International Workshop on Visual Analytics (2011)
S. Miksch and G. Santucci (Editors)

Adaptive Interactive Multi-Resolution Computational Steering for Complex Engineering Systems

K. Matković,[†1] D. Gračanin,[‡2] M. Jelović,[§3] Y. Cao,[¶2]

[1] VRVis Research Center in Vienna, Austria
[2] Virgina Tech, Blacksburg, USA
[3] AVL AST Croatia d.o.o, Zagreb, Croatia

Abstract

Computational steering integrates modeling, computation, data analysis, visualization, and data input components of a simulation. Since the simulation space is, in general, very large and continuous, selecting discrete simulation points that can reasonably present the whole simulation space is difficult. We need to interpolate the "missing" values and cover a continuous region of interest in the simulation space. We describe an approach that, in an iterative manner, allows a domain expert to interactively select data points (design of experiments), approximate the values in a continuous region of the simulation space (regression) and automatically find the "best" points in that continuous region based on the specified constraints and objectives (optimization), using the regression and aggregated data. Once the objectives are found, data points in the neighborhood of the objective are generated by the simulation tool thus providing denser coverage of the regions of interest.

1. Introduction and Related Work

Computational steering integrates modeling, computation, data analysis, visualization, and data input components of a simulation. However, the interaction between simulation and visualization can be a very difficult problem. Since the computational cost of a single simulation run can be high, we need a framework where simulation is done in an adaptive, iterative fashion to reduce the overall number of simulation runs by focusing on "interesting" cases. Since computational steering is a highly interactive process, the user interface is a critical component [MGJH08].

In each iteration of computational steering a user defines a region of interest that has to be explored in more details. Additional simulation runs needed to cover that region of interest constitute 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 good cover of the region of interest [Kle07].

While support for user controlled simulation is at the very core of computational steering, there is very limited support for user controlled optimization [BP10]. Very often there is no clear or unique optimal solution and the user has to analyze, in an interactive fashion, trade-offs and interdependencies between objectives [PGR99]. Using an analytical representation of the objective function the user can be presented the values of the objective function in the region of interest [MM06]. Such values can dynamically updated in all views and brushes (selections) [PTMB09]

The type of the objective function determines the nature of the optimization process. In the case of linear or quadratic programming there are efficient and fast algorithms that provide the optimal solution [BV04]. In a general case we are dealing with nonlinear optimization where gradient based optimization methods are used to find a heuristic solution.

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

† Matkovic@VRVis.at
‡ gracanin@vt.edu
§ mario.jelovic@avl.com
¶ yongcao@vt.edu

a surrogate (regression) model that approximates simulation results over the region of interest. Machine learning techniques (e.g., Support Vector Machines [BGV92]) can create a linear, quadratic or nonlinear surrogate models.

2. Computational Steering Framework

We describe an approach that, in an iterative manner, allows a domain expert to interactively select data points (design of experiments), approximate the values in a continuous region of the simulation space (regression) and automatically find the optimal points in that continuous region based on the specified constraints and objectives (optimization), using the regression and aggregated data. The data points around the optimal solutions are then generated by the simulation tool.

When the simulation space is very large, the iterative process can be time consuming and tedious. We would like to automatize that process as much as possible and help the domain expert. Given the simulation model of the problem being explored, this process has four stages (Figure 1):

Design: Since the simulation space is, in general, very large and continuous, selecting representative discrete simulation points is difficult. While the domain expert can select "interesting" values, we can provide automatic methods, such as randomization or blocking, to determine a set of input values that can be used for a sequence of simulation runs.

Simulation: The specified input values and the develop simulation model are used by the simulation tool to generate the corresponding output. The input values and the corresponding output values represent a point in the simulation space. A collection of these points (a collection of simulation runs) are then evaluated using an evaluation (visualization) tool.

Evaluation: The evaluation can be conducted using an information visualization tool [KMG*06] (coordinated multiple views, composite brushes, etc.). Since discrete simulation points do not provide full "coverage" of the simulation space, we can interpolate the "missing" values and cover the whole space. Various interpolation, regression, and machine learning techniques can be used to extrapolate the data.

Optimization: The evaluation tries to determine some "interesting" points. That can be described as an optimization problem with an objective function over some subset of the simulation space. The objective function can be a single parameter, a combination of several parameters, or a multiple-objective function. Various optimization methods can then be applied. The result of the optimization may then trigger the next iteration where the additional simulation points are defined in the next design stage.

Simulation and visualization can be combined in a single framework allowing a user to conduct computational steering. Once a region of interest is detected, a new simulation run (design of experiment) is conducted to adaptively increase the resolution of the simulation points in that region.

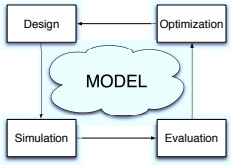

Figure 1: *Exploring large simulation spaces.*

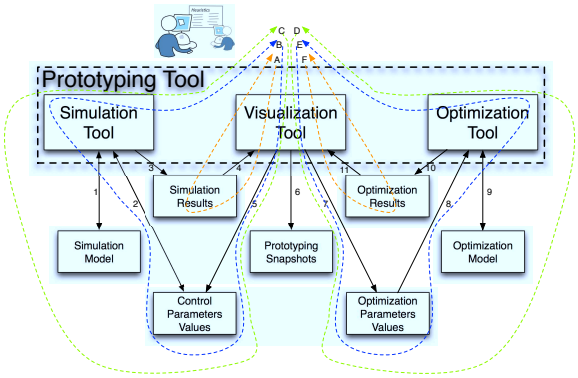

Figure 2: *An iterative approach to prototyping using a combination of simulation, visualization and optimization tools.*

We can increase the analytic capabilities provided to the user by the framework by extending the framework with the support for interactive optimization. An iteration within the extended framework consists of the following steps:

- Conduct simulation runs based on the design of experiment in the previous iteration (for the first iteration create an initial design of experiment).
- Integrate the new simulation runs with the existing simulation data.
- Visually analyze the data and determine the objective function.
- Create a surrogate model, explore the objective function.
- Identify the region of interest.
- Use optimization to determine optimal value(s).
- If the optimal value is not satisfactory, create a new design of experiment with increased resolution.

Figure 2 shows the iterative approach to prototyping, a combination of simulation, visualization, and optimization.

Loops A, B, and C describe combinations of simulation and visualization. Loop A describes direct visualization of simulation results. Loop B describes a combination of simulation and visualization using a set of control parameters values. Loop C also allows for changes in the simulation model.

Loops D, E, and F describe combinations of visualization and optimization. Loop F describes direct visualization of simulation results. Loop E describes a combination of visualization and optimization using a set of optimization param-

eters values. Loop D also allows the user to determine a new objective function, the corresponding surrogate model, and to specify the optimization constraints.

Determining the surrogate model is, in general, a trial-and-error process. More complex models, such as support vector model regression using polynomial or radial basis, can predict the objective function values using the simulation data $\mathbf{w}$, however finding the correct model parameters can be challenging. Once the surrogate model $y(\mathbf{x};\mathbf{w})$ is determined (Equation 1), it can be used to present additional simulation points and different resolutions to determine the region of interest ($K(\mathbf{x},\mathbf{x}_i)$ is a kernel function).

$$y(\mathbf{x};\mathbf{w}) = \sum_{i=1}^{N} w_i K(\mathbf{x},\mathbf{x}_i) + w_0 \qquad (1)$$

Once the region of interest is identified, the corresponding optimization constraints are determined and the values provided by the surrogate model can be used in non-linear optimization techniques. The optimization results are then visualized and used for further analysis.

3. Illustrative Example

We illustrate the above described system using an example from automotive industry. Together with a domain expert we explored a modern, common rail, Diesel car engine injection system. A high pressure in the rail is used to inject fuel into cylinders. Electronically controlled actuators open and close injectors and precisely control the injection.

Due to high pressures and quick changes in the system some phenomena, not typical for classical fluid mechanics, appear during injection. Furthermore, in a common rail system, each cylinder, or injector, is influenced by others through the rail. All of these requires careful re-thinking of traditional approaches in injection system design. One of the great challenges of a common rail injection system design is to understand and prevent unwanted pressure oscillations which can lead to an unexpected system operation. This results in the reduced efficiency and increased emission, exactly the opposite of the design goals.

Modern simulation software can be used to simulate injection systems. The data used in this example were computed using the AVL Hydsim simulation software. The whole system (four injectors) was simulated. We focus on the high-pressure pipe geometry and the common-rail itself. We analyze their contribution to the oscillations in pressure and we try to tune the system to minimize these oscillations. Due to the overall system complexity we need advanced tools to comprehend behavior of the whole system.

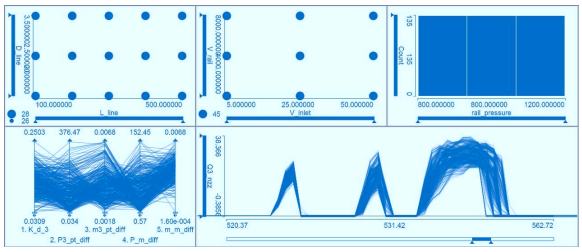

Figure 3: *Two scatter plots show four control parameters, the histogram shows rail pressure (three variations used). The parallel coordinates show five aggregates used in the analysis, and the curve view shows injection curves. 405 initial simulation runs are shown.*

3.1. Iterative Analysis - a combination of automatic and interactive process

The analysis is based on results from multiple simulation runs. There are many parameters which has to be set in the simulation model. We varied five most relevant parameters in the optimization process, the high pressure pipe geometry, *L_line* and *D_line* (the length and diameter), the common-rail characteristics, *V_rail* and *rail_pressure* (the volume of the rail and the pressure inside the rail), and the inlet geometry, *V_inlet* (the volume of a junction between the rail and high pressure pipe). For each simulation run more than 30 output parameters are computed, all of them being time series, i.e., functions of the crank angle. As automatic optimization methods expect each point in a multidimensional space to have scalar dimensions, our data does not fit. We have to aggregate time series outputs in order to use automatic optimization. The more aggregates we have, a curve is better described, and the optimization and the regression model will be more precise. At the same time the regression model and the optimization become more and more complex as number of parameters increases.

We first varied every control parameter in several steps (405 simulation runs). This is a starting point for exploration and optimization. We use a coordinated multiple views tool to explore results. We can easily select (*brush*) some simulation runs in any view and see all attributes highlighted in all other views. We use a curve view to depict time dependent attributes. Figure 3 shows the initial 405 simulation runs.

We use the same tool to specify optimization criteria. We interactively brush (using composite brushing if necessary) the constraints — we define an area in the whole space where the solution should be. We use a special view to define target optimization goals. Based on the constraints and already computed regression model the system suggests an optimum point. As this point is based on a regression model which is computed using curve aggregates, we compute the actual data for the computed optimum. Moreover, we also compute a set of points in the neighborhood of the point using origi-

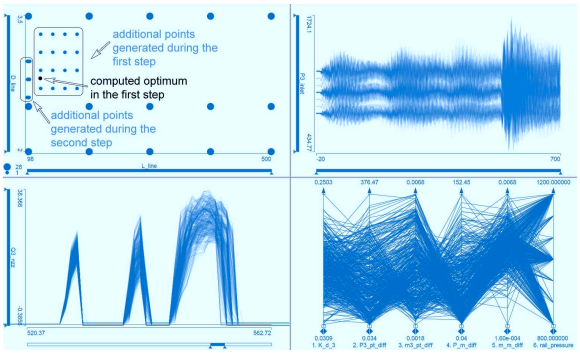

Figure 4: *After the user selects constraints and defines optimization goal the optimization module computes the optimum. The optimum is depicted in a different color (black in the scatterplot), and additional points in the constrained space are computed. The original simulation is started for each new point, including optimum. All views show results from two iterations. Note the optimum on the border of the subspace indicating a possible direction for a new iteration.*

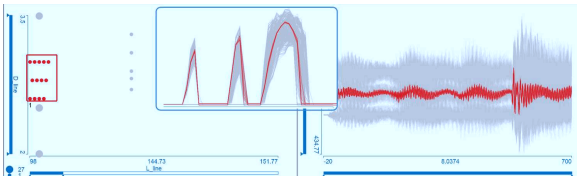

Figure 5: *Results from the second iteration (the scatterplot is zoomed in). The curves are of the desired shape. Moving the subspace further to the left (smaller L_line) is not possible due to the physical constraints of the system.*

nal simulation model rather then regression model. Note that regression model and optimization is done using a reduced subset based on curves aggregates. We most often use maximum value, the length of injection interval, the slope of the curve as parameters. Still, we have to check the shape of the suggested curve, sometimes it has some unwanted characteristics, although all computed curve parameters are within wanted range. We have reduced the possible solution space using interactive visual analysis and the system computed the optimum point. Figure 4 shows the newly computed point (upper left scatter plot, black point). We computed additional points in the neighborhood, explore the curve shapes (computed using simulation software), changed the constraints now, computed new regression model and computed new optimum. The results after two iterations are shown in Figure 5. The process continues for other parameters.

4. Conclusion

Analysis and optimization of complex systems can be done either automatically or using interactive visualization. However, when dealing with complex systems with many parameters and complex data models, neither approach works well. We described a computational steering framework that integrates simulation (Hydsim), visualization (ComVis), and optimization (Matlab) tools. The domain expert (one of the co-authors of the paper) successfully used this implementation on a real-world case study. The proposed workflow can be applied to any exploration or optimization problem of complex systems. There are two computational bottlenecks, simulation and optimization. In either case, the massive data parallel processing power of GPUs can provide large performance leap, thus, greatly shortening the turn-around time of the iterative system design process.

5. Acknowledgments

The simulation data is courtesy of AVL. Part of this work was done in the scope of the SemSeg project and the K1 program at the VRVis. The project SemSeg acknowledges the financial support of the Future and EmergingTechnologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number 226042. This work was supported in part by Science Foundation Ireland grant 03/CE2/I303_1 to Lero-The Irish Software Engineering Research Centre.

References

[BGV92] BOSER B. E., GUYON I. M., VAPNIK V. N.: A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT '92)* (New York, 1992), ACM, pp. 144–152. 2

[BP10] BERGER W., PRINGER H.: Interactive visual analysis of multiobjective optimizations. In *Proceedings of the 2010 IEEE Symposium on Visual Analytics Science and Technology* (24–29 Oct. 2010), pp. 215–216. 1

[BV04] BOYD S., VANDENBERGHE L.: *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004. 1

[Kle07] KLEIJNEN J. P. C.: *Design and Analysis of Simulation Experiments*. Springer, New York, 2007. 1

[KMG*06] KONYHA Z., MATKOVIĆ K., GRAČANIN D., JELOVIĆ M., HAUSER H.: Interactive visual analysis of families of function graphs. *IEEE Trans. on Visualization & Computer Graphics 12*, 6 (Nov./Dec. 2006), 1373–1385. 2

[MGJH08] MATKOVIĆ K., GRAČANIN D., JELOVIĆ M., HAUSER H.: Interactive visual steering - rapid visual prototyping of a common rail injection system. *IEEE Trans. on Visualization & Computer Graphics 14*, 6 (Nov.-Dec. 2008), 1699–1706. 1

[MM06] MIETTINEN K., MÄKELÄ M. M.: Synchronous approach in interactive multiobjective optimization. *European J. of Operational Research 170*, 3 (May 2006), 909–922. 1

[PGR99] PIRKUL H., GUPTA R., ROLLAND E.: VisOpt: a visual interactive optimization tool for P-median problems. *Decision Support Systems 26*, 3 (Sept. 1999), 209–223. 1

[PTMB09] PIRINGER H., TOMINSKI C., MUIGG P., BERGER W.: A multi-threading architecture to support interactive visual exploration. *IEEE Trans. on Visualization & Computer Graphics 15*, 6 (Nov.–Dec. 2009), 1113–1120. 1