Visual Analytics for Complex Engineering Systems: Hybrid Visual Steering of Simulation Ensembles

Krešimir Matković, *Member, IEEE CS*, Denis Gračanin, *Senior Member, IEEE*, Rainer Splechtna, *Member, IEEE CS*, Mario Jelović, Benedikt Stehno, Helwig Hauser, *Member, IEEE CS*, Werner Purgathofer, *Member, IEEE CS*

Abstract—In this paper we propose a novel approach to hybrid visual steering of simulation ensembles. A simulation ensemble is a collection of simulation runs of the same simulation model using different sets of control parameters. Complex engineering systems have very large parameter spaces so a naïve sampling can result in prohibitively large simulation ensembles. Interactive steering of simulation ensembles provides the means to select relevant points in a multi-dimensional parameter space (design of experiment). Interactive steering efficiently reduces the number of simulation runs needed by coupling simulation and visualization and allowing a user to request new simulations on the fly. As system complexity grows, a pure interactive solution is not always sufficient. The new approach of hybrid steering combines interactive visual steering with automatic optimization. Hybrid steering allows a domain expert to interactively (in a visualization) select data points in an iterative manner, approximate the values in a continuous region of the simulation space (by regression) and automatically find the "best" points in this continuous region based on the specified constraints and objectives (by optimization). We argue that with the full spectrum of optimization options, the steering process can be improved substantially. We describe an integrated system consisting of a simulation, a visualization, and an optimization component. We also describe typical tasks and propose an interactive analysis workflow for complex engineering systems. We demonstrate our approach on a case study from automotive industry, the optimization of a hydraulic circuit in a high pressure common rail Diesel injection system.

Index Terms—Interactive Visual Analysis, Integrated Design Environment, Simulation, Visual Steering, Automatic Optimization

1 Introduction

Recent advances in computation technologies provide an opportunity to compute large simulation ensembles — multiple simulation runs of the same simulation model using different sets of control parameters. Parameter spaces of complex engineering systems, if not carefully sampled, can result in prohibitively large simulation ensembles.

Current emission regulations and efficiency goals are great challenges for automotive systems designers. In order to meet strict time constraints and reduce the time to market, system designers of modern automotive systems need powerful design tools to understand the systems, their behavior, and their responses to changes of the design parameters. In this paper, we present a case study dealing with an injection system, i.e., one of the key components of modern car engines. Target users of the proposed solution are designers of complex systems that are based on simulation ensembles. This paper is a result of a long-term collaboration between visualization and simulation experts. We, a team of visualization, simulation, and injection experts, developed the proposed approach, inspired by the actual application in the automotive industry. Our collaboration included numerous interviews and common sessions. We had regular meetings on a weekly basis for more than six months. One of the injection experts with more then 15 years experience in simulation, also coauthors the paper. Additionally we observed and interviewed four more simulation experts. Hence, when we say *we* throughout the paper, we mean the whole team, the visualization and the simulation experts. In our opinion, neither group alone could come to such a solution. The new approach is the result of a long-term research effort to address and overcome the described obstacles in the design of a complex system. Although we developed the newly proposed approach with experts from the automotive industry we are confident that the proposed approach can be used in other domains where complex simulation data in high dimensional parameter spaces have to be explored and understood.

Properly specifying the simulation parameters is a tedious task, and, at the same time, crucial for the effective utilization of simulation. There is an inherent trade-off between the simulation accuracy and its speed. Better accuracy requires more simulation points (i.e., simulation runs) to better cover the parameter space. More simulation runs increase the simulation time and lengthen the design process. A system designer needs help to navigate the simulation space and explore the most promising combinations of simulation parameters. With proper support, the designer can be more efficient and productive.

Simulation results often have a complex structure and a simplified representation. A common workflow includes the extraction of certain scalar features prior to the analysis. These features are then studied in the automatic and interactive analysis. Current state of the art techniques also support the consideration of complex data in interactive studies [17], but these techniques do not support an automatic analysis at the same time. Our work targets the interactive hybrid visual steering of a simulation ensemble which combines simulation and optimization with interactive visual steering to provide an integrated design environment.

The identified tasks for an integrated, hybrid steering environment are summarized in Table 1. These tasks are abstractions of the observed real-world practices and concrete tasks/activities in the automotive design workflow. Supporting these tasks is the key requirement that guided the development of our integrated, hybrid steering environment and our hybrid visual steering approach.

The main contributions of this paper are: **(1)** A case study demonstrating Hybrid Visual Steering, a novel simulation ensembles steering and exploration approach. This approach combines interactive explo-

- *Krešimir Matković is with VRVis Research Center, Vienna, Austria. E-mail: Matkovic@VRVis.at.*
- *Denis Gračanin is with Virginia Tech, Blacksburg, VA USA. E-mail: gracanin@vt.edu.*
- *Mario Jelović is with AVL-AST Zagreb, Croatia. E-mail: mario.jelovic@avl.com.*
- *Rainer Splechtna is with VRVis Research Center, Vienna, Austria. E-mail: Splechtna@VRVis.at.*
- *Benedikt Stehno is with VRVis Research Center, Vienna, Austria. E-mail: Stehno@VRVis.at.*
- *Helwig Hauser is with University of Bergen, Norway. E-mail: Helwig.Hauser@uib.no.*
- *Werner Purgathofer is with Vienna University of Technology, Austria. E-mail: wp@cg.tuwien.ac.at.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

Table 1. Hybrid steering tasks abstractions.

A	Explore and Analyze the Ensemble	
A1	**Parameters' Sensitivity**	Identify simulation results for certain control parameters and explore the parameters' sensitivity.
A2	**Model Reconstruction**	Identify control parameters for a desired output.
A3	**Comparison**	Compare output results related to different areas of the parameter space.
R	**Compute the Regression Model**	
R1	**Model Validation**	Show the model accuracy across the parameter space.
R2	**Model Definition**	Partition the parameter space and define relevant parts for the regression model building.
R3	**Automatic Optimization**	Automatic optimization using the regression model.
D	**Generate the Data**	
D1	**Initial Parameter Space Sampling**	Select regions in the parameter space to be initially sampled.
D2	**Interactive Refinement**	Select regions in parameter space which have to be resampled.
D3	**Automatic Optimization Refinement**	Choose refinement regions based on automatic optimization.

ration and analysis with automatic optimization based on regression models; **(2)** The task abstractions (Table 1) and the supporting visualization system, including two improved views, the Parameters Exploration View and Regression Exploration View. **(3)** The tight integration of all relevant components in an interactive workflow; and **(4)** An evaluation of the proposed approach based on a case study from the automotive industry including user feedback.

We build on our previous work [17, 18, 19, 21, 20] which integrates multiple simulation runs and visualization and focuses exclusively on the interactive exploration and steering. Here we introduce the adaptive exploration of the simulation space, based on regression modeling and the use of optimization to find an optimum within a subset of the simulation space. This new approach covers the spectrum between a fully automatic simulation and the manual adjustment of simulation parameters.

2 Related Work

Simulations are usually computationally intensive and are often combined with interpolation for sensitivity analysis and optimization. An example is the Kriging interpolator, representing a global metamodel that covers the whole experimental area [40]. However, we can also iteratively refine the simulation model, in addition to the refinement of the simulation parameter values.

If data analysis is a postprocessing step after a simulation batch is completed, errors invalidating the results of the entire simulation may be detected too late [25]. Computational steering and interactive visualization started in 1980s and 1990s as useful visualization paradigms for the computational sciences [11] enabling users to interactively steer computations, change simulation parameters and instantly see the simulation results. The simulation results are usually presented using scientific visualization methods [14].

Computational steering integrates modeling, computation, data analysis, visualization, and data management components of a simulation [25]. However, integrating simulation within computational steering can be a very difficult problem. We need to address four facets of the problem [15]: control structures, data distribution, data presentation, and user interfaces. Since computational steering is a highly interactive process, the user interface is a critical component [23]. This early simulation steering approaches usually deal with a single simulation run which lasts for a long time. The idea is to monitor simulation execution and to change some parameters if preliminary results seem to be wrong. We do rely on basic simulation steering principles, but we deal with simulation ensemble steering. Our simulation can be computed relatively fast, and we steer the ensemble creation, not a single simulation run.

In each iteration of computational steering the user can define a region of interest in the parameter space that should be explored in more detail. Additional simulation runs are needed to cover that region, constituting a new "simulation experiment". The design of such an experiment, i.e., the selection of the simulation points in the region of interest, is very important since we would like to reduce the number of simulation runs while providing a good coverage of the region of interest [16, 24].

While the support for a user controlled simulation is at the very core of computational steering, there is very limited support for user controlled optimization [3]. Very often there is no clear or unique optimal solution. A user has to analyze, in an interactive fashion, trade-offs and interdependencies between objectives [29, 32, 34]. Using an analytical representation of the objective function the user can explore the values of the objective function in the region of interest [22]. Such values can be dynamically updated in all views and brushes (selections) [28]. All these solutions are not integrated in an interactive steering environment. They focus on optimization based on a batch of precomputed simulation runs. In our case, we use optimization as a guideline in interactive steering in a fully integrated workflow.

The simulation data consists of discrete simulation points while the region of interest is usually a continuous space. We can use the simulation points to "span" that space using a surrogate (regression) model that approximates simulation results over the entire region of interest. The number of grid points in full grid methods depends exponentially on the number of dimensions. However, using sparse grids can reduce the dimensionality problem under some smoothness conditions. The sparse grid method, originally developed for the solution of partial differential equations [42], is also used for interpolation and approximation. The properties of the hierarchical representation and approximation properties of sparse grids are discussed by Bungartz and Griebel [9]. Improvements over the classical sparse grid approach include spatially adaptive refinement, modified ansatz functions, and efficient regularization techniques [26].

Simulation steering and dealing with ensemble simulations require control over multiple heterogeneous simulation runs. World lines [41] integrate simulation, visualization and computational steering to deal with the extended solution space by representing simulation runs as causally connected tracks that share a common time axis. The user has to select parameter combinations for new runs, there is no automatic support in selection of the new design points. Konyha et al. [17] and Matkovic et al. [19, 21] use interactive visual analysis for engineering problems with large parameter spaces. This is a purely interactive solution without an automatic support for steering. Berger et al. [2] employ statistical learning methods to predict results in real-time at any user-defined point and its neighborhood. The user is guided to potentially interesting parameter regions and the uncertainty of predictions is shown using 2D scatterplots and parallel coordinates. Booshehrian et al. [7] present a parameter space exploration approach from the fishery domain. These systems are not coupled with simulation, they operate on a set of predefined simulation runs. Engel et al. [13] describe a novel interactive visual framework for dimensionality reduction of high-dimensional single particle mass spectrometry data. Bergner et al. [4] present ParaGlide, a visualization system designed for interactive exploration of parameter spaces of multidimensional simulation models. They do initiate new data generation from the visualization, but the selection of points is based solely on user input, there is no support from automatic methods.

Machine learning techniques such as support vector machines [8, 12, 33] or relevance vector machines [36] can be used to create linear, quadratic or nonlinear surrogate models. The validation of a surrogate model is difficult in general [27]. We assume that our regression models are validated.

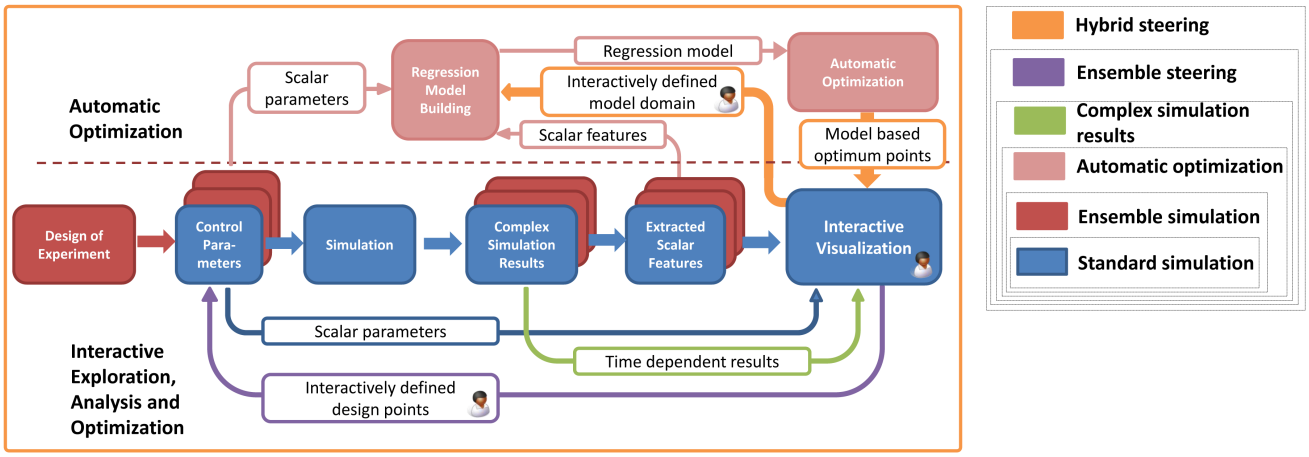

Figure 1. Overview of the proposed approach. **Standard simulation**: A collection of control parameter values is used for a single simulation run to determine and visualize simulation results (extracted scalar features). **Ensemble simulation**: Design of experiment methods are used to create several collections of control parameters. The resulting output values are aggregated and visualized together with the control parameter values. **Automatic optimization**: Aggregated parameter values are used to create a regression model which is used for optimization using the defined optimization constraints. This approach is usually decoupled from visual analysis, or visualization is used to show optimization results only. **Complex simulation results**: All complex simulation results are visualized. **Ensemble steering**: During visual exploration additional control parameter values for new simulation runs are selected by means of visualization. **Hybrid steering**: A unified approach which enables the exploration of parameters, complex results, extracted features and optimization results. Furthermore, it uses results from automatic optimization to guide the user during interactive visual steering. The hybrid steering also supports regression model building and optimization constraints and goals specification, all within the same framework.

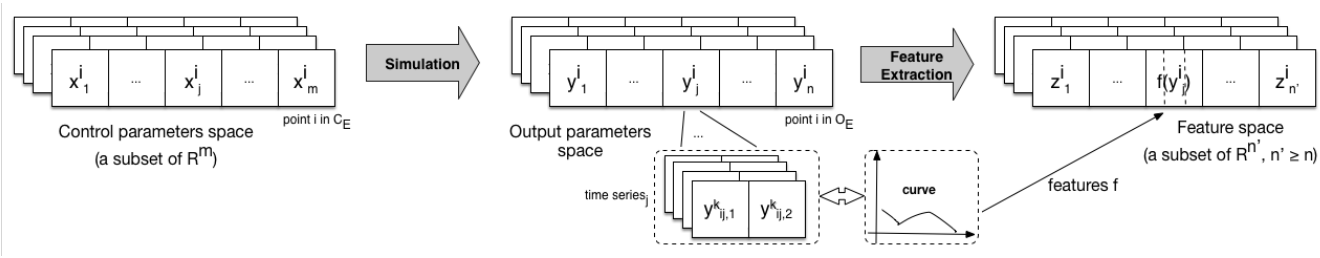

Figure 2. Simulation ensemble data model: control data points, output data points and features. For each output data point $\mathbf{y}^i$, there can be an output value y^i_j that is a time series (a curve). The time series is replaced by one or more scalar values (feature f) in the feature space.

Although the related work covers parts of our proposed solution, none of these approaches, according to our best knowledge, integrates all components in a unified framework.

3 Simulation and Visualization

Figure 1 illustrates our Hybrid Visual Steering approach and its evolution. The basic workflow in simulation includes model definition, setting of control parameters, simulation, feature extraction from complex simulation results, and the visualization of the extracted features. Feature extraction is necessary if the simulation produces complex data that is not suitable for standard direct visualization. The blue parts in Figure 1 correspond to such a traditional procedure. Advances in computation make it possible to compute many runs for the same simulation model with different sets of control parameters — a simulation ensemble.

In this case the parameter space has to be sampled and different combinations of control parameters have to be chosen. This is a well-known problem (design of experiment) for which there are several available techniques. A simulation ensemble results in a combination of multiple complex simulation results and multiple scalar features (red parts in Figure 1). If an automatic optimization is desired, a regression model can be computed based on the control parameters and the extracted scalar features. This step is usually decoupled from the visualization (light red parts in Figure 1). In our previous work we have demonstrated that also complex simulation results can be integrated in the visual analysis [17, 19] (green parts in Figure 1). Ensemble steering makes it possible to select new sets of control parameters from the visualization [20] (purple parts in Figure 1) in an iterative, interactive manner.

When the simulation space is very large, the iterative design process can be time consuming and tedious. We would like to help the domain expert by automatizing this process as much as possible. Therefore, we couple automatic optimization with the visualization in a hybrid visual steering environment (orange parts in Figure 1). Our framework supports all identified tasks for a complex system design.

Integrated design environments are not readily available for industrial design. Tools are used separately or as partially integrated tools which significantly reduces efficiency. The integrated design environment we developed for the common rail injection design resulted, according to the domain expert, in a speed up factor of at least ten compared to the conventional approach where all tools are used separately. We also talked with four more domain experts at the AVL company working on optimization, timing drive, hybrid vehicle, and crankshaft design. They informally evaluated our integrated design environment prototypes and estimated a similar potential for speed up.

3.1 Formal Background

We often model the simulation process as a function S that maps the control parameters $\mathbf{x} = (x_1, \ldots, x_m)$ (a control data point in $\mathbb{R}^m$) to the output values $\mathbf{y} = (y_1, \ldots, y_n)$ (an output data point in $\mathbb{R}^n$) where m is

the number of control parameters and n is the number of outputs. Due to the physical constraints of the simulated system, the set of feasible control data points C is a subset of $\mathbb{R}^m$ and the set of feasible output data points O is a subset of $\mathbb{R}^n$. A simulation ensemble E is a set of pairs of data points $(\mathbf{x},\mathbf{y})$, $\mathbf{x} \in C$ and $\mathbf{y} \in O$.

Traditional data analysis approaches, such as statistics or OLAP techniques [35], usually use a relatively simple multi-dimensional model [10, 31] (simple with respect to the types of data dimensions) using data tables to capture relations among control parameter values for the same simulation run. While the control parameters are almost always numerical, scalar values, the output values also often include time/data series so O is no longer a subset of $\mathbb{R}^n$.

Since simple data tables [10] are not sufficient, we need an adequate simulation ensemble data model to deal with more complex data. Our simulation ensemble data model uses a two-level data hierarchy for the output data points (Figure 2).

For each output data point $\mathbf{y}^i$ and y^i_j that is a data series, we have a separate set of "sub-points" with its own length and number of dimensions (parameters). Our discussion is limited to two dimensions. We can select one of the data series dimensions as the independent variable (e.g., time) and the other dimension as the dependent variable. Such data series can be considered a function of one variable and represented as a curve. The curve is replaced by one or more scalar values (features f) to create a feature point $\mathbf{z}^i$ in the feature space F, a subset of $\mathbb{R}^{n'}$, $n' \geq n$.

In other words, the scalar values y^i_s from $\mathbf{y}^i$ are directly used in the feature space as the corresponding values z^i_s in $\mathbf{z}^i$ while each data series value y^i_j from $\mathbf{y}^i$ is replaced by one or more scalar values (features f) in $\mathbf{z}^i$.

3.2 Regression Model

If we can approximate the mapping S from the control parameter values to the simulation results using a regression model, we can estimate the simulation results much faster, compared to actually running the simulation. In order to estimate S, a number of simulation runs are executed to get a set of input-feature pairs (simulation ensemble), $\{(\mathbf{x}^i,\mathbf{z}^i)\}$ as training data. A regression model R is built from the training data as the surrogate model of the simulation.

After the regression model is trained, we can then use it to estimate the simulation results for arbitrary input values. Or, more importantly, it can be applied in an optimization process, of which the goal is to obtain a set of control parameters so that the simulation results satisfy a set of desired constraints, such as the maximization of a linear combination of the output variables.

4 Integrated Interactive Steering Workflow

Hybrid Visual Steering integrates simulation, optimization, and visualization in a unified framework enabling the user to conduct simulation ensemble steering. After the computation of a set of initial runs the user explores the simulation ensemble, and detects a region of interest. New simulation runs are conducted to adaptively increase the resolution of the simulation points in that region and augment the simulation ensemble. Automatic optimization supports the user in identifying regions of interest. However, since the proposed optimum values are based on regression models (approximation) built based on scalar features extracted from actually more complex simulation data (again, an approximation), we can not rely on them. The user needs a hint on the regression model accuracy in the detected region, and dependent on this accuracy more or fewer additional runs will be computed. The overall workflow can be summarized as:

- Conduct simulation runs based on the design of experiment in the preceding iteration (for the first iteration create an initial design of experiment).
- Integrate the new simulation runs with the existing data (if there are any).
- Visually analyze the data and select an objective function.
- Set a regression model to explore the objective function.
- Identify a region of interest.
- Use optimization to determine (seemingly) optimal value(s).
- If the result of this optimization is not satisfactory, create a new design of experiment with an increased resolution.

The realization of a hybrid visual steering framework also represents a technical challenge. All components do exist in current design processes, but they are not coupled in a unified framework. Furthermore, not all of the components support the identified tasks, and they have to be extended. The following components are needed to realize an integrated hybrid visual steering environment:

- **Design of Experiment (DOE) Component**: supports the specification of a subspace of possible input values (a shape in a multi-dimensional space) and the specification of a distribution of points within this space.
- **Simulation Component**: simulates the phenomena of interest at a comparably accurate level.
- **Analysis and Exploration Component**: supports feature extraction, advanced interaction, and the visualization of complex and scalar simulation results. In the case of steering, it also supports the interactive selection of subspaces in the parameter space, and the specification of new design points. Finally, it also controls regression model building, evaluates it, and shows optimization results.
- **Regression Model Building Component**: builds a regression model from (a subset of) the already computed simulation runs.
- **Automatic Optimization Component**: computes an optimum using the regression model, subject to the interactively specified constraints.

We use AVL's [1] Design Explorer which supports DOE definition, regression model building, and automatic optimization and AVL's CruiseM for simulation. We extend the existing interactive analysis and exploration component to support all analysis tasks listed in Table 1. We exploit the well-known principle of coordinated multiple views and integrate all components in a unified framework.

5 Tasks and Steering Design

We have identified three main groups of tasks (Table 1). Each group of tasks has specific requirements. The main questions are how to visualize control parameters and simulation results, how to design the interaction, both for ensemble exploration and for optimization constraints as well as goals definition, and how to specify new points in the parameter space. Based on our accumulated experience and the current state of the art in exploratory visualization, we rely on the coordinated multiple views principle. The main idea is to depict multiple dimensions using several views and to allow the user to interactively select (brush) subsets of the data in a view. Consequently, all the corresponding data items in all linked views will be consistently highlighted.

5.1 Control Parameters Visualization

There are several possibilities for sampling the parameter space. The parameter space can be continuous, or discrete. However, even if the parameter space is continuous, we only select a limited, discrete set of parameters, resulting in a discrete parameter sample.

As we deal with a multi-dimensional parameter space, well-known techniques from visualization can be used. In the case of a continuous parameter space, parallel coordinates are often used [27]. Projections to 2D using scatterplots are also frequently used. Furthermore, histograms and bar charts are also regularly used to show values of different parameters [7].

We needed a view which supports the identified tasks (Table 1, tasks A1–A3, R1–R3, D1–D3). We can abstract the tasks on a finer level when it comes to the parameter space visualization and identify general task requirements:

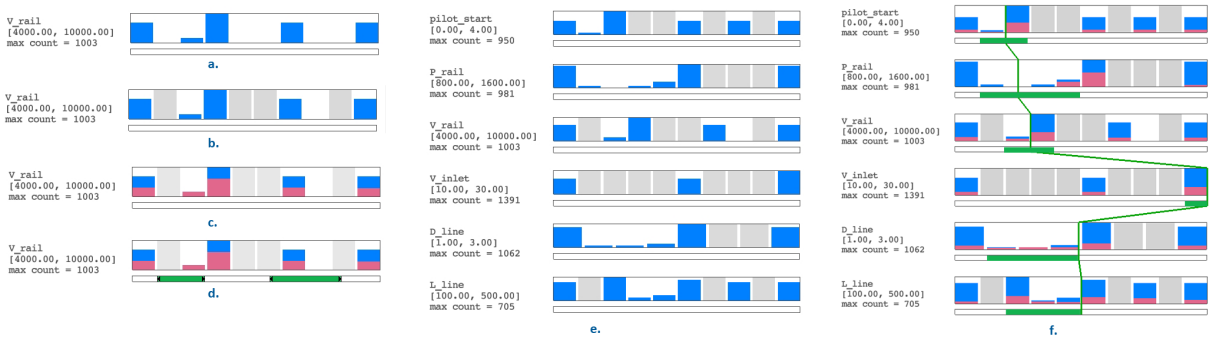

Figure 3. Parameters Exploration View: **a)** Histogram for one parameter. Note that bins which are shown empty might also contain just a few items which are not visible if the display resolution is too low. **b)** Empty bins are shown in a different color (grey). This design was preferred by domain experts over highlighting non-empty bins. **c)** Histogram showing brushed runs (red) in relation to the overall distribution (blue). **d)** Constraints bar below the histogram that is used to specify optimization constraints. **e)** Six parameters shown in the view. **f)** Six parameters with constraints and optimum values. Some items are brushed and they are shown in red.

- Show the distribution of each parameter (A1–A3, R2, D1, D2).
- Show the distribution of brushed simulation runs (A1, A2, A3).
- Support the interactive specification of optimization constraints for the parameters (R3).
- Initiate regression model building (R2).
- Show the automatically computed optimum values (A3, R3).
- Keep track of the automatic optimization process (R3).

None of the standard views supports all these requirements. We designed a new view — the Parameters Exploration View which meets the design requirements. It is inspired by the attribute explorer [39]. We add additional bars for specifying optimization constraints and additional interaction capabilities.

The basic idea is to show each parameter as a histogram with a user defined bin count. Figure 3a shows the histogram for one parameter. Basic information, like the parameter name, its range, and the maximum number of counts across all bins are shown on the left. We do not show this information under the histogram as we stack histograms vertically. There are five bins in the histogram in Figure 3 that seem to be empty. As the number of runs per bin can vary, and the histograms have a limited vertical space, some non-empty bins can occupy only one or even less than one pixel. We realized that it is important to mark really empty bins.

During an informal user study with five engineers at the AVL company, we presented them with two alternatives, explicitly marking empty bins and explicitly marking non-empty bins. All engineers preferred the solution where empty bins are marked with a gray rectangle. Figure 3b shows such a solution. We see that there are only four empty bins. There is a bin with just a few runs, which is not marked as empty. This information cannot be seen in Figure 3a.

The view is fully integrated in the coordinated multiple views environment and also shows the brushed runs. Figure 3c shows one histogram showing brushed runs (red) in addition to the overall distribution (blue).

When designing a complex system the user wants to test various hypotheses and adjust optimization constraints. The optimization constraints change during the analysis process. We add a constraints bar below the histogram (green bars in Figure 3d) to support the interactive specification of constraints. The constraints bar is used to specify constraints for each parameter. As we binned the parameters a simple click in the constraints bar sets the constraints to the specified bin. The user can extend the constraints bar by additional clicks or simply by dragging one side of any rectangle in the bar. The dragging includes additional bins per default, but it can also be set to specify ranges not aligned with the borders of the bins.

Figure 3e shows the Parameters Exploration View for six control parameters. The computation of a new regression model is also initiated from the Parameters Exploration View. In the left part of the view, which we call view control (not shown in Figure 3), there are two buttons which start the process. One initiates the use of all simulation runs and the other computes the regression model based on the brushed runs only. The user can also choose a regression building method.

We also want to show the computed optimum values. We depict them as a polyline, passing through all histograms, similar to a parallel coordinates polyline. The line passes through the constraints bar as well. The lines are depicted in two colors, depending if they are a result from a regression model that was computed based on all runs or only on a subset of all runs. As the user hovers over the optimum line, corresponding constraints are shown in the constraints bar. The user can also hide the optimum lines if they are not of interest. There is a list of all computed optimum values (not shown in the figures) with user specified names, so the user can activate hidden optima on demand. Figure 3f shows the view for six parameters with an automatically computed optimum. The view does not only show brushed values but can be used to brush as well. A simple click on a bin selects the corresponding parameter values.

5.2 Simulation Results Visualization

Both parameters and the simulation results have to be shown together. As described in Section 5.1, we deal with a complex data model where scalars and curves are considered as elementary data units. In the case of an ensemble, this means that for each dimension we have a collection of scalars (as usual) or a collection of curves. We call all curves in an ensemble that belong to the same dimension a family of curves [17].

When it comes to the exploration and analysis of ensemble simulation data (tasks A1, A2, and A3), we have to show the simulation results. Standard views, such as scatterplots, parallel coordinates, or histograms are most often used during our study. For more complex outputs, in particular for the families of curves, we use a curve view which depicts all curves simultaneously also employing a certain density mapping, when needed. Brushing is extensively used in the exploration of such ensemble simulation data. Having all complex outputs depicted, we can easily realize the reconstruction of a model (task A2) which is a tedious or impossible task using analytical methods only.

We use regression models that are evaluated and tuned. There are multiple methods for evaluating and tuning a regression model (a detailed description is beyond the scope of this paper). We are dealing with a high-dimensional space and many time-dependent state variables are represented by aggregates. Although the model is tuned, it is practically impossible to find a model which fits all simulation points. Therefore, we also need to show the accuracy of the regression model (task R1). If an automatically computed optimum belongs to an area

where the regression model is not a good approximation of the simulation results, we need more simulation runs.

In order to show the model's accuracy, we simultaneously show simulation results and approximated results (from the regression model) in the Regression Exploration View. The Regression Exploration View uses a projection of the high-dimensional output (solution) space (simulation results and regression model results) onto a plane determined by two selected output values (a scatter plot). During the design process a designer should always explore several such projections (different pairs of output values).

There are several ways to visualize pairs of points in the scatterplot. We showed different designs to five experts from the AVL company. We did not end with a uniquely preferred solution, as different tasks require different approaches. Throughout the following examples the simulation based points are orange and the regression model based points are blue. Of course, the user can also configure these colors.

The first idea is to show the pairs of points and connecting lines. Figure 4a shows such a case. In an ideal case the orange and the blue points would overlap and there would be no lines. In a realistic case the lines depict the accuracy of the regression model. The lines help in deciding if a computed point is a good enough approximation in a certain region. In case of many points (Figure 4a) there is a large overlapping problem and the user gets just a rough impression of accuracy. However, during a drill-down process in the analysis, the number of relevant pairs is reduced and the view becomes more useful. Figure 4b shows the same view for a subset of points. The lines are easier to identify now, and there is in general less clutter.

We also enable the user to show only simulation or only approximation points, and then, to use color coding and point size to encode the accuracy (the line length in the other solution). The view is less cluttered as it has less points (Figures 4c and 4d). We also offer a design where only color or only point size is used, but all domain experts preferred the combined approach. Figure 4e shows the same data as in Figure 4c using color only. The user can turn the context on or off. The context use was very task specific, users sometimes preferred to see the context and sometimes they wanted no context.

Furthermore, the users are also considering the direction and magnitude of the lines. Inspired by the work of Turkay et al. [38, 37], we also show only the deviation lines. In this case all start points are moved to the origin, and the lines and the end points are shown. In an ideal case there would be no lines, again. With this setup, it is easy to brush all lines that have a certain slope and/or magnitude. Figure 4f shows such a visualization. Figure 5 shows two different areas, one where the model is more accurate (Figure 5a) and one where differences between the simulation and the regression are much bigger (Figure 5b). Figure 5b also shows the context.

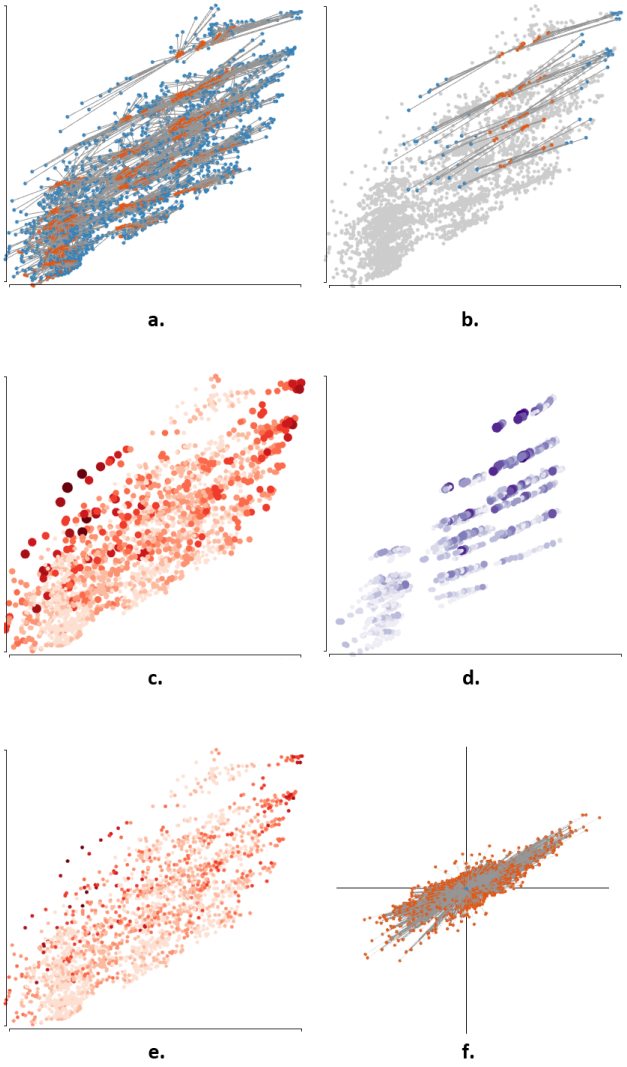

Figure 4. Regression Exploration View: A projection of the high-dimensional data points on a plane determined by two output values, X_max and Q_inj_max. The simulation results are shown in orange. The regression model results are shown in blue. **a)** Simulation points and corresponding regression model points are connected by a line to make the difference between them visually explicit. The view is cluttered due to the large number of points. **b)** Only brushed points are visualized. As there are much less points, clutter is also reduced. The context shown in gray can be turned off. **c)** and **d)** Alternative visualizations where only simulation points (**c**), or only regression points (**d**) are shown. The length of the lines (inverse accuracy of the model) is coded using color and point size at the same time. All points are shown (as in **c**). **e)** Only color is used to code the accuracy, the view **c** is preferred by domain experts. **f)** All lines are drawn from the origin which eases to brush points having a large deviation, or which deviate in a certain direction.

6 Case Study

We use an example from the automotive industry to evaluate our proposed approach. Together with five domain experts we conducted several analysis sessions in order to optimize a common rail Diesel injection system. All of these experts have a long (10–20 years) experience in automotive simulation. The sessions were conducted at AVL [1], one of the worldwide leaders in powertrain measurement and simulation systems.

Modern simulation software can be used to simulate injection systems and to help engineers to understand and tune injection parameters. Many phenomena can not be easily measured experimentally and the only way to get more information is through computational simulation. We used the AVL CruiseM simulation software to simulate a complete injection system [1]. We simulate an injection system consisting of four injectors. The injectors are produced by only few manufacturers. Hence the engine manufacturers have to tune them according to their specific needs. The whole system, including the rail and high-pressure pipes, is also different for each engine. The simulation model has more than 200 elements. Each element in the simulation model has several control and state variables. In this case study we focus on 6 parameters and explore different simulation results.

The main principle of a common rail system is the use of a high-pressure rail, common to all cylinders. The high pressure in the rail is used to precisely inject fuel into the cylinders. Electronically controlled actuators open and close the injectors. Sometimes, in one cycle, the main injection is preceded by a pilot injection, or even several of them. All this happens at least several hundred times per second. A more detailed description of common rail injection is beyond the scope of this paper [5, 6].

Due to high pressures and quick changes in the system, a modern common rail injection system operates in a condition which cannot be described sufficiently precise using classical fluid mechanics. Further-

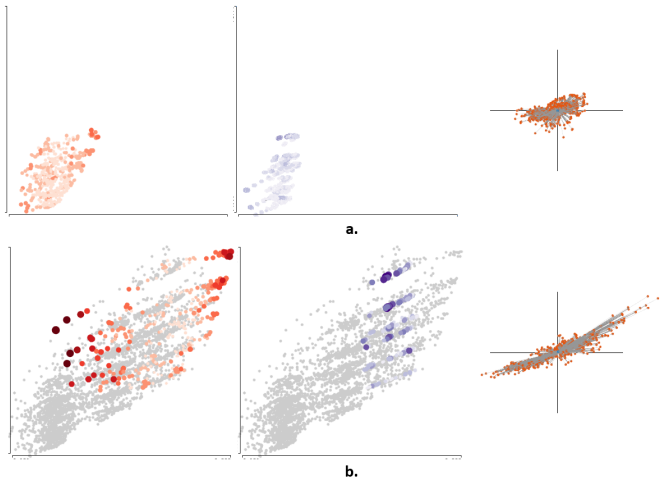

Figure 5. Two interesting areas are brushed. **a)** The regression model is quite accurate here. **b)** A much less accurate area is identified.

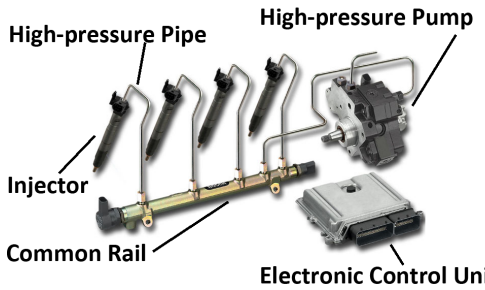

Figure 6. A modern common rail injection system used in Diesel engines of cars. There is one injector for each cylinder, and a common rail which utilizes high pressure (over 1500 bar) to precisely control the injectors' opening and closing.

more, in a common rail system each cylinder and injector is influenced by the others through the rail. This requires a careful rethinking of traditional system design. Figure 6 shows a modern injection system for a four-cylinder engine.

When tuning an injection system, engineers have a set of goals they have to meet. If the process values are not within a certain range, the engine will either run inefficiently or not at all. These results include mass injection rates, injection pressures (for each injector and the overall system), and the pilot and main injections time intervals. Corresponding control parameters for the desired ranges have to be explored (task A2, model reconstruction). A poor injection automatically leads to poor combustion, thereby increasing consumption, pollution, and power loss. An additional challenge in the design of a common rail injection system is to understand and prevent the appearance of unwanted pressure oscillations. Modern systems operate at a pressure of over 1500 bar. The oscillations can lead to amplitude jumps of over 400 bar. Such large oscillations can cause an undesirable behavior of individual injectors and introduce differences among the injectors. This results in a reduced efficiency and an increased emission — exactly the opposite of the design goals.

In this case study, we are interested in the high-pressure pipe geometry and the common rail itself. We have studied the geometry of the common rail and the high pressure pipes and the influence of the common-rail pressure and the pilot injection timings on the overall system performance.

There are four injectors with pilot and main injections, four high-pressure pipes and the common rail. We need advanced tools to comprehend the behavior of such a complex system. We assume that the individual injectors are tuned, and we study (for two pilot injections and the main injection) the injection pressure, the amount of injected fuel, the needle opening velocity and the needle closing velocity.

We are exploring the system at the individual cylinder level and at the overall system level. At the cylinder level we aim at the following:

- Small differences among injection pressures of the individual pilot injections.
- Small differences among the amounts of the injected fuel during the individual pilot injections.
- Maximum possible needle opening and closing velocities for two pilot injections and the main injection.
- Good damping of pressure oscillations that can occur within the high-pressure pipe.

On the overall system level we are looking for minimum possible differences among injection pressures for the individual cylinders and among the injected amounts of fuel for the individual cylinders. Besides these goals, the injection curves have to be of certain shapes, depending on the engine operation regime.

6.1 Iterative Analysis

After creating the model we need to decide which parameters will be varied. We focus on the six most relevant parameters: *L_line* and *D_line* (the length and diameter of the high pressure pipe), *V_rail* and *rail_pressure* (the volume of the rail and the pressure inside the rail), *V_inlet* (the volume of a junction between the rail and the high pressure pipe) and *pilot_start* (the starting time of the first pilot injection measured in degrees of crankshaft rotation).

We vary each control parameter and compute 2700 simulation runs. One simulation run takes approximately 200 milliseconds on a standard single core desktop PC. Several simulations can run simultaneously when multiple cores are available.

More than 30 output values (time series — functions of the crank angle) are computed per run. Once the simulation values are computed, more than 30 features per run are computed as well. Some of the features are computed from one curve (e.g., its maximum), and some are based on several curves. Table 2 shows a small subset of the computed scalar features.

All values of each run are computed for two revolutions of the crankshaft (720 degrees). The result is a complex data set where each record has some scalar attributes and some time series attributes (all state parameters). Since the computation of the regression model and the optimization expect scalar values, extracted scalar features are used to approximate the model.

After the computation of the initial set of runs we started the data exploration and analysis. Not all of the runs generate feasible results. Some the high pressure parameter combinations result in a too low injected fuel mass, for example. Figure 7 shows the initial setup of our analysis. The Parameters Exploration View is placed in the upper left corner and it remained there during the whole session. The other views are often reconfigured, depending on the current analysis goals.

Table 2. Scalar features of time series simulation values.

State parameter	Explanation
P_m_diff	Absolute difference between the maximum pressure during the main injection for the first and the third cylinder.
m_m_diff	Absolute difference between the injected fuel mass during the injection into the first and the third cylinder.
K_d_3	A damping value.
P3_pt_diff	Absolute difference between the maximum pressure for the first and the second pilot injection (third cylinder).
m3_pt_diff	Absolute difference between the injected fuel mass during the first and the second pilot injection (third cylinder).

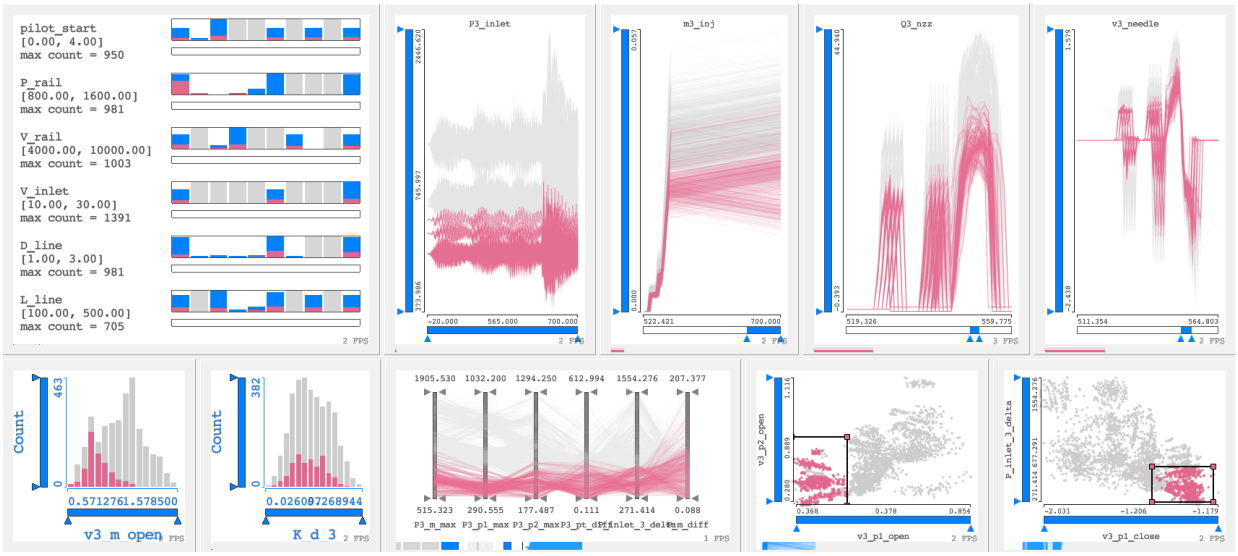

Figure 7. The initial view configuration (more than 2700 records). The Parameters Exploration View is shown in the upper left corner. The other views depict selected state parameters studied during the analysis.

Three of the five users had experience with interactive visual analysis prior to this case study. One user reported after the study that *"Seeing all results at once, although some are not feasible, represents a great advantage. I can see results in context now, and not each run separately as up to now."* In order to explore how outputs change when control parameters change (task A1) we use brushing. By moving the brush across the parameter space all corresponding outputs are highlighted consistently. Another user commented *"Seeing how outputs change when I move the brush adds additional quality to conventional sensitivity analysis. I get a much better impression now."*

The model reconstruction task focuses on finding parameters which result in desired values for scalar outputs, and certain curve shapes for curve outputs. Curve shapes are crucial for an efficient injection. For model reconstruction (task A2), we simply select the desired values of the scalar outputs and the desired curve shapes. This is practically impossible to do analytically becomes very intuitive now. We brush desired shapes and undesired shapes as well. Some oscillations occur for some combinations of control parameters and it is important to understand when this happens. We also use multiple composite brushing. The user can select different subsets in different colors and interactively change any selection. This proved to be perfect for various comparisons (task A3), as stated by one user *"Comparing several scenarios is straightforward using multiple brushes. Similar comparisons are simply impossible using a conventional workflow where we analyze each run separately, and compare the results afterwards."*

We identify the area in the parameter space where a possible optimum could be. Automatic optimization is used to find the optimum instead of trying to regularly fill in the range with new design points (which would result in many additional simulation runs). The automatic optimization can be started from within the tool. The parameters view is used to specify which runs are included and the regression model is created. From now on all scalar values can be computed using the regression model. We specify the optimization constraints and compute the optimum based on the regression model. All these steps are performed in the integrated environment that was appreciated by a user: *"I could never set up optimization so fast. I also see all results together with the initial runs."*

We run the simulation (not the regression model) for the optimum point. The simulation tool is started automatically, a new run is computed, and the simulation results are loaded in the visualization tool. All outputs are computed by the simulation for the computed optimum point. Of course, these values differ from the values computed using the regression model. While the regression model is only an approximation we know it is precise enough to indicate the region of the optimum in the specified subspace. However, we are aware that the optimum is computed using an approximation, so we propose to compute additional points in the neighborhood of the optimum. For each additional point a simulation run is needed. The number of additional points can be specified dependent on model accuracy in the optimum's neighborhood. In this case we generate 244 new points around the optimum. The points are mostly in the constrained space, but we allow a slight deviation if the optimum is on a border. The points are generated using the full-factorial algorithm [30].

The computed optimum point is depicted in the Parameters Exploration View and in the other views. If better points exist, they are displayed and can be selected by the user. One user commented: *"The suggestions of where to refine the parameter space based on optimization speeds up the steering. The model accuracy shown indicates the quality of the optimization results. Seeing all runs all the time is simply unmatched in a conventional workflow. I can also see all curves which are normally not available when an optimization based on scalar features is conducted. I would estimate a speedup of at least an order of magnitude when designing complex systems."*

Figure 8 illustrates a typical workflow from an analysis session.

6.2 Regression Model Modification

The following example illustrates how the regression models influence the computed optimum. At one stage, we had 3188 simulation runs in the ensemble, as we had continuously added runs to the ensemble. We computed an optimum and additional points in the neighborhood. We wanted to minimize *P_m_diff*. The scatterplot in Figure 9a shows the scatterplot where *P_m_diff* is on the *x* axis. The computed minimum is dark green and all points from the optimum's neighborhood are pink. We clearly see that there are better points in the neighborhood. The same data in the parallel coordinates (Figure 9b) shows two clusters among the pink items on the first axis, *P_m_diff*. The optimum is rather high, and it seems as if the upper cluster is attracting the optimum.

It is possible that we have a high gradient which influences the regression model. A new regression model is computed based on a subset of the points only. The points with high *P_m_diff* values are excluded, and new optimum is computed. The scatterplot (Figure 9c) and parallel coordinates (Figure 9d) show the original optimum and the new optimum in light green. The new model fits much better and the new optimum is smaller (remember that we wanted to minimize *P_m_diff*). We also computed the optimum using a model based on even more neighboring points but the outcome was almost the same.

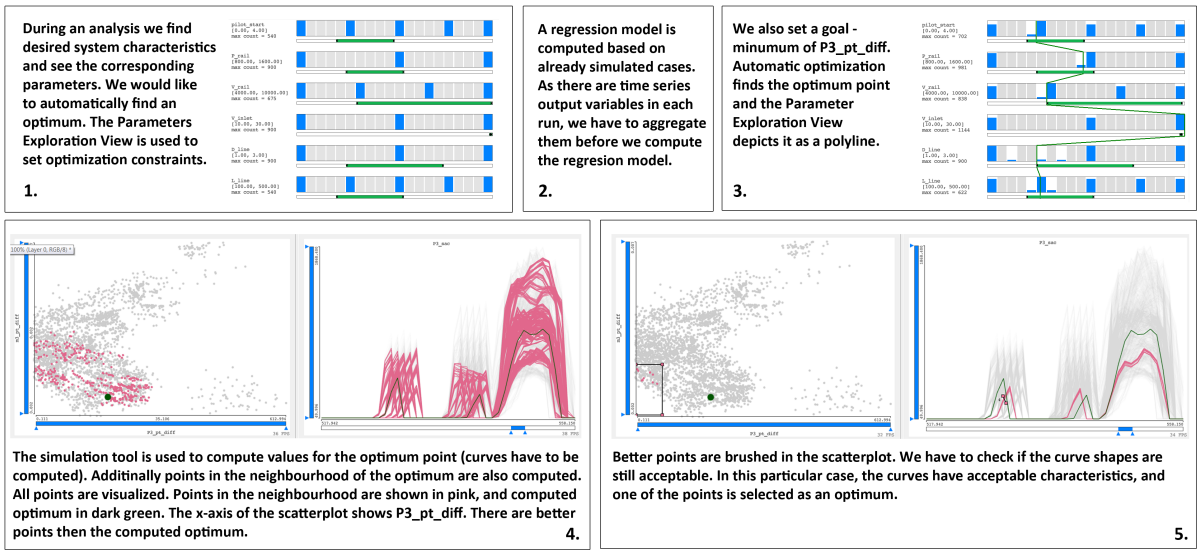

Figure 8. An example of a typical analysis iteration. **1.** The Parameters Exploration View is used to set the initial ranges of the control parameters. **2.** A regression model is computed based on the already computed simulation runs. Time series data are aggregated. **3.** The Parameters Exploration View shows the first computed optimum. The green poly-line connects the optimal control parameters. **4.** Left: the points in the neighborhood of the computed optimum are simulated and depicted in pink. Right: the corresponding curves. It is obvious that there are better points than the automatically suggested one. **5.** We additionally check if the curves have acceptable shapes. Left: the selected points and the optimum are highlighted. Right: the corresponding curves.

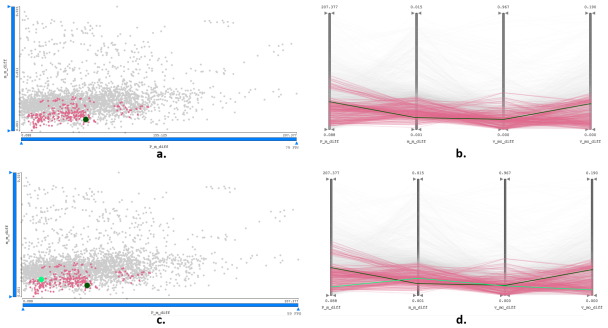

Figure 9. **a)** A scatterplot showing the first optimum (*P_m_diff*), dark green. **b)** The parallel coordinates showing the first optimum, dark green. **c)** The scatterplot showing the first and the second optimums (*P_m_diff*), light green. **d)** The parallel coordinates showing the first and the second optimums, light green.

The curve shapes for the new optimum are examined and verified by users: *"Hybrid steering makes us aware of automatic optimization limits. We can easily see if results are right or not, and we can quickly refine the model if needed. I used steering before, but the addition of automatic optimization improves it significantly."*

7 Discussion and Conclusions

The new Hybrid Visual Steering approach represents an integrated design environment for simulation, visualization, and optimization. The development of this approach would have been impossible without a close collaboration with domain experts through numerous sessions over several months. The improvements and time savings are significant when compared to the conventional approach. The integrated design environment manages complex data (no tedious file conversions), keeps track of the process and of all optima found during the process.

We illustrated the approach on the common rail injection system design but the approach is not limited to the injection design only. We talked to the domain experts working on different automotive systems (timing drive, crankshaft balancing, and overall driving comfort) and they anticipate similar potential speed ups for their design problems. The approach can be applied whenever there is a complex system that can be represented by a simulation model with a multidimensional parameter space (either continuous or discrete), and relatively fast simulation. The simulation could be any algorithm that computes something based on control parameters. We are currently exploring applying the approach to image segmentation in the medical domain. We are also planning to extend the system for air flow simulation and traffic simulation.

The initial discussions we had with experts from all these domains make us confident that Hybrid Visual Steering can be applied in many scientific and engineering domains. Of course, the individual components of the system (simulation tool, regression model building tool, optimization, and visualization) should be modified according to the specific domain, but the main methodology and the workflow (see Section 4) remains the same. The modifications should be done in consultation with the domain experts as every domain has its own requirements, conventions and standards.

The interplay between the parameters of complex systems is so intricate that the expert's intuition and knowledge can not be represented by an automatic system. Hence it is important to have an interactive system. Only human experience, knowledge, and imagination, supported by automatic methods can yield the best-possible results.

In this paper we described our experiences with the injection system design. Initially, the common rail injection system design process was done using a number of isolated tools. It was necessary to create an integrated design environment (Figure 1) that supports simulation, visualization and optimization. The evaluation of the proposed approach by five domain experts demonstrates the viability of the proposed approach, advantages over the existing design practice, and its usefulness in everyday industrial design. The integrated design environment which was deployed in the context of the case study is currently used by AVL. The intent is to make it a standard part of AVL's commercially available software suite.

Acknowledgments

Part of this work was done in the scope of the K1 program at the VRVis Research Center.

REFERENCES

[1] AVL. AVL List GmbH. http://www.avl.com/, 2013. [last accessed 25 June 2014].

[2] W. Berger, H. Piringer, P. Filzmoser, and E. Gröller. Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Computer Graphics Forum*, 30(3):911–920, June 2011.

[3] W. Berger and H. Pringer. Interactive visual analysis of multiobjective optimizations. In *Proceedings of the 2010 IEEE Symposium on Visual Analytics Science and Technology*, pages 215–216, 24–29 Oct. 2010.

[4] S. Bergner, M. Sedlmair, T. Möller, S. N. Abdolyousefi, and A. Saad. ParaGlide: Interactive parameter space partitioning for computer simulations. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1499–1512, Sept. 2013.

[5] F. Boecking, U. Dohle, J. Hammer, and S. Kampmann. Passenger car common rail systems for future emissions standards. *MTZ worldwide*, 66(7–8):552–557, 2005.

[6] W. Boehner and K. Hummel. Common Rail Injection System for Commercial Diesel Vehicles. *SAE Transactions*, (SAE 970345), 1997.

[7] M. Booshehrian, T. Möller, R. M. Peterman, and T. Munzner. Vismon: Facilitating analysis of trade-offs, uncertainty, and sensitivity in fisheries management decision making. *Computer Graphics Forum*, 31(3pt3):1235–1244, June 2012.

[8] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT '92)*, pages 144–152, New York, 1992. ACM.

[9] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[10] S. K. Card, J. Mackinlay, and B. Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Interactive Technologies. Morgan Kaufmann, 1999.

[11] J. X. Chen, D. Rine, and H. D. Simon. Advancing interactive visualization and computational steering. *IEEE Computational Science & Engineering*, 3(4):13–17, Winter 1996.

[12] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[13] D. Engel, K. Greff, C. Garth, K. Bein, A. Wexler, B. Hamann, and H. Hagen. Visual steering and verification of mass spectrometry data factorization in air quality research. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2275–2284, Dec. 2012.

[14] H. Hagen, A. Ebert, R. H. van Lengen, and G. Scheuermann. Scientific visualization: methods and applications. In *Proceedings of the 19th spring conference on Computer graphics (SCCG '03)*, pages 23–33, New York, NY, USA, 2003. ACM.

[15] C. Johnson, S. G. Parker, C. Hansen, G. L. Kindlmann, and Y. Livnat. Interactive simulation and visualization. *IEEE Computer*, 32(12):59–65, Dec. 1999.

[16] J. P. C. Kleijnen. *Design and Analysis of Simulation Experiments*. International Series in Operations Research & Management Science. Springer, New York, 2007.

[17] Z. Konyha, K. Matković, D. Gračanin, M. Jelović, and H. Hauser. Interactive visual analysis of families of function graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1373–1385, Nov./Dec. 2006.

[18] K. Matković, M. Djuras, D. Gračanin, R. Splechtna, B. Stehno, and H. Hauser. Interactive visual analysis in the concept stage of a hybrid-vehicle design. In *Proceedings of the EuroVA 2013 - EuroVis Workshop on Visual Analytics*, 17–18 June 2013.

[19] K. Matković, D. Gračanin, M. Jelović, A. Ammer, A. Lež, and H. Hauser. Interactive visual analysis of multiple simulation runs using the simulation model view: Understanding and tuning of an electronic unit injector. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1449–1457, Nov.-Dec. 2010.

[20] K. Matković, D. Gračanin, M. Jelović, and H. Hauser. Interactive visual steering — rapid visual prototyping of a common rail injection system. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1699–1706, Nov.-Dec. 2008.

[21] K. Matković, D. Gračanin, M. Jelović, and H. Hauser. Interactive visual analysis supporting design, tuning, and optimization of diesel engine injection. In *VisWeek 2011: Discovery Exhibition*, 23–28 Oct. 2011.

[22] K. Miettinen and M. M. Mäkelä. Synchronous approach in interactive multiobjective optimization. *European Journal of Operational Research*, 170(3):909–922, May 2006.

[23] J. D. Mulder, J. J. van Wijk, and R. van Liere. A survey of computational steering environments. *Future Generation Computer Systems*, 15(1):119–129, 1999.

[24] L. C. Onyiah. *Design and Analysis of Experiments: Classical and Regression Approaches with SAS*. Statistics: Textbooks and Monographs. CRC Press, Boca Raton, FL, 2009.

[25] S. G. Parker, C. J. Johnson, and D. Beazley. Computational steering: Software systems and strategies. *IEEE Computational Science & Engineering*, 4(4):50–59, Oct.-Dec. 1997.

[26] D. Pflüger, B. Peherstorfer, and H.-J. Bungartz. Spatially adaptive sparse grids for high-dimensional data-driven problems. *Journal of Complexity*, 26(5):508–522, Oct. 2010.

[27] H. Piringer, W. Berger, and J. Krasser. HyperMoVal: Interactive visual validation of regression models for real-time simulation. *Computer Graphics Forum*, 29(3):983–992, June 2010.

[28] H. Piringer, C. Tominski, P. Muigg, and W. Berger. A multi-threading architecture to support interactive visual exploration. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1113–1120, Nov.–Dec. 2009.

[29] H. Pirkul, R. Gupta, and E. Rolland. VisOpt: a visual interactive optimization tool for P-median problems. *Decision Support Systems*, 26(3):209–223, Sept. 1999.

[30] F. Pukelsheim. *Optimal Design of Experiments*, volume 50 of *Classics in Applied Mathematics*. SIAM, Philadelphia, 2006.

[31] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Elsevier, Amsterdam, 2006.

[32] J. Seo, M. Bakay, Y.-W. Chen, S. Hilmer, B. Shneiderman, and E. P. Hoffman. Interactively optimizing signal-to-noise ratios in expression profiling: project-specific algorithm selection and detection p-value weighting in Affymetrix microarrays. *Bioinformatics*, 20(16):2534–2544, 2004.

[33] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[34] S. Tarkkanen, K. Miettinen, and J. Hakanen. Interactive poster: Interactive multiobjective optimization — a new application area for visual analytics. In *Proceedings of the 2009 IEEE Symposium onVisual Analytics Science and Technology (VAST 2009)*, pages 237–238, Oct. 2009.

[35] E. Thomsen. *OLAP Solutions: Building Multidimensional Information Systems*. John Wiley & Sons, Inc., New York, 1997. CD-ROM included.

[36] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, June 2001.

[37] C. Turkay, A. Lundervold, A. Lundervold, and H. Hauser. Hypothesis generation by interactive visual exploration of heterogeneous medical data. In *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data*, volume 7947 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2013.

[38] C. Turkay, J. Parulek, and H. Hauser. Dual analysis of DNA microarrays. In *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies*, pages 26:1–8, New York, 2012. ACM.

[39] L. Tweedie, B. Spence, D. Williams, and R. Bhogal. The attribute explorer. In *Conference Companion on Human Factors in Computing Systems (CHI'94)*, pages 435–436, New York, 1994. ACM.

[40] W. C. M. van Beers and J. P. C. Kleijnen. Kriging interpolation in simulation: a survey. In *WSC '04: Proceedings of the 36th conference on Winter simulation*, pages 113–121. Winter Simulation Conference, 2004.

[41] J. Waser, R. Fuchs, H. Ribičić, B. Schindler, G. Blöschl, and M. E. Gröller. World lines. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1458–1467, Nov.–Dec. 2010.

[42] C. Zenger. Sparse grids. In W. Hackbush, editor, *Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar, Kiel, 1990*, volume 31 of *Notes on Numerical Fluid Mechanics*, pages 241–251. Vieweg-Verlag, 1991.