Using Polynomial Approximations to Discover Qualitative Models

Reha K. Gerçeker
reha@computer.org
Dept. of Computer Engineering, Boğaziçi University
Bebek 34342, Istanbul, Turkey

A. C. Cem Say
say@boun.edu.tr
Dept. of Computer Engineering, Boğaziçi University
Bebek 34342, Istanbul, Turkey

Abstract

Automating the discovery of qualitative models from observations is a difficult problem of machine learning and various algorithms have been proposed for the solution of this problem in the literature. In this paper, we present a new algorithm called LYQUID, which uses polynomials fitted on observed numerical data as approximations to the underlying real world functions; constraint discovery is then performed over those polynomials. LYQUID is shown to be a fast and successful learning algorithm even in the presence of a high level of noise.

Introduction

Formulating and solving differential equation systems belonging to real world problems lie at the heart of many, if not all, engineering disciplines. However, situations arise frequently where it is difficult to even formulate an ODE for a given system, let alone solve it to reveal the exact functions for its variables. In such cases, moving into the qualitative domain helps not only with the modeling problem but also allows simulation of the system with less precise information.

System identification deals with the problem of discovering the mathematical model of a system from observed input and output variables. The main problem in system identification problems is the size of the initial search space. Narrowing down the huge search space into descriptions of possible models is called structural identification; deciding for the precise model from that subset is a problem of parameter estimation. In this respect, qualitative modeling is a really expressive tool in representing structure in the structural identification stage. Using qualitative modeling, it is possible to express relationships between system variables without ever having to find out what they precisely are. Therefore, discovering the qualitative model of a physical system is an important step in understanding the underlying mechanism which determines the system's behavior.

Different aspects of qualitative reasoning and benefits of qualitative reasoning based approaches for system identification are examined thoroughly by Travé-Massuyès, Ironi and Dague (Travé-Massuyès, Ironi, & Dague 2004). Similarly, Bratko and Šuc have reviewed different techniques used in learning qualitative models in detail (Bratko & Šuc 2004).

This paper proposes a new methodology, which is called LYQUID, for automated discovery of a qualitative model from observed numerical data. Constraints on the properties of the observed data are lowered to a minimum; neither the observed quantities must be sampled with the same frequency nor the samples have to be equally spaced in time. Even absence of samples at some intervals could be tolerated. Moreover, the intuition behind the algorithm proves to be very strong against noise over the samples. The same intuition also lowers computational complexity; LYQUID's complexity at the model discovery stage is free of the number of observed samples.

A New Algorithm: LYQUID

Recalling the Taylor series, it is possible to approximate arbitrary functions with polynomials and there are various ways of fitting polynomials over sampled data using least square error techniques (Ralston & Rabinowitz 2001). With this intuition, LYQUID first approximates observed variables using polynomials that are functions of time and then uses the fitted polynomials during the identification process. In fact, the name LYQUID arises from a combination of the word poLYnomial with QUalitative system IDentification.

LYQUID uses orthogonal polynomials, which results in better numerical solutions, when fitting polynomials to sampled data. We have adopted an algorithm for curve fitting from Ralston and Rabinowitz (Ralston & Rabinowitz 2001), who have discussed least square error techniques in great detail. Similar mathematical texts should be consulted for details of polynomial curve fitting using orthogonal polynomials.

LYQUID requires quantitative data, which are sampled over time from the observed system, as input. It is neither mandatory to have equal number of samples for each variable nor necessary to have the samples equally spaced over time for the purpose of the algorithm. The samples together with the timepoints of measurements are input to the algorithm. The unit of each observed variable and a multiplication table of the specified units are also provided to the algorithm as inputs.

The output of LYQUID is a qualitative model expressed in QSIM's well–known vocabulary (Kuipers 1994). Con-

sidering the availability of numerical data for processing, LYQUID enhances its output by suggesting landmark values for constants and functions for monotonic relationships. In other words, it is very much possible to use LYQUID's output with semi–quantitative extensions of QSIM like Q2 (Kuipers 1994) and Q3 (Berleant & Kuipers 1997).

Fitting Piecewise Polynomials

One of the parameters of LYQUID is the degree of polynomials used in curve fitting. When this degree increases, the accuracy of the fit automatically increases; remember that the degree of Taylor series expansion tends to infinity. However, with high degree approximations, there is also the possibility of overfitting unnecessarily to noisy data. Therefore, this parameter should be tuned according to the complexity of input data; it is referred to as d for the rest of the paper. d determines the accuracy of the approximation and there are two main factors that hinder accuracy:

- d being insufficiently small for the input time interval. As an example to the problem here, consider the sine wave. Using perfectly clean samples from the sine wave and setting $d = 5$, unfortunately, it is not possible to have a polynomial fit with an acceptable square error over the whole $[-\pi,\ \pi]$ interval; a single fifth degree polynomial approximates the sine wave with reasonable error only on $[-\frac{1}{2}\pi,\ \frac{1}{2}\pi]$ and additional polynomials are needed in order to approximate the rest of the wave. A single seventh degree polynomial, on the other hand, gives a low error approximation over the whole $[-\pi,\ \pi]$ interval.
- Noise over the samples. A polynomial of degree d may actually be sufficient to fit perfectly clean samples on a time interval. However, data are never such clean and square error of the approximation could still be high because of noise over data samples.

Inaccuracy caused by the first problem is not tolerable, whereas, inaccuracy related to noise must be allowed to some extent. When d is fixed to some value, the only way to increase accuracy of the fit is to increase the number of polynomials used in approximation, by splitting the large input time interval into smaller sub–intervals. This corresponds to using piecewise polynomials rather than fitting a single polynomial on all of the samples.

LYQUID uses Algorithm 1 to divide a large time interval into smaller sub–intervals to increase the accuracy of approximation. A parameter called $Tolerance$ is used to decide whether the error of approximation is acceptable or not. When the error is not acceptable, LYQUID splits the time interval at the midpoint and fits two polynomials on both the new sub–intervals. In order for this split to be accepted, the accuracy after the split must be improved; parameters $Improvement$ and $SplitImprovement$ determine the acceptable rate of improvement. Otherwise, the split is rejected and a single polynomial is fitted on the whole interval. If the actual cause of inaccuracy in an approximation is the noise over the data, then this mechanism of accepting or rejecting a split makes sure that the split is really necessary. As a final remark, it should be noted that the partitioning algorithm does not require that the fitted piecewise polynomial be continuous at transition points between sub–intervals.

Algorithm 1 Interval partitioning algorithm

```
partition is a {start, stop, poly, error} tuple
intervals and result are stacks of partition
P ← d–degree polynomial on S[1..N]
e ← square error of P on S[1..N]
intervals.push {1, N, P, e}
while intervals.empty is false do
  interval ← intervals.pop
  e ← interval.error
  c ← interval.stop - interval.start + 1
  if (e/c) ≤ Tolerance then
    result.push interval
  else
    ini ← interval.start
    fin ← interval.stop
    mid ← ⌊(ini+fin)/2⌋
    P1 ← d-degree polynomial on S[ini..mid]
    e1 ← square error of P1 on S[ini..mid]
    P2 ← d-degree polynomial on S[mid + 1..fin]
    e2 ← square error of P2 on S[mid + 1..fin]
    chgInTotalErr ← (e − (e1 + e2)) / e
    chgInInterval1 ← (e − 2 e1) / e
    chgInInterval2 ← (e − 2 e2) / e
    if chgInTotalErr ≥ Improvement or
       chgInInterval1 ≥ SplitImprovement or
       chgInInterval2 ≥ SplitImprovement
     then
      intervals.push {mid + 1, fin, P2, e2}
      intervals.push {ini, mid, P1, e1}
    else
      result.push interval
    end if
  end if
end while
```

Polynomial Operations

After piecewise polynomial approximations are computed for every observed variable, LYQUID no longer uses the numerical data samples for any purpose. In other words, LYQUID performs all identification functions based on the computed polynomials. Therefore, this subsection is devoted to a brief discussion of polynomial operations.

Being able to check equality of polynomials is the fundamental requirement in LYQUID. Even though two polynomials are equal only when their coefficients are equal, it is generally the case that curve fitting do not produce such exact matches. In other words, some margin of error must be tolerated. When comparing two polynomials $p(t)$ and $q(t)$ on a time interval $[a, b]$, LYQUID uses this error criterion:

$$e = \frac{\int_a^b [p(t) - q(t)]^2 dt}{b - a}$$
$$equal(p, q) = \begin{cases} true & \text{if } e \leq iTolerance \\ false & \text{otherwise} \end{cases} \quad (1)$$

$iTolerance$ is another parameter of the algorithm. Similar to $Tolerance$, which appeared in Algorithm 1, $iTolerance$

specifies the tolerable square error per unit time during model discovery. It is possible to set these parameters to different values.

During identification, LYQUID has to perform different types of operations on polynomials. These operations are addition, subtraction, multiplication, translation, synthetic division, differentiation and integration. All of these operations require a basic a manipulation of polynomial coefficients. Even though it is also dependant on the polynomial representation, it is straightforward to implement listed operations in $O(d^2)$ time, where d is the degree of the operated polynomials.

For monotonic function constraints, LYQUID needs to check whether or not a polynomial is positive or negative on a time interval $[a, b]$. This check translates into checking existence of polynomial roots on $[a, b]$. It is possible to find the number of roots of a polynomial on an interval $[a, b]$ by using properties of a mathematical concept called the Sturm sequences. It is also possible to create a Sturm sequence from an arbitrary polynomial in $O(d^2)$ time (Ralston & Rabinowitz 2001).

Generation of Hidden Variables

Most of the real world qualitative models involve relationships between observable variables and some hidden variables. Discovering such relationships is the essence of learning qualitative models. This subsection explains how LYQUID generates hidden variables; they are generated from polynomial representations of observed variables. Note that, among the generated variables, only the ones which are involved in the discovered constraints are included in the output model.

Apart from looking for equalities between polynomials using the error criterion in (1), LYQUID requires that units of the compared polynomials match in order to conclude that an equality exists. Units of the observed variables are provided by the user, whereas, units of the generated variables must be automatically assigned by LYQUID.

The units of polynomials change only when they are differentiated or multiplied. Units are stored by LYQUID in such a way that time derivatives of the same unit are maintained as an ordered list. Therefore, whenever LYQUID generates a derivative, unit of the polynomial is lowered one level in the list; a new entry is appended to the list if necessary. Multiplication table of units, which is provided as input by the user, is used in the case of multiplication. LYQUID allows a multiplication only if the result of the multiplied units exist in this table.

LYQUID first generates derivatives of observed variables by differentiating their polynomials. First $o \leq d$ derivatives are computed, where o is a parameter of the algorithm. Whenever a constant is found in the derivative chain, LYQUID stops seeking into higher orders.

Composites are then generated as products of sums. LYQUID allows only a single appearance of an observed variable or one of its derivatives in a composite variable. Moreover, another parameter $c \leq n$ is used to limit the size of composite variables, where n is the number of observed variables. Setting $c > n$ is equivalent to setting $c = n$ because of the single appearance limitation. When generating sums and differences, all possible combinations of variables with matching units are taken into consideration. When generating products as composites, only the ones whose units could be determined from the multiplication table are created.

At the time of constraint discovery, size of the set of variables plays an important role in the complexity of the procedure. The problem here is to determine the number of variables after hidden variable generation when starting with n observed variables. In the worst case, all observed variables have a common unit and when $c = n$, the number of variables after generation of sums is equal to $\frac{1}{2}(o + 1)(3^n - 1)$. When $c < n$, the number of variables could be written with a tighter bound: $O(n^c(o + 1))$. Assuming that the multiplication table allows multiplication of a sequence of at most m units, it is definite that the size of the final set of variables is $O((n^c(o + 1))^m)$. Remember that when $c = n$, the size of variables is inevitably exponential with n.

Model Discovery

LYQUID's model discovery is a generate–and–test procedure where every possible constraint is tested to see if it truly exists. For LYQUID, checking existence of a constraint is equivalent to checking whether or not two polynomials with common units are equal according to Equation 1. This is basically done in three steps: (*i*) create an intermediate polynomial according to the nature of the constraint, (*ii*) determine the unit of this polynomial, (*iii*) if units of the intermediate and target polynomials match, compare the polynomials. If an equality is found at the end, proposed constraint is added to the output model.

Assuming that their units are common, when checking existence of an *add* constraint between three polynomials $p(t)$, $q(t)$ and $r(t)$, for example, LYQUID creates the intermediate polynomial by adding $p(t)$ and $q(t)$. Then, the target polynomial, which will be used in comparison with this intermediate polynomial, is $r(t)$. LYQUID also checks the other two possibilities in which the intermediate polynomial is obtained by adding $p(t)$ and $r(t)$ or adding $q(t)$ and $r(t)$. The target polynomials in these cases are $q(t)$ and $p(t)$ respectively. If one of the comparisons results in an equality of the intermediate and target polynomials, an *add* constraint is included in the output model. Other constraint types require a different way of obtaining the intermediate polynomial, however, the main idea is always the same.

Monotonic function constraints, on the other hand, are a little different. First of all, unit checking is not done when testing M^+ and M^- constraints. As a first requirement, the product of the derivatives of the involved polynomials must be positive or negative on the questioned time interval. This should remind the reader of the discussion about polynomial roots appearing in the section about polynomial operations. Secondly, if the involved polynomials in the constraint are $p(t)$ and $q(t)$, there should be a third polynomial $f(t)$ such that $p(t) = f(q(t))$ satisfying the error criterion. This requirement is inherent from QSIM's definition of M^+ and M^- relationships. Given $p(t)$ and $q(t)$, the degree of $f(t)$ is

first decided from the degrees of $p(t)$ and $q(t)$; unkown coefficients of $f(t)$ are then calculated by setting partial derivatives of the error criterion in Equation 1 to zero.

Since variables are approximated by piecewise polynomials, LYQUID has to test existence constraints on multiple time intervals. Assuming that piecewise polynomials consist of k sub–parts at maximum, k has to appear as a linear factor in the complexity of the discovery stage. Therefore, it is important to put an upper bound on k. In the general case, it is impossible to estimate k precisely; however, it is possible to put a bound on k using the maximum number of extrema taken on by any one of the observed variables.

Assume that one of the observed variables attains at most x local extrema during the observation interval $[a, b]$. Then, it is possible to say that every observed variable consists of at most $x + 1$ monotonically increasing or decreasing sections, which connect subsequent local extrema. Fitting polynomials on such increasing or decreasing sections generally yields very good approximations. Therefore it is reasonable to conclude that k is $O(x)$ if not exactly $x + 1$.

Evaluation of constraints in multiple time intervals results in three levels of strength for constraints. These levels are determined by the user through the use of three ratios, $R_P \leq R_L \leq R_D$, each of which are real numbers in $[0, 1]$. Assume that a constraint exists on s of the total k sub–intervals; LYQUID calculates the ratio $R = s/k$ and decides that (*i*) constraint is definite if $R \geq R_D$, (*ii*) constraint is likely if $R_L \leq R < R_D$, (*iii*) constraint is possible if $R_P \leq R < R_L$. If $R < R_P$, then the constraint does not exist in LYQUID's terms.

LYQUID in Pseudo–Code

Algorithm 2 makes references to subroutines starting with $Generate$. They are related to the generation of hidden variables which is discussed in the related section. There are also references to subroutines starting with $Check$. These routines work with the principles explained in the previous section and they return a set of constraints if a qualitative relationship is found between the input variables and an empty set otherwise. The discovered qualitative relationships are appended to a list of constraints, which contains the qualitative model of the observed system once the algorithm stops.

When a constraint is newly discovered by one of the $Check$ subroutines and the new constraint involves an automatically generated hidden variable, LYQUID appends the qualitative constraints, which are necessary to create the automatic variable, to the list of constraints. This makes sure that the qualitative constraints that result in creation of a new automatic variable do not appear in the output qualitative model when the new variable is not bound to the model with a different constraint. LYQUID also makes sure not to append artificial discoveries to the list of constraints. For example, when a generated variable $auto1$ is the sum of two observed variables x and y, $CheckSum(x, y, auto1)$ would return a valid qualitative relationship without LYQUID's internal redundancy checking.

Overall Complexity

The number of partitions generated by Algorithm 1 is $O(x)$, which means that the main while loop of the algorithm will run $O(x)$ times. The polynomial fitting algorithm adopted from Ralston and Rabinowitz (Ralston & Rabinowitz 2001) takes $O(Nd^2)$ time, where N is the number of processed samples. Then, overall complexity of this initial stage is $O(n\,x\,Nd^2)$.

Complexity of the hidden variable generation stage is $O((n^c(o+1))^m x\,d^2)$. If the number of variables prior to constraint discovery is v, then there could be at most $O(v^2)$ binary constraints and $O(v^3)$ ternary constraints. Even though the dominant term arises from binary constraints and actual complexity is much lower because of unit checking, it is safe to express complexity of the discovery stage as $O(v^3 x\,d^2)$. As a result, the overall complexity of LYQUID is

$$O(n\,x\,Nd^2 + (n^c(o+1))^{3m} x\,d^2) \tag{2}$$

The dominating term here is the second operand of the sum inside the brackets and it belongs to the model discovery procedure. This term is a polynomial of n whose power is determined by parameters of the algorithm. It is important not to have N as a factor in this dominating term because N is capable of growing quite large.

Algorithm 2 LYQUID

$V \leftarrow \emptyset$ *(set of variables)*
for $k = 1$ to n **do**
  $ppoly \leftarrow$ new piecewise polynomial fitted to $\left\{t_i^{(k)}, x_i^{(k)}\right\}_{i=1}^{N^{(k)}}$
  $var \leftarrow$ new variable, whose unit is $u^{(k)}$, represented by $ppoly$
  $V \leftarrow V \cup var$
end for
$V \leftarrow V \cup GenerateDerivatives(V, o)$
$V \leftarrow V \cup GenerateSums(V, c)$
$V \leftarrow V \cup GenerateProducts(V, c, M)$
$C \leftarrow \emptyset$ *(list of constraints)*
for all $(v_1, v_2) \in V$ **do**
  $C_{temp} \leftarrow CheckEquality(v_1, v_2)$
  if $C_{temp} \neq \emptyset$ **then**
    $V_{prune} \leftarrow v_2 \cup \{v \mid v$ is derived from v_2 or involves $v_2\}$
    $V \leftarrow V - V_{prune}$
    $C \leftarrow C \cup C_{temp}$
  end if
end for
for all $v \in V$ **do**
  $C \leftarrow C \cup CheckConstant(v)$
end for
for all $(v_1, v_2) \in V$ **do**
  $C \leftarrow C \cup CheckDerivative(v_1, v_2)$
  $C \leftarrow C \cup CheckMonotonic(v_1, v_2)$
end for
for all $(v_1, v_2, v_3) \in V$ **do**
  $C \leftarrow C \cup CheckSum(v_1, v_2, v_3)$
  $C \leftarrow C \cup CheckProduct(v_1, v_2, v_3)$
end for

Related Work

In terms of input and output of the algorithm, LYQUID resembles the QMN algorithm (Džeroski & Todorovski 1995). It is another generate–and–test based learning method which extracts qualitative models from numerical behavior. The logic behind QMN is very similar to LYQUID's logic; however, rather than working on polynomials, QMN decides by working on individual data samples. One of the disadvantages of QMN is that its complexity involves N and N^2 as linear factors in front of expressions which are exponential with n in the worst case. This is an important drawback when N is large. LYQUID's complexity involves N only as a factor of n at the curve fitting stage and removes N from the possibly exponential complexity of the discovery stage. This means that the dominant term of complexity of LYQUID is not dependant on N. Another disadvantage of QMN is that its computation is based on individual data samples making the algorithm prone to noise over the data; on the contrary, LYQUID is a more robust algorithm which uses an identification approach based on a polynomial interpolation of samples. Moreover, QMN requires samples to be made at the same timepoints in order for them to be compared, whereas, LYQUID is capable of using samples which are measured at different timepoints. One final advantage of LYQUID over QMN is that it uses units of variables in its reasoning, which reduces the size of the variable and constraint spaces significantly.

LAGRANGE (Džeroski & Todorovski 1993; 1995) is another identification algorithm which outputs ODEs rather than QDEs. Its specification is very similar to QMN and LYQUID in the way it introduces new variables. Differing from QMN in its discovery stage, LAGRANGE generates and tests numerical equations over its variable set by linear regression. This feature is reminiscent of LYQUID's polynomial regression. An upgrade of this method is LAGRAMGE (Todorovski *et al.* 2000), which uses multi–dimensional polynomial approximations in calculating partial derivatives of variables. However, unlike LYQUID, these approximations bind variables between themselves, rather than expressing them as functions of time. LYQUID seems to be unique in that respect.

The algorithms discussed upto this point, including LYQUID, use numerical data samples in a rather quantitative manner to produce their output. SQUID (Kay, Rinner, & Kuipers 2000), on the other hand, also starts with numerical data as input but does not perform what it calls raw data matching; it rather focuses on qualitative aspects of the data like trends and extrema, or envelopes that bound the trajectory of variables. It then works by consequent forward and reverse simulations to reduce size of the initially large model space by refutation.

There are some algorithms which use qualitative, rather than quantitative descriptions of behavior for identification. Those methods, even if they accept numerical data as input, convert their input into qualitative states before starting identification. GENMODEL (Hau & Coiera 1993) is one of the earliest of such methods, which constructs all possible constraints consistent with the input qualitative samples. It suffers from underconstrained models because it lacks creation of hidden variables. QSI (Say & Kuru 1996) solves the problem introducing hidden variables and iteratively does the same thing with GENMODEL until the discovered model allows only the observed qualitative behavior. Another method Q_{OPH} (Coghill, Garrett, & King 2002) relies on Inductive Logic Programming, which is a rather formal way deducing a model (hypothesis) that satisfies observations (evidence as qualitative states) given some background knowledge (QSIM state transition rules). Such a general formulation of the problem allows usage of a general purpose ILP technique as a solution.

Usage of units, commonly called dimensional analysis, also appears in GENMODEL. MISQ (Richards, Kraan, & Kuipers 1992) and QSI use a different type of dimensional analysis in testing consistency of generated models. This type of dimensional analysis checks for inconsistencies in constraints involving time derivatives rather than units of variables.

Experimental Results

LYQUID was tested on the U–tube, cascaded tanks and coupled tanks models which are used as benchmark systems in related literature (Kuipers 1994; Coghill, Garrett, & King 2002). The numerical data required for the experiments were obtained by simulating physically realistic systems of the three kinds. Apart from the clean data, noisy datasets were obtained by adding Gaussian noise over the original samples and LYQUID has been tested on both clean and noisy data samples. Other types of experiments involve datasets in which, (*i*) samples are obtained with different or irregular sampling frequencies, (*ii*) there are time intervals where some variables are not sampled.

Experiments on Clean Data

U–Tube U–tube data used in the experiments belong to a cylindrical U–tube with 1.25 and 0.8 units of radii. Initial heights of the liquid columns in the tubes were 2.5 and 0.5 units respectively and the levels were sampled over time with 0.01 second intervals for a total of 3 seconds. Recall at this point that all U–tubes obey the following qualitative model, which is unknown to LYQUID at any point:

d/dt($level_A$*, *$rise_A$*)* $\quad$ *M*$^-$*(levelDiff, *$rise_A$*)* (0, 0)
d/dt($level_B$*, *$rise_B$*)* $\quad$ *M*$^-$*(*$rise_A$*, *$rise_B$*)* (0, 0)
add($level_B$*, levelDiff, *$level_A$*)*

Using fifth degree polynomials for approximations and 6×10^{-5} as *iTolerance*, LYQUID discovered the constraints given in Table 1. Although the listed constraints are numerous when compared to the general U–tube model, all of the listed monotonic function constraints are correct for the specific numerical dataset. The functional relationships suggested for the monotonic function constraints are not provided but the landmark pairs given in the table are the intercepts of those unlisted functions. Note that the hidden variables *rise*$_A$, *rise*$_B$ and *levelDiff* are discovered as `dA`, `dB` and `auto2` respectively; the two monotonic function constraints which bind these hidden variables are also discovered. Being able to capture this detailed qualitative model from only two observed variables is remarkable.

d/dt(A dA)			M+(B auto1)	(-4.5811 0.0000)	(0.0000 2.7048)
d/dt(B dB)			M-(B auto2)	(1.9188 0.0000)	(0.0000 2.7048)
add(A B auto1)			M-(dA dB)	(0.0000 0.0000)	
minus(B -B)			M+(dA auto1)	(-99.6365 0.0000)	(0.0000 3.8394)
add(A -B auto2)			M-(dA auto2)	(-0.0121 0.0000)	(0.0000 -0.0040)
M-(A B)	(2.7048 0.0000)	(0.0000 6.6033)	M-(dB auto1)	(243.2531 0.0000)	(0.0000 3.8394)
M-(A dA)	(1.9177 0.0000)	(0.0000 97.3223)	M+(dB auto2)	(0.0297 0.0000)	(0.0000 -0.0040)
M-(A auto1)	(4.5813 0.0000)	(0.0000 6.6035)	M-(auto1 auto2)	(3.8377 0.0000)	(0.0000 9.1624)
M+(A auto2)	(1.9188 0.0000)	(0.0000 -6.6034)	M+(A dB)	(1.9177 0.0000)	(0.0000 -237.5864)
M+(B dA)	(1.9217 0.0000)	(0.0000 -2.4451)	M-(B dB)	(1.9217 0.0000)	(0.0000 5.9695)

Table 1: Discoveries from the clean U–tube dataset

Cascaded Tanks In a cascaded tanks system, there are two separate tanks with holes at their bases. There is an inflow of liquid into the first tank and outflow from this first tank pours into the second tank. The most general qualitative model for a cascaded tanks system is the following:

d/dt(amount$_A$, netflow$_A$)
d/dt(amount$_B$, netflow$_B$)
add(outflow$_A$, netflow$_A$, inflow)
add(outflow$_B$, netflow$_B$, outflow$_A$)
M$^+$(amount$_A$, outflow$_A$) (0, 0)
M$^+$(amount$_B$, outflow$_B$) (0, 0)

Dataset for the cascaded tanks experiment was created by simulating a cascaded tanks system with two cylindrical tanks whose radii are 1 and 0.75 units. The initial levels of liquid in the tanks were 1 units each and *amounts* of liquid in the tanks were sampled over time with 0.01 second intervals for a total of 4 seconds. Inflow of liquid into the first tank, which is the exogenous variable, was taken to be $e - e^{-t}$. Unit of inflow was specified as the derivative of the unit of amounts. The sampled levels and inflow were supplied as input to the algorithm.

LYQUID was tested on the cascaded tanks dataset with fifth and eleventh degree polynomials separately. In both cases $iTolerance$ was set to 7×10^{-5}. Results that are listed in Table 2 show that the hidden variables *netflow$_A$*, *netflow$_B$*, *outflow$_A$* and *outflow$_B$* are discovered as `dA`, `dB`, `auto4` and `auto16` respectively. The monotonic function relationships between the amounts and outflows are also discovered. Considering nature of the exponential function that is used as inflow, the monotonic function constraint that relates inflow and its derivative is also correct. Inspection of the simulation data revealed that the three other monotonic function constraints are also correct discoveries.

An important observation from Table 2 is the increase in the accuracy of landmark pairs when eleventh degree polynomials are used. Not only the accuracy of LYQUID's numerical suggestions improve but also LYQUID discovers another true monotonic function constraint which was left unrevealed otherwise. This improvement is obviously caused by the increase of the accuracy of the polynomial approximations.

Some monotonic function constraints given in Tables 1 and 2 are redundant in the presence of some other ones. LYQUID retains those redundant constraints in its output because extra landmark pairs provided with them might be necessary to eliminate possible spurious behaviors from the qualitative simulation of the output model. It should also be noticed that one of the monotonic function constraints lacks landmark pairs in Table 2; this is because the intercepts of the discovered relationship are inconsistent with the M^-.

Coupled Tanks In a coupled tanks system, two tanks are interconnected with a horizontal pipe; there is an inflow of liquid to one of the tanks and there is an outflow of liquid from the other tank. A general qualitative model for the described system is the following one:

d/dt(amount$_A$, netflow$_A$)
d/dt(amount$_B$, netflow$_B$)
add(flow$_{AB}$, netflow$_A$, inflow)
add(outflow$_B$, netflow$_B$, flow$_{AB}$)
add(level$_B$, levelDiff, level$_A$)
M$^+$(amount$_B$,outflow$_B$) (0, 0)
M$^+$(amount$_A$, level$_A$) (0, 0)
M$^+$(amount$_B$, level$_B$) (0, 0)
M$^+$(levelDiff, flow$_{AB}$) (0, 0)

It is also possible to write the same model using pressures as the hidden variables instead of levels. Notice that, be it levels or pressures, these variables are introduced to the model through monotonic function constraints. Since LYQUID introduces hidden variables only through addition or multiplication, without observing either the levels or pressures, it is not possible to make LYQUID extract this general model. This is why both levels and amounts of liquid in the tanks were sampled while generation of the coupled tanks dataset.

The dataset was created by simulating a coupled tanks system with two cylindrical tanks whose radii are 1 and 0.8 units. The tanks were initially empty. Amounts and levels of water in the tanks were sampled over time with 0.01 second intervals for a total of 10 seconds. The exogenous inflow variable was taken to be constant. Unit of inflow was specified as the derivative of the unit of amounts and unit of levels was specified as a different unit. Amount, level and inflow samples were input to LYQUID.

Using eleventh degree polynomials for approximations and 7×10^{-5} as $iTolerance$, LYQUID discovered a total of 190 monotonic function constraints involving 20 different variables. Five of these 20 variables are the observed amounts and levels; other 15 variables are the hidden vari-

Using fifth degree polynomials			Using eleventh degree polynomials		
d/dt(A dA)			d/dt(A dA)		
d/dt(B dB)			d/dt(B dB)		
d/dt(I dI)			d/dt(I dI)		
add(A B auto1)			add(A B auto1)		
add(I dA auto3)			minus(B -B)		
minus(dA -dA)			add(A -B auto2)		
add(I -dA auto4)			add(I dA auto3)		
minus(dB -dB)			minus(dA -dA)		
add(I -dA -dB auto16)			add(I -dA auto4)		
M+(A auto4)	(-0.1082 0.0000)	(0.0000 1.2634)	minus(dB -dB)		
M-(B auto3)	(1.7135 0.0000)	(0.0000 2.1635)	add(I -dB auto6)		
M+(B auto16)	(-0.0625 0.0000)	(0.0000 1.2885)	add(I -dA -dB auto16)		
M-(I dI)	(2.7181 0.0000)	(0.0000 2.6350)	M+(A auto4)	(0.0001 0.0000)	
M-(dA auto1)	(1.0462 0.0000)	(0.0000 1.8080)	M-(B auto3)	(1.7133 0.0000)	(0.0000 2.1838)
M-(auto3 auto16)			M+(B auto16)	(0.0000 0.0000)	
			M-(I dI)	(2.7183 0.0000)	(0.0000 2.7183)
			M-(dA auto1)	(1.0606 0.0000)	(0.0000 1.8085)
			M-(auto2 auto6)	(3.6566 0.0000)	(0.0000 3.3998)
			M-(auto3 auto16)		

Table 2: Discoveries from the clean Cascaded Tanks dataset

ables generated by LYQUID. Those hidden variables include *netflow*$_A$, *netflow*$_B$, *flow*$_{AB}$, *outflow*$_B$ and *levelDiff*. Although the suggested landmark pairs deviate slightly from the origin, the monotonic function constraints that appear in the general qualitative model also existed in LYQUID's output.

126 of the total 190 monotonic function discoveries were correct for the dataset that was used; other 64 discoveries were incorrect. The decision about the correctness of each of the constraints is made by looking at the plots of the original numerical samples belonging to the variables involved in the constraint. The incorrect discoveries were actually functions that made a sudden jump during the first second of sampling; if these initial portions were disregarded, all of the discovered monotonic function relationships would have been consistent with the observed behavior.

Experiments on Noisy Data

U–Tube A noisy dataset for the U–tube was generated by adding *N*(0, 1) Gaussian noise over the clean samples that were used for the first experiment. Figure 1 is the plot of the noisy dataset together with the original clean samples. Table 3 lists the constraints which are discovered by LYQUID on this noisy U–tube dataset. The only difference in LYQUID's settings between the two U–tube experiments is the value of $iTolerance$; $iTolerance$ was set to 6×10^{-2} for the experiment on the noisy dataset.

Obviously, Tables 1 and 3 differ only in their landmark value pairs. Many constraints in the second experiment lack landmark pairs because intercepts of the discovered relationships were not consistent with the M^+ or M^-. The M^+ between dB and auto2, namely *rise*$_B$ and *levelDiff*, originally passes through the origin and this landmark is a critical one for elimination of unrealistic behavior. LYQUID was able to estimate landmark pairs close to the origin, which is satisfactory considering the amount of noise. The M^- between *rise*$_A$ and *rise*$_B$ also has a critical landmark pair at the origin; this pair is implicit, even though not perfectly, from the monotonic function constraints between dA, dB and auto1. Shortly, discoveries listed in Table 3 is a strong evidence of how tolerant LYQUID is against noise.

Cascaded Tanks A noisy cascaded tanks dataset was created by adding *N*(0, 0.5) noise over the clean samples which were used in the first cascaded tanks experiment. The clean and noisy datasets are plotted together in Figure 2. Seventh degree polynomials were used in this experiment and $iTolerance$ was increased to 8×10^{-3} to allow more error in comparisons.

In this experiment, LYQUID discovered 55 monotonic function constraints which involve 3 observed and 9 hidden variables. All constraints listed in Table 2 are successfully discovered, even though the landmark pairs are not so accurate as in the U–tube case. In fact, all of the discovered constraints are correct when the observed numerical behavior is considered; the reason why these constraints were not discovered in the clean dataset experiment is related to the value of $iTolerance$. In the clean dataset experiment, $iTolerance$ was set to a very low value, which in turn resulted in rejection of actually correct relationships. The constraints in Table 2 are the ones which were able to survive under a very low margin of error. As a result, although it discovered too many constraints, LYQUID was able to capture the necessary constraints which determine the behavior of the system.

Other Experiments

Arbitrarily Sampled Data In this experiment, LYQUID was tested on numerical U–tube data again. This time, however, the data samples were taken arbitrarily from the whole 3 second observation interval. 300 samples were taken from levels of water in the tubes; even though a specific sam-

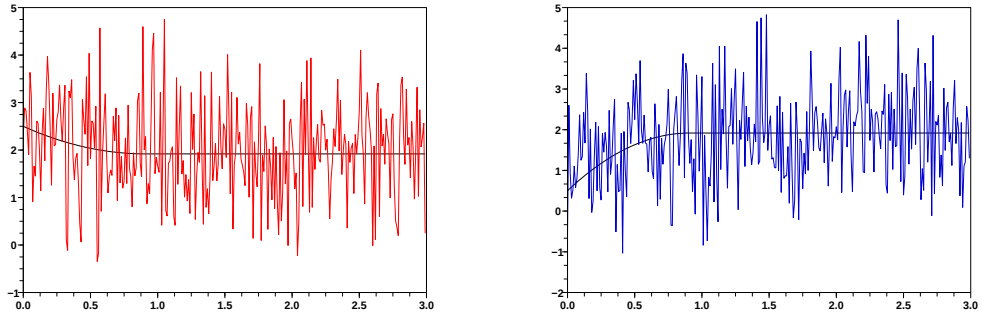

Figure 1: U–tube samples with $N(0, 1)$ Gaussian noise: $level_A$ (left), $level_B$ (right).

d/dt(A dA)			M−(dA dB)		
d/dt(B dB)			M+(dA auto1)	(−2.2032 0.0000)	(0.0000 3.8270)
add(A B auto1)			M−(dA auto2)		
minus(B −B)			M−(dB auto1)	(120.3892 0.0000)	(0.0000 3.8643)
add(A −B auto2)			M+(dB auto2)	(−0.0397 0.0000)	(0.0000 0.0122)
M−(A B)			M−(auto1 auto2)	(3.8735 0.0000)	(0.0000 45.6008)
M−(A auto1)			M−(A dA)	(1.9604 0.0000)	(0.0000 49.1951)
M+(A auto2)			M+(A dB)		
M+(B auto1)	(−22.8004 0.0000)	(0.0000 4.0153)	M+(B dA)		
M−(B auto2)	(1.9367 0.0000)	(0.0000 4.0153)	M−(B dB)		

Table 3: Discoveries from the noisy U–tube dataset

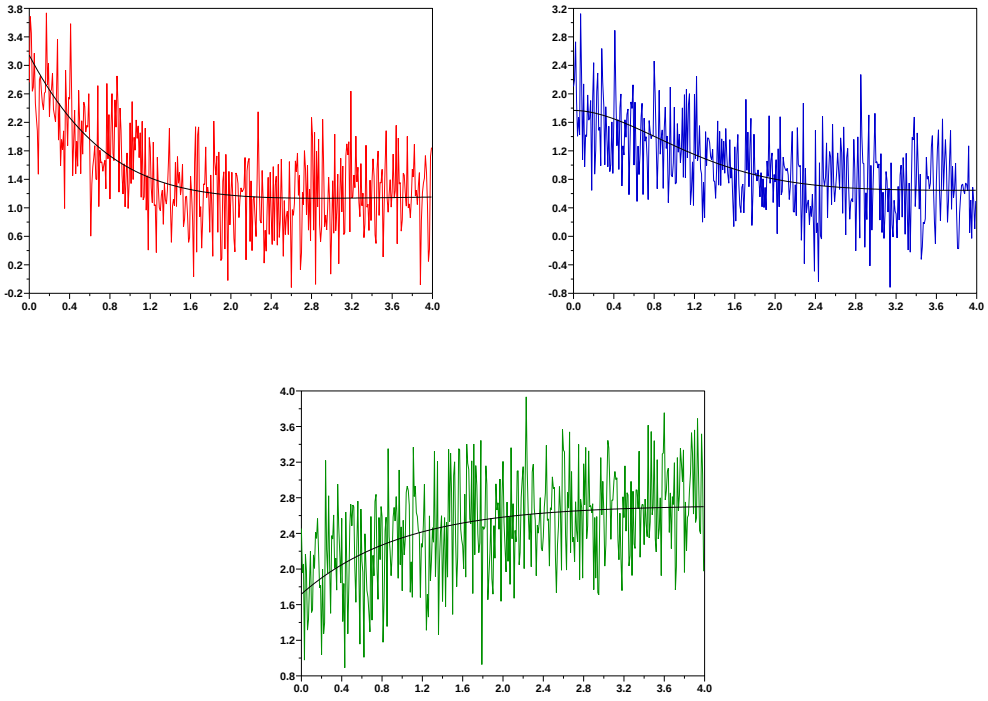

Figure 2: Cascaded tanks samples with $N(0, 0.5)$ Gaussian noise: $amount_A$ (top left), $amount_B$ (top right), *inflow* (bottom).

pling frequency was not used, the samples were still well distributed on the observation period. The data samples were left clean without artificial noise.

Using the same tolerance with the first U–tube experiment, LYQUID discovered the constraints listed in Table 4. The discovered constraints are identical to the discoveries on U–tube datasets with equi–distant samples. This is a clear evidence of how capable LYQUID is when working with arbitrarily sampled data. It shows that LYQUID neither requires data samples to be equally spaced in time nor sampling timepoints of different variables have to coincide.

Missing Data For the final type of experiment, numerical U–tube simulation was done with the usual settings; however, this time the samples were taken from only the $[0, 0.5s]$ and $[1.5s, 2.5s]$ time intervals. There were a total of 150 samples which were equally spaced in the specified time intervals. Note that the timepoint of reaching the equilibrium lies inside the $[0.5s, 1.5s]$ time interval, which was left unobserved. Increasing LYQUID's tolerance slightly to 8×10^{-5}, it was possible to end up with the discoveries which are also listed in Table 4. Note that, this time also, although some data samples were missing right in between the observation interval, LYQUID was able to capture all of the necessary constraints with appropriate landmark value pairs.

Conclusion

In this paper, LYQUID has been proposed as a new algorithm intended for the automatic extraction of qualitative models from observed numerical data. The mathematical background is very simple; samples from observations are used to make polynomial approximations to the real world functions hidden behind the observations. Polynomials are known to be powerful tools of approximation; operating on polynomials and making decisions based on those operations are also computationally cheap.

Using approximations is a firm technique against noise because approximations are generated from a group of samples, which softens the effect of noise on individual samples. It also relaxes the conditions on sampling properties; with LYQUID, unlike similar algorithms, it is possible to sample every variable with a different sampling frequency but still use measurements belonging to different timepoints in the discovery process. The technique could even be extended to work in the absence of data at certain time intervals; missing parts of the data could be filled in by extending neighboring approximations over to the unobserved time interval. All of these described properties have been demonstrated on realistic numerical data. LYQUID was shown to be capable of discovering constraints from noisy, arbitrarily sampled and partially observed data.

LYQUID was shown to possess a better complexity when compared to similar algorithms. Complexity of the model discovery stage of the algorithm is free from the number of samples. This is a favorable property because the exponentiality of the problem lies behind the generate–and–test methodology of the discovery stage. This improvement of complexity also arises from the computational ease of operating on polynomials.

As future work, LYQUID's multi–interval processing of samples needs to be improved. That is, LYQUID is not so successful in computing monotonic functions and landmark value pairs when more than one polynomial is fitted over the samples. Discovery of constraints on multiple intervals, on the other hand, still works correctly.

Another idea, which should also be treated as future work, is to run LYQUID on several datasets of the same system with different settings and then to combine output of all experiments into a single output model. As an example, consider the U–tube: first, several datasets with different amounts of water in the tubes will be created. Then, LYQUID will run on all of the datasets and discover constraints. When combining these output models, the algorithm will retain only the constraints which exist in every output model; it will also require that landmark values or value pairs be equal with some tolerated margin of difference. The constraints that remain in the end will be constraints which are truly related to the U–tube model and free from the experimental settings.

References

Berleant, D., and Kuipers, B. J. 1997. Qualitative and quantitative simulation: bridging the gap. *Artificial Intelligence* 95(2):215–255.

Bratko, I., and Šuc, D. 2004. Learning qualitative models. *AI Magazine* 24(4):107–119.

Coghill, G. M.; Garrett, S. M.; and King, R. D. 2002. Learning qualitative models in the presence of noise. In *Proceedings of the 16th International Workshop on Qualitative Reasoning: QR'02*, 27–35.

Džeroski, S., and Todorovski, L. 1993. Discovering dynamics. In *International Conference on Machine Learning*, 97–103.

Džeroski, S., and Todorovski, L. 1995. Discovering dynamics: From inductive logic programming to machine discovery. *Journal of Intelligent Information Systems* 4(1):89–108.

Hau, D. T., and Coiera, E. W. 1993. Learning qualitative models of dynamic systems. *Machine Learning* 26:177–211.

Kay, H.; Rinner, B.; and Kuipers, B. 2000. Semi-quantitative system identification. *Artificial Intelligence* 119(1-2):103–140.

Kuipers, B. 1994. *Qualitative reasoning: modeling and simulation with incomplete knowledge*. Cambridge, MA, USA: MIT Press.

Ralston, A., and Rabinowitz, P. 2001. *A First Course in Numerical Analysis*. Mineola, New York: Dover Publications, 2^{nd} edition.

Richards, B. L.; Kraan, I.; and Kuipers, B. 1992. Automatic abduction of qualitative models. In *National Conference on Artificial Intelligence*, 723–728.

Say, A. C. C., and Kuru, S. 1996. Qualitative system iden-

Arbitrarily Sampled Data			**Missing Data**		
d/dt(A dA)			d/dt(A dA)		
d/dt(B dB)			d/dt(B dB)		
add(A B auto1)			add(A B auto1)		
minus(B -B)			minus(B -B)		
add(A -B auto2)			add(A -B auto2)		
M-(A B)	(2.6920 0.0000)	(0.0000 6.4480)	M-(A B)	(2.7047 0.0000)	(0.0000 6.5978)
M-(A dA)	(1.9173 0.0000)	(0.0000 18.0090)	M-(A dA)	(1.9136 0.0000)	(0.0000 20.3705)
M-(A auto1)	(4.3603 0.0000)	(0.0000 6.4480)	M-(A auto1)	(4.5771 0.0000)	(0.0000 6.5978)
M+(A auto2)	(1.9192 0.0000)	(0.0000 -6.4480)	M+(A auto2)	(1.9188 0.0000)	(0.0000 -6.5978)
M+(B dA)	(1.9239 0.0000)	(0.0000 -1.5053)	M+(B dA)	(1.9314 0.0000)	(0.0000 -1.1418)
M+(B auto1)	(-4.3603 0.0000)	(0.0000 2.6920)	M+(B auto1)	(-4.5771 0.0000)	(0.0000 2.7047)
M-(B auto2)	(1.9192 0.0000)	(0.0000 2.6920)	M-(B auto2)	(1.9188 0.0000)	(0.0000 2.7047)
M-(dA dB)	(-0.0004 0.0000)	(0.0000 -0.0009)	M-(dA dB)	(0.0000 0.0000)	
M+(dA auto1)			M+(dA auto1)		
M-(dA auto2)	(-0.0162 0.0000)	(0.0000 -0.0067)	M-(dA auto2)	(-0.0355 0.0000)	(0.0000 -0.0178)
M-(dB auto1)			M-(dB auto1)		
M+(dB auto2)	(0.0386 0.0000)	(0.0000 -0.0066)	M+(dB auto2)	(0.0865 0.0000)	(0.0000 -0.0178)
M-(auto1 auto2)	(3.8384 0.0000)	(0.0000 8.7207)	M-(auto1 auto2)	(3.8376 0.0000)	(0.0000 9.1542)
M+(A dB)	(1.9173 0.0000)	(0.0000 -33.6867)	M+(A dB)	(1.9136 0.0000)	(0.0000 -49.7620)
M-(B dB)	(1.9238 0.0000)	(0.0000 3.7529)	M-(B dB)	(1.9314 0.0000)	(0.0000 2.7876)

Table 4: Discoveries in the last two type of experiments

tification: deriving structure from behavior. *Artificial Intelligence* 83(1):75–141.

Todorovski, L.; Džeroski, S.; Srinivasan, A.; Whiteley, J.; and Gavaghan, D. 2000. Discovering the structure of partial differential equations from example behavior. In *Proceedings of the 17th International Conf. on Machine Learning*, 991–998. Morgan Kaufmann, San Francisco, CA.

Travé-Massuyès, L.; Ironi, L.; and Dague, P. 2004. Mathematical foundations of qualitative reasoning. *AI Magazine* 24(4):91–106.