# Qualitatively Faithful Quantitative Prediction

Dorian Šuc [a,*], Daniel Vladušič [a], Ivan Bratko [a]

[a] *Faculty of Computer and Information Science, University of Ljubljana*
*Tržaška 25, 1000 Ljubljana, Slovenia*

**Abstract**

We describe an approach to machine learning from numerical data that combines both qualitative and numerical learning. This approach is carried out in two stages: (1) induction of a qualitative model from numerical examples of the behaviour of a physical system, and (2) induction of a numerical regression function that both respects the qualitative constraints and fits the training data numerically. We call this approach $Q^2$ learning, which stands for Qualitatively faithful Quantitative learning. Induced numerical models are "qualitatively faithful" in the sense that they respect qualitative trends in the learning data. Advantages of $Q^2$ learning are that the induced qualitative model enables a (possibly causal) explanation of relations among the variables in the modelled system, and that numerical predictions are guaranteed to be qualitatively consistent with the qualitative model which alleviates the interpretation of the predictions. Moreover, as we show experimentally the qualitative models guidance of the *quantitative* modelling process leads to predictions that may be considerably more accurate than those obtained by state-of-the-art numerical learning methods. The experiments include an application of $Q^2$ learning to the identification of a car wheel suspension system – a complex, industrially relevant mechanical system.

*Key words:* Automated model building, System identification, Machine learning, Qualitative reasoning, Learning qualitative models, Numerical regression, Inductive learning

* Corresponding author.
*Email addresses:* dorian.suc@fri.uni-lj.si (Dorian Šuc), daniel.vladusic@fri.uni-lj.si (Daniel Vladušič), ivan.bratko@fri.uni-lj.si (Ivan Bratko).

## 1 Introduction

It is generally accepted that qualitative models are easier to understand and reason about than quantitative models. Qualitative models thus provide a better basis for the explanation of phenomena in a modelled system than numerical models. In this paper we describe an approach to automated modelling which combines the learning of qualitative and quantitative models from behaviour data of the modelled system. In our approach, the quantitative learning is constrained by the induced qualitative model so that the resulting quantitative model is in a sense guaranteed to be qualitatively consistent with the learning data. We call this approach $Q^2$ learning, which stands for Qualitatively faithful Quantitative learning.

$Q^2$ learning has the following advantages, some expected and some possibly less expected. The induced qualitative model enables nice causal interpretation of the relations among the variables in the system. This is precisely as one would expect from a qualitative model. More surprisingly, however, we also show by experiments with $Q^2$ learning that the qualitative model guides the *quantitative* modelling process in such a way that usually the resulting numerical predictions are considerably more accurate than those provided by state-of-the-art numerical modelling methods.

Thus the main message of this paper is that a combination of methods for qualitative and quantitative system identification has good chances to attain significant improvements over numerical system identification techniques, including techniques of numerical machine learning, such as regression trees [5] and model trees [18], or locally weighted regression [1]. The potential improvements are in two respects: first, the predictions are consistent with the qualitative properties of the modelled system, and in addition they are also numerically more accurate. The qualitative consistency is particularly important for the expert interpretation of induced models. So a domain experts explanation of how the system works is not obscured by qualitative inconsistencies in numerical predictions.

The idea of combining qualitative and quantitative machine learning for system identification is in this paper carried out in two stages. First, induce a qualitative model from system's numerical behaviour data (training data) with program QUIN (overviewed in Section 3). Second, induce a numerical regression function that both respects the qualitative constraints and fits well the training data numerically. We call this second stage the Qualitative to Quantitative transformation, or Q2Q for short. Our Q2Q method is described in Section 4.

To underline the importance of qualitative fidelity, we illustrate in Section 2

some problems that numerical learners typically have in respect of qualitative consistency. Sections 3 and 4 describe program QUIN and the Q2Q transformation. Section 5 presents $Q^2$ learning in the cannonball flight domain and further illustrates qualitative difficulties of numerical learners. In Section 6 we present a case study in applying $Q^2$ learning to the problem of modelling a car wheel suspension system. This is a complex mechanical system, and its modelling has good practical relevance in car design industry. Section 7 contains discussion and comments on related work, including approaches to learning qualitative models that may enable alternative approaches to the realization of $Q^2$ learning.

## 2 Qualitative difficulties of numerical learning

Consider a simple container of cylindrical shape and a drain at the bottom. If we fill the container with water, the water drains out. Water level monotonically decreases, until it reaches zero. Suppose we fill the container with water, and measure initial outflow $\Phi_0 = \Phi(t_0)$ and the time behaviour of water level $h(t)$. Since this is a rather simple behaviour, one would naturally expect that standard techniques of numerical machine learning should be able to fairly accurately predict time behaviour of water level if enough learning examples are given. Quite surprisingly, even in such simple cases, the usual numerical predictors can give strange and qualitatively unacceptable predictions. We illustrate the problems with a simple experiment, using well-known techniques for numerical prediction: model trees and locally weighted regression.

In our experiment we used container outflow simulation data to evaluate how different numerical predictors learn the time behaviour of water level. The outflow from a container is $\Phi(t)=c\sqrt{h(t)}$, where $c$ is a parameter depending on the area of the drain. For simulation we used Euler integration $h(t+\Delta t) = h(t)$ - $\Phi(t)\Delta t$, where $\Delta t$=0.1 seconds and $c$=1.25. We used six example sets, generated with different initial water levels and initial outflows $\Phi_0$=5 + $ic$, $i$=1, .., 6. Each example set had 20 examples ($t$, $h(t)$, $\Phi_0$) corresponding to 19 seconds of simulation.

We used the Weka [30] implementations of locally weighted regression [1] (LWR, for short), and M5 regression and model trees [18] to learn the time behaviour of level $h$ given the initial outflow, i.e. $h(t)=f(t, \Phi_0)$. We carried out a 6-fold cross validation. Each time one example set was used as a test set, and the other five sets ($5 \times 20 = 100$ examples) for learning. As an example consider the case when the test set was the set with $\Phi_0$=7.5. In this case M5 with the default parameters induced a model tree with 9 leaves. Figure 1 shows the learning examples and the M5 prediction of level $h(t)$ on the test set. Note that M5 predicts that water level *increases* between time points $t$=9 and

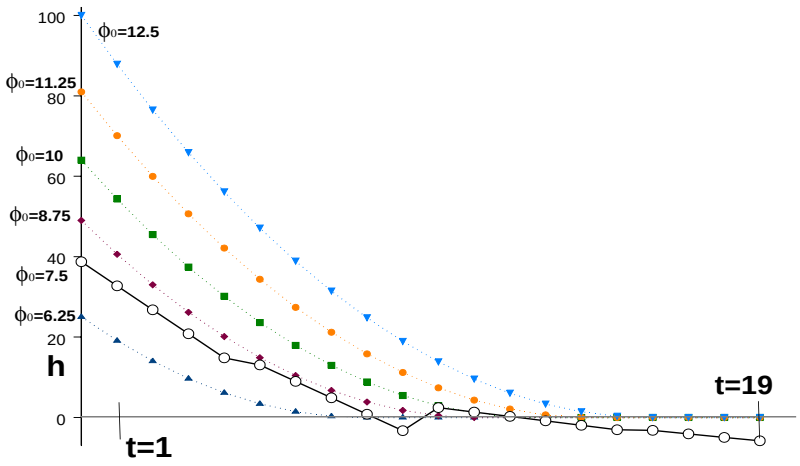



Fig. 1. M5 predictions of water outflow: empty circles are M5 predictions of level $h(t)$ on the test set with $\Phi_0 = 7.5$. Other symbols indicate the learning examples. Note that M5 predicts that water level *increases* between time points 9 and 10.

$t$=10, and that the water level is below zero at the end. The same happens if we change the pruning parameter of M5. These predictions are of course qualitatively unacceptable as water level can never increase. Also, there are no learning examples where water level increases.

One might think that this is an isolated weird case or or a bug in M5. But it is not. LWR makes a similar qualitative error on the test set with $\Phi_0 = 11.25$, when it predicts that water level increases at $t$=5. Of course, LWR predictions depend on the parameters of the LWR method, i.e. the number of nearest neighbors and the weighting function. However, often the appropriate LWR parameter setting that does not produce qualitative errors on one container, results in qualitative errors when learning from similar data but with different area of the drain. Often, the errors are even more obvious if we make predictions at the edges of the space covered by the learning examples, i.e. using as test set $\Phi_0 = 6.25$ or $\Phi_0 = 12.5$. Regression trees make similar qualitative errors.

Another example where numerical predictors make similar qualitative errors is described in Section 5.2. Based on this and other experiments, we believe that other numerical predictors also make qualitative errors, at least in more complex domains. This might be quite acceptable in applications where we just want to minimize *numerical* prediction errors. But often it is also important to respect qualitative relations that are either given in advance or hidden in the data. Ignoring them results in clearly unrealistic predictions that a domain expert would find disturbing and impossible to explain. The idea of this paper is to use qualitative constraints, either given or induced from data, to avoid qualitatively incorrect predictions. As it turns out, such constraints typically improve the accuracy of numerical prediction as well.

## 3 Qualitative learning with QUIN

QUIN (Qualitative Induction) is a learning program that looks for qualitative patterns in numerical data. QUIN expresses such qualitative patterns by *qualitative trees*. In this section we give a brief introduction to QUIN, sufficient for the understanding of the rest of the paper. More detail can be found in [27–29].

Induction of qualitative trees is similar to the induction of decision trees. The difference is that in decision trees, the leaves are labelled with values of the dependant variable, whereas in qualitative trees the leaves are labelled with what we call *qualitatively constrained functions* that define qualitative constraints on the dependant variable. The dependant variable is also called class and the other variables are called attributes.

### *3.1 Qualitatively constrained functions*

Qualitatively constrained functions (QCFs for short) are a kind of monotonicity constraints that are widely used in the field of qualitative reasoning [15,11,17,12,16]. QCFs are also a generalization of the qualitative constraint $M^+$, as used in QSIM [17].

A *qualitatively constrained function* $M^{s_1,\ldots,s_m}$ where $s_i \in \{+,-\}$, stands for an arbitrary function of $m$ continuous attributes that respects the qualitative constraints given by signs $s_i$. The qualitative constraint given by sign $s_i = +$ ($s_i = -$) requires that the function is strictly increasing (decreasing) in its dependence on the $i$-th attribute. We say that the function is *positively related* (negatively related) to the $i$-th attribute. $M^{s_1,\ldots,s_m}$ represents any function which is, for all $i = 1,\ldots,m$ positively (negatively) related to the $i$-th attribute, if $s_i = +$ ($s_i = -$).

A simple example of QCF is $M^+(X)$ that stands for an arbitrary monotonically increasing function of $X$. In general, QCFs can have more than one argument. For example, $M^{+,-}(X,Y)$ stands for a function that monotonically increases in $X$ and monotonically decreases in $Y$. If $Z{=}M^{+,-}(X,Y)$ and both $X$ and $Y$ increase, then according to this constraint, $Z$ may increase, decrease or stay unchanged. In such a case, a QCF cannot make an unambiguous prediction of the qualitative change in $Z$.

QCFs are similar to $\propto_Q$-all predicate suggested by Forbus [11] and multivariate monotonic function constraints by Wellman [31].

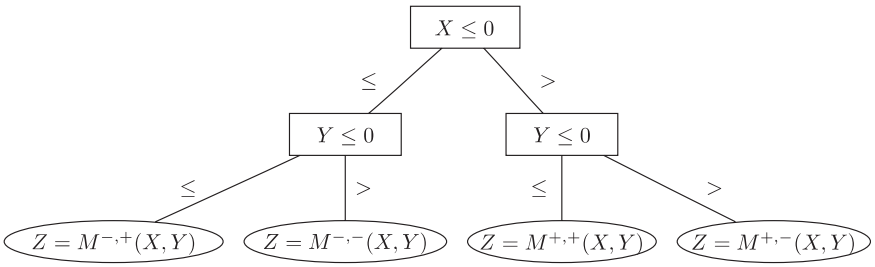



Fig. 2. A qualitative tree induced from a set of examples for the function $Z = X^2 - Y^2$. The rightmost leaf, applying when attributes $X$ and $Y$ are positive, says that $Z$ is strictly increasing in its dependence on $X$ and strictly decreasing in its dependence on $Y$.

### *3.2 Learning qualitative trees with QUIN*

QUIN takes as input a set of numerical examples and looks for qualitative patterns in the data. More precisely, QUIN looks for regions in the data space where monotonicity constraints hold. Such a set of qualitative patterns are represented in terms of a qualitative tree. As in decision trees, the internal nodes in a qualitative tree specify conditions that split the attribute space into subspaces. In a qualitative tree, however, each leaf specifies a QCF that holds among the input data that fall into that leaf. For example, consider a set of data points ($X$,$Y$,$Z$) where $Z = X^2 - Y^2$ possibly with some noise added. When QUIN is asked to find in these data qualitative constraints on $Z$ as a function of $X$ and $Y$, QUIN generates the qualitative tree shown in Figure 2. This tree partitions the data space into four regions that correspond to the four leaves of the tree. A different QCF applies in each of the leaves. The tree describes how $Z$ qualitatively depends on $X$ and $Y$.

QUIN constructs a tree in the top-down greedy fashion, similarly to decision tree induction algorithms. At each internal node of the tree, QUIN considers all possible splits, that is conditions of the form $X \leq T$ for all the attribute variables $X$ and effectively all possible thresholds $T$ with respect to $X$. Each such condition partitions the training data into two subsets. QUIN finds the "best" QCF for each subset according to an error-cost measure for QCFs. Then the best split is selected according to the minimum description length principle [21], which minimizes an error-cost of QCFs. The error-cost of a QCF with respect to an example set S is defined so that it takes into account the encoding complexity of the QCF, the consistency of the QCF with S, and the "ambiguity" of the QCF with respect to the data in S (the more unambiguous qualitative predictions the QCF can make in S the better).

A more elaborate description of the QUIN algorithm and its evaluation on a set of artificial domains is given elsewhere [27–29]. Empirical results show that QUIN can handle noisy data and, at least on simple domains, produces quali-

tative trees that correspond to the human intuition. In [27,24] QUIN has been used for qualitative reconstruction of human control strategies in dynamic domains such as controlling a crane or riding a bicycle. In both domains some surprising and non-trivial aspects of human control skill have been induced. In [25] QUIN was used to reverse engineer a complex industrial controller for gantry cranes. Experimental comparison with induction of regression and model trees show advantages of the qualitative approach both in terms of explanation of the studied artefact, and in terms of the control performance of the induced model of the artefact.

## 4 Q2Q transformation

In this section we describe one approach to the qualitative-to-quantitative transformation (Q2Q for short). Given a set of numerical data and a qualitative tree, Q2Q attempts to find a regression function that fits the data well numerically, and also respects the qualitative constraints in the tree. We say that such a regression function is *qualitatively consistent* with the tree. In fact, Q2Q finds a qualitatively consistent regression function with good fit separately for each leaf in the tree. The overall regression function is then obtained by gluing together the regression functions for the leaves.

### *4.1 Reifying QCFs*

Q2Q transforms the QCF in each leaf into a numerical function that respects the qualitative constraints imposed by the QCF in the leaf. We also call such transformation *reifying*, and the resulting regression function is called *reification of a QCF*. Of course, there are infinitely many possible reifications of a QCF, but we seek for reifications that fit numerical data well. For simplicity we consider only piece-wise linear reifications. This allows to learn numerical function values at a number of equidistant points in the attribute space and use linear interpolation to predict function values at other points. This procedure is described in the following section.

An interesting question is which attributes should be used in a QCF reification. Namely, a QCF usually mentions only a subset of the attributes that appear in the learning data. Generally, QCFs only mention those attributes that suffice to explain qualitative changes of the class variable. When reifying a QCF, a reification of the QCF should obviously be a regression function of all the attributes that appear in the QCF. But, as we want to optimize the numerical fit of the regression function with the data, it might appear that the inclusion of other attributes could improve the fit with the data. Our Q2Q procedure,

explained later in this section, however only produces reifications that include the attributes that appear in the QCF. This is justified by the somewhat surprising *continuous reifications theorem:* considering continuous regression functions only, the only possible reifications of a QCF are functions of only those attributes that appear in the QCF. The sketch of the proof is given below. Our choice to limit reifications to continuous functions only is justified by the fact that all our application domains were continuous.

For the sketch of the *continuous reifications theorem* we will consider the QCF $z = M^+(x)$. We will show that, even if there is another variable $y$ in the modelled system, continuous reifications of the QCF can only be functions of $x$, and not $y$ as well. We will show that reifications of the form $z = f(x, y)$, where function $f$ is continuous in $x$, are not possible. We show that such functions cannot satisfy the monotonicity constraint.

For $f$ to really be a function of both $x$ and $y$, there must exist some values $x_1$ of $x$ and $y_1$, $y_2$ of $y$ such that: $f(x_1, y_2) \neq f(x_1, y_1)$. That is: $f(x_1, y_2) = f(x_1, y_1) + d, d \neq 0$.

The monotonicity constraint $z = M^+(x)$ requires that: $\forall x_1, x_2, y_1, y_2 : x_1 < x_2 \Rightarrow f(x_1, y_2) < f(x_2, y_1)$. Let $x_2 = x_1 + \epsilon$, $\epsilon > 0$, and let us choose $y_1$ and $y_2$ so that: $f(x_1, y_2) = f(x_1, y_1) + \mathrm{d}$, $\mathrm{d} > 0$. The monotonicity constraint requires: $\forall \epsilon > 0 : f(x_1, y_1) + d < f(x_1 + \epsilon, y_1)$. Since $f$ is continuous in $x$, $f(x_1 + \epsilon, y_1) = f(x_1, y_1) + \delta$ where $\delta$ is a function of $\epsilon$ such that if $\epsilon$ approaches 0 then $\delta$ also approaches 0. The monotonicity constraint can thus be written as: $\forall \epsilon > 0 : f(x_1, y_1) + d < f(x_1, y_1) + \delta$. We had $d > 0$, so: $\forall \delta : 0 < d < \delta$. This is obviously false because $\delta$ can be made arbitrarily small by choosing $\epsilon$ sufficiently small.

### *4.2 Q2Q transformation procedure*

The Q2Q procedure is as follows. First, we partition the learning examples according to the leaves of the qualitative tree. These subsets are then used for learning qualitatively consistent functions in the corresponding leaves. Let us focus on learning a qualitatively consistent function for some particular leaf. Suppose we have a leaf with the qualitative constraint $y{=}M^+(x)$. We then have to find a function that is monotonically increasing in $x$ and fits the data well. One straightforward solution, used in Q2Q, is to divide the range of variable $x$ with a number of equidistant points (i.e. $\{x_1, x_2, ..., x_n\}$) in which we learn from the given data the function values $y$. The points $x_1$ and $x_n$ coincide with the $x$-wise extreme learning data points that fall in the leaf. The result is a set of pairs $\{(x_1, y_1), ..., (x_n, y_n)\}$ that defines a piece-wise linear function which can be easily checked for compliance with the given qualitative

constraint. It is only necessary to check if the following constraint is satisfied: $\forall i \in \{1, n-1\} : x_i < x_{i+1} \Rightarrow y_i < y_{i+1}$. If this constraint is satisfied then we have an acceptable regression function. If the constraint is not satisfied then we retry the learning of the $y$-values by modifying the parameters of the learning method, etc. This procedure can be generalized to qualitative constraints of any dimension.

In our implementation the function values $y_i$ are determined with a standard version of locally weighted regression (LWR) [1], which uses Gaussian weighting function. Thus the transformation has two parameters: the number of equidistant points per dimension $N$ and the kernel size of the Gaussian weighting function $K$. In the experiments in this paper we limited the choice of $N$ to $N \in \{3, 4, 5, 6\}$ and $K$ to $K \in \{0.3, 0.4, .., 0.8\}$. The choice of these candidate sets for $N$ and $K$ is discussed in Section 4.4. All possible combinations of these two parameters ($4 \times 6 = 24$) define the space of all candidate piece-wise linear functions for each leaf that is exhaustively searched by Q2Q to find functions that satisfy given qualitative constraints. For each leaf, Q2Q selects among these qualitatively consistent piece-wise linear functions one that has the best fit with the data that fall into this leaf. In order to avoid over-fitting and to determine the quality of the candidate functions, we used internal 4-fold cross-validation for each pair of $N$ and $K$ in every leaf. The cumulative error (defined as the root mean squared error or *RMSE* for short) over all 4 test folds defines the quality of the particular pair of parameters.

This regression method may be viewed as LWR guided by the constraints imposed by a qualitative tree. These constraints enforce an appropriate degree of fit with the data, and the relative importance of near neighbours vs. distant training data points. This aims at preventing overfitting or underfitting. An allowed degree of fit is such that the resulting regression function preserves the qualitative features in the original data.

Theoretically, this method has a problem with the guarantee of actually finding a solution. It is theoretically possible that none of the candidate regression functions is qualitatively consistent with the QCF in the leaf. This is discussed in the following section.

### *4.3 Qualitative consistency of Q2Q regression functions*

In general, the regression procedure above is not guaranteed to find a qualitatively consistent regression function. There is no explicit mechanism in this procedure to guarantee the existence of a qualitatively correct solution in all situations under conditions specified above. Of course, if a solution is found, it is guaranteed to be qualitatively consistent with the given qualitative tree.

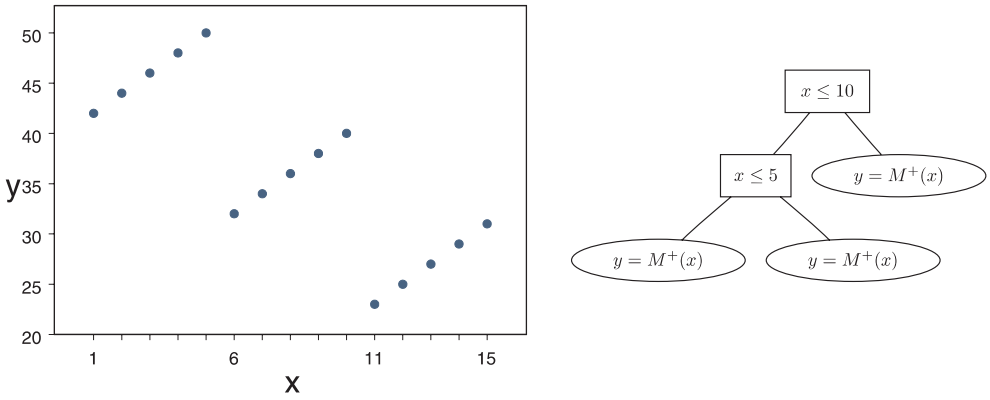



Fig. 3. Data and the corresponding QUIN's tree. Although the consistency of the induced qualitative tree with the data is perfect, we can also observe the globally dominant decreasing trend in the function value. This observation gives us two possible qualitative explanations: the induced qualitative tree (shown on the right) and a hand-crafted one. The latter is based on our observation of the overall decreasing trend, and consists of a single leaf with the QCF $y = M^-(x)$.

In our experiments, however, a situation in which this procedure would fail to find a qualitatively consistent regression function never occurred. To assess the robustness, we also tried to synthetically construct critical test situations in which our Q2Q procedure would fail. In one such test we sampled the learning data from three straight lines which have a special property that was intended to confuse the Q2Q procedure: all three lines are locally increasing, but the overall trend of the data is decreasing. The data and the corresponding qualitative tree induced by QUIN are shown in Figure 3. As QUIN tries to find locally monotonic trends it correctly identifies the splits between the three lines and marks all the functions in the leaves as monotonically increasing. Finding qualitatively consistent functions was in this case trivial for our Q2Q method. We now tried the more difficult case: we discarded QUIN's qualitative tree and replaced it by the overall decreasing trend in the data. This decreasing trend, coded as a qualitative tree, consists of a single leaf with the QCF $y = M^-(x)$. Using this qualitative constraint and all the 15 data points with our Q2Q method, also results in a qualitatively consistent regression function. Regression functions obtained in both experiments are presented in Figure 4.

This experiment illustrates how qualitatively consistent regression functions in the leaves of a tree can be successfully found by the presented Q2Q method for different qualitative interpretations of the same data. In the foregoing example we presented two possible qualitative explanations for the same data set. Different explanations may also be caused by noise in data – a situation which is quite common in real-world modelling. In the first explanation the qualitative tree was fully consistent with the underlying data, which made the task of finding qualitatively consistent regression functions in the leaves trivial. On the other hand, the hand-crafted global qualitative constraint was not fully consistent with the underlying data. In this case LWR which is used for the

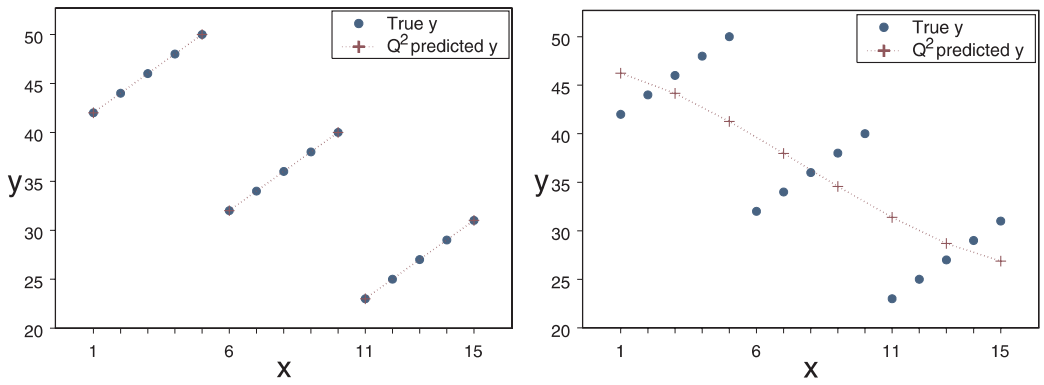



Fig. 4. The graphs present the data and the Q2Q-learned regression functions based on two different qualitative explanations of the data. Left, the case with a three leaf qualitative tree; right, the case with a single leaf qualitative tree saying $y = M^{-}(x)$.

learning of the regression function values at the points on the underlying grid has to overcome the conflict between the global decreasing constraint and the locally increasing trends in the data. Nevertheless the search through the space of possible LWR parameters yielded qualitatively consistent regression function.

It is instructive to study how the changes in $K$ and $N$ effect the qualitative behaviour of LWR functions. In the first case – the qualitative tree with three leaves, all combinations of $N$ and $K$ produce qualitatively consistent regression functions. This result is expected, as the data in all three leaves are completely qualitatively consistent with the corresponding QCFs. In the second case, where we only have one (global) qualitative constraint, we noticed three regions in the space of $N$ and $K$ regarding qualitative correctness of regression functions. In the first region, where $K$ and $N$ are small, LWR constructs qualitatively correct reifications. In the second region, where $K$ is still small and $N$ is larger, the algorithm does not produce any qualitatively correct reifications. In the third region, where $K$ is larger, reifications are qualitatively consistent for all $N$.

Explanation for such behaviour of Q2Q is as follows. In the first region where $K$ and $N$ both small, we only have one regression point for each of the three locally increasing sections of the learning data. Hence the constructed reifications are qualitatively correct (the overall trend is decreasing). In the second region, where $N$ is larger, we have more regression points for each locally increasing section of the learning data. That is the reason for qualitatively incorrect reifications. In the third region, we are taking advantage of larger $K$ that enables global view of the data (decreasing), therefore reifications are qualitatively correct again.

This example and experience from other cases provide evidence that the presented Q2Q method, although simple, is suitable for most practical cases. In

spite of this, it is still theoretically possible that the described Q2Q procedure would fail to find a qualitatively consistent regression function. Although in our experiments this never happened, we also developed an alternative, more complex Q2Q method that is guaranteed to find a qualitatively consistent regression function [26].

### *4.4 Complexity of Q2Q procedure*

Let us consider the time complexity of reifying a QCF with $D$ attributes where values of parameters $N$ and $K$ are chosen from the sets $\mathcal{N}$ and $\mathcal{K}$ of possible values for $N$ and $K$ respectively. Let the time required to answer one LWR query be $t_{\mathrm{LWR}}$. Note that reifying a QCF involves a 4-fold cross-validation. So the time complexity of reifying one QCF is: $Q_D = 4{\cdot}|\mathcal{K}|{\cdot}\sum_{N \in mathcal N} t_{\mathrm{LWR}}{\cdot}N^D$. As we have no influence on the number of independent attributes $D$, the selection of plausible value sets for $K$ and $N$, and efficient implementation of the LWR algorithm become very important. We now discuss the choice of candidate sets $\mathcal{K}$ and $\mathcal{N}$ in our experiments.

The candidate value sets $\mathcal{K}$ and $\mathcal{N}$ were chosen so as to maximize the probability of constructing a qualitatively correct piece-wise linear regression function and to minimize the computation time. We did not include extreme values of $K$ in $\mathcal{K}$, for the following reasons. In the case of very small $K$, the probability of qualitatively correct QCF is rather small, as only extremely local trends in data are used for prediction. In the case of very high $K$, the numerical fit is likely to be unsatisfactory, as we are virtually simulating ordinary linear regression. The values in the set $\mathcal{N}$ were chosen in accordance with two constraints, namely, we wanted sufficient resolution of the piece-wise linear functions and acceptable computation time. Both constraints are addressed with the use of interval $\mathcal{N} = \{3, 4, 5, 6\}$, as the number of grid points increases exponentially with dimension of QCF. Hence, maximum number of grid points for QCF with one attribute is 6, with two attribute 36, three 216 and four 1296. Limiting the upper value to six is now evident, as 1296 is already in the order of a typical number of training examples. Increasing resolution even further would not improve numerical accuracy nor yield significant number of qualitatively correct piece-wise linear functions. It might be better to use different sets $\mathcal{N}$ for different dimensionalities of QCFs, but we have not explored this possibility.

## 5 An extended example: cannonball flight

As a detailed example, we here describe experiments with predicting the flight distance of a cannonball fired at some elevation angle and initial velocity. First,

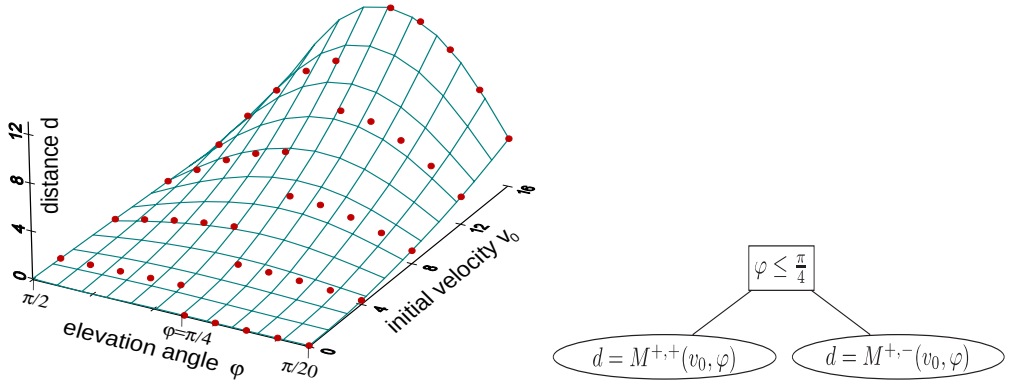



Fig. 5. Cannonball flight learning examples and induced qualitative tree. The dots in the left graph denote the learning examples. The figure on the right is the qualitative tree for the flight distance induced by QUIN from the learning examples.

we illustrate qualitative errors made by the usual numerical predictors. Then we describe experiments with the $Q^2$ method.

### *5.1 Cannonball flight domain*

Consider a cannonball fired at some elevation angle $\varphi$ and initial velocity $v_0$. We measure the flight distance $d$ of the cannonball. Of course, a greater initial velocity results in a greater flight distance. However, flight distance also depends on the elevation angle. The maximal flight distance is achieved by firing the cannonball at elevation angle $\varphi = \frac{\pi}{4}$. Qualitatively, flight distance monotonically increases with elevation angle if $\varphi \leq \frac{\pi}{4}$, and decreases with elevation angle if $\varphi > \frac{\pi}{4}$. In both cases, flight distance monotonically increases with initial velocity. One would expect that numerical predictions would respect these qualitative constraints when learning from noise-free examples.

Ignoring air resistance and assuming the start and the landing position of the cannonball are at the same ground level, the flight distance is given by $d = \frac{{v_0}^2}{g} \sin\varphi \cos\varphi$. The learning task is to learn flight distance, given the initial velocity and elevation angle, i.e. $d{=}f(v_0, \varphi)$. The learning examples are the points with $v_0 = 2j$, where $j$=0, 1, 2, .., 8 and $\varphi{=}\frac{\pi}{20}, \frac{2\pi}{20}, .., \frac{5\pi}{20}$ for $j$ even, and the points with $\varphi{=}\frac{6\pi}{20}, .., \frac{10\pi}{20}$ for $j$ odd. This gives 45 learning examples, described with attributes $v_0$ and $\varphi$ and class $d$. There was no noise in this data. These examples are illustrated in Figure 5. Numerical accuracy and qualitative correctness of the predictions were tested on various test example sets. Each test example set corresponds to one initial velocity $v_0$. A test set consists of the examples that have the same initial velocity, but different elevation angle: $\varphi{=}\frac{\pi}{36}, \frac{2\pi}{36}, .., \frac{18\pi}{36}$.

A nice qualitative description of the domain is given by the qualitative tree, in-

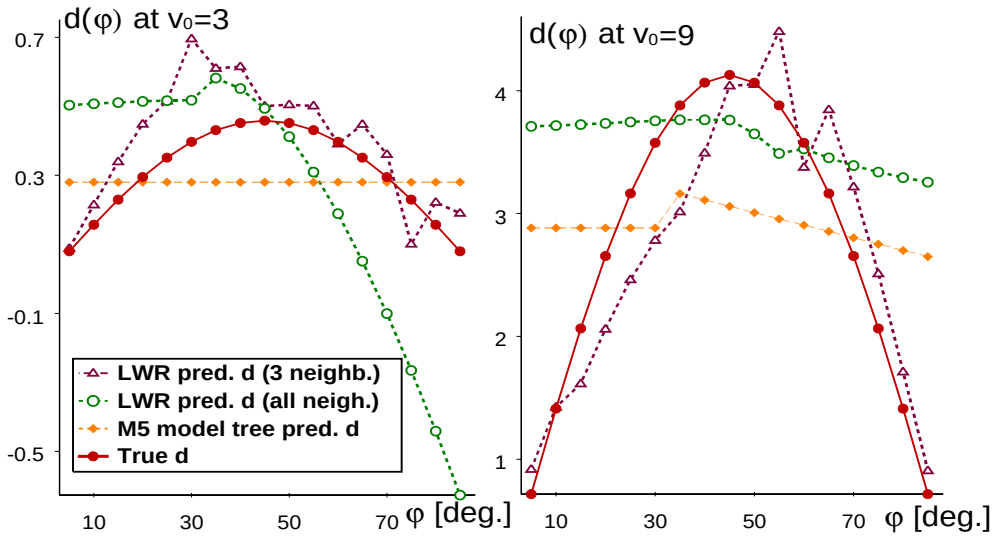



Fig. 6. Typical problematic predictions by M5 and LWR for cannonball flight distance. The left graph gives predictions at $v_0$=3 and the right one at $v_0$=9. The symbols denote M5 model tree predictions, LWR predictions using 3 and all nearest neighbors, and the true value of flight distance. LWR using all neighbors on test example set with $v_0$=3 correctly predicts that flight distance is first increasing and then decreasing in $\varphi$. However it does not correctly predict the point of the maximum, and later predicts negative distance. All other predictions are qualitatively wrong.

duced by QUIN from the described learning examples. This qualitative tree is given in Figure 5. The left leaf that applies when $\varphi \leq \frac{\pi}{4}$ has QCF $M^{+,+}(v_0, \varphi)$. This says that flight distance monotonically increases with initial velocity and with elevation angle. If $\varphi > \frac{\pi}{4}$, then the right leaf with QCF $M^{+,-}(v_0, \varphi)$ applies. Flight distance still monotonically increases with initial velocity, but decreases with elevation angle.

### *5.2 Qualitative difficulties of M5 and LWR*

We used the Weka implementations of LWR and M5 regression and model trees to learn flight distance, given the initial velocity and elevation angle, i.e. $d=f(v_0, \varphi)$. Numerical accuracy and qualitative correctness of the predictions was tested on various test sets. A general observation was that respecting the monotonicity in initial velocity was usually not a problem for model trees and LWR. However, none of the induced predictors respected the qualitative dependence on the elevation angle. These experiments are described in the following paragraphs.

By changing the pruning parameter, M5 induced three model trees with up to eight leaves. All of these model trees gave qualitatively wrong predictions. They predicted that flight distance is monotonically decreasing with elevation

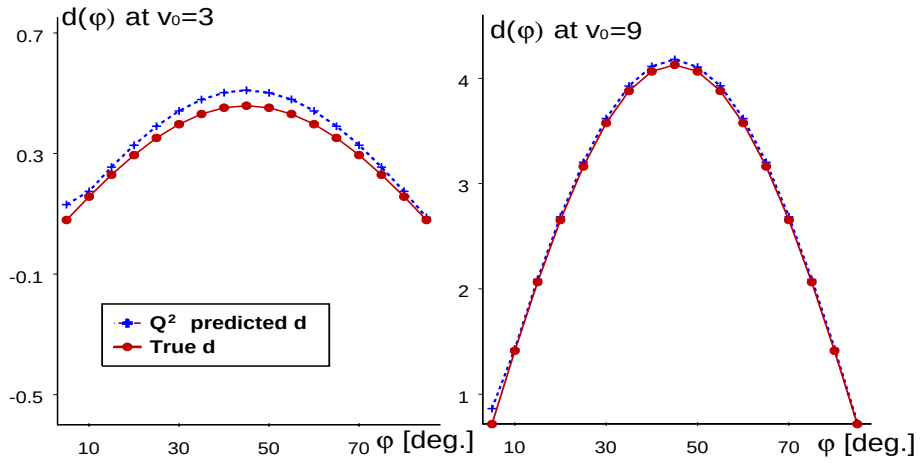



Fig. 7. $Q^2$ predictions for cannonball flight distance. The left graph gives predictions at $v_0$=3 and the right one at $v_0$=9. The predictions are qualitatively correct and also numerically more accurate than LWR and M5 predictions.

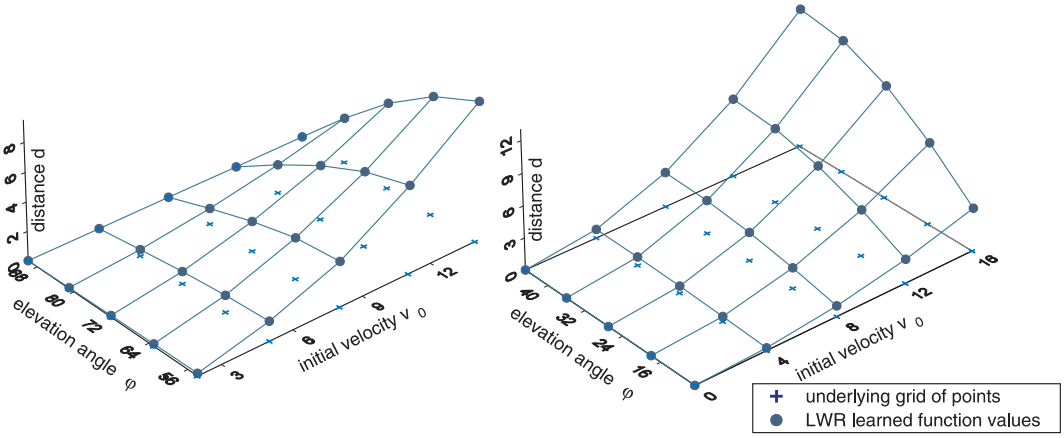



Fig. 8. QCF reifications for the Cannonball example using the qualitative tree of Figure 5. The left graph corresponds to the left leaf ($\varphi_0 > \frac{\pi}{4}$) and the right graph for the right leaf of the qualitative tree.

angle for initial velocity $v_0$=6. These model trees also make other qualitative errors with different initial velocities. This is illustrated in Figure 6 that shows predictions of a model tree with eight leaves on test example sets with $v_0$=3 and $v_0$=9. This model tree predicts that flight distance is constant at $v_0$=3 and has a local minimum near $\varphi = \pi/6$ at $v_0$=9. We also experimented with M5 regression trees. They give piece-wise constant predictions and make even more qualitative errors than model trees.

Predictions by LWR significantly depend on its parameters, i.e. the number of nearest neighbours and the weighting function. For this reason we used different settings of LWR parameters to make sure that the observed problems are not due to an unfortunate choice of LWR parameters. We used all three possible weighting functions, i.e. linear, inverse and Gaussian, and the number of nearest neighbours that was set to three, five, ten and all. This gives al-

Table 1
The comparison of $Q^2$ learning, LWR and M5 on four different test sets. The $Q^2$ learning is better than LWR and M5 not only in terms of qualitative correctness (see Figures 6 and 7), but also in terms of numerical accuracy. Note that the parameters of M5 and LWR were fitted to the particular test sets. In this way M5 and LWR were given an advantage over $Q^2$ that used only learning data to set its parameters.

| *RMSE* at | $Q^2$ | M5 | LWR |
|---|---|---|---|
| $v_0 = 1.0$ | 0.0267 | 0.0270 | 0.0741 |
| $v_0 = 3.0$ | 0.0385 | 0.1427 | 0.1036 |
| $v_0 = 6.0$ | 0.0020 | 0.6079 | 0.0602 |
| $v_0 = 9.0$ | 0.0539 | 0.8278 | 0.0645 |

together 12 different parameter settings. Given a particular initial velocity, it was usually possible to find a parameter setting that resulted in qualitatively correct prediction. However, the same LWR parameters did not give qualitatively correct predictions for different initial velocities. For example, none of the 12 parameter settings resulted in qualitatively correct prediction for initial velocities 1, 6.5 and 9. Figure 6 gives typical LWR predictions at $v_0$=3 and $v_0$=9. We also experimented with our implementation of LWR that optimizes the number of nearest neighbours at each prediction point separately, but it also did not produce qualitatively correct predictions.

### *5.3 $Q^2$ learning in the Cannonball domain*

We assume the qualitative tree of Figure 5. As described in Section 4, the learning data is first partitioned by Q2Q according to the qualitative tree. Then QCFs of the two leaves are reified using the data in the corresponding leaves to find qualitatively consistent regression functions for both leaves. These two QCF reifications are shown in Figure 8.

To compare numerical accuracy of $Q^2$ learning and Weka implementations of LWR and M5 we used different initial velocities ($v_0$=1, $v_0$=3, $v_0$=6 and $v_0$=9) that define four test sets. As described in previous section we performed an exhaustive search through a selection of LWR and M5 parameters to find the parameters which result in smallest *RMSE* on the particular test set. Note that in this way M5 and LWR were given an advantage over our Q2Q method. The best results of LWR and M5 are compared to the results obtained with $Q^2$ in Table 1. Even with fitting of the parameters to the particular test sets LWR and M5 perform inferior to $Q^2$ learning.

## 6 Learning about car wheel suspension

### *6.1 Intec wheel model*

In this section we present an application of $Q^2$ learning to the modelling of a car wheel suspension system. This is a complex mechanical system of industrial relevance. The model and simulation software used in this experiment were provided by Intec, a German car simulation company. The main role of the application in this paper is to provide a controlled experiment to assess the potentials of $Q^2$ learning on a modelling problem of industrial complexity. However, although the target model was already known and developing such a model was not an issue of practical relevance, this case study was nevertheless motivated by a practical objective. Namely, the complexity of Intec's model is so high that on the present simulation platform, the simulation cannot be run in real time. Therefore the practical objective of the application of $Q^2$ learning was to speed up the wheel simulation. The goal was thus to obtain a simplified wheel model that would still be sufficiently accurate and at the same time significantly simpler than the original model to allow real-time simulation. Indeed, the simplified model obtained with $Q^2$ is computationally trivial compared with the original model.

The Intec wheel model (shown in Figure 9) is a multi-body model of a front wheel suspension built in compliance with the physical model assuming no car-body movement and no wheel-spin. In fact, the suspension system is modelled as if the car-body was fixed. The flexible joints in the multi-body suspension system that links the wheel to the car-body allow displacements in several directions. The wheel position is given by $x$, $y$ and $z$ coordinates of the wheel center, and the rotation angles about axes $x$, $y$ and $z$. These are called camber $\beta$, enforced wheel-spin angle $\gamma$, and toe angle $\alpha$.

The multi-body simulation software Simpack [23] was used to set up the model and to generate simulation traces. During simulation, a number of forces and moments are acting upon the tyre: two horizontal forces $F_x$ and $F_y$, vertical movement (measured as elevation of the road $R$) and rotational moment $M_z$. For example, $F_x$ is acting upon the tyre when braking, $F_y$ when driving through corners (centripetal force) and rotational moment $M_z$ when parking the car.

### *6.2 Details of experiments*

During the simulation, input and output variables are logged to a file called simulation trace. We used traces of wheel simulation with different trajectories of input variables. Each trace lasted for 70 seconds, and was sampled with

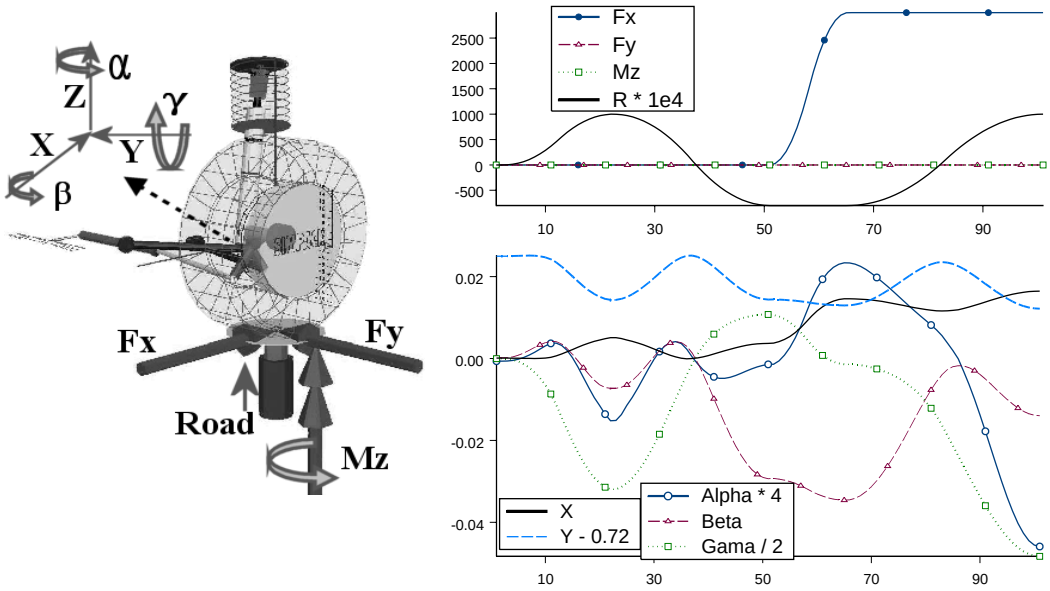



Fig. 9. Intec wheel model: the variables in the model (on the left) and a typical simulation trace (on the right). Wheel position is given by $x$, $y$ and $z$ coordinates of the wheel center, and rotation angles about axes $x$, $y$ and $z$. These are affected by horizontal forces $F_x$ and $F_y$, elevation of the road $R$ and rotational moment $M_z$ that act upon the tyre. The right figure shows a typical simulation trace. The input variables are on the upper graph. The output variables (except $z$ that changes the same as road $R$) are on the lower graph. On $x$-axis is time in steps $dt$=0.7 seconds. Note the complex behaviour of the output variables resulting from changes in $F_x$ and road $R$.

$dt$=0.7 seconds. In this way a trace gives 100 examples, each example contains 10 values, corresponding to the values of four input and six output variables at a given time. Figure 9 shows a typical simulation trace. It should be noted that all these traces correspond to very slow changes of input variables, and as a result the traces are illustrative mainly of the kinematics of the mechanism, but not also of its dynamics. Dynamic effects in these slow behaviours are negligible. Accordingly, the goal of learning in these experiments was a kinematics model of the system.

The experiments reported in this paper were done using a black-box approach. We did not use any knowledge of the model, and we did not have a direct access to the model. The simulation traces were provided by our partners from the Czech Technical University in the European project Clockwork (see Acknowledgement).

In all the experiments we used 7 traces for learning with the same road profile as in the trace of Figure 9. In the first learning trace all other three input variables were zero. In the next three traces two of the other three input variables were zero and one other variable ($F_x$,$F_y$ or $M_z$) was changing. Figure 9 shows one such trace. The remaining three traces were similar, but the trajectory of the changing variable was different, i.e. it first increased, stayed

unchanged for 20 seconds, and than slowly decreased to zero. Each trace gives 100 examples, giving altogether 700 learning examples with 10 variables.

The task was to learn each of the six output variables as a function of input variables. In this way we have six learning problems, where an output variable is the class and the input variables are the attributes. For example, angle $\alpha$ was learned as $\alpha{=}f(R, F_x, F_y, M_z)$.

The prediction accuracy was tested on 8 test traces, denoted by $T_1$, $T_2$,..,$T_8$. The first six traces traces have the same road profile as the traces used for learning, but different profiles of one or two other input variables ($F_x$, $F_y$ and $M_z$). Test trace $T_7$ has the same road profile as the learning traces and other three input variables change similar as $F_x$ in the trace in Figure 9. This trace was recommended as a critical test trace by the domain expert, who considered it far more difficult (all four input variables change) than the first six test traces where one or two input variables were always zero. At the final assessment of the developed models during our visit at Intec, a domain expert suggested the last and the most difficult test trace. In this trace, denoted by $T_8$, all of the input variables have a different and a more complex profile than the variables in the learning traces. The traces $T_7$ and $T_8$ are later also referred to as the critical test traces.

### *6.3 Inducing a qualitative wheel model with QUIN*

QUIN was used to induce a qualitative tree for each of the six output variables, where the input variables were the attributes. All of the induced qualitative trees had over 99 % consistency on the learning set of examples. We say that a QCF is consistent with a pair of example points if the QCF's qualitative prediction of the change in the dependent variable does not contradict the direction of change between the two example points. The level of consistency of a qualitative tree with the examples is the percentage of the examples with which the tree is consistent. Consistency of 99% indicates that the induced qualitative model fits the data nearly perfectly.

The simplest qualitative tree was induced for translation in the $z$-axis. This tree only has one leaf with QCF $z{=}M^+(R)$. This tree has a simple and obvious explanation. It says that $z$ changes in the direction of the road change. If road increases then $z$ increases, i.e. the wheel center moves upwards. None of the other variables has a significant effect on qualitative changes in $z$.

Qualitative trees for translations in $x$ and $y$ axes are a bit more complicated. Since they have similar explanations we will present just the qualitative tree for $x$ translations, given in Figure 10. Note that $x$ is measured in the opposite direction to usual, i.e. positive $x$ means wheel center moving in the direction

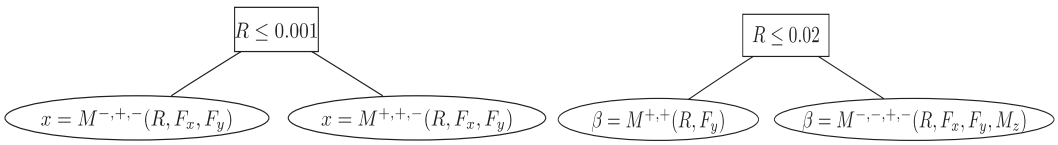



Fig. 10. Induced qualitative trees for $x$ position of the wheel center (left qualitative tree) and camber angle $\beta$, i.e. the rotation around $x$-axis (right qualitative tree).

of car driving backwards. Both leaves of the tree have the same qualitative dependence on $F_x$ and $F_y$, but differ in qualitative dependence on road $R$. The qualitative tree says that $x$ is positively related to force $F_x$ that acts in the direction of $x$. Obviously, the wheel center position $x$ changes (wheel moves backward or forward) in the direction of force in $x$ direction. Second, $x$ is negatively related to force $F_y$. This means that if we push the wheels together (we apply force in the $y$ direction), the wheels will move forward ($x$ decreasing). This is not so obvious, but can be understood if we consider the usual mechanics of wheel suspension. The qualitative dependence on road $R$ is a bit more complicated. The qualitative tree of Figure 10 says that $x$ is negatively related to $R$ when $R \leq 0.001$. Otherwise $x$ is positively related to $R$. Consider for example that road $R$ is negative and increasing. Since $x$ is negatively related to $R$ (the leftmost leaf applies) $x$ will decrease, i.e. wheel moving forward. Therefore, when $R$ increases from its minimum to its maximum, $x$ will first decrease and then increase, i.e. the wheel center will first move forward and than backward.

Rotations about axes $x$, $y$ and $z$ are measured by enforced wheel-spin $\gamma$, camber $\beta$ and toe angle $\alpha$, respectively. For enforced wheel-spin $\gamma$, QUIN induced a simple one-leaf tree that says $\gamma{=}M^{-,-,+}(R, F_x, F_y)$. Note that $\gamma$ changes in the direction of the tyre rotation when driving forward. Consider for example the dependence of $\gamma$ on force $F_x$ that is positive during braking. Since $\gamma$ is negatively related to $F_x$, increasing $F_x$ causes $\gamma$ to decrease, i.e. during braking enforced wheel spin angle changes in the direction of the tyre rotation.

For camber angle $\beta$ QUIN induced the qualitative tree given in the right hand side of Figure 10. Similar to qualitative trees for $x$ and $y$ translations, the dependence of $\beta$ on road $R$ differs in the two leaves. When $R \leq 0.02$, $\beta$ is positively related to road $R$ and negatively related to $R$ otherwise. In both leaves $\beta$ is positively related to $F_y$. As we would expect, positive force from the outside of the tyre causes positive camber.

The toe angle $\alpha$, i.e. the rotation about $z$-axis is effected by all input variables and is the most complicated. The induced tree is given in Figure 11. We will omit explanation of this qualitative tree since it requires complex understanding of the flexible nature of the multi-body suspension system that links the wheel to the car-body.

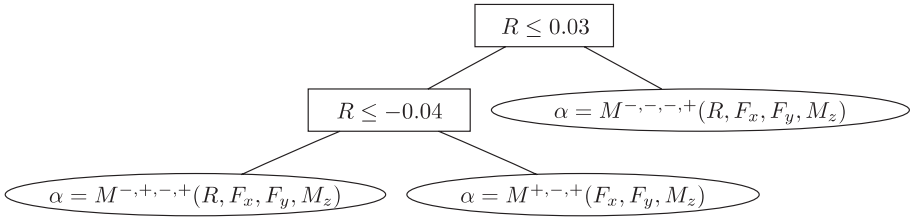



Fig. 11. Induced qualitative tree for toe angle $\alpha$, i.e. the rotation around $z$-axis.

Overall, as judged by our domain expert, these qualitative trees give a good explanation of the wheel suspension system behaviour. They explain the qualitative relations between input and output variables. Moreover, they provide a qualitative model of wheel suspension system that enables qualitative simulation. In this way, they enable to predict all possible qualitative changes of output variables over an arbitrary time interval given qualitative changes of all or some input variables. Consider for example that the car is driving over a bump, that is road $R$ is first increasing and then decreasing. For simplicity, let us assume that during that period all other input variables are unchanged and we are only interested in $\beta$. The qualitative tree for $\beta$ (see Figure 10) gives the following possible behaviours of $\beta$. If $R > 0.02$ then $\beta$ is decreasing until the road is increasing, and is increasing afterwards, when road is decreasing. If $R \leq 0.02$ and the top of the bump does not exceed $R = 0.02$ then $\beta$ is first increasing and decreasing afterwards. If $R \leq 0.02$ and the top of the bump exceeds $R = 0.02$ then $\beta$ is first increasing, then decreasing after $R > 0.02$ as long as the road is increasing. Afterwards, when road is decreasing $\beta$ is first increasing, and decreasing after $R \leq 0.02$.

This kind of qualitative simulation enables to predict all possible qualitative changes of output variables in time given qualitative changes of all or some input variables. Such qualitative behaviour in time is valid for all possible numerical changes of input variables. Numerical simulation, on the other hand, gives precise numerical results, but these are valid just for the one given profile of the input variables in time.

Besides explanation and qualitative simulation, the induced qualitative model enables to improve numerical predictions, as described in the following subsections.

### *6.4 Qualitative correctness of numerical predictors*

In this section we compare qualitative correctness of $Q^2$ predictions with the predictions of other typical numerical learners in wheel suspension modelling. These numerical learners are: the Weka implementation of the M5 model trees and two versions of LWR. The first version is the Weka implementation of LWR with the default parameters. The second version of LWR uses a similar internal

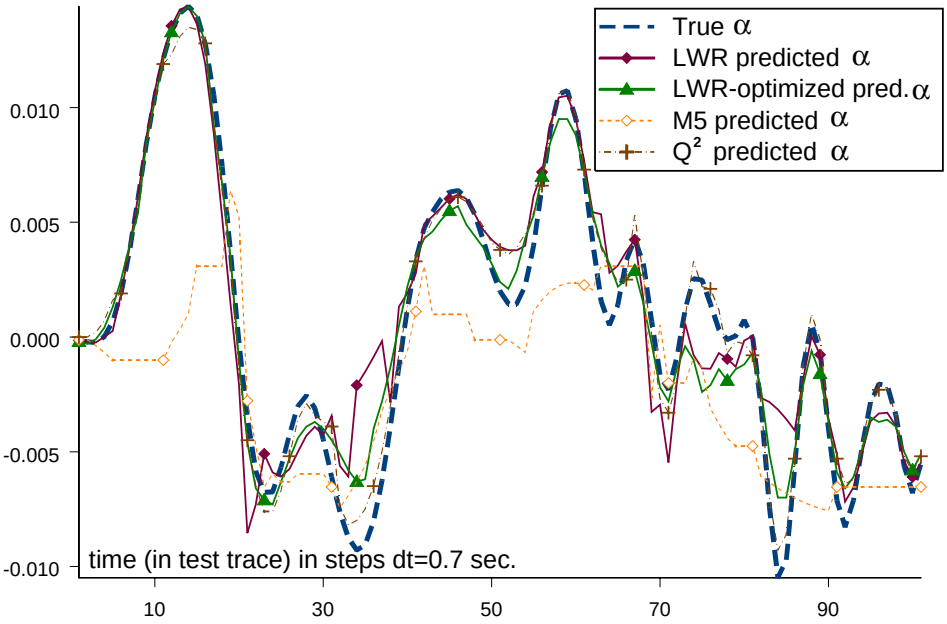
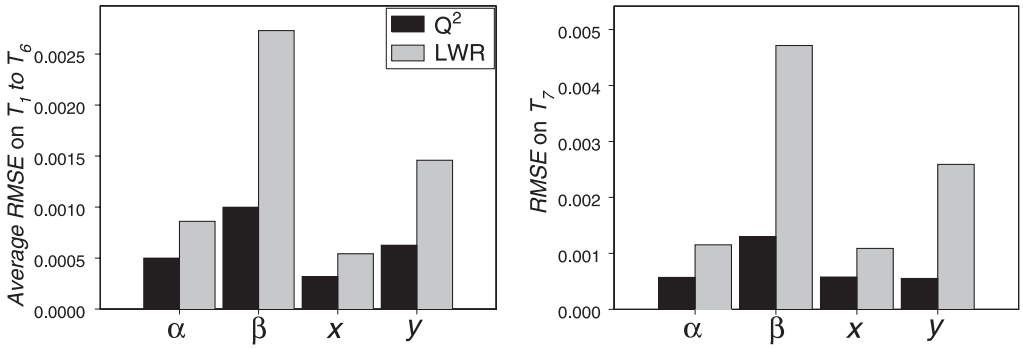



Fig. 12. LWR, LWR with optimized parameters, M5 and $Q^2$ predictions of $\alpha$ on the critical test trace $T_8$. With each method, $\alpha$ at time step $t_i$ was predicted according to the values of input variables at time $t_i$ in the test trace.

4-fold cross-validation as used by Q2Q and described in Section 4.2. This 4-fold cross-validation is used to set the value of LWR parameter $K$. Namely, at each point of prediction four nearest training points are considered and at each of the four neighbouring points the value of $K$ that minimizes *RMSE* is selected. The mean value of $K$, weighted according to the distance of the neighbouring point from the prediction point, is then used to predict the class value at the prediction point.

Figure 12 shows $\alpha$ predicted with M5 model tree, LWR and LWR with optimized parameters, on the critical test trace $T_8$, where all the input variables are changing simultaneously. The figure shows that both M5 and LWR sometimes make large errors. Moreover these errors are not only numerical, but also qualitative. Consider for example the M5 predictions at the beginning of the trace. Here the predicted $\alpha$ is decreasing, but the true $\alpha$ is *increasing*. This error could be avoided by considering the induced qualitative tree for $\alpha$ given in Figure 11. At the beginning of this test trace, road $R$ is near zero, $F_x$ and $M_z$ are increasing, and $F_y$ is decreasing. Since $R$ is near zero, the middle leaf of the qualitative tree applies. Its QCF $\alpha{=}M^{+,-,+}(F_x, F_y, M_z)$ requires increasing $\alpha$ since $F_x$ and $M_z$ are increasing, and $F_y$ is decreasing.

As can be observed in Figure 12, M5 and LWR often make qualitative errors. $Q^2$ predictions are qualitatively correct. The use of a qualitative model enables $Q^2$ to better generalize in the areas sparsely covered by the training examples, resulting in better numerical accuracy.




| | $\alpha$ | | $\beta$ | |
|---|---|---|---|---|
| | $Q^2$ | opt. LWR | $Q^2$ | opt. LWR |
| avg. *RMSE* ($T_1$-$T_6$) | 0.00050 | 0.00086 | 0.00100 | 0.00273 |
| *RMSE* $T_7$ | 0.00057 | 0.00115 | 0.00130 | 0.00418 |
| *RMSE* $T_8$ | 0.00094 | 0.00153 | 0.00120 | 0.00317 |
| | $x$ | | $y$ | |
| | $Q^2$ | opt. LWR | $Q^2$ | opt. LWR |
| avg. *RMSE* ($T_1$-$T_6$) | 0.00032 | 0.00054 | 0.00063 | 0.00146 |
| *RMSE* $T_7$ | 0.00058 | 0.00109 | 0.00055 | 0.00259 |
| *RMSE* $T_8$ | 0.00039 | 0.00066 | 0.00067 | 0.00179 |

Fig. 13. Comparing accuracy of $Q^2$ and LWR with optimized parameters: the graph on the left shows the average *RMSE* on test traces $T_1$ to $T_6$, the graph on the right shows the *RMSE* on the more difficult test trace $T_7$. These results are also given in the table below the graphs.

### *6.5 Numerical accuracy of the induced models*

Here we compare the numerical accuracy of LWR, M5 model trees, and $Q^2$ learning. All the methods learned from 7 learning traces (also used for the learning of qualitative trees) and were tested against 8 test traces described in Section 6.2. Each of the six output variables gives one learning problem and the accuracy was measured by *RMSE*. For each of the six learning problems, corresponding to six output variables, $Q^2$ used a corresponding qualitative tree induced by QUIN as described in Section 6.3. Q2Q transformation, described in Section 4, was used to transform qualitative trees into numerical regression functions.

When experimenting with M5, we noticed that it was grossly inferior both in terms of qualitative acceptability as well as numerical error. This is illustrated also in the previous section. Attempts at optimizing M5's parameters did not help noticeably. Weka implementation of LWR did a bit better. LWR with optimized parameters, described in the previous section, was the best

among standard numerical learners. For this reason, our presentation of experimental results largely concentrates on comparison between $Q^2$ and LWR with optimized parameters.

We give the results for variables $\alpha$, $\beta$, $x$ and $y$. The predictions of the remaining variables $\gamma$ and $z$ were not much affected by induced qualitative constraints and for this reason the improvements of $Q^2$ were smaller.

The test results are divided in two groups. The first group consists of results on simpler test traces $T_1$, .., $T_6$. The results on the critical test traces $T_7$ and $T_8$ are in the second and third group respectively. A domain expert considered these two traces far more difficult (all four input variables are changing simultaneously) than the six test traces in the first group. The average *RMSE* on the six simpler test traces are presented in the left graph of Figure 13. We can see that even our simple Q2Q method improves the numerical prediction for all the variables (compared to LWR). The improvements in numerical accuracy are even greater on the critical traces $T_7$ and $T_8$ that are less similar to the learning traces. The results on test trace $T_7$ are presented on the right graph in Figure 13. All the results are also given in the table bellow the graphs.

## 7 Discussion

### *7.1 Summary of method and experiments*

In this paper we introduced a new approach to machine learning in numerical domains, which we call $Q^2$ learning (qualitatively faithful quantitative learning). This combines the induction of qualitative properties from numerical data and numerical regression that respects the induced qualitative properties. With *continuous reifications theorem* and various experiments, $Q^2$ learning was studied empirically and theoretically. We showed by an experimental case study that $Q^2$ learning may lead to the following advantages compared to the usual numerical learning:

(1) Induced models tend to be *qualitatively* consistent with the data and therefore have better chances to correspond to the qualitative mechanisms in the domain of modelling. For example, if the amount of water in a container is decreasing, the level of water cannot be increasing. This is important with respect to the interpretation of induced models and explanation of phenomena based on these models.
(2) Qualitative consistency of induced models with learning data also affects the accuracy of the model's *numerical* predictions: numerical accuracy may be considerably improved. This was illustrated by the experimental

results.

$Q^2$ learning does not make any assumptions regarding the linearity of the modelled system. Thus it is appropriate for the identification of non-linear systems. QUIN takes into account only the ordering of the values of the variables, and not their magnitudes. For this reason it can be expected to be more robust against noise in the data, and not sensitive to transformations of the data that preserve the order of values. For example, the function $y = x$ gives rise to the same qualitative tree as $y = 5x^3$, $y = e^x$, etc.

In respect of numerical prediction accuracy, in our case study $Q^2$ overall outperformed all competing numerical learners. Among these, locally weighted regression (LWR) with optimized parameters (through internal cross validation on the training set) performed best in terms of mean squared error. However its performance may sharply degrade under more difficult circumstances. Consider LWR-optimized performance on a difficult test set (Figure 12). It achieves excellent accuracy on the first part of this trace which is similar to the data in the training sets. LWR-optimized accuracy there is actually better than that of $Q^2$. However, problems begin for LWR in the second part of this trace where the input variables start to deviate considerably from the training data, and LWR's predictive error increases sharply. In this part of the trace, $Q^2$ manages to largely preserve qualitative consistency with the true behaviour, and maintains the numerical accuracy at a comparable level as in the area densely populated by training examples. A similar phenomenom can be observed in Figure 13, where the improvements of $Q^2$ are greater on more difficult test traces $T_7$ and $T_8$ than on other test traces that are more similar to the learning examples.

LWR-optimized was the best among standard numerical learners, and therefore our presentation of experimental results largely concentrated on comparison between $Q^2$ and LWR. The performance of M5 was grossly inferior both in terms of qualitative acceptability as well as numerical error. Optimizing M5's parameters did not help noticeably.

It should be noted that qualitatively faithful regression as carried out by the Q2Q program is actually inferior to LWR as a regression method. Struggling to satisfy qualitative consistency, Q2Q is limited to piece-wise linear regression with a small number of linear segments. This numerical inferiority of Q2Q usually turns out to be more than compensated by preserving qualitative consistency.

In this paper, qualitative constraints for Q2Q were induced from training data with QUIN. Alternatively, such constraints can be defined directly by a domain expert. In such a case, studied also in [26], the $Q^2$ learning can be viewed as an approach that enables the use of expert's qualitative knowledge in system

identification.

Among the limitations of our realization of $Q^2$, the rather basic numerical regression method in Q2Q should be noted. This method allows sharp changes in variable values (discontinuities in the variables' derivatives) at the borders between leaves of a qualitative tree. Future work should include a method for smoothing such discontinuities. A recently developed Q2Q transformation method, called Qfilter [26] and based on quadratic programming addresses this problem. Qfilter has some advantages over the Q2Q method presented in this paper, but it has higher computational complexity and its implementation is more demanding.

In the present paper, $Q^2$-learning was applied to automated system modelling from data. In our earlier work [27,25], a very simple kind of Q2Q transformation was applied to transforming qualitative control strategies into executable control strategies.

### *7.2 Related work*

The idea of QM-FS modelling [2,3] is close in spirit to $Q^2$ learning introduced in this paper. QM-FS modelling integrates qualitative modelling techniques with fuzzy logic systems. In QM-FS modelling, a domain expert supplies a QDE model (Qualitative Differential Equation model). This model is is used for qualitative simulation to produce possible qualitative behaviours of the modelled system. These qualitative behaviours are automatically converted into a set of fuzzy rules. These rules model the input output relations in the system. By estimating the parameters of the membership functions from experimental data, these fuzzy rules are transformed into a fuzzy system that can be used for numerical predictions. The fitting of fuzzy rules to learning data can be viewed as a particular kind of Q2Q transformation. A comparison with black-box approaches shows the advantages of using a qualitative model in terms of prediction accuracy. Improvements in terms of qualitative consistency are also evident although the authors do not explicitly address this. A notable difference between QM-FS modelling and $Q^2$ learning introduced in this paper is that $Q^2$ learning does not require a QDE model. Instead, a qualitative model is automatically induced from data.

There are several approaches to learning qualitative models that may support alternative approaches to $Q^2$ learning. Most of these approaches learn qualitative models in the form of QDEs that use qualitative relationships as *add* or *deriv* to describe dependencies among the system variables. In QDEs the states of the system variables are described as pairs of qualitative values and directions of change, and qualitative relationships are defined over se-

quences of such state descriptions, i.e. qualitative behaviours. Relevant early work in learning QDE models is described in [7,4,20,9,22]. These systems typically learn QDEs from qualitative behaviours, but can be extended to enable learning from numerical data by translating them to qualitative behaviours. They learn a model for one single operation region of the system and cannot learn models described by multiple QDEs. MISQ-RT [19] heuristically breaks the behaviours into segments and can learn multiple QDEs, corresponding to different operation regions and is in this respect similar to QUIN. QMN [8] uses a simple search procedure to find QDEs that, within some tolerance, fit the data. QMN, like QUIN, learns a qualitative model from numerical data directly, without translation to qualitative behaviours.

GENMODEL [7], the earliest work on learning QDE models, was later extended in [10] and demonstrated impressive results on real-valued experimental data. Systems QSI [22] and QOPH [6] use more sophisticated mechanisms in learning QDE models. Both systems introduce new variables if necessary and can handle noisy data.

Most of the mentioned systems may support alternative approaches to $Q^2$ learning. For that, a Q2Q method for reifying QDEs would be needed. However, reifying QDEs seems to require additional mechanisms to those in reifying QUIN's QCFs. SQUID [13] does a kind of Q2Q transformation from a different perspective. SQUID uses numerical data to form an envelope around the functional relationships in the model. It assumes that a semi-quantitative model of the observed system is given in the form of a semi-quantitative differential equation and refines it using numerical data. MSQUID [14] uses a neural network to fit the numerical data to a monotonic function of one variable and is in this respect simpler than here described Q2Q method. Similar to our $Q^2$ learning, monotonicity constraints enables MSQUID to make more accurate predictions.

## Acknowledgments

The work reported in this paper was partially supported by the European Fifth Framework project Clockwork and the Slovenian Ministry of Education, Science and Sport. We thank A. Eichberger and W. Rulka of Intec, for providing the wheel suspension model and the Simpack simulation software for this study, and for acting as domain experts. M.Valašek and P. Steinbauer of the Czech Technical University also helped in the Intec case study.

## References


[1] C. Atkeson, A. Moore, S. Schaal, Locally weighted learning, Artificial Intelligence Review 11 (1997) 11–73.

[2] R. Bellazzi, L. Ironi, R. Guglielmann, M. Stefanelli, Qualitative models and fuzzy systems: an integrated approach for learning from data, Artificial Intelligence in Medicine, Artificial Intelligence in Medicine 14 (1998) 5–28.

[3] R. Bellazzi, R. Guglielmann, L. Ironi, A hybrid input-output approach to model metabolic systems: An application to intracellular thiamine kinetics, Journal of Biomedical Informatics 24 (2001) 221–248.

[4] I. Bratko, S. Muggleton, A. Varšek, Learning qualitative models of dynamic systems, in: Proceedings of the 8th International Workshop on Machine Learning, 1991.

[5] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Wadsworth, Belmont, California, 1984.

[6] G. Coghill, S.M. Garret, R.D. King, Learning qualitative models in the presence of noise, in: Proceedings of the QR'02 Workshop on Qualitative Reasoning, 2002, Sitges, Spain.

[7] E. Coiera, Generating qualitative models from example behaviours, Technical report 8901, University of New South Wales (1989).

[8] S. Džeroski, L. Todorovski, Discovering dynamics: from inductive logic programming to machine discovery, J. Intell. Information Syst. 4 (1995) 89–108.

[9] S. Džeroski, L. Todorovski, Discovering dynamics, in: Proceedings of the 10th International Conference on Machine Learning, Morgan Kaufmann, 1993, pp. 97–103.

[10] D. Hau, E. Coiera, Learning qualitative models of dynamic systems, Machine Learning Journal 26 (1997) 177–211.

[11] K. Forbus, Qualitative process theory, Artificial Intelligence 24 (1984) 85–168.

[12] K. Forbus, Qualitative reasoning, in: A. Tucker (Ed.), CRC Computer Science and Engineering Handbook, CRC Press, 1997, pp. 715–733.

[13] H. Kay, B. Rinner, B. Kuipers, Semi-quantitative system identification, Artificial Intelligence 119 (2000) 103–140.

[14] H. Kay, L. H. Ungar, Estimating monotonic functions and their bounds, American Institute of Chemical Engineering (AIChE) Journal 46 (12) (2000) 2426–2434.

[15] J. de Kleer, J. Brown, A qualitative physics based on confluences, Artificial Intelligence 24 (1984) 7–83.

[16] B. Kuipers, Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge, MIT Press, Cambridge, Massachusetts, 1994.

[17] B. Kuipers, Qualitative simulation, Artificial Intelligence 29 (1986) 289–338.

[18] J. Quinlan, Learning with continuous classes, in: Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, World Scientific, Singapore, 1992, pp. 343–348.

[19] S. Ramachandran, R. Mooney, B. Kuipers, Learning qualitative models for systems with multiple operating regions, in: Working Papers of the 8th International Workshop on Qualitative Reasoning about Physical Systems, 1994, Nara, Japan.

[20] B. Richards, I. Kraan, B. Kuipers, Automatic abduction of qualitative models, in: Proceedings of the National Conference on Artificial Intelligence, AAAI/MIT Press, 1992.

[21] J. Rissanen, Modelling by shortest data description, Automatica 14 (1978) 465–471.

[22] A. Say, S. Kuru, Qualitative system identification: deriving structure from behavior, Artificial Intelligence 83 (1996) 75–141.

[23] Simpack software, Intec, 2002, www.simpack.com.

[24] D. Šuc, I. Bratko, Qualitative induction, in: Proceedings of the 15th International Workshop on Qualitative Reasoning, Stoughton: The Printing House, 2001, pp. 13–20, San Antonio, Texas.

[25] D. Šuc, I. Bratko, Qualitative reverse engineering, in: C. Sammut, A. Hoffmann (Eds.), Proceedings of the 19th International Conf. on Machine Learning, Morgan Kaufmann, 2002, pp. 610–617.

[26] D. Šuc, I. Bratko, Improving numerical accuracy with qualitative constraints, in: Proceedings of the 14th European Conference on Machine Learning, Springer, 2003, pp. 385–396, Dubrovnik, Croatia.

[27] D. Šuc, Machine reconstruction of human control strategies, Ph.D. thesis, Faculty of Computer and Information Sc., University of Ljubljana, Slovenia, 2001. Also published by IOS Press, Amsterdam (2003).

[28] D. Šuc, Machine Reconstruction of Human Control Strategies, Vol. 99 of Frontiers in Artificial Intelligence and Applications, IOS Press, Amsterdam, The Netherlands, 2003.

[29] D. Šuc, I. Bratko, Induction of qualitative trees, in: L. De Raedt, P. Flach (Eds.), Proceedings of the 12th European Conference on Machine Learning, Springer, 2001, pp. 442–453, Freiburg, Germany.

[30] I. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, San Francisco, 2000, Ch. 8: Machine Learning algorithms in Java, pp. 265–320.

[31] M. Wellman, Qualitative simulation with multivariate constraints, in: J. Allen, R. Fikes, S. E. (Eds.), Proc. 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, 1991.