Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 00 (2011) 1–10
Procedia Computer Science

International Conference on Computational Science, ICCS 2011

Towards High-Dimensional Computational Steering of Precomputed Simulation Data using Sparse Grids

Daniel Butnaru*, Dirk Pflüger*, Hans-Joachim Bungartz*

Institut für Informatik, Technische Universität München,
Boltzmannstr. 3, 85748 Garching, Germany

Abstract

With the ever-increasing complexity, accuracy, dimensionality, and size of simulations, a step in the direction of data-intensive scientific discovery becomes necessary. Parameter-dependent simulations are an example of such a data-intensive tasks: The researcher, who is interested in the dependency of the simulation's result on a set of input parameters, changes essential parameters and wants to immediately see the effect of the changes in a visual environment. In this scenario, an interactive exploration is not possible due to the long execution time needed by even a single simulation corresponding to one parameter combination and the overall large number of parameter combinations which could be of interest.

In this paper, we present a method for computational steering with pre-computed data as a particular form of visual scientific exploration. We consider a parametrized simulation as a multi-variate function in several parameters. Using the technique of sparse grids, this makes it possible to sample and compress potentially high-dimensional parameter spaces and to efficiently deliver a combination of simulated and precomputed data to the steering process, thus enabling the user to interactively explore high-dimensional simulation results.

Keywords: Computational Steering, CFD Simulations, Sparse Grids, High Dimensionalities

1. Introduction

In the last decades the computational branch of science has made significant progress in both modeling and in performing accurate simulations of very complex phenomena. A consequence of such a step is the availability of large amounts of data generated by various simulations, and the focus now moves on to how to manage such data and explore it in a convenient way for the researcher. Another unfortunate characteristic inherent to simulations is that a higher accuracy (resolution) of the simulation demands a higher computational effort and thus significantly slows down the exploration process.

In this paper we consider (visual) data exploration due to parameter variation, see Fig. 1. Inside a computational steering environment, a researcher observes the effects of changes of the simulation's main parameters to the simulation results. The goal is, for example, to identify correlated as well as unimportant parameters, or to discover new and

*Corresponding author
Email addresses: butnaru@in.tum.de (Daniel Butnaru), pflueged@in.tum.de (Dirk Pflüger), bungartz@in.tum.de (Hans-Joachim Bungartz)

unexpected patterns in the data. Due to storage costs and computational effort, it is however not possible to generate and store simulation results for every parameter combination of interest, especially in high-dimensional settings. Just consider that taking only ten distinct values for each of five parameters into account would require to compute and store 10^5 different simulation results. In order to deal with this problem, we introduce a method of sampling and compressing the simulation's parameter space based on sparse grids [1]. By reducing the size of the data, fast exploration of certain multi-dimensional data sets is enhanced, while sufficiently good accuracy of the stored simulation results is preserved.

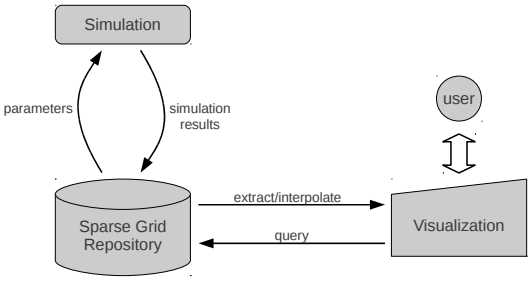

Figure 1: Workflow for computational steering for high-dimensional simulations.

The main idea is to consider the simulation as a function of various simulation parameters. Such a multi-dimensional function can be numerically represented and treated; however, classical discretizations of the parameter space suffer with increasing dimensionality from the so-called curse of dimensionality, the exponential dependency of the effort on the number of dimensions. Sparse grids enable us to mitigate the curse of dimensionality to some extent, allowing to tackle dimensionalities that are of interest in engineering settings, where models depend on a moderate number of variables. Instead of running a simulation for every parameter combination—an unrealistic task in itself—the sparse grid sampling dictates which parameter combinations actually need to be examined and stored. The associated sparse grid interpolation scheme then offers access to all other parameter combinations.

To demonstrate our method, we show results from a simple computational fluid dynamics (CFD) simulation, the so-called driven cavity. In this scenario, the fluid (e.g. water) in a cavity is stimulated by the cavity's moving lid. Location and shape of the emerging vortices mainly depend on the velocity of the lid and the viscosity of the fluid. We assume these two parameters to be continuous within a certain range, and together with time they form a three-dimensional parameter space. We then use a sparse grid to sample and approximate the flow in the considered parameter range, without the need to run new simulations for every combination in between. This enables an efficient real-time visualization of simulation results, well-suited for interactive computational steering.

2. The Sparse Grid Technique

Sparse grids help to overcome the curse of dimensionality to a great extent. Interpolating on a regular grid with a resolution of N grid points in one dimension, they enable one to reduce the number of grid points significantly in d dimensions from $O(N^d)$ to $O(N(\log N)^{d-1})$ while maintaining a similar accuracy as in the full grid case. The only requirement is that the underlying function f has to be sufficiently smooth [1]. Note that it has been shown that even functions that do not meet the smoothness requirements can be successfully dealt with if adaptive refinement is employed [2]; we will address adaptive refinement in Sec. 3.3. The notion sparse grids was coined in 1990 [3] for the solution of high-dimensional partial differential equations, and they have meanwhile been successfully employed in a whole range of applications, ranging from astrophysics and quantum chemistry to data mining and computational finance, see, e.g., [1, 2] and the references cited therein. In the following, we briefly describe sparse grids and the main principles they base upon, a hierarchical representation of the one-dimensional basis and the extension to the d-dimensional setting via a tensor product approach; for further details, see [1, 2] again.

We consider the representation of a piecewise d-linear function $f_N : \Omega \to \Gamma$ for a certain mesh-width $h_n := 2^{-n}$ with some discretization level n. The function $f_N(\underline{x})$ thus maps a set of parameters $\underline{x}$ out of the parameter space Ω to a simulation result Γ. For the parameter space Ω we consider rectangular domains which we scale to $\Omega := [0, 1]^d$.

To obtain an interpolant f_N as an approximation to some function f, we discretize Ω and employ basis functions ϕ_i which are centered at the grid points stemming from the discretization. f_s is thus a weighted sum of N basis functions, $f_s := \sum_{j=1}^{N} \alpha_j \phi_j$, with coefficients α_j.

The underlying principle is a hierarchical formulation of the basis functions. In one dimension, we use the standard hierarchical basis

$$\Phi_l := \left\{ \varphi_{l',i} : l' \leq l, i \leq 2^{l'} - 1 \wedge i \text{ odd} \right\}.$$

with piecewise linear ansatz functions $\varphi_{l,i}(x) := \varphi\left(x \cdot 2^l - i\right)$ and $\varphi(x) := \max(1 - |x|, 0)$ for some level $l \geq 1$ and an index $1 \leq i < 2^l$. The basis functions are centered at grid points $x_{l,i} = 2^{-l} i$ at which we interpolate f, see Fig. 2 (left) for the basis functions up to level 3. Note that all basis functions on one level have pairwise disjoint supports and cover the whole domain.

The hierarchical basis functions can be extended to d dimensions via a tensor product approach as

$$\varphi_{\underline{l},\underline{i}}(\underline{x}) := \prod_{j=1}^{d} \varphi_{l_j,i_j}(x_j),$$

with multi-indices $\underline{l}$ and $\underline{i}$ indicating level and index of the underlying one-dimensional hat functions for each dimension. The d-dimensional basis

$$\Phi_{W_{\underline{l}}} := \left\{ \varphi_{\underline{l},\underline{i}}(\underline{x}) : \; i_j = 1, \ldots, 2^{l_j} - 1, \; i_j \text{ odd}, \; j = 1, \ldots, d \right\}$$

span hierarchical subspaces $W_{\underline{l}}$. As before, the basis functions for each $W_{\underline{l}}$ have pairwise disjoint, equally sized supports and cover the whole domain. The full-grid space of piecewise d-linear functions V_n can be obtained as a direct sum of $W_{\underline{l}}$,

$$V_n := \sum_{l_1=1}^{n} \cdots \sum_{l_d=1}^{n} W_{(l_1,\ldots,l_d)} = \bigoplus_{|\underline{l}|_\infty \leq n} W_{\underline{l}},$$

but the hierarchical scheme of subspaces allows one to choose those subspaces that contribute most to the approximation. With respect to the L^2-norm, this leads to the sparse grid space $V_n^{(1)}$,

$$V_n^{(1)} := \bigoplus_{|\underline{l}|_1 \leq n+d-1} W_{\underline{l}},$$

e.g. The tableau of subspaces in $2d$ is shown in Fig. 2 (right) for $n = 3$.

To obtain non-zero values on the boundary, the one-dimensional basis of level 1 can be extended by the two basis function $\varphi_{0,0}$ and $\varphi_{0,1}$. Unfortunately, even for a very coarse grid with a resolution of $h_1 = 1/2$ this requires to obtain 3^d simulation results—$3^d - 1$ parameter combinations being located on the boundary of the parameter space Ω. For our application of computational steering, we assume that we start from a reasonable choice of Ω and that these extreme parameter combinations are of less interest than the inner part of Ω. We therefore choose to interpolate only in the inner part and to extrapolate towards the boundary, and use in the following the one-dimensional basis functions

$$\varphi_{l,i}(x) := \begin{cases} 1 & \text{if } l = 1 \wedge i = 1, \\ \left\{ \begin{array}{ll} 2 - 2^l \cdot x & \text{if } x \in \left[0, \frac{1}{2^{l-1}}\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1 \wedge i = 1, \\ \left\{ \begin{array}{ll} 2^l \cdot x + 1 - i & \text{if } x \in \left[1 - \frac{1}{2^{l-1}}, 1\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1 \wedge i = 2^l - 1, \\ \varphi\left(x \cdot 2^l - i\right) & \text{else} \end{cases}$$

see Fig. 2 (second left).

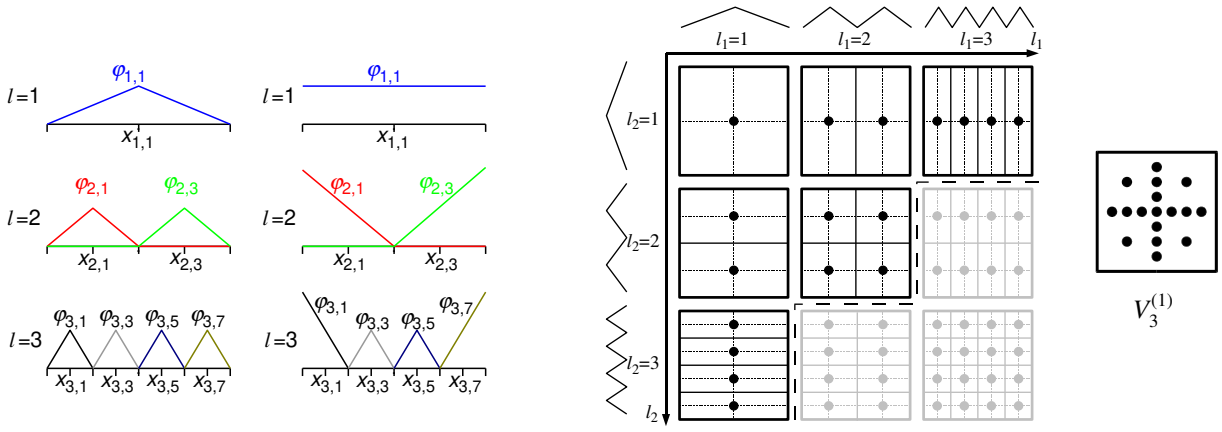

Figure 2: Classical one-dimensional hierarchical basis functions up to level 3 (left) and their modified, extrapolating counterparts (second left), and the tableau of subspaces $W_{\underline{l}}$ up to level 3 in two dimensions together with the resulting sparse grid for $n = 3$ (right).

3. The Sparse Grid Repository

The scenario we consider is a scientist studying the effects of changes in the parameters of some physical phenomena on the simulation results, a classical parameter study. It is typically done by running a large number of simulations on a massively parallel system and storing the results which are later analyzed. Common tasks making use of parameter studies are optimization problems where a certain optimal choice of parameters for a measure of error or quality is searched for. The higher the number of parameters, the more difficult and costly this task gets.

Computational steering was initially defined as the interactive control over a computational process during execution [4], with the purpose of significantly reducing the time between changes to parameters and the viewing of the results. Approaches to computational steering can be split into application specific computational steering systems tailored to a very specific simulation and scenario [5], domain specific computational steering systems working with various scenarios for a specific application, and more generally applicable computational steering environments which act as problem-solving environments. VASE [6], SCIRun [7], and CUMULVS [8] are examples of the latter ones. All of them display data obtained from a live-running simulation and their interactivity depends on the how fast the simulation can deliver data. Other approaches to extract data from multi-dimensional precomputed simulation data are query-driven visualizations [9] based on bitmap indices data structures [10]. Such approaches answer multi-variate, multi-dimensional queries by first indexing a set of precomputed datasets and then using these specially tailored indices to efficiently return one of them as an answer to a query.

The approach presented in this paper moves away from delivering only presimulated data, trying to extract as much information as possible from precomputed simulations. For any parameter combination in a certain range, an approximation can be immediately delivered without having to wait for the exact simulation to finish execution. This can be supported by simulations running in the background to increase the accuracy in regions of interest in an incremental way.

3.1. Compressing a Multi-Dimensional Parameter Space

Computational steering, as we are considering, enables a researcher to interactively study parameter-dependent simulation results. This allows researchers to qualitatively identify the influence of one or several parameters on the considered model by changing them (steering) and visually assessing the changes in the corresponding result. Experts can demonstrate dependencies to non-experts, and non-experts can obtain an intuition about the behavior of a certain model. A fundamental criterion for computational steering is the speed with which the data is delivered to the visualization in order to make the exploration process acceptable for the user.

Most realistic, high resolution simulations just cannot produce the data in time in order to guarantee a smooth exploration. We therefore propose to use a repository of precomputed simulation results based on a sparse grid

discretization of the parameter space at hand. Due to the incremental nature of sparse grids, this even allows to combine the precomputed results with results from simulations running in the background during steering.

The repository thus consists of simulation results dependent on a sparse grid discretization of the parameter space which is determined by the sparse grid structure introduced in the previous section. Figure 3 (left) shows a sparse grid for $n = 3$ in the two-dimensional parameter space.

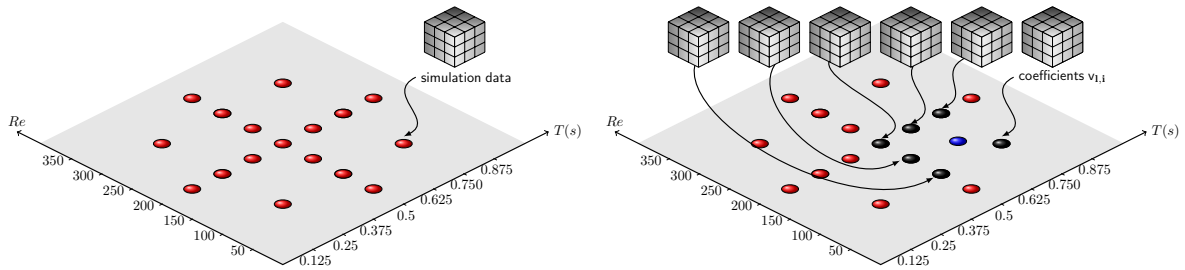

Figure 3: A 2D sparse grid discretization (left) requires the function value (simulation data) only for the parameter combinations for the sparse grid's points. A 2D interpolation on a sparse grid (right) reconstructs the simulation result for a previously unsimulated parameter combination, here $(0.7, 90)$, by a combination of stored simulation results.

To set up the repository, the following steps are performed:

1. **Discretize the parameter space**. All the parameter combinations belonging to a suitable sparse grid are generated up to the predefined level. The higher the dimensionality is, the more parameter combinations can be saved compared to a classical full grid discretization.
2. **Generate initial data**. For all parameter combinations, a simulation is executed and the result is stored in the repository. If time is one of the parameters, a single time-dependent simulation can provide multiple results.
3. **"Compress"**. Working in-place, the compression process is specific to the sparse grid method. For each grid point j, the corresponding hierarchical coefficient α_j has to be computed, a process which is called hierarchization. Knowing them, approximations for new parameter combinations can be computed, see below.

3.2. The Data Extraction Process: Interpolation

The whole purpose of the repository is to deliver simulation data as fast as possible to the visualization tool in order to speed up the exploration process. Instead of running new simulations, the visualization receives approximated data obtained by interpolating in the multi-dimensional space of pre-computed simulations. This is just the evaluation of the underlying sparse grid function (d-linear interpolation). Figure 3 (right) depicts which stored results are required to obtain an interpolation for the new parameter combination $(0.7, 90)$. Following the notation from Sec. 2, first the hierarchical basis functions $\phi_{\underline{l},\underline{i}}$ with support over the point of interest are identified, evaluated and weighted with the hierarchical coefficients $\alpha_{\underline{l},\underline{i}}$ computed during the set-up of the repository. The sum of all these weighted contributions is the interpolated value at the desired point of interest $\underline{x} \in \Omega$:

$$f_N(\underline{x}) := \sum_{|\underline{l}|_1 \leq n+d-1} v_{\underline{l},\underline{i}} \cdot \phi_{\underline{l},\underline{i}}(\underline{x}),$$

Basically, the interpolation is reduced to several constant-vector multiplications followed by a summation. This can be executed very efficiently on accelerator cards or other parallel environments and guarantees a delivery of data to the visualization which is fast enough for interactive interaction [11].

3.3. Increasing the Accuracy of the Repository with Adaptivity

The initial construction of the repository presented in Sec. 3.1 requires to run several initial simulations. Starting from that, the steering (exploration) process can begin. However, the accuracy of the initial interpolation may not be accurate enough to capture certain features in the parameter space of the simulation.

For our task, a very convenient property of the sparse grid method is its hierarchical and incremental structure. New sparse grid points (simulations) can be added to the grid by just extending the storage and computing their weight (hierarchical coefficient). In our current environment, the user can assist the growth of the repository during the exploration process by specifying where new refinements have to be performed, i.e., where a higher accuracy is desired.

Figure 4 presents such a refinement in a two-dimensional parameter space. Each refinement triggers a series of simulations which are executed in background and whose results are automatically integrated in the repository as soon as available, without interrupting the exploration process. A cue in the visualization environment signals the researcher (who might have changed the parameters of the current view in the meantime) that data with a higher accuracy is available and can be displayed. Upon exiting the exploration environment the current repository is stored to disk and reloaded at the start of a new steering task. This way, results from previous explorations are not lost but contribute to an ever increasingly-accurate simulation repository.

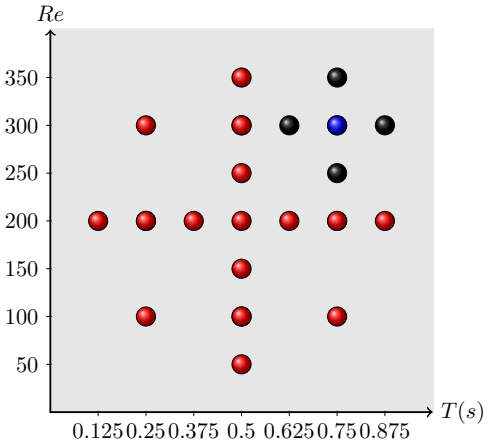

Figure 4: Refinement in a two-dimensional parameter space. Each refinement introduces new grid points in the parameter space. The positions of these points are restricted to the sparse grid's structure. In this example, four additional points are created, each corresponding to a combination of the two parameters (Re, t). For each of them, a simulation is started, and the results are added to the repository.

4. Application to a CFD scenario

As a test-case, we consider a well-known benchmark problem for viscous incompressible fluid flow, the lid-driven cavity [12]. To describe the flow, the $3d$ Navier-Stokes equations are used with an explicit scheme for the time steps. The geometry at stake is shown in Fig. 5 in two dimensions: a square cavity consisting of three rigid walls with no-slip conditions and a lid moving with a given tangential velocity. The movement of the lid influences the fluid which is stagnant at the beginning. (The lower left corner of Ω always has a reference static pressure of zero.) With time passing by, this results in a series of vortices at several locations and with different rotation directions. Starting from a state of rest, the flow tends to some steady behaviour, even though the patterns during this process are quite complex. The two main target values of interest computed by the underlying fluid simulation are the pressure and the velocity at each point in the three-dimensional coordinate system.

4.1. Parameters

The lid-driven cavity is well-suited as a proof-of-concept scenario for our computational steering application. There are three parameters of interest which influence position, shape, and size of the vortices; they exhibit characteristics that are different enough to raise interesting problems and challenges; and grids in a three-dimensional parameter space can still be visualized. The parameters are

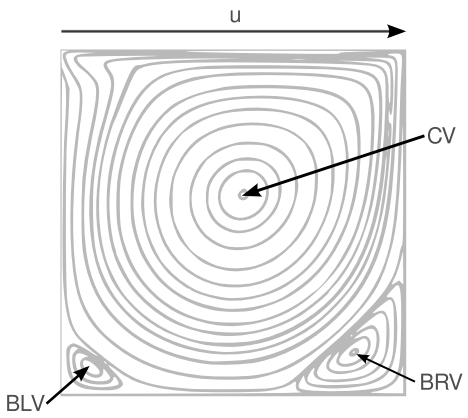

Figure 5: Central (CV), bottom left (BLV), and bottom right (BRV) vortices of a two-dimensional lid-driven cavity in the stationary case for a random parameter combination.

Reynolds number Re**.** The Reynolds number expresses the ratio of inertial (resistant to change or motion) forces to viscous (heavy and gluey) forces. For example, a lower Reynolds number is characteristic for a viscous fluid (such as honey). We consider Reynolds numbers in a range spanning from 50 to 450.

Lid velocity u**.** With increasing lid-velocity the main vortex takes shape faster, is flattened and pushed to the right. In the following, the range of the lid-velocity ranges from 1 m/s to 5 m/s.

Simulation time t**.** While not a parameter in itself as are the previous ones, the abilility to navigate in simulation time and instantly being able to observe the changes in the simulation state is very useful during the steering process. Furthermore, the influence of the parameter time on the flow field is completely different from the other ones, providing a good test case. Starting at $t = 0$, the state of rest quickly changes, while after some time, here for $t = 1$, a stationary flow field evolves. Even more, simulation data for a certain time step can be costly to obtain as all previous time steps have to be simulated. For a certain choice of Reynolds number and lid velocity, the whole time span of interest has to be simulated, but only time steps belonging to sparse grids points have to be stored, thus significantly reducing the size of the repository. And queries for new parameter combinations during steering can be answered by a fast interpolation rather than having to simulate for many time steps.

For the lid-driven cavity in three dimensions, both parameter and simulation space are three-dimensional and can thus be visualized which helps to interprete and understand the results obtained during the computational steering process. And finally, the scenario is well-understood, which enables one to focus on the demands and challenges building up a sparse grid compressed repository for real-time steering.

4.2. Results

Figure 6 shows snapshots (interpolated simulation data retrieved from the sparse grid repository) in three spatial dimensions from our visualization environment for three different Reynolds numbers and lid velocities at time $t = 0.5\,s$. The flow in the cavity is displayed using particle tracing. The particles are released from probes, which could be moved by a user, at one or two locations of interest. After a probe is placed, any of the three parameters can be changed in real-time as the corresponding flow-field data is delivered instantly from the sparse grid repository. In the top row, the Reynolds number is increased from left to right. As one might expect, the central vortex is pushed in the flow direction further to the right with decreasing viscosity. In the bottom one, the velocity of the lid is changed from slow to fast. The bottom-right vortex takes shape earlier for higher velocities, which can be clearly seen.

The sparse grid method to compress the multi-dimensional parameter space allows a fast delivery of interpolated simulation data to the visualization environment. Neither a simulation is required nor many time steps have to be considered. And, as a further development, the simulation results could be compressed by an (adaptive) sparse grid-based sampling, too.

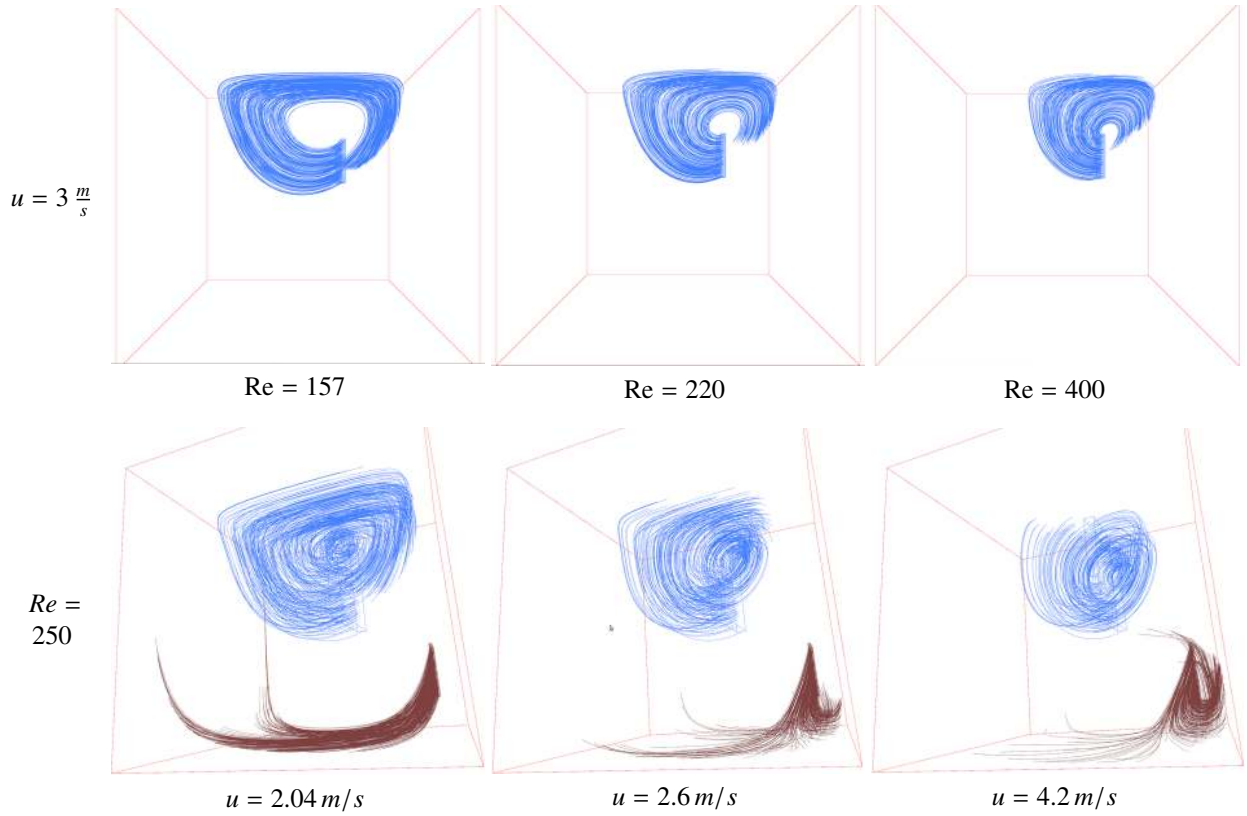

Figure 6: The interpolated flow field in a three-dimensional lid-driven cavity for different parameter values. In the top row, a probe is placed in the central vortex and the Reynolds number is increased from left to right. As expected, the central vortex is pushed in the direction of the flow. In the bottom row, the lid velocity is increased from left to right causing the bottom-right vortex to take shape: the higher the lid-velocity the sooner the bottom-right vortex appears.

To examine how well the sparse grid sampling of the parameter space is suited for the interactive exploration process, we now consider the approximaton accuracy of the flow fields obtained by the evaluation of a sparse grid repository function f_N. We measure the L^2-norm of the approximation error of the velocity field for each parameter combination. To this end, the parameter space Ω was sampled with level three, leading to a resolution of 15 inner grid points in each dimension: for 15×15 parameter combinations of Re and u, 15 time steps are computed each. The simulations were performed with OpenFOAM[1]. For a repository based on a full grid sampling of the parameter space, all 3375 flow fields would have to be computed and stored. In our case, we only have to store a small fraction (31) of the simulation results, and we would have to simulate only for 17 combinations of Reynolds number and lid velocity rather than for 225 different ones. The advantages of the sparse grid sampling are obvious.

In Fig. 7, the relative error is plotted for all 15^3 parameter combinations of Reynolds number, lid velocity, and time. The size of each block correponds to the L^2-norm of the error between the interpolated flow field and the exact one obtained via simulation for the corresponding parameter set. Even though the sampling is rather coarse (level three), the approximation results are rather good; high errors are only obtained towards one side of Ω.

The increase of the error towards $t = 0$ is not surprising: On the one hand, we extrapolate towards the boundary, and the largest errors can thus be expected close to the domain's boundary. On the other hand, the situation for $t = 0$ is physically impossible, leading to a singularity, as the lid moves with a certain velocity whereas the fluid right next to it is still motionless. Figure 7 (right) shows the L^2-norm error plot of three traversals in the most critical parameter time starting from three random grid points (indicated on the left) through the parameter space. Where the traversals

[1] www.openfoam.com

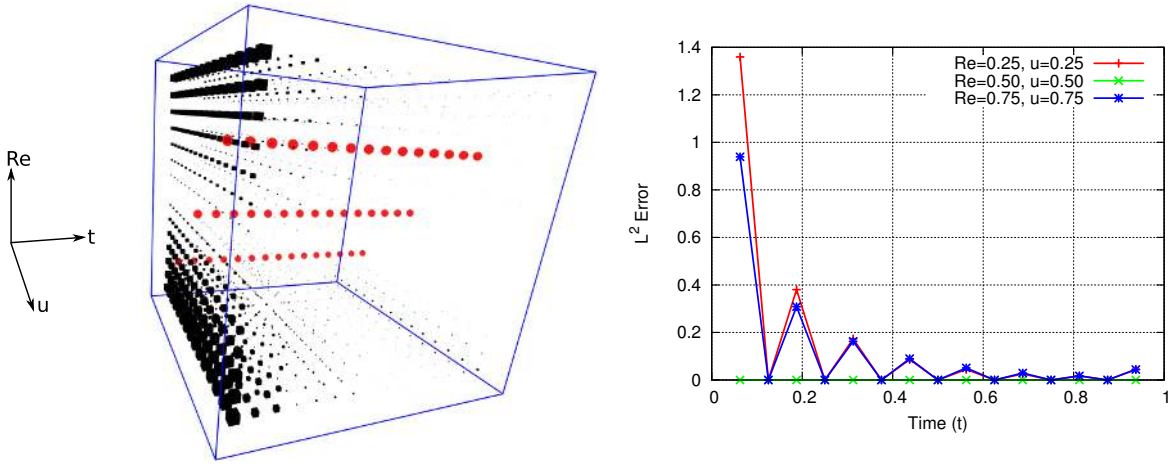

Figure 7: Left, L^2-norm of the approximation error for all 15^3 parameter combinations. Right, L^2-norm of the error for three traversals through the three-dimensional parameter space. The path of each traversal is indicated with spheres on the left.

coincide with grid points contained in the sparse grid, the repository contains the exact solution. In-between, an interpolation error is obtained which increases with decreasing t. Close to the boundary, the error increases in both directions, as the simulation results are obtained by extrapolation rather than by interpolation there.

If the overall error has to be reduced, or if the user of the steering application is interested in the exact properties of the flow field towards $t = 0$, a refinement of the sparse grid can be triggered. This can be done either automatically or semi-automatically, either by a standard criterion for adaptive refinement based on the coefficients in the hierarchical representation [2], e.g., or by the user specifiying to refine around the current point of interest. Refining a grid point results in new parameter combinations for which simulation results have to be obtained. As soon as they are available, the current view can be updated. Here, we refined the single grid point with the smallest t-value (and the highest hierarchical coefficient), resulting in six new parameter combinations. The refinement significantly reduces the interpolation error towards $t = 0$, see Fig. 8, demonstrating the feasability of our approach. Note that we would still obtain relatively high errors due to extrapolation for $t < 1/16$ in the right plot.

5. Conclusions and Future Work

In this paper, we have studied a first scenario for an interactive real-time computational steering environment for the exploration of parameter-dependent simulation data. A repository of pre-computed simulation results based on a sparse grid sampling of the parameter space under consideration allows to treat more parameters than with conventional approaches and to quickly obtain approximations for new parameter combinations via interpolation. We have applied the method to the lid-driven cavity, a well-known scenario in CFD, and have obtained good results in terms of accuracy already for coarse discretizations. Parameters with completely different influences on the simulation results can be dealt with. We have demonstrated how the results in regions of greater interest or with a higher dependency on the simulation parameters can be improved by refining appropriate grid points. Furthermore, adaptivity enables one to control the error in regions where the underlying multi-dimensional function looses its smoothness.

The three-dimensional interactive steering performed for the lid-driven cavity serves as a proof of concept and will be generalized to higher-dimensional scenarios. The long-term goal is to perform computational steering for a simulation of carbon dioxide sequestration which involves multi-phase flows and geological properties. The simulations can depend on up to ten parameters of interest, and they are typically time-consuming to compute even on massively parallel systems. Using the approach presented here, we aim to be able to explore a ten-dimensional parameter space interactively as accurate as possible.

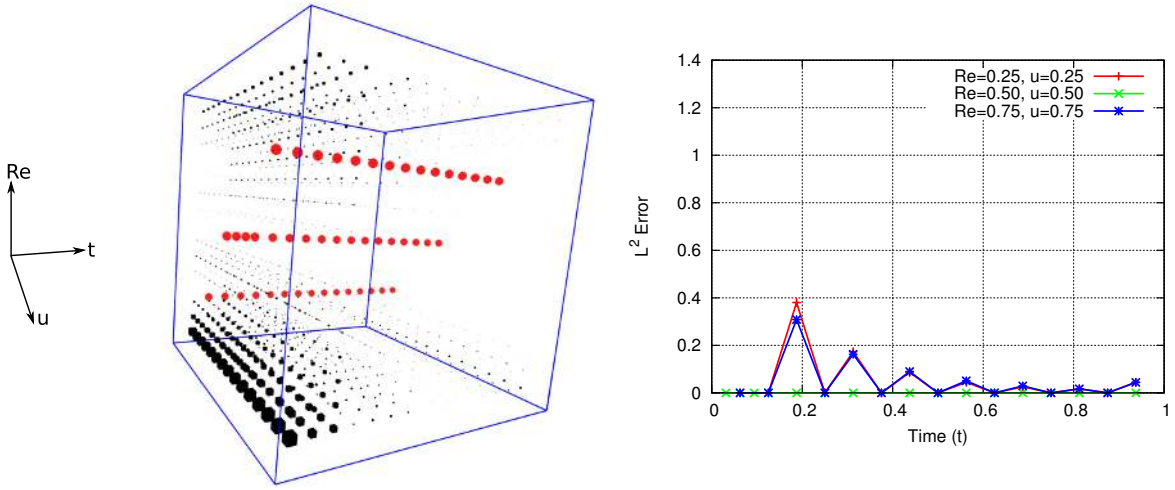

Figure 8: Left, L^2-norm of the approximation error for all 15^3 parameter combinations after the refinement of a single grid point. Right, L^2-norm of the error for the same three traversals of the three-dimensional parameter space as before.

With the increase in dimensionality, the scalability of storing and retrieving (interpolating) simulation data becomes crucial to ensure interactive steering. Efficient algorithms performing the interpolation on accelerator cards such as GPGPUs are already under development [11], while dynamic extensions of the parameter ranges and automatic refinement based on user behavior and error measure need to be addressed in the near future. The quality of the visualization and user interaction are also critical for the success of any visual computational steering process, and thus in the focus of our project. Finally, a large-scale visualization in a CAVE environment for high-resolution simulation data is also under development.

Acknowledgement : This publication is based on work supported by Award No. UK-C0020, made by King Abdullah University of Science and Technology (KAUST).

References

[1] H.-J. Bungartz, M. Griebel, Sparse grids, Acta Numerica 13 (2004) 147–269.
[2] D. Pflüger, Spatially Adaptive Sparse Grids for High-Dimensional Problems, Verlag Dr. Hut, München, 2010.
[3] C. Zenger, Sparse grids, in: W. Hackbusch (Ed.), Parallel Algorithms for Partial Differential Equations, Vol. 31 of Notes on Numerical Fluid Mechanics, Vieweg, 1991, pp. 241–251.
[4] J. D. Mulder, J. J. van Wijk, R. van Liere, A survey of computational steering environments, Future Gener. Comput. Syst. 15 (1999) 119–129.
[5] R. Marshall, J. Kempf, S. Dyer, C.-C. Yen, Visualization methods and simulation steering for a 3d turbulence model of lake erie, SIGGRAPH Comput. Graph. 24 (1990) 89–97.
[6] R. Haber, B. Bliss, D. Jablonowski, C. Jog, A distributed environment for run-time visualization and application steering in computational mechanics, Computing Systems in Engineering 3 (1-4) (1992) 501 – 515, high-Performance Computing for Flight Vehicles.
[7] S. G. Parker, C. R. Johnson, Scirun: a scientific programming environment for computational steering, in: Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM), Supercomputing '95, ACM, New York, NY, USA, 1995.
[8] G. I. James, G. A. Geist, I. James, A. Kohl, P. M. Papadopoulos, Cumulvs: Providing fault-tolerance, visualization and steering of parallel applications, International Journal of High Performance Computing Applications 11 (1996) 224–236.
[9] K. Stockinger, J. Shalf, K. Wu, E. Bethel, Query-driven visualization of large data sets, in: Visualization, 2005. VIS 05. IEEE, 2005, pp. 167 – 174. doi:10.1109/VISUAL.2005.1532792.
[10] K. Wu, E. Otoo, A. Shoshani, On the performance of bitmap indices for high cardinality attributes, in: Proceedings of the Thirtieth international conference on Very large data bases - Volume 30, VLDB '04, VLDB Endowment, 2004, pp. 24–35.
[11] A. Murarasu, J. Weidendorfer, G. Buse, D. Butnaru, D. Pflüeger, Compact data structure and parallel alogrithms for the sparse grid technique, in: 16th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2011, to be published.
[12] M. Griebel, T. Dornseifer, T. Neunhoeffer, Numerical simulation in fluid dynamics: a practical introduction, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998.