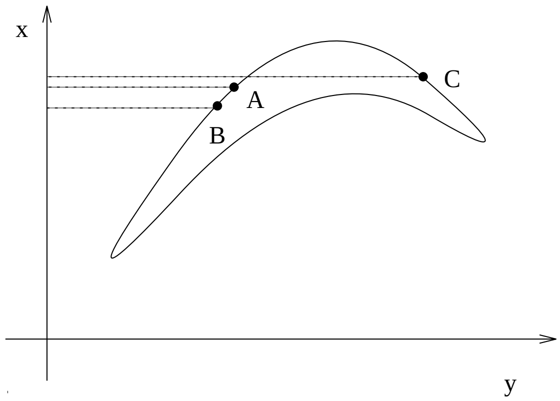


Fig. 5. The geometric basis of the FNN algorithm: the points labeled A and B are true near neighbors in the x-projection, while A and C are false near neighbors.

bor, and then re-embeds the data with $m = 2$. If the point separations change abruptly between the 1- and 2-embeddings, then the points were *false* neighbors (like A and C in the x-projection of Figure 5). The FNN algorithm continues adding dimensions until an acceptably small[9] number of false near neighbors remain, and returns the last m-value as the estimated dimension. We use a K-D tree implementation[11] to reduce the complexity of the nearest-neighbor step from $O(N^2)$ to $O(N \log N)$, where N is the length of the time series.

As both FNN and P-S are based on heuristics, their estimates of the embedding dimension m are not necessarily the same. Since both algorithms provide conservative estimates, PRET chooses the minimum of the two results as an upper bound for the dimension of the model.

[9] An algorithm that removes *all* false near neighbors can be unduly sensitive to noise.

4 Status and Discussion

In order to demonstrate the functions of PRET's intelligent data analysis module, this section presents a real-world modeling example: a PRET run on a parametrically driven pendulum — a solid aluminum arm that rotates freely on a standard bearing. The pendulum vertex is driven up and down in a sinusoidal pattern by a motor and a simple linkage. An actuator controls the pendulum's drive frequency and a sensor (an optical encoder) measures its angular position. The behavior of this apparently simple device is really quite complex and interesting: for low drive frequencies, it has a single stable fixed point, but as the drive frequency is raised, the attractor undergoes a series of bifurcations. In the sensor data, this manifests as interleaved chaotic and periodic regimes[6]. This system is also interesting from a modeling standpoint; at high resolutions, the backlash in the bearings invalidates the standard textbook model. Modeling these effects is critical, for instance, to the accurate deployment of the space shuttle's manipulator arm[16].

Figure 6 shows how a user instructs PRET to build an ODE model of this system. The full details and implications of both input and output syntax (and the GUI that facilitates entry of the `find-model` call) are covered elsewhere[4, 13]; here, we will concentrate on the parts of the call that pertain to the intelligent data analyzer. The first line of the `find-model` call specifies the domain of the problem and causes the program to instantiate the associated domain theory – here, a single rule specifying that forces at a point sum to zero. Recall that PRET uses domain theory only to combine hypotheses into models, so this theory is far more compact than the ODE theory used to test candidate models against observations. The point of this is to make it easy for a user to extend PRET to new domains. The next two lines of Figure 6 identify `<theta>` as a state variable that is a coordinate associated with a point. The remainder of the `find-model` call consists of three types of information about the target system: `specifications`, `hypotheses`, and `observations`. `specifications` prescribe resolutions and ranges to and over which the model is required to be valid.

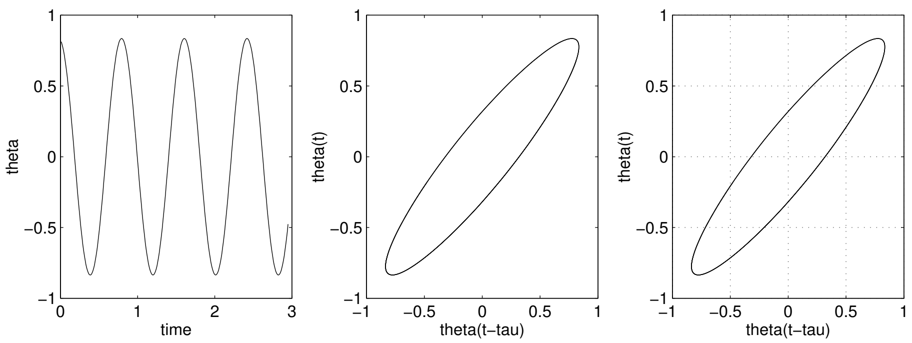
```
(find-model
  (domain mechanics)
  (state-variables <theta>)
  (point-coordinates <theta>)
  (hypotheses  (<force> (* A1 (sin <theta>)))
               (<force> (* A2 (deriv <theta>)))
               (<force> (* A3 (cos (* A4 <time>))))
               (<force> (* A5 (deriv (deriv <theta>))))  )
  (observations (nonautonomous)
                (numeric (<time> <theta>) ((0.00 0.812) ... )) )
  (specifications (<theta> absolute-resolution 0.5)
                  (<theta> range -1.570796327 1.570796327) ))
```

Fig. 6. PRET's input: finding an ODE model of the driven pendulum. `<theta>` is the bob angle.

`Hypotheses` are ODE fragments from which PRET constructs the model. If the user's hypotheses are inadequate, PRET uses power-series techniques, the standard technique to which an engineer would resort in this situation, to generate ODE fragments from scratch. The search space is exponentially complex in the number of hypotheses, however, so blind — and rapid — power-series enumeration is far less desirable than a hypothesis list intelligently selected by a user.

`Observations` range from the purely qualitative to the purely quantitative. The former are a powerful target for QR techniques: using a qualitative observation, PRET's inference engine can quickly discard broad classes of candidate models. Moreover, to a human expert, making qualitative observations is a natural part of the modeling process, so this provides a smooth user interface. *Quantitative* observations, the specific focus of the methods described in this paper, are typically sensor measurements; the source of the `numeric` observation in this `find-model` call, for instance, is the optical encoder on the pendulum shaft. Direct inferences from such observations are computationally expensive and of limited utility, but qualitative features extracted from them, as described below, can be leveraged in an abstract and powerful manner by the inference engine.

As a first step in the modeling process, PRET's intelligent data analysis reasoner distills a variety of qualitative information from the `numeric` observation in the `find-model` call, which is shown in part (a) of Figure 7. It first recon-



The pendulum data set (a) time series (b) reconstruction (c) grid discretization.

Fig. 7. The `numeric` observation in the `find-model` call of the previous figure. The source of these measurements is an angle sensor on the pendulum shaft.

structs the dynamics, beginning by invoking P-S and FNN on this data set. The former returns $\tau = 15$ and $m = 5$; the latter also returns $m = 5$, corroborating the estimates, so PRET embeds the data with those parameters, yielding the reconstruction-space trajectory shown in part (b) of the figure. Normally, the reconstructed state space is discretized based upon user resolution and range specifications; in Figure 7(c), for example, the cell size (0.5) is taken from the first line of the `specifications` and the mesh boundary ($\pm\pi/2$) from the second line. In some cases, however, the classification heuristics may alter the mesh parameters dynamically after the first pass (e.g., if a high-level ODE rule makes it clear that the data are only valid in some subset of the region) in order to refine its results. The `limit-cycle` heuristic in Section 2 recognizes the repeating sequences of cells in the discretized trajectory in part (c) of the figure and classifies it as a limit cycle, so a `limit-cycle` fact is added to the list[10] of `observations` that a successful model must match:

- `(nonautonomous)`
- `(numeric (<time> <theta>) ((0.00 0.812) ... )) )`
- `(limit-cycle)`

Note that automatic data analysis results may conflict with the user's observations. In this case, PRET assumes a higher confidence level in the former.

Guided by this augmented observation set, PRET searches the space of `hypothesis` combinations. Inferences drawn from the two qualitative observations, `nonautonomous` and `limit-cycle` — one specified by the user and one inferred automatically by the intelligent data analyzer — play a critical role in search-space reduction. The `nonautonomous` fact lets PRET's inference engine rule out all models that have no `<time>`-dependent terms. The knowledge that the system exhibits a limit cycle is equally powerful, since `limit-cycle` implies that any linear model must be of at least second order. The core of the internal representation of the main rule involved in this reasoning process is shown below:

```
(<- (constraint >= (var N) 2)
    ((linear-model)
     (limit-cycle (var State-Var-1) (var State-Var-2))
     (order (var State-Var-1) (var N))))
```

The rules are in the form `(<-` *head body*`)`[11], where the conjunction of the formulae in *body* implies *head*. `State-Var-1` and `State-Var-2` are logical variables that are instantiated with the state variables (logical constants) $\theta(t)$ and $\theta(t-\tau)$ of Figure 7 (b) and (c). This inference progression eventually guides PRET to the model $A_5\ddot{\theta} - A_3\cos[A_4 t] - A_1\sin\theta - A_2\dot{\theta} = 0$. To determine values for the unknown coefficients A_i, PRET invokes its QR-based *nonlinear parameter*

[10] This list is initially composed of the `observations` in the `find-model` call.

[11] Variables in the formulae have the form `(var` *symbol*`)`; they are (implicitly) universally quantified.

estimation reasoner or NPER[3], which returns $A_i = (2.5, 98.0, 80.0, 9.0)$. With these coefficient values, this model meets the requirements in the `find-model` call, and so it is returned as PRET's output.

This example has been simplified for presentation purposes; for instance, we loosened the resolution so as to avoid modeling the backlash in the bearings. Normally, too, the `find-model` call would contain many more hypotheses and qualitative observations. Most of the models constructed from the former are quickly discarded using fast, inexpensive qualitative reasoning on the latter.

Note that the angular velocity state variable $\dot{\theta} = \omega$ was neither identified nor measured in this example. In the `mechanics` domain, PRET automatically generates a new state variable from the (symbolic) first derivative of each known state variable[12]; here, this step led directly to ω, effectively solving the identification part of the observer problem. PRET's NPER solves the other part of this problem: using a combination of qualitative reasoning and numerical analysis, it automatically synthesizes starting values and initial search directions for a non-linear least-squares solver routine that performs the coefficient/data regression. If PRET cannot build a successful model with θ and ω, it may be possible to synthesize state variables from scratch using geometric reasoning on the data, as embedded using the P-S and FNN results. This is a current focus of our research effort.

The work described here is only the first half of the full input-output analysis that expert engineers apply to modeling problems; the next step in this research is to incorporate actuators as well as sensors, as shown in Figure 8. The overall goal of PRET's automatic sensor/actuator interpretation and control module, of which the intelligent data analyzer described here is a component, is to autonomously generate and refine observations and hypotheses, possibly even constructing and using observations that transcend the user's knowledge of physics. There are a variety of potential problems with this; for instance, the type and number of automatically generated hypotheses will have to be

[12] more generally, the conjugate momentum for each generalized coordinate

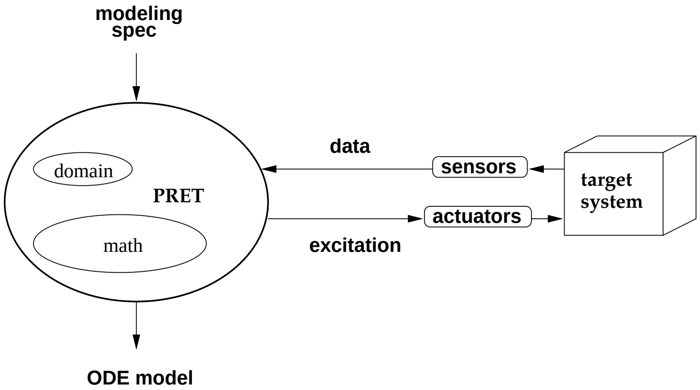


Fig. 8. The next step in PRET's development. Automatic actuator control will be added to the sensor data analysis techniques described in this paper, allowing PRET to autonomously generate, refine, and use new observations and hypotheses.

limited, lest the search space become unmanageable and/or the models overfit the data. There are two fundamental limitations on automated actuator manipulation: *controllability* and *reachability.* The combined actuator and system properties implicitly govern what kinds of experiments can be performed, and PRET's intelligent actuator control module will have to reason about these issues explicitly and efficiently. Automating the experimental observation process gives rise to some interesting AI issues involving representation and reasoning: how to identify and express the appropriate properties, and how to reason from that information in order not only to decide what experiments are possible, but also to determine which of those experiments yield the most powerful and useful inferences.

5 Relationship to Related Work

Any type of reasoning about a dynamic system requires a model. The precise form of that model is governed by the form of the reasoning: quantitative analysis requires exact equations, while a qualitative understanding can rest purely on abstract notions like "y is a monotonic function of x." Automated modeling tools reflect this difference in approaches; the models that they produce span the spectrum from precise mathematical descriptions of a system to highly abstract representations of its physics. Most of the work in the very active AI/QR modeling community, including the PRET project, focuses on qualitative models. These tools typically generalize a set of descriptions of the state into a higher-level abstraction — *qualitative states*[5, 9]. Many QR modeling tools reason about equations at an abstract level by combining model fragments. The abstraction levels and representations of these fragments vary; QSIM[19] models, for instance, are *qualitative* differential equations (QDEs) like `y = M+(x)` (which expresses that `y` is a monotonic function of `x`), whereas PRET's models are the mathematician's standard ODEs. Like many of the more physics-based modeling systems (e.g., QPT/QPE[8]), PRET's inputs are expressed in the high-level descriptive concepts and terms (e.g. `linear, autonomous`, etc.) that are typically used by scientists and engineers, and its internal reasoning is guided and governed by the standard ideas of physics. PRET also resembles these physics-based modeling tools in how it orchestrates the data flow between the modeler and the target system. Physical measurements, for example, often must be interpreted with respect to a particular situation. QPT solves this problem by concentrating on partitioning time-series data into logical segments[9]. While these types of observations are important to PRET, our research concentrates more on the control-theoretic issues of this problem — attempting to observe the important states of a system directly, identifying ways to observe them indirectly, or changing sensor configurations. PRET differs from systems like QPT in several important ways; among other things, it does not focus on explanation and causality (e.g. "What happens when a block is dropped from a table"). Perhaps the most significant difference between the work described here and the rest of the QR modeling literature, however, is that PRET takes a practical engineering approach, working with noisy, incomplete sensor data from real-world systems and attempting not to "discover" the underlying physics, but rather to find the simplest ODE that can account for the given observations.

PRET also shares goals, ideas, and tactics with several other fields. In particular, it solves the same problems as traditional system identification[17] — albeit in an automated fashion — and therefore it rests upon many of the standard methods found in basic control theory texts. Finally, it incorporates many of the same ideas that appear in the data analysis literature[7], but it adds a layer of AI techniques, such as symbolic data representation and logical inference, that let it automate many of the higher-level reasoning tasks normally performed by human experts[13].

6 Conclusion

Geometric reasoning and delay-coordinate embedding allow the automatic data analyzer described in this paper to infer, from a quantitative data set, exactly the kinds of qualitative information that effectively guide the PRET automated system identification tool through the complex search space of ordinary differential equation models. These methods provide a partial solution to the observer problem, allowing PRET to infer some internal system state variable properties from incomplete output sensor data. Importantly, this intelligent data analyzer automates many of the higher-level tasks normally performed by human experts — the choice of what lower-level technique to use in a given phase of the SID process, how to invoke that technique, how to interpret its results, and how to leverage that information later in the process.

The ultimate goal of the PRET project is a tool that can construct internal ODE models of high-dimensional black-box systems in a variety of domains — with minimal human guidance or forethought. Because of this, PRET is designed

[13] PRET's task is far too complex for many traditional AI approaches; neither a state-driven rule application strategy nor a left-right, depth-first theorem prover strategy were powerful enough for the reasoning demanded by system identification. In order to address these complex reasoning issues, our framework provides meta-level control constructs that allow control information to be expressed declaratively and separately from the ODE theory. See [13] for more details.

for easy extension to new domains, and a current focus of our research is to test this extensibility by solving modeling problems in visco-elastic materials and electronic circuits. The initial results have been good, but some subtleties remain: for instance, topology plays a critical role in descriptions of networks of discrete components, and PRET's syntax and reasoning do not yet handle these requirements smoothly. Moreover, human experts in both of these domains — and others — rely heavily on input/output studies such as impulse or frequency response. Intelligent automation of PRET's sensor interaction, the topic of this paper, is only one part of this process; the details of the dynamics cannot be truly exposed without an interactive input-output analysis, which will require intelligent actuator manipulation as well.

Acknowledgments: Joe Iwanski, with some help from Josh Stuart (the virtual integer implementation), coded the algorithms described in Section 3. Reinhard Stolle contributed the description of the internal workings of PRET's inference engine. Apollo Hogan, and Brian LaMacchia also contributed code and/or ideas to this project, and the IDA reviewers' comments helped focus the content and presentation of this document.

References

1. H. Abarbanel. *Analysis of Observed Chaotic Data*. Springer, 1995.
2. E. Bradley. Autonomous exploration and control of chaotic systems. *Cybernetics and Systems*, 26:299–319, 1995.
3. E. Bradley, A. O'Gallagher, and J. Rogers. Global solutions for nonlinear systems using qualitative reasoning. *Annals of Mathematics and Artificial Intelligence*. In press.
4. E. Bradley and R. Stolle. Automatic construction of accurate models of physical systems. *Annals of Mathematics and Artificial Intelligence*, 17:1–28, 1996.
5. J. de Kleer and J. S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.
6. D. D'Humieres, M. R. Beasley, B. Huberman, and A. Libchaber. Chaotic states and routes to chaos in the forced pendulum. *Physical Review A*, 26:3483–3496, 1982.
7. A. Famili, W.-M. Shen, R. Weber, and E. Simoudis. Data preprocessing and intelligent data analysis. *Intelligent Data Analysis*, 1(1), 1997.
8. K. D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.
9. K. D. Forbus. Interpreting observations of physical systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3):350–359, 1987.
10. K. D. Forbus. Qualitative reasoning. In J. A. Tucker, editor, *The Computer Science and Engineering Handbook*. CRC Press, Boca Raton, FL, 1997.

11. J. Friedman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3:209–226, 1977.
12. B.-L. Hao. Symbolic dynamics and characterization of complexity. *Physica D*, 51:161–176, 1991.
13. A. Hogan, R. Stolle, and E. Bradley. Putting declarative control to work. In *AAAI-98*. In review.
14. C. S. Hsu. A theory of cell-to-cell mapping dynamical systems. *Journal of Applied Mechanics*, 47:931–939, 1980.
15. C. S. Hsu. *Cell-to-Cell Mapping*. Springer-Verlag, New York, 1987.
16. J. Iwanski and E. Bradley. Modeling nonlinear/chaotic phenomena: Comparing models. In preparation.
17. J.-N. Juang. *Applied system identification*. Prentice Hall, Englewood Cliffs, N.J., 1994.
18. M. Kennel, R. Brown, and H. Abarbanel. Determining minimum embedding dimension using a geometrical construction. *Physical Review A*, 45:3403–3411, 1992.
19. B. J. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29(3):289–338, 1986.
20. D. Lind and B. Marcus. *Symbolic Dynamics and Coding*. Cambridge University Press, Cambridge, 1995.
21. N. Packard, J. Crutchfield, J. Farmer, and R. Shaw. Geometry from a time series. *Physical Review Letters*, 45:712, 1980.
22. F. J. Pineda and J. C. Sommerer. Estimating generalized dimensions and choosing time delays: A fast algorithm. In *Time Series Prediction: Forecasting the Future and Understanding the Past*. Santa Fe Institute Studies in the Sciences of Complexity, Santa Fe, NM, 1993.
23. R. Stolle and E. Bradley. A customized logic paradigm for reasoning about models. In *Proceedings of the Tenth International Workshop on Qualitative Reasoning about Physical Systems*, 1996. Stanford Sierra Camp, CA.
24. R. Stolle and E. Bradley. Opportunistic modeling. In *IJCAI-97 Workshop on Engineering Problems for Qualitative Reasoning*, 1997.
25. F. Takens. Detecting strange attractors in fluid turbulence. In D. Rand and L.-S. Young, editors, *Dynamical Systems and Turbulence*, pages 366–381. Springer, Berlin, 1981.
26. L. Travé-Massuyès and R. Milne. Application oriented qualitative reasoning. *Knowledge Engineering Review*, 10(2):181–204, 1995.
27. K. Yip. *KAM: A System for Intelligently Guiding Numerical Experimentation by Computer*. Artificial Intelligence Series. MIT Press, 1991.
28. F. Zhao. Computational dynamics: Modeling and visualizing trajectory flows in phase space. *Annals of Mathematics and Artificial Intelligence*, 8:285–300, 1993.