

Dimension Reduction of Chemical Process Simulation Data

Gábor Janiga¹ and Klaus Truemper²

¹Laboratory of Fluid Dynamics and Technical Flows,
University of Magdeburg “Otto von Guericke”,
39106 Magdeburg, Germany

²Department of Computer Science,
University of Texas at Dallas,
Richardson, TX 75083, U.S.A.

Corresponding author:

G. Janiga
janiga@ovgu.de

Abstract. In the analysis of combustion processes, simulation is a cost-efficient tool that complements experimental testing. The simulation models must be precise if subtle differences are to be detected. On the other hand, computational evaluation of precise models typically requires substantial effort. To escape the computational bottleneck, reduced chemical schemes, for example, ILDM-based methods or the flamelet approach, have been developed that result in substantially reduced computational effort and memory requirements.

This paper proposes an additional analysis tool based on the Machine Learning concepts of Subgroup Discovery and Lazy Learning. Its goal is compact representation of chemical processes using few variables. Efficacy is demonstrated for simulation data of a laminar methane/air combustion process described by 29 chemical species, 3 thermodynamic properties (pressure, temperature, enthalpy), and 2 velocity components. From these data, the reduction method derives a reduced set of 3 variables from which the other 31 variables are estimated with good accuracy.

Key words: dimension reduction, subgroup discovery, lazy learner, modeling combustion

1 Introduction

Virtually all transportation systems and the majority of electric power plants rely directly or indirectly on the combustion of fossil fuels. Numerical simulation has become increasingly important for improving the efficiency of these combustion processes. Indeed, simulation constitutes a cost-efficient complement of experimental testing of design prototypes.

A key demand is that simulation be precise since otherwise subtle differences of complex processes escape detection. For example, the analysis of quantitative problems like predictions of intermediate radicals, pollutant emissions, or ignition/extinction limits requires complete chemical models [6]. Until recently, most models have been 1- or 2-dimensional due to the huge computational effort required for 3-dimensional models; see [11, 9] for the computational effort carried out for some 3-dimensional models.

Reduced chemical schemes, for example, ILDM-based methods or the flamelet approach [17], lead to a considerable speed-up of the computations when compared with a complete reaction mechanism, with small loss of accuracy. They drastically reduce required computational memory as well. These chemical schemes are based on an analysis of chemical pathways that identifies the most important ones. The other reactions are then modeled using the main selected variables.

This paper proposes an additional analysis tool called REDSUB that is based on Machine Learning methods, specifically Subgroup Discovery [12, 19] and the concept of Lazy Learner [15]. It analyses simulation data of chemical processes using a complete reaction scheme and searches for a compact representation that uses fewer variables while keeping errors within specified limits.

To demonstrate efficacy, a laminar methane/air burner is considered that occurs in many practical applications. The numerical simulation evaluates a 2-dimensional model using detailed chemistry [10]. For each grid point, the simulation produces values for 29 chemical species, 3 thermodynamic properties (pressure, temperature, enthalpy), and 2 velocity components. Given this information, REDSUB selects 3 variables from which the values of the remaining 31 variables are estimated with good accuracy.

We begin with a general definition of the reduction problem. Let X be a finite collection of vectors in R^n , the n -dimensional real space. In the present setting, X contains the results of a simulation model for some process. For efficient use of X , for example, for optimization purposes, we want to select a small subset J of $\{1, 2, \dots, n\}$ and construct a black box so that the following holds. For any vector of $x \in X$, the black box accepts as input the $x_j, j \in J$, and outputs values that are close estimates of the $x_j, j \notin J$.

For a simplified treatment of numerical accuracy and tolerances, we translate, scale, and round values of vectors $x \in X$ so that the entries become uniformly comparable in magnitude. Thus, we select translation constants, scaling, and rounding so that all variables have integer values ranging from 0 to 10^6 , with both bounds achieved on X . From now on we assume that the vectors of X themselves are of this form.

In the situations of interest here, we are given some additional information. Specifically, with each vector of $x \in X$ an associated m -dimensional vector y of R^m and the value of a real function $F(x)$ are supplied. For given input $x_j, j \in J$, the above black box must also output a close estimate of the function value $F(x)$. On the other hand, the black box need not supply estimates of the values of the vector y .

The reader may wonder why we introduce the function value $F(x)$, since it could be declared to be an additional entry in the vectors of X . In a moment we introduce an assumption about the function $F(\cdot)$, and for that reason it is convenient to treat its values separately.

As a final condition that is part of the problem definition, the index set J selected in the reduction process must be a subset of a specified nonempty set $I \subseteq \{1, 2, \dots, n\}$ and must contain a possibly empty subset $M \subseteq I$. For example, the x_j , $j \in I$, may represent values that are easily measured in the laboratory and thus support a simplified partial verification of the full model via the reduced model. The subset M contains the indices of variables that we require in the reduced model for any number of reasons.

In the example combustion process discussed in Section 3, each vector $x \in X$ contains $n = 33$ values for various chemical components, temperature, pressure, and two velocities. The function $F(x)$ is the enthalpy. The associated vector $y \in R^m$ has $m = 2$ and defines the point in the plane to which the values of x and $F(x)$ apply. Problem of that size are easily handled by REDSUB. Indeed, in principle the method can handle cases where the vectors x of X have up to several hundred entries. The size of m is irrelevant.

Once REDSUB has identified the index set J , the Lazy Learner of REDSUB can use J and the set X to estimate for any vector where just the values x_j , $j \in J$ are given, the values for all x_j , $j \notin J$, and $F(x)$. This feature allows application of the results in settings similar to that producing X .

We introduce two assumptions concerning the vectors y and the function $F(x)$. They are motivated by the applications considered here. The first assumption says that the vectors y constitute a particular sampling of a compact convex subspace of R^m . The second one demands that $F(\cdot)$ is close to being a one-to-one function on the data set X .

Assumption 1 *Collectively, the vectors y associated with the vectors $x \in X$ constitute a grid of some compact convex subset of R^m .*

Assumption 2 *$F(\cdot)$ is close to being a one-to-one function, in the sense that the number of distinct function values $F(x)$, $x \in X$, is close to the number of vectors in X .*

We motivate the assumptions by the combustion processes we have in mind. There, the vectors y are the coordinate values of the 1-, 2-, or 3-dimensional simulation. Since the simulation generates the points of a grid, Assumption 1 is trivially satisfied. Assumption 2 demands that the function distinguishes between almost all points $x \in X$. In the example application discussed later, the function is the enthalpy of the process and turns out to satisfy Assumption 2.

The paper proceeds as follows. The next section, 2, describes Algorithm REDSUB. Section 3 covers application of the method to a methane combustion process. Section 4 has details of the subgroup discovery method used in REDSUB. Section 5 summarizes the results.

2 Algorithm REDSUB

In a traditional approach for the problem at hand, we could (1) estimate the function values with ridge regression [7, 8]; (2) derive the index set J via the estimating coefficients of the ridge regression equations of (1) and covariance considerations; and (3) use ridge regression once more to compute estimating equations that estimate values for the variables $j \notin J$, and $F(x)$ from those for the variables x_j , $j \in J$. The approach typically requires that the user specifies nonlinear transformations of the variables for the regression steps (1) and (3). Here, we propose a new approach that is based on two Machine Learning concepts and does not require user-specified nonlinear transformations. Specifically, we use *Subgroup Discovery* [12, 19] to select *interesting* indices j from the set I ; from these indices, the set J is defined. Then we use the concept of *Lazy Learner* to predict values for the x_j , $j \notin J$, and $F(x)$.

2.1 Subgroup Discovery

The Subgroup Discovery method, called SUBARP, constructs polyhedra that represent important partial characterizations of *level sets* $\{x \in X \mid F(x) \geq c\}$ of $F(\cdot)$ for various values c . SUBARP has been constructed from the classification method Lsquare [2–5] and the extension of [18], using the general construction scheme of [14] for deriving subgroup discovery methods from rule-based classification methods. Details are included in Section 4.

The key arguments for use of the polyhedra are as follows.

If a variable occurs in the specification of a polyhedron, then control of the value of that variable is important if function values are to be in a certain level set. This is a direct consequence of the fact that Subgroup Discovery constructed the polyhedron. Accordingly, the frequency q_j with which a variable x_j occurs in the definition of the various polyhedra is a measure of the importance the variable generally has for inducing function values $F(\cdot)$.

The frequency q_j is computed as follows. Define the *length* of an inequality to be the number of its nonzero coefficients. For a given polyhedron, the contribution to q_j is 0 if x_j does not occur in any inequality of the polyhedron, and otherwise is $1/k$ where k is the length of the shortest inequality containing x_j .

For a simplified discussion, momentarily assume that indices j of high frequencies q_j constitute the set J . We know that the variables x_j , $j \in J$, are essential for determining the level sets of $F(\cdot)$. By Assumption 2, $F(\cdot)$ is close to being a one-to-one function; thus, function values $F(x)$ or, equivalently, level sets may be used to estimate x vectors. Since variables x_j , $j \in J$, are important for determining level sets of $F(\cdot)$, we know that such x_j are also important for determining entire x vectors, in particular, for estimating values of the variables x_j , $j \notin J$.

The actual construction of the set J is a bit more involved and involves iterative steps carried out by REDSUB. Details are discussed after the description of the Lazy Learner. Here we mention only that REDSUB splits the input data set X , which from now on we call a *learning set*, 50/50 into a *training set* S and

testing set T . The split is done in such a way that the y vectors corresponding to the vectors of S constitute a coarser subgrid of the original grid. SUBARP utilizes both sets S and T , while the Lazy Learner only uses S .

2.2 Lazy Learner

A Lazy Learner is a method that, well, never learns anything. Instead, it directly derives from a given training set an estimate for the case at hand and then discards any insight that could be gained from those computations [15]. In the present situation, given index set J and training set S , the Lazy Learner estimates for a given vector x the entries x_j , $j \notin J$, and $F(x)$ from the x_j , $j \in J$. This section describes that estimation process, which is a particular Locally Weighted Learning algorithm; see [1] for an excellent survey.

For the given vector x , let v be the subvector of x indexed by J , and define $k = |J|$. We first find a vector $x^1 \in S$ whose subvector z^1 indexed by J is closest to v according the *infinity norm*, which defines the distance between any two vectors g and h to be $\max_j \{|g_j - h_j|\}$. We use that distance measure throughout unless distance is explicitly specified to be Euclidean.

If $z^1 = v$, x^1 and $F(x^1)$ supply the desired estimates. Otherwise, we use the following 5-step interpolation process. (1) Let L be the line passing through v and z^1 , and denote by d the distance between the two points. Define p^1 and p^2 to be the two points on L whose distance from v is equal to d and $2d$, respectively, such that on L the point v lies between z^1 and p^1 as well as between z^1 and p^2 ; see Figure 1. (2) Let Z be the set of vectors z such that z is a subvector of some

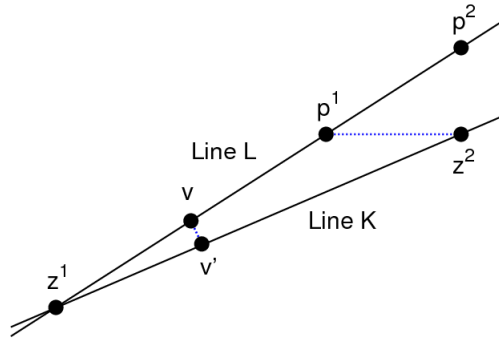


Fig. 1. Lazy Learner Interpolation Process

$x \in S$ and is within distance d of p^2 . (3) If Z is empty, x^1 and $F(x^1)$ supply the desired estimates. Otherwise, let z^2 be the vector of Z that is closest to p^1 , and let x^2 be any vector of S with z^2 as subvector; break ties in the choices of z^2 and

x^2 arbitrarily. (4) Project v onto the line K passing through z^1 and z^2 , getting a point v' . The projection uses Euclidean distance. (5) If the point v' lies on the line segment of K defined by z^1 and z^2 and thus is a convex combination of z^1 and z^2 , say $v' = \lambda z^1 + (1-\lambda)z^2$ for some $\lambda \in [0, 1]$, then the vector $\lambda x^1 + (1-\lambda)x^2$ and $\lambda F(x^1) + (1-\lambda)F(x^2)$ provide the desired estimates. Otherwise, use x^1 and $F(x^1)$ to obtain the estimates.

The above choices of distance measure and interpolation were derived via experimentation. Indeed, the infinity norm emphasizes maximum deviations, and with that measure, the interpolation seemingly is effective in producing estimates of uniform accuracy for the entire range of variables and the function $F(x)$.

The estimation errors for any vector x of the testing set T is the distance between the vector $[x, F(X)]$ and the corresponding vector of estimated values. The *average error* is the sum of the estimation errors for the set T , divided by $|T|$. It is used next in Section 2.3.

2.3 Iterations of REDSUB

Algorithm REDSUB constructs the set J in an iterative process from the given learning set X , where in iteration $i \geq 1$ a set J_i is selected. Recall that $I \subseteq \{1, 2, \dots, n\}$ contains the indices j of variables x_j that may occur in the reduced model, and that M has the indices j for which x_j must occur in that model. Furthermore, the set X has been split 50/50 into a training set S and a testing set T .

Iteration $i \geq 1$ begins with the set J_{i-1} ; for $i = 1$, the set $J_{i-1} = J_0$ is defined to be the set I of variables that may be considered for the reduced representation.

Before iteration 1 commences, the Lazy Learner uses the index set J_0 and the training set S to compute estimates for the testing set T as well as the average error of those estimates. If the average error exceeds a user-specified bound, the method declares that a reduced model with acceptable estimation errors cannot be found, and stops.

Iteration $i \geq 1$ proceeds as follows. If $J_{i-1} = M$, then no further reduction is possible, and REDSUB outputs J_{i-1} as the desired set J and stops; otherwise, SUBARP computes frequencies q_j , $j \in J_{i-1}$, as described in Section 2.1 except that J_{i-1} plays the role of J . Let q^* be the minimum of the q_j with $j \in (J_{i-1} - M)$. REDSUB processes each $j \in (J_{i-1} - M)$ for which q_j is at or near the minimum q^* , as follows.

First, the index j is temporarily removed from J_{i-1} , resulting in a set J' , and the Lazy Learner computes the average error as above, except that J' is used instead of J_0 .

Second, let $j^* \in (J_{i-1} - M)$ be the index for which the average error is smallest. If that smallest error is below a user-specified bound, the associated $J' = J_{i-1} - \{j^*\}$ is defined to be J_i , and iteration $i + 1$ begins; otherwise, the process stops, and J_{i-1} is declared to be the desired set J .

If $q^* = 0$, which typically happens in early iterations i , then the method accelerates the process by also considering the case where all $j \in (J_{i-1} - M)$

with $q_j = 0$ are simultaneously deleted. If the average error of that case is below the user-specified bound, then that case defines J_i , and iteration $i + 1$ begins; otherwise, the above process using j^* is carried out.

Computational Complexity

REDSUB is polynomial if suitable polynomial implementations of SUBARP and the Lazy Learner are used. We should mention, though, that SUBARP of our implementation uses the Simplex Method for the solution of certain linear programming instances. Since the Simplex Method is not polynomial, our implementation of REDSUB cannot be claimed to be polynomial. But the Simplex Method is well known for solving linear programs rapidly and reliably, and for this reason it is used in SUBARP.

The next section presents computational results of REDSUB for a methane combustion process.

3 Application

In [10], a simulation model represents a partially-premixed laminar methane/air combustion process in 2-dimensional space. Thus, $m = 2$. We skip a review of the model and only mention that the simulation has $n = 34$ variables measuring important characteristics such as mixture composition, temperature, pressure, two velocities, and enthalpy. The mixture composition covers 29 gases, among them H_2 , H_2O , O_2 , CO , CH_4 , CO_2 , and N_2 . The 29 gases define the set I from which the set J is to be selected. The set M of mandated selections is empty.

For the 7,310 grid points y shown in Figure 2, the simulation supplies vectors x , each with values for the 33 variables. Thus, the set X consists of 7,310 vectors. The function $F(x)$ is the enthalpy. We split the data 50/50 into a learning set X from which the reduced model is derived via SUBARP and the Lazy Learner, and a *verification set* V . The split is so done that the y vectors corresponding to the vectors of X constitute a coarser subgrid of the original grid produced by the simulation. That way the y vectors associated with X satisfy Assumption 1. The learning set X has 3,655 vectors. Of these, 3,412 vectors (= 93%) have distinct function values $F(x)$. Thus, $F(x)$ is close to one-to-one on X , and Assumption 2 is satisfied.

Recall that the *length* of an inequality is the number of nonzero coefficients. One option of SUBARP controls the complexity of the polyhedra via a user-specified upper bound on the length of the defining inequalities. The option relies on the method of [18]. In principle, the upper bound on length can be any value 2^l , $l \geq 0$, but experimental results cited in the reference indicate that $l \leq 2$ is preferred. We run SUBARP with each of the corresponding three bounds of 1, 2, and 4 on the length of the inequalities. In subgroup applications of SUBARP, we typically use 20 distinct values for any *target* variable, which is a variable to be investigated. Here, the target variable is $F(x)$, and in agreement with prior uses of SUBARP we employ 20 distinct c values to define the level sets

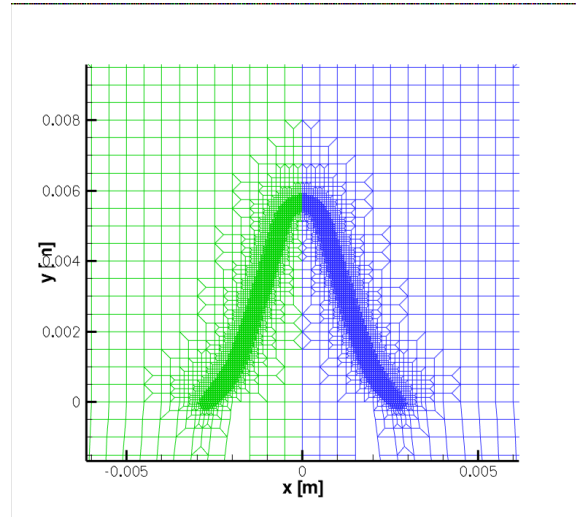


Fig. 2. Numerical Grid in the Vicinity of the Flame Front

$\{x \in X \mid F(x) \geq c\}$. For each of these level sets, SUBARP may find polyhedra contained in the level set as well as in the complement. Hence, a number of polyhedra may be determined for given level set. Of these, we choose for each level set the polyhedron with highest significance for the computation of the frequencies q_j as described in Section 2.1.

We run REDSUB once for each of the three bounds on the length of inequalities. For each run, the overall error is the ratio of the average error computed by the Lazy Learner, divided by the range of the variable values. Due to the transformation described in Section 1, the latter range is uniformly equal to 10^6 for each variable. REDSUB stops the reduction process when the overall error begins to exceed a user-specified bound e .

Using $e = 0.5\%$, each of the three REDSUB runs produces the same solution. It consists of the three variables H_2 , H_2O , and N_2 . To evaluate the effectiveness of the solution variables, we employ the entire set X in the Lazy Learner and use it as black box to estimate values for the verification set V . The overall error turns out to be 0.26% . Indeed, the estimated values are quite precise. For a demonstration, Figs. 3–8 show sample pictures for six variables, specifically, for C_2H_2 , CH_4 , CO , HCO , the enthalpy, and the temperature. In each picture, the area of the flame is shown. On the right-hand side of the center vertical axis, the colors encode the values determined by the simulation. On the left-hand side, colors encoding the estimated values are displayed. Due to the symmetry, consistency of the color patterns of the two sides implies accurate estimation, and conversely.

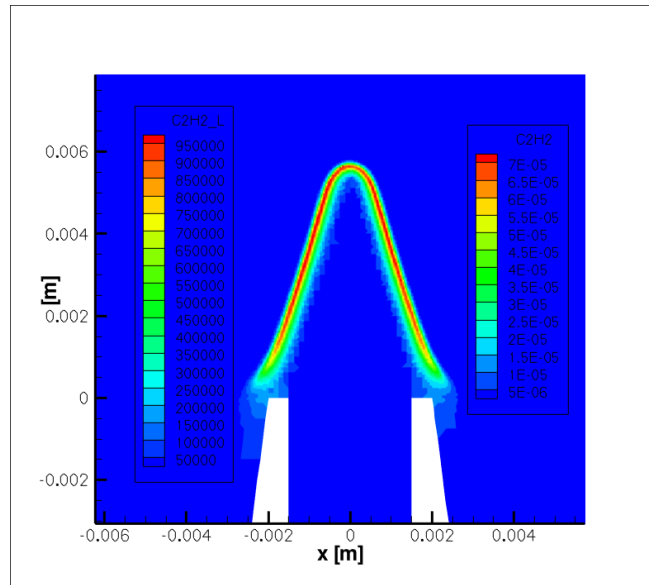


Fig. 3. 3-Variable Solution: Accuracy for C_2H_2

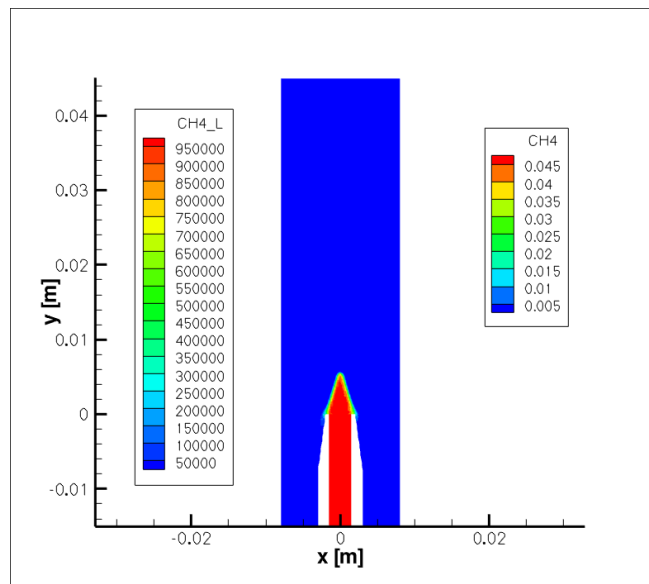


Fig. 4. 3-Variable Solution: Accuracy for CH_4

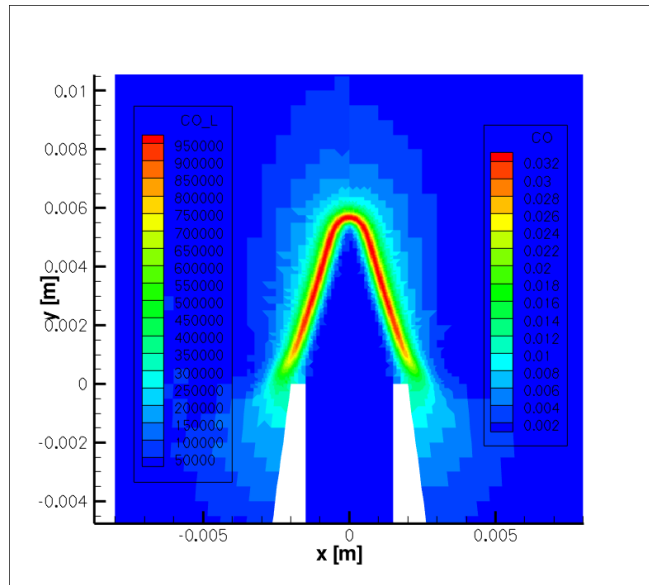


Fig. 5. 3-Variable Solution: Accuracy for CO

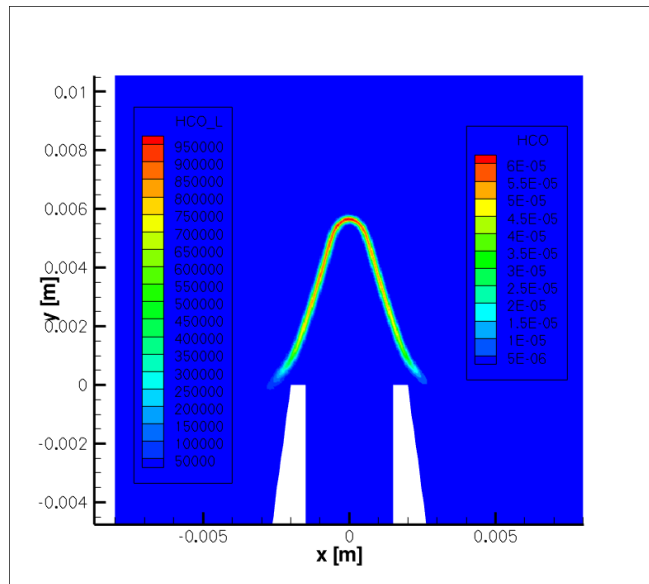


Fig. 6. 3-Variable Solution: Accuracy for HCO

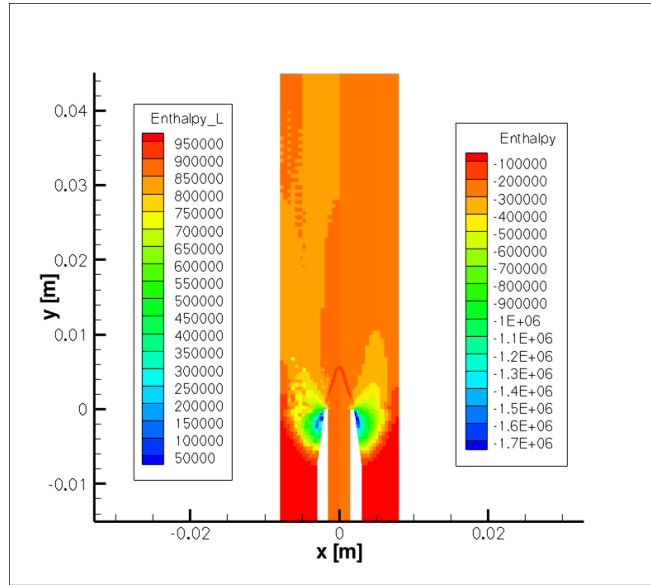


Fig. 7. 3-Variable Solution: Accuracy for Enthalpy

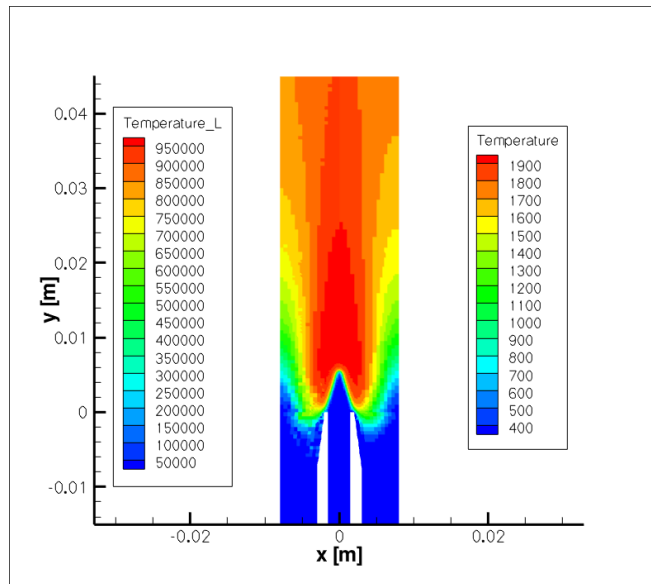


Fig. 8. 3-Variable Solution: Accuracy for Temperature

We include a table that shows the correlation between estimated and actual values. Since H_2 , H_2O and N_2 are used to produce the estimates, their correlation values are trivially equal to 1.0. Clearly, estimated and actual values are highly correlated for all variables.

Table 1. Correlation of Actual and Estimated Values Using H_2 , H_2O and N_2

Variable	Correlation	Variable	Correlation
u -velocity	0.9942	CH_2O	0.9998
v -velocity	0.9924	CH_3	0.9998
Pressure	0.9923	CH_3O	0.9996
Temperature	0.9989	CH_2OH	0.9998
H	0.9999	CH_4	0.9999
OH	0.9998	C_2H	0.9996
O	0.9998	HCCO	0.9996
HO_2	0.9996	C_2H_2	0.9998
H_2	1.0000	CH_2CO	0.9998
H_2O	1.0000	C_2H_3	0.9997
O_2	0.9999	C_2H_4	0.9997
CO	0.9999	C_2H_5	0.9997
CO_2	0.9999	C_2H_6	0.9997
CH	0.9997	C	0.9996
HCO	0.9998	C_2	0.9996
CH_2S	0.9998	N_2	1.0000
CH_2	0.9997	Enthalpy	0.9929

For the variable HO_2 and the enthalpy, whose correlation values approximately cover the range exhibited in Table 1, we include graphs that show the relationship between actual and estimated values. Considering that in the two graphs almost all of the 3,655 points occur on or very close to the diagonal, estimation errors are clearly low. There are a few exceptions, most noticeably for a few small values of the enthalpy.

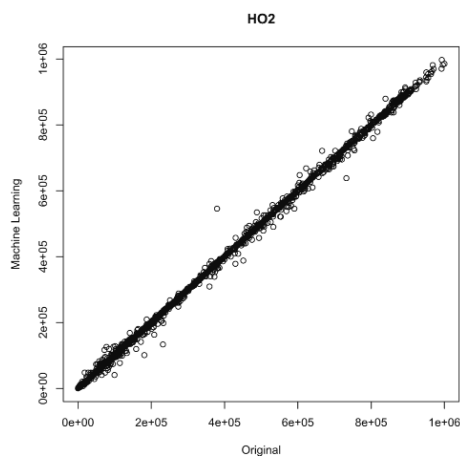


Fig. 9. 3-Variable Solution: HO₂ actual versus estimated values

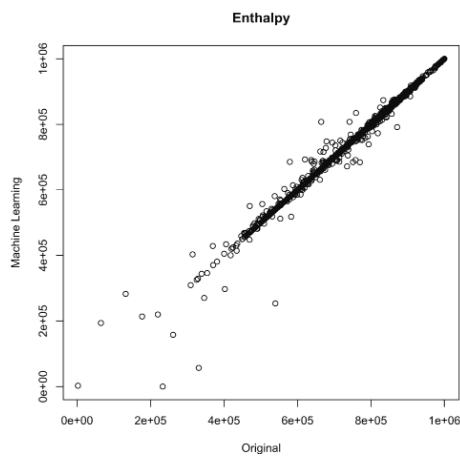


Fig. 10. 3-Variable Solution: Enthalpy actual versus estimated values

We repeat the above evaluation except that we use the slightly larger error bound of $e = 1\%$. Each of the three runs selects the two variables H₂ and H₂O. The overall error for the verification set V is 0.33%. Figs. 11–16 show the corresponding pictures. They demonstrate that reduction to the two variables H₂ and H₂O still produces potentially useful results, but at a lesser level of accuracy,

as is evident from occasional and moderate disparities of the color patterns on the left-hand side versus the right-hand side.

In agreement with the results shown for the 3-variable case, we also include a table showing the correlation of actual and estimated values for all variables and enthalpy, and two graphs providing details for HO_2 and enthalpy.

The reader may wonder whether a much simpler approach could produce the same results. For example, a greedy method could use in each iteration minimum average error computed by the Lazy Learner als deletion criterion and thus would avoid the subgroup discovery process. We have implemented that method and obtained the following results. First, for the case of $e = 0.5\%$, the simplified process obtains a solution with the three variables OH , H_2 , and O_2 . The overall error is virtually identical to that obtained earlier, 0.25% now versus 0.26% earlier. But for $e = 1\%$, the simplified method finds a solution with the two variables H_2 and O_2 . The average error turns out to be 0.94% , which is almost triple of the 0.33% error for the earlier derived solution with two variables. Indeed, the solution obtained by the simplified method is essentially unusable.

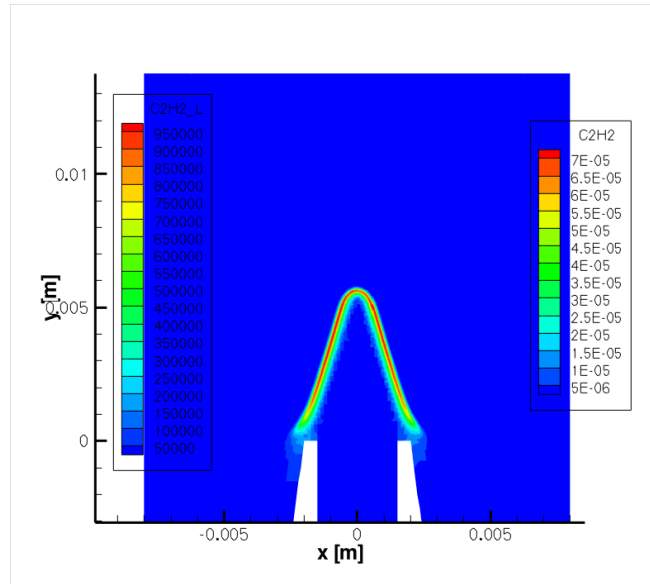


Fig. 11. 2-Variable Solution: Accuracy for C_2H_2

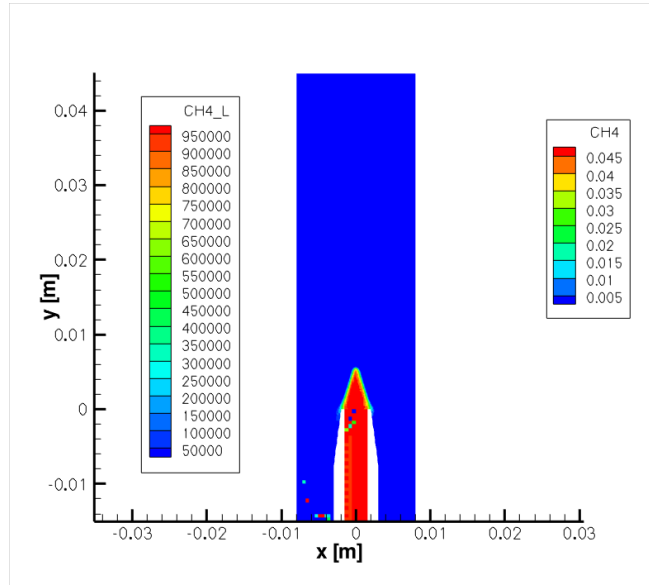


Fig. 12. 2-Variable Solution: Accuracy for CH₄

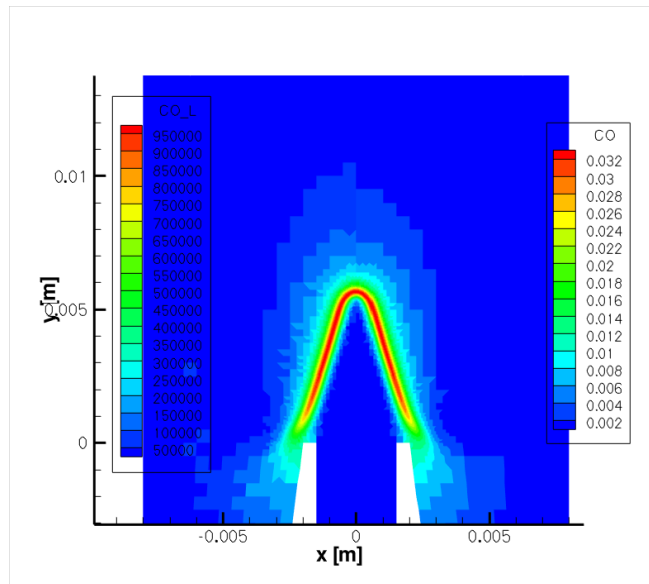


Fig. 13. 2-Variable Solution: Accuracy for CO

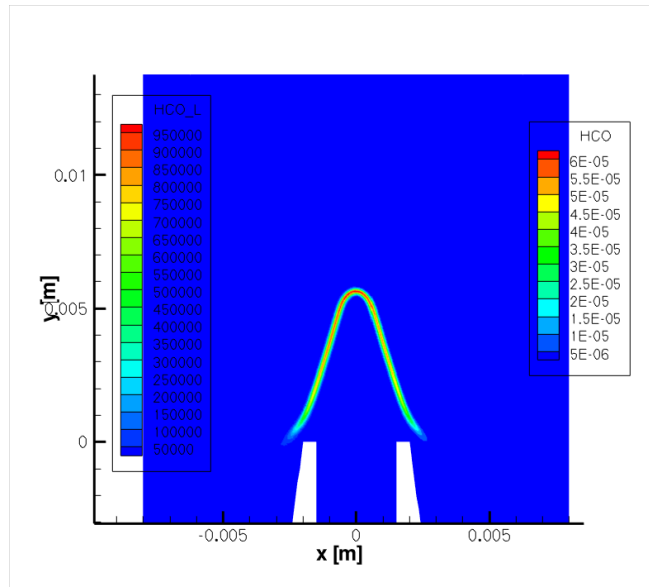


Fig. 14. 2-Variable Solution: Accuracy for HCO

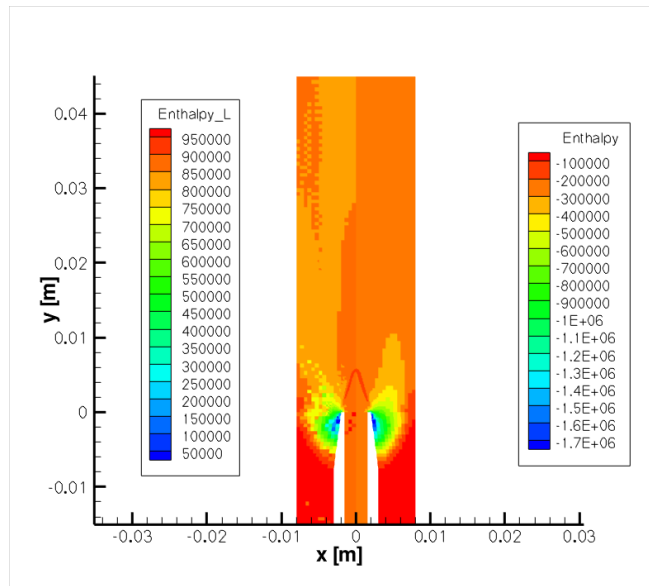


Fig. 15. 2-Variable Solution: Accuracy for Enthalpy

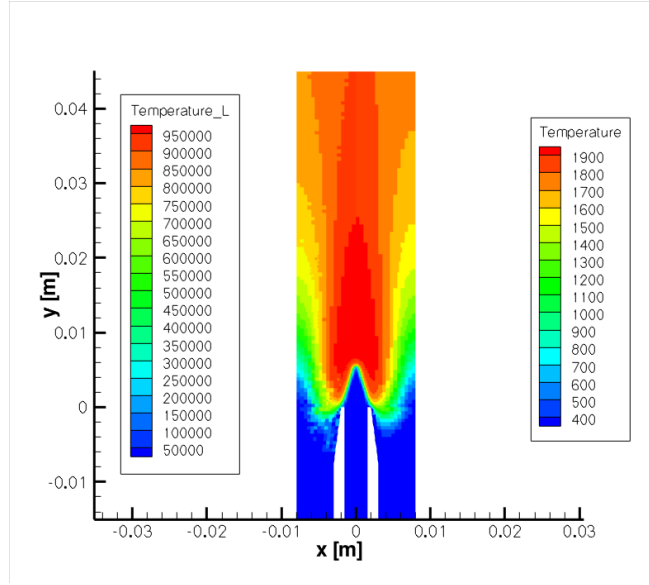


Fig. 16. 2-Variable Solution: Accuracy for Temperature

Table 2. Correlation of Actual and Estimated Values Using H₂ and H₂O

Variable	Correlation	Variable	Correlation
<i>u</i> -velocity	0.9942	CH ₂ O	0.9998
<i>v</i> -velocity	0.9890	CH ₃	0.9998
Pressure	0.9845	CH ₃ O	0.9996
Temperature	0.9988	CH ₂ OH	0.9998
H	0.9999	CH ₄	0.9663
OH	0.9997	C ₂ H	0.9997
O	0.9998	HCCO	0.9996
HO ₂	0.9996	C ₂ H ₂	0.9998
H ₂	1.0000	CH ₂ CO	0.9998
H ₂ O	1.0000	C ₂ H ₃	0.9997
O ₂	0.9997	C ₂ H ₄	0.9998
CO	0.9999	C ₂ H ₅	0.9997
CO ₂	0.9996	C ₂ H ₆	0.9997
CH	0.9997	C	0.9996
HCO	0.9998	C ₂	0.9996
CH ₂ S	0.9998	N ₂	0.9519
CH ₂	0.9997	Enthalpy	0.9881

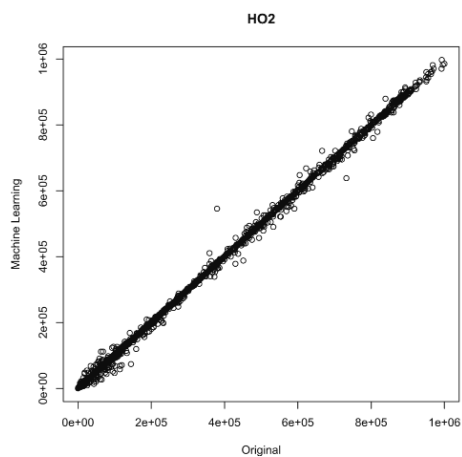


Fig. 17. 2-Variable Solution: HO₂ actual versus estimated values

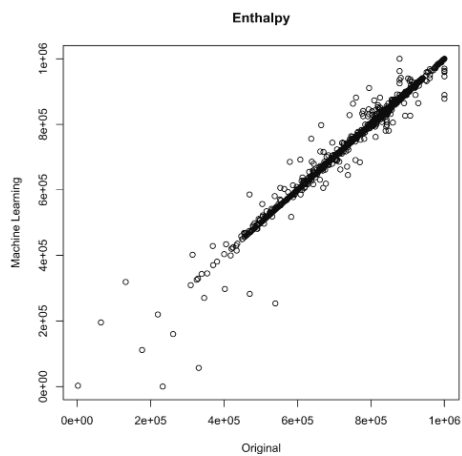


Fig. 18. 2-Variable Solution: Enthalpy actual versus estimated values

How close are the two solutions obtained above for two and three variables to optimal solutions with the same number of variables, assuming that the average error computed by the Lazy Learner is to be minimized? Since the number of variables is small, the question can be settled by enumeration of cases. For the 3-variable case, the optimal solution has the variables OH, H₂, and O₂, which

is precisely the solution obtained by the greedy method. As stated earlier, the accuracy for that case is essentially the same as that attained by the REDSUB solution. For two variables, the optimal solution has the variables H_2 and H_2O , which are precisely the variables selected by REDSUB. We conclude that for this application, REDSUB has obtained solutions that are very close to or actually optimal.

4 Algorithm SUBARP

Recall that I is the index set of variables from which the variables are to be selected, and that the learning set X is split 50/50 into a training set S and a testing set T . Let Z be the collection of subvectors of the $x \in S$ defined by the index set I . With each record of Z , we have a value t called the *target value* of the record, which is the function value $F(x)$ of the associated $x \in S$. *Subgroup Discovery* can find important relationships that link the vectors of Z and the target t ; see [12, 19]. The subset of records defined by any such relationships is a *subgroup*. In the present setting, each relationship is defined by a logic formula involving linear inequalities, and each subgroup consists of a subset of vectors of Z lying in a certain polyhedron.

There are a number of Subgroup Discovery methods; for a survey that unifies Subgroup Discovery with the related tasks of *Contrast Set Mining* and *Emerging Pattern Mining*, see [16]. In particular, any standard classification rule learning approach can be adapted to Subgroup Discovery [14]. SUBARP is one such method, adapted from the classification method Lsquare [2–5] and the extension of [18]. For completeness we include an intuitive summary of SUBARP that is specialized for the case at hand.

4.1 Target Discretization

We want to find relationships connected with the level sets of the function $F(\cdot)$ or their complements. Accordingly, we define a number of values c of possible target values for the target t . For each c , we declare the set $\{t \mid t \geq c\}$ to be a *range* of the target t .

For given target range R , let A be the set of given records whose target values t are in R , and define B to be the set of remaining records. SUBARP tries to explain the difference between the sets A and B in a multi-step process. The steps involve feature selection, computation of two formulas, factor selection, and subgroup evaluation. They are described in subsequent sections. The first two steps use Lsquare cited above.

We need to introduce the notion of a *logic formula* that to a given record of Z assigns the value *True* or *False*. The concept is best explained using an example. Suppose that a record has g and h among its attributes, and that the record values for these attributes are $g = 3.5$ and $h = 4.7$. The logic formula consists of *terms* that are linear inequalities involving the attributes, and the operators \wedge (“and”) and \vee (“or”). For a given record, a term evaluates to *True*

if the inequality is satisfied by the attribute values of the record, and evaluates to *False* otherwise. Given *True/False* values for the terms, the formula is evaluated according to the standard rules of propositional logic.

For example, $(g \leq 4)$ and $(h > 3)$ are terms. A short formula using these terms is $(g \leq 4) \wedge (h > 3)$. For the above record with $g = 3.5$ and $h = 4.7$, both terms of the formula evaluate to *True*, and hence the formula has that value as well. As a second example, $(g \leq 4) \vee (h > 5)$ produces the value *True* for the cited record, since at least one of the terms evaluates to *True*.

4.2 Feature Selection

This step, which is handled by Lsquare, repeatedly partitions the set A into subsets A_1 and A_2 ; correspondingly divides the set B into subsets B_1 and B_2 ; finds a logic formula that achieves the value *True* on the records of A_1 and *False* on those of B_1 ; and tests how often the formula achieves *True* for the records A_2 and *False* for those of B_2 . In total, 20 formulas are created. In Lsquare, these formulas are part of an ensemble voting scheme to classify records. Here, the frequency with which a given attribute occurs in the formulas is used as an indicator of the importance of the attribute in explaining the differences between the sets A and B . Using that indicator, a significance value is computed for each attribute. The attributes with significance value beyond a certain threshold are selected for the next step.

4.3 Computation of Two Formulas

Using the selected attributes, Lsquare computes two formulas. One of the formulas evaluates to *True* for the records of A and to *False* for those of B , while the second formula reverses the roles of A and B .

Both formulas are in *disjunctive normal form* (DNF), which means that they consist of one or more *clauses* combined by “or.” In turn, each clause consists of linear inequality terms as defined before and combined by “and.” An example of a DNF formula is $((g \leq 4) \wedge (h > 3)) \vee ((g < 3) \wedge (h \geq 2))$, with the two clauses $((g \leq 4) \wedge (h > 3))$ and $((g < 3) \wedge (h \geq 2))$. Later, we refer to each clause as a *factor*.

Next, using the general construction approach of [14], we derive factors, and thus implicitly subgroups, from the first DNF formula. The same process is carried out for the second DNF formula, except for reversal of the roles of A and B .

4.4 Factor Selection

Recall that the first DNF formula evaluates to *False* for the records of B . Let f be a factor of that formula. Since the formula consists of clauses combined by “or,” the factor f also evaluates to *False* for all records of B . Let Q be the subset of records for which f evaluates to *True*. Evidently, Q is a subset of A .

We introduce a direct description of Q using an example. Suppose the factor f has the form $f = ((g \leq 4) \wedge (h > 3))$. Let us assume that A was defined as the subset of records for which the values of target t fall into the range $R = \{t \geq 9\}$. Then Q consists of the records of the original data set satisfying $t \geq 9$, $g \leq 4$, and $h > 3$. But we know more than just this characterization. Indeed, each record of B violates at least one of the inequalities $g \leq 4$ and $h > 3$. Put differently, for any record where t does not satisfy $t \leq 9$, we know that at least one of the inequalities $g \leq 4$ and $h > 3$ is violated.

So far, we have assumed that the formula evaluates to *False* for all records of B . In general, this goal may not be achieved, and the formula produces the desired *False* values for most but not all records of B . Put differently, Q then contains a few records of B . For the above example, this means that for some records where the target t does not satisfy $t \geq 9$, both inequalities $x \leq 4$ and $y > 3$ may actually be satisfied.

Potentially, the target range R and the factor f characterize an important configuration that corresponds to an interesting and useful subgroup. On the other hand, the information may be trivial and of no interest. To estimate which case applies, SUBARP computes a significance value for each factor that lies in the interval $[0, 1]$. The rule for computing the significance value is related to those used in other Subgroup Discovery methods, where the goal of *interestingness* [13] is measured with quality functions balancing conflicting goals such as (1) the size of the subgroup, (2) the length of the pattern or rule defining the subgroup, and (3) the extent to which the subgroup is contained in the set A . Specifically, SUBARP decides significance using the third measure and the probability with which a random process could create a set of records M for which the target t is in R , the factor f evaluates to *True*, and the size of M matches that of Q . We call that random process an *alternate random process* (ARP).

Generally, an ARP is an imagined random process that can produce an intermediate or final result of SUBARP by random selection. SUBARP considers and evaluates several ARPs within and outside the Lsquare subroutine, and then structures decisions so that results claimed to be important can only be achieved by the ARPs with very low probability. The overall effect of this approach is that subgroups based on factors with very high significance usually turn out to be interesting and important.

Up to this point, the discussion has covered polyhedra representing subgroups contained in a level set. When the same computations are done with the roles of A and B reversed, polyhedra are found that represent subgroups contained in the complement of a level set. It is easily seen that the arguments made earlier about the importance of variables based on the earlier polyhedra apply as well when the latter polyhedra are used.

4.5 Evaluation of Subgroups

Once the significance values have been computed for the subgroups, the associated target ranges and factors are used to compute a second significance value using the testing records of T , which up to this point have not been employed

by SUBARP. The average of the significance values obtained from the training and testing data is assigned as *overall significance* value to each subgroup.

In numerous tests of data sets, it has turned out that only subgroups with very high significance value are potentially of interest and thus may produce new insight. Based on these results, only subgroups with significance above 0.90 are considered potentially useful.

Finally, we relate the output of highly significant subgroups to the original problem.

4.6 Interpretation of Significant Subgroups

Let f be the factor of a very significant subgroup and $R = t \geq c$ be the associated target range. Define P to be the polyhedron defined by $P = \{x_j, j \in I \mid \text{factor } f \text{ evaluates to } True\}$. Then the polyhedron is an important partial characterization of the level set $\{x \mid F(x) \geq c\}$ of the function $F(\cdot)$. As discussed in Section 2, the variables used in the definition of P can therefore be considered an important part of the characterization of that level set. That fact supports the construction of the sets J_i described in Section 2 according to the frequencies q_j with which each variable $x_j, j \in I$, occurs in the definitions of the polyhedra of very significant subgroups.

5 Summary

This paper introduces a method called REDSUB for the reduction of combustion simulation models. REDSUB is based on the Machine Learning techniques of Subgroup Discovery and Lazy Learner. Use of the method does not require any user-specified transformations of simulation data or other manual effort. The effectiveness of REDSUB is demonstrated with simulation data of a methane/air combustion process, where 34 variables representing 29 chemical species of the combustion mixture, 3 thermodynamic properties, and 2 velocity components are reduced to 3 variables. For the remaining 31 variables, the values are estimated with good accuracy over the entire range of possible values. An even smaller model using 2 variables still has reasonable accuracy.

References

1. C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.
2. S. Bartnikowski, M. Granberry, J. Mugan, and K. Truemper. Transformation of rational and set data to logic data. In *Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques*. Springer, 2006.
3. G. Felici, F. Sun, and K. Truemper. Learning logic formulas and related error distributions. In *Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques*. Springer, 2006.
4. G. Felici and K. Truemper. A MINSAT approach for learning in logic domain. *INFORMS Journal of Computing*, 14:20–36, 2002.

5. G. Felici and K. Truemper. The lsquare system for mining logic data. In *Encyclopedia of Data Warehousing and Mining*, pages 693–697. Idea Group Publishing, 2005.
6. R. Hilbert, F. Tap, H. El-Rabii, and D. Thévenin. Impact of detailed chemistry and transport models on turbulent combustion simulations. *Progress in Energy and Combustion Science*, 30:61–117, 2004.
7. A. E. Hoerl. Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58:54–59, 1962.
8. A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.
9. G. Janiga, O. Gicquel, and D. Thévenin. High-resolution simulation of three-dimensional laminar burners using tabulated chemistry on parallel computers. In *2nd ECCOMAS Thematic Conference on Computational Combustion*, 2007.
10. G. Janiga, A. Gordner, H. Shalaby, and D. Thévenin. Simulation of laminar burners using detailed chemistry on parallel computers. In *European Conference on Computational Fluid Dynamics (ECCOMAS CFD 2006)*, 2006.
11. G. Janiga and D. Thévenin. Three-dimensional detailed simulation of laminar burners on parallel computers. In *Proceedings of the European Combustion Meeting ECM2009*, 2009.
12. W. Klösgen. EXPLORA: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996.
13. W. Klösgen. Subgroup discovery. In *Handbook of Data Mining and Knowledge Discovery*. Oxford University Press, 2002.
14. N. Lavrač, B. Čestnik, D. Gamberger, and P. Flach. Decision support through subgroup discovery: Three case studies and the lessons learned. *Machine Learning*, 57:115–143, 2004.
15. T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
16. P. K. Novak, N. Lavrač, and G. I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403, 2009.
17. N. Peters. *Turbulent Combustion*. Cambridge University Press, 2000.
18. K. Truemper. Improved comprehensibility and reliability of explanations via restricted halfspace discretization. In *Proceedings of International Conference on Machine Learning and Data Mining (MLDM 2009)*, 2009.
19. S. Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proceedings of First European Conference on Principles of Data Mining and Knowledge Discovery*, 1997.