

New Approaches for Particle Tracing on Sparse Grids

Christian Teitzel and Thomas Ertl

Computer Graphics Group, University of Erlangen
Am Weichselgarten 9, 91058 Erlangen, Germany
{teitzel,ertl}@informatik.uni-erlangen.de
<http://www9.informatik.uni-erlangen.de>

Abstract. Flow visualization tools based on particle methods continue to be an important utility of flow simulation. Additionally, sparse grids are of increasing interest in numerical simulations. In [14] we presented the advantages of particle tracing on uniform sparse grids. Here we present and compare two different approaches to accelerate particle tracing on sparse grids. Furthermore, a new approach is presented in order to perform particle tracing on curvilinear sparse grids. The method for curvilinear sparse grids consists of a modified Stencil Walk algorithm and especially adapted routines to compute, store, and handle the required Jacobians. The accelerating approaches are on the one hand an adaptive method, where an error criterion is used to skip basis functions with minor contribution coefficients, and on the other hand the so-called combination technique, which uses a specific selection of small full grids to emulate sparse grids.

1 Introduction

The idea of the sparse grid technique was developed in the 1960s by Babenko [1] and Smolyak [12]. In 1990 sparse grids were introduced to the field of numerical computation by Zenger [15]. By means of these grids, it is possible to reduce the total amount of data points or the number of unknowns in discrete partial differential equations. Due to these benefits, sparse grids are more and more used in numerical simulations nowadays [2, 3, 5–7].

On the other hand, it is rather difficult to visualize the results of the simulation process directly on sparse grids, since evaluation and interpolation of function values is quite complicated. Because of this, the results of numerical simulations on sparse grids are usually interpolated to the associated full grid. Then all known visualization algorithms on full grids can be performed, e.g. particle tracing, iso-surface extraction, and volume rendering. However, a major drawback of this procedure is the fact that the advantage of low memory consumption of sparse grids comes to nothing using the associated full grid for the visualization step.

Therefore, visualization tools working directly on sparse grids are an important topic of research. Recently, Heußner and Rumpf presented an algorithm for

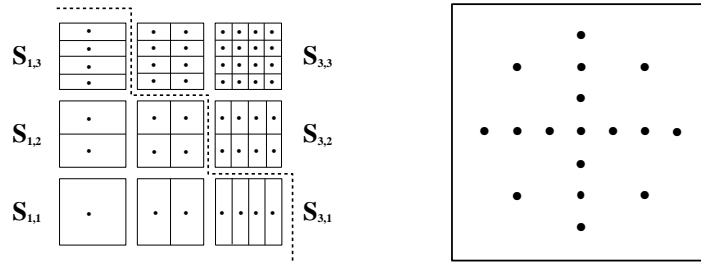


Fig. 1. On the left hand side a two-dimensional hierarchical subspace decomposition is shown and on the right hand side you can see the respective sparse grid.

iso-surface extraction on sparse grids [9]. In a previous work [14], we introduced particle tracing on uniform sparse grids and showed that sparse grids can be used for data compression in order to visualize huge data sets even on workstations with a limited amount of main memory. Now we present two major improvements of particle tracing on sparse grids. In order to accelerate sparse grid particle tracing, an adaptive approach for the interpolation process (Subsect. 2.1) and the so-called combination technique (Subsect. 2.2) are explained. Afterwards, we are going to introduce particle tracing on curvilinear sparse grids (Subsect. 3.3).

2 Sparse Grids

In this section a very brief introduction to the basic ideas of sparse grids is given. For a detailed survey of sparse grids we refer to [2, 15], for a brief summary to [14]. We describe only three-dimensional grids, whereas the sketches always reveal the two-dimensional situation.

If a smooth function f is used in a numerical computation it has to be discretized, which means that only function values at certain positions of a spatial grid are stored. On a uniform mesh this can be done by a hierarchical basis decomposition where piecewise tri-linear finite elements are used as basis functions. In the two-dimensional situation these basis functions are bi-linear hat-functions reaching their maxima at the dots on the left hand side of Fig. 1 and with disjoint supports denoted by the rectangles. Then, the interpolated function \hat{f}_n is given by

$$\hat{f}_n = \sum_{i_1=1}^n \sum_{i_2=1}^n \sum_{i_3=1}^n f_{i_1, i_2, i_3} \quad (1)$$

$$\text{where } f_{i_1, i_2, i_3} = \sum_{k_1=1}^{2^{i_1-1}} \sum_{k_2=1}^{2^{i_2-1}} \sum_{k_3=1}^{2^{i_3-1}} c_{k_1, k_2, k_3}^{(i_1, i_2, i_3)} \cdot b_{k_1, k_2, k_3}^{(i_1, i_2, i_3)} \quad (2)$$

The values $c_{k_1, k_2, k_3}^{(i_1, i_2, i_3)}$ are called contribution coefficients and $b_{k_1, k_2, k_3}^{(i_1, i_2, i_3)}$ denotes the mentioned basis functions.

The concept of sparse grids is to calculate the interpolated function \tilde{f}_n by using only certain basis functions:

$$\tilde{f}_n = \sum_{i_1+i_2+i_3 \leq n+2} f_{i_1, i_2, i_3} \quad . \quad (3)$$

It can be shown that by interpolating this way a very small loss of accuracy is rewarded with a huge amount of saved storage. The number of nodes of the underlying grids are given by $O(2^{3n})$ in case of full and by $O(2^n \cdot n^2)$ in case of sparse grids (compare Section 4).

2.1 Adaptive Evaluation of Sparse Grids

In order to improve on the rather time consuming standard sparse grid interpolation as described above, an adaptive approach for the function evaluation is presented in this subsection. Since our goal is to decrease the computing time of the sparse grid interpolation, we introduce an adaptive traversal of the standard sparse grid in order to compute function values. The idea is to omit contribution coefficients with a norm below a given error criterion during the interpolation process. Going into the details, we have to distinguish between adaptivity with regard to the L^2 and the L^∞ norm. Although they generate the same sparse grid, these norms lead to slightly different adaptive approaches.

Analyzing the situation with respect to the L^∞ norm, we find that the contribution of one basis element of subspace S_{i_1, i_2, i_3} to the function value is given by (compare Eq. (2))

$$\left\| c_{k_1, k_2, k_3}^{(i_1, i_2, i_3)} \cdot b_{k_1, k_2, k_3}^{(i_1, i_2, i_3)} \right\|_\infty = \left| c_{k_1, k_2, k_3}^{(i_1, i_2, i_3)} \right| \cdot \left\| b_{k_1, k_2, k_3}^{(i_1, i_2, i_3)} \right\|_\infty = \left| c_{k_1, k_2, k_3}^{(i_1, i_2, i_3)} \right| \quad . \quad (4)$$

Then, the greatest possible contribution of subspace S_{i_1, i_2, i_3} is

$$\max_{k_1, k_2, k_3} \left| c_{k_1, k_2, k_3}^{(i_1, i_2, i_3)} \right| \quad \text{with} \quad 1 \leq k_j \leq 2^{i_j-1} \quad (5)$$

and the maximum contribution of level n is

$$\sum_{i_1+i_2+i_3=n+2} \left(\max_{k_1, k_2, k_3} \left| c_{k_1, k_2, k_3}^{(i_1, i_2, i_3)} \right| \right) \quad . \quad (6)$$

For vector fields the absolute value $|\cdot|$ has to be replaced by an appropriate norm of the Euclidean space \mathbf{R}^m and we apply the maximum norm of \mathbf{R}^m in order to ensure maximum accuracy for all components of the vector field.

The actual concept of the adaptive grid traversal is that basis functions that have contribution coefficients with an absolute value below a given error bound are left out during the interpolation process. This results in a function evaluation that considers the local structure of the data set. That is, regions with a high variation in data values and, therefore, large contribution coefficients primarily contribute to the result, whereas coefficients of smooth regions are likely to be omitted.

As a second modification of the plain sparse grid algorithm, we have integrated a preprocessing step, which computes and stores the maximum contribution of each subspace (see Eq. (5)). This preprocessing step is performed during the conversion of a full grid to the appropriate sparse grid. This kind of adaptive grid traversal leads to a function evaluation with direction dependent accuracy, because different subspaces of the same level exhibit different resolutions in the three coordinate directions (reconsider the hierarchical subspace decomposition on the left hand side of Fig. 1). Omitting an entire level of the sparse grid (compare Eq. (6)) is totally independent of the spatial structure of the data set and, therefore, a contradiction to adaptive approaches in general.

Now let us discuss the adaptive approach based on the L^2 norm. A straightforward calculation shows that the contribution of one basis element of subspace S_{i_1, i_2, i_3} to the function value is given by

$$\left\| c_{k_1, k_2, k_3}^{(i_1, i_2, i_3)} \cdot b_{k_1, k_2, k_3}^{(i_1, i_2, i_3)} \right\|_2 = \left| c_{k_1, k_2, k_3}^{(i_1, i_2, i_3)} \right| \sqrt{(3 \cdot 2^{i_1 + i_2 + i_3 - 1})^{-3}} \quad . \quad (7)$$

Since $i_1 + i_2 + i_3 - 1 = n + 1$ with n denoting the current level, the square-root term only depends on the level and is, therefore, constant for all subspaces of the same level. Hence, this term is also a factor of the maximum contribution of the corresponding subspaces.

In contrast to the L^∞ norm, the L^2 norm generates an adaption strategy that considers not only the absolute value but also the level of a contribution coefficient. Contribution coefficients of higher levels are more likely to be omitted than coefficients of lower levels. Later on in Subsect. 4, we are going to see that the different properties of these adaptive grid traversals yield different results.

2.2 The Combination Technique

Since both the standard and the adaptive sparse grid interpolation of function values are quite complicated and rather time consuming, we have also implemented the so-called combination technique, which was introduced by Griebel, Schneider, and Zenger in 1992 [6]. Actually, the combination method has been used in numerical simulations in order to combine partial solutions computed on smaller, suitable full grids to the desired sparse grid solution. However, we start with a data set given on a sparse grid and decompose the grid such that the data set is represented on certain uniform full grids of low resolution. Now the fast and simple tri-linear interpolation can be performed on each of these full grids. The resulting value is computed by linear combination of the tri-linear interpolated full grid results. Specifically, it can be proven that the three-dimensional interpolated function $\tilde{f}_n \in \tilde{L}_n$ is given by

$$\tilde{f}_n = \sum_{i_1 + i_2 + i_3 = n+2} f_{i_1, i_2, i_3}^c - 2 \cdot \sum_{i_1 + i_2 + i_3 = n+1} f_{i_1, i_2, i_3}^c + \sum_{i_1 + i_2 + i_3 = n} f_{i_1, i_2, i_3}^c \quad (8)$$

where f_{i_1, i_2, i_3}^c denotes the tri-linear interpolation of function values on the respective full grid. Figure 2 reveals the two-dimensional situation, which also

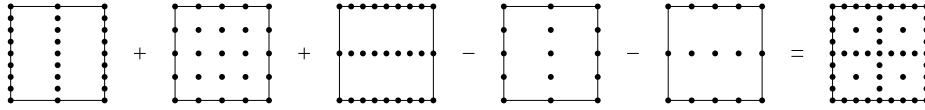


Fig. 2. A two-dimensional sparse grid of level 3 can be reconstructed by linear combination of five full grids of low resolution.

shows that the used full grids consist of the same nodes as the corresponding sparse grid.

Investigating the benefits of the combination technique, we find that the total number of summands of the standard sparse grid interpolation on a three-dimensional sparse grid of level n is given by $\frac{1}{6}n(n+1)(n+2)$ (compare Eq. (3)), whereas the total number of tri-linear interpolations of the combination method adds up to $\frac{3}{2}n(n-1) + 1$ (see Eq. (8)). That is, the number of tri-linear interpolations of the combination method is one order of magnitude lower than the number of summands of the standard interpolation. However, the main advantage of the combination technique is the fact that uniform full grids are used. Thus, it is possible to implement the interpolation routine in terms of tight `for`-loops (see Sect. 3.2), which makes the combination technique an order of magnitude faster than the standard approach even for the lower levels.

3 Particle Tracing

Lagrange visualization techniques of vector fields are based on the numerical solution of an initial value problem for the ordinary differential equation: $d\mathbf{x}/dt = \mathbf{v}(\mathbf{x}, t)$. Usually, a numerical integration method is used to obtain a solution. All such methods have in common that they must evaluate the vector field \mathbf{v} at certain positions, which are in general not at grid points. Therefore, the value of \mathbf{v} at such a position has to be interpolated. As mentioned in Subsect. 2, this interpolation on sparse grids is different from that one on full grids, whereas most other parts of the particle tracing algorithm can remain unchanged. Further exceptions are the routines required for handling curvilinear grids (see Subsect. 3.3).

Our sparse grid particle tracing modules are implemented as IRIS Explorer modules (compare also [14]). In order to visualize the particles, we have chosen lines, bands, tubes, balls, and tetrahedra as geometrical primitives. Of course, all kinds of traces can visualize an additional scalar value by means of color coding. Moreover, balls and tetrahedra can reveal another scalar value by their size. Besides that, bands and tetrahedra display the local vorticity of the flow via rotating around the actual streak line (see colored figures in the Appendix). As integration methods for the particle tracing algorithm, we use the integration schemes that we have already implemented in our full grid particle tracer. A comparison of these schemes can be found in [13]. An adaptive Runge-Kutta method of order 3 (RK3(2)) is used for the tests described in Sect. 4.

In contrast to tri-linear full grid interpolation, sparse grid interpolation does not operate locally, because one basis function in every subspace contributes to the function value. Since the tri-linear interpolation is one of the most time consuming operations during the particle tracing process on full grids [10], the complicated sparse grid interpolation is all the more time consuming. Therefore, it is important to execute the interpolation as fast as possible.

The contribution coefficients of the sparse grid are usually stored in a binary tree [2, 3, 9]. Then, a recursive tree traversal has to be performed in order to interpolate the function value. This tree traversal is very slow. Although caching strategies can increase the efficiency of the traversal [9], the computation of the values remains rather time consuming. In order to avoid the tree traversal and to accelerate the access to the contribution coefficients, we have developed a very efficient data structure based on arrays (see [14]). In the next two subsections our new approaches for further cutting the interpolation time are described.

3.1 Adaptive Grid Traversal

In order to perform an adaptive grid traversal as described in Subsect. 2.1, our former class hierarchy [14] has been slightly modified. The interpolation process has been enhanced in such a way that contribution values smaller than a given error bound are omitted. In addition, the preprocessing step for creating the actual sparse grid has been modified. During this process the contribution coefficients are computed from an analytic function or a full grid data set, or they are directly read from a sparse grid data set. Because we often deal with vector fields, each basis function does not contain a single contribution coefficient but an array of coefficients. For the purpose of adaptive function evaluation, the mentioned array has been extended by one component in order to store the maximum absolute value of the contribution coefficients. Moreover, a variable has been added for storing the maximum contribution coefficient of the entire subspace. Of course, all these additional variables storing maximum contribution coefficients are initialized during the creation of the sparse grid.

3.2 Combination Technique

Equation (8) determines the interpolation process for the combination technique, which combines full grids of low resolution to a resulting sparse grid. Since these full grids are uniform grids, the function values can be stored in three-dimensional arrays and derived by tri-linear interpolation. Thus, the interpolation method of the appropriately derived class can address the necessary function values in a tight loop. This fact makes the combination technique an order of magnitude faster than the previously described sparse grid interpolation even for low levels.

3.3 Curvilinear Sparse Grids

The underlying concept of curvilinear sparse grids is the same as for curvilinear full grids. In the case of uniform full grids, only the function values are stored

in an array, whereas in case of curvilinear grids, the function values and the coordinates of the grid points as well are saved. If a curvilinear sparse grid is considered, the contribution coefficients of the coordinates of the grid points are stored as additional components of the basis functions. For the combination technique, the coordinates of the grid points are stored in the arrays of the small full grids accordingly.

Particle tracing in arbitrary non-uniform grids requires the so-called *point location* to be performed for each integration step, in order to find the cell containing the actual particle position. For the case of curvilinear grids, particle tracing algorithms can be divided into P-space and C-space methods. Sadarjoen et al. [11] showed that P-space algorithms are in general preferable to C-space methods. Hence, we have implemented a P-space algorithm appropriately adapted for sparse grids. The *stencil walk* algorithm introduced by Buning [4] has been modified in the following way. First of all, we initialize the desired C-space position \mathbf{r}_c by starting in the center of our volume in C-space. In order to improve this guess, the C-space position is transformed into P-space. This is done by a sparse grid interpolation using either plain sparse grids, adaptive sparse grids, or the combination technique. If the difference of the transformed guess and the current position in P-space is small enough, we accept the C-space position. Otherwise the difference is transformed back into C-space via the inverse Jacobian and then added to the previous guess. Thereafter, the procedure is iterated until the appropriate position in C-space is located. On full grids, the stencil walk algorithm usually needs less than five iterations to find the correct C-space position.

As yet, the modifications of the stencil walk algorithm seem to be very moderate. But the main question is how to calculate the inverse Jacobian. On full grids, this is done on the fly by tri-linear interpolation of the eight Jacobians at the vertices of the current cell and subsequent matrix inversion. The Jacobians at the vertices are computed by finite differences. However, tri-linear interpolation is not possible on sparse grids. Thus, we have to use sparse grid interpolation and we have to store the inverse Jacobian, i.e. the respective contribution coefficients, at each sparse grid point. This memory overhead can only be justified with the fact that sparse grids themselves are very storage efficient.

Now we describe how the contribution coefficients of the inverse Jacobians are computed and stored. In case of uniform sparse grids, we compute the contribution coefficients in a preprocessing step from the input data and store the coefficients in the sparse grid structure, which is done by the `setCoeff(...)` methods. The data sets usually do not contain the Jacobians explicitly, thus, the Jacobians and their inverse matrices have to be calculated as well as their contribution coefficients. Since a contribution coefficient can not be computed from a single function value but from a specific collection of function values, the inverse Jacobians have to be stored somewhere. We have modified the `setCoeff(...)` methods in such a way that in a first pass the contribution coefficients of the function values and the inverse Jacobians are computed and stored in the sparse grid structure. In a second pass, the contribution coefficients of the inverse Ja-

Table 1. Computing times in CPU-seconds for integrating nine stream ribbons over 55 time steps in an analytic vortex flow using an adaptive RK3(2) scheme.

level	3	4	5	6	7	8
# of full grid points	9^3	17^3	33^3	65^3	129^3	257^3
uniform full grid	0.67 s	1.18 s	1.89 s	2.28 s	2.66 s	
uniform sparse grid	0.24 s	0.33 s	0.68 s	0.93 s	4.51 s	5.91 s
uniform combination	0.07 s	0.12 s	0.20 s	0.30 s	1.15 s	1.61 s
curvilinear full grid	0.70 s	1.30 s	2.58 s	5.28 s	10.59 s	
curvilinear sparse grid	1.56 s	3.28 s	6.82 s	9.31 s	22.72 s	31.16 s
curvilinear combination	0.64 s	1.19 s	2.02 s	3.02 s	6.05 s	8.49 s

cobians are computed and stored over the original components of the inverse Jacobians. The second pass traverses the levels beginning with the highest level and ending with level 1. It is done this way because the contribution coefficients only depend on the current function value and on the function values of lower levels. Hence, it is possible to overwrite the original components of the inverse Jacobians successively. Notice that level 0 is not part of the mentioned second pass because in level 0 contribution values and function values coincide.

4 Results

In order to compare our sparse grid particle tracing modules with a full grid particle tracer, several data sets were used. For uniform grids we used the data set of a cavity flow, which was provided by S. H. Enger from the Department of Fluid Mechanics and is given on a full grid with 129^3 nodes, which corresponds to level 7 in sparse grid terminology. The data set contains the velocity, pressure, and temperature at each vertex requiring more than 40 MB. The same data set with a resolution of 257^3 (level 8) would need more than 320 MB, which is probably too much for most workstations. On the other hand, this data set stored on a sparse grid consumes only 175 kB for level 7 and 415 kB for level 8 (compare Table 2). For curvilinear grids the NASA blunt fin data set was used (see Fig. 4). In addition, we used several analytic test data sets on uniform and different curvilinear grids. Therefore, we were able to create sparse and full grids in any resolution only limited by the main memory of the used machine. These vector fields were chosen for our quantitative performance tests, with the results being comparable to the ones obtained from the discrete given data sets. All tests were performed on an SGI with a 250 MHz R10000 processor. For each experiment nine stream ribbons consisting of about 500 particles were integrated. The CPU-times were measured in seconds and are listed in Table 1. The measured times show that interactive particle tracing is possible even on sparse grids of level 8 by using the combination technique.

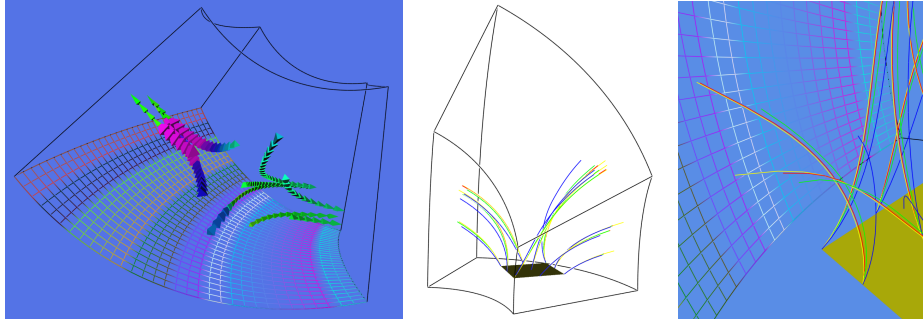


Fig. 3. In the image on the left hand side, streak tetrahedra in an analytically given flow on a curvilinear sparse grid of level 5 are shown. In the next two pictures streak lines depict a vortex flow field; yellow lines display the flow on a full grid, blue, green, and red lines on a curvilinear sparse grid of level 2, 3, and 4 respectively. In the closeup on the right hand side, it can be seen that the traces computed on the full grid and the sparse grid of level 4 are almost identical.

Investigating the accuracy of sparse grid particle tracing, the traces computed on sparse grids are compared with their counterparts resulting from full grids. Theory tells us that the difference should be rather small. In fact, the results of particle tracing on the analytic data set confirm this estimation, since the lines computed on uniform full and sparse grids coincide on screen for levels greater than 3 (compare [14]) and for levels greater than 4 on curvilinear grids (see Fig. 3). However, for the derivation of the upper bounds for the interpolation errors, a certain smoothness of the data was a prerequisite. Since discrete data sets are not smooth at all, these estimations do not hold in this case. Indeed, for discrete data we found out [14] that the particle traces computed on sparse grids converge rather slowly to the full grid solution. Nevertheless, due to the great advantage of low memory consumption, it is possible to use a sparse grid of a sufficiently high level to overcome this problem.

For the adaptive grid traversal, several experiments have shown that it is important whether the L^2 norm is used for the adaptive traversal or the L^∞ norm. Employing the L^∞ norm leads to a marginal decrease in computing time but to a significant loss in accuracy. However, by using the L^2 norm, it is possible to decrease computing time by about 20 per cent and to achieve nearly the quality of the corresponding plain sparse grid.

The next approach for accelerating particle tracing on sparse grids is the combination technique. The first advantage of this technique compared to adaptive grid traversal is the fact that there is no loss in accuracy at all. Combination technique and plain sparse grid interpolation create exactly the same particle path. The second and more important benefit is that the combination technique is almost by a factor of four faster than plain sparse grid interpolation (compare Table 1). That is, particle tracing based on the combination method is a lot faster and also more accurate than particle tracing based on adaptive sparse grid

Table 2. Memory consumption of a typical data set.

level	5	6	7	8	9	10
uniform full grid	640 kB	5 MB	40 MB	320 MB	2.5 GB	20 GB
uniform sparse grid	29 kB	73 kB	175 kB	415 kB	970 kB	2.2 MB
uniform combination	110 kB	295 kB	760 kB	1.8 MB	4.5 MB	10.5 MB
curvilinear full grid ¹	1 MB	8 MB	64 MB	512 MB	4 GB	32 GB
curvilinear sparse grid ²	99 kB	248 kB	595 kB	1.4 MB	3.2 MB	7.5 MB
curvilinear combination ²	375 kB	1 MB	2.5 MB	6.1 MB	15.3 MB	35.7 MB

¹ including coordinates but excluding inverse Jacobians

² including coordinates and inverse Jacobians

interpolation. Hence, the combination technique should be used for interpolation on sparse grids.

Now let us turn to curvilinear sparse grids. We have implemented curvilinear grids with all three sparse grid interpolation methods, but in consideration of the last paragraph we usually employ the combination technique in connection with curvilinear sparse grids. For a first test of particle tracing in curvilinear sparse grids we have used the NASA blunt fin data set (see Fig. 4). Additionally, our module has been verified with several analytic data sets (compare Fig. 3). On the one hand side these tests have confirmed that smooth data sets are more appropriate for using sparse grid methods than discontinuous data. On the other hand these tests have revealed that particle traces calculated on curvilinear sparse grids converge slower to the corresponding full grid trace than particle traces computed on uniform sparse grids. The reason for this decline in accuracy might be due to a less accurate point location caused by an intensive use of sparse grid interpolation in the stencil walk algorithm. Finally, time measurements have shown that particle tracing on curvilinear sparse grids is about five times slower than tracing on uniform grids. This is roughly the same deceleration as on full grids.

The great advantage of the sparse grid technique is the low number of required grid points. Table 2 demonstrates this benefit listing the memory consumption for various grid levels on the assumption that a typical data set resulting from a numerical flow simulation is given. Such data usually contain five floating point values, namely three velocity components, pressure, and temperature, at each grid node. Then, these floating point values add up to 20 bytes per node. In case of curvilinear grids three more floating point numbers for the coordinates and nine additional values for the inverse Jacobians are stored at each grid node. Nevertheless, sparse grids are very suitable for compressing huge data sets. This opens up the potential to visualize them even on workstations with a limited amount of main memory.

5 Conclusion

We have introduced particle tracing on curvilinear sparse grids and presented competing approaches for accelerating the time consuming sparse grid interpolation process. Technically, we have implemented adaptive sparse grids with error monitoring and the combination technique. This allows to carry out flow visualization directly on sparse grids without prior transformation to the associated full grids. This is an important step for the broader application of the sparse grid method, since in real applications it is often impossible to load full grids of more than 128^3 nodes into the main memory of a workstation for visualization purposes. Furthermore, the sparse grid approach can be used as a compression method in order to realize particle tracing in huge data sets on workstations with a small amount of main memory.

Feasible directions of future work are texture based algorithms and iconic methods combined with feature extraction. In addition, our sparse grid particle tracer could be extended to multi-block data sets in the same way as it was done in our full grid particle tracing module [8]. In a parallel work we are investigating further visualization techniques on sparse grids, namely hardware assisted volume rendering and fast iso-surface extraction.

References

1. K. I. Babenko. Approximation by trigonometric polynomials in a certain class of periodic functions of several variables. *Soviet Mathematics*, 1:672–675, 1960. Translation of Doklady Akademii Nauk SSSR.
2. H.-J. Bungartz. *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung*. PhD thesis, TU Munich, 1992.
3. H.-J. Bungartz and T. Dornseifer. Sparse Grids: Recent Developments for Elliptic Partial Differential Equations. In *Multigrid Methods V*, Lecture Notes in Computational Science and Engineering. Springer-Verlag, 1998.
4. P. Buning. Numerical Algorithms in CFD Post-Processing. In *Computer Graphics and Flow Visualization in Computational Fluid Dynamics*, number 1989-07 in Lecture Series, Brussels, Belgium, 1989. Von Karman Institute for Fluid Dynamics.
5. M. Griebel, W. Huber, U. Råde, and T. Störtkuhl. The combination technique for parallel sparse-grid-preconditioning or -solution of PDE's on multiprocessor machines and workstation networks. In L. Bougé, M. Cosnard, Y. Robert, and D. Trystram, editors, *Second Joint International Conference on Vector and Parallel Processing*, pages 217–228, Berlin, 1992. CONPAR/VAPP, Springer-Verlag.
6. M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In P. de Groen and R. Beauwens, editors, *International Symposium on Iterative Methods in Linear Algebra*, pages 263–281, Amsterdam, 1992. IMACS, Elsevier.
7. M. Griebel and V. Thurner. The efficient solution of fluid dynamics problems by the combination technique. *Int. J. Numer. Methods Heat Fluid Flow*, 5:251–269, 1995.
8. R. Grosso, M. Schulz, J. Kraheberger, and T. Ertl. Flow Visualization for Multi-block Multigrid Simulations. In P. Slavick and J. J. van Wijk, editors, *Virtual*

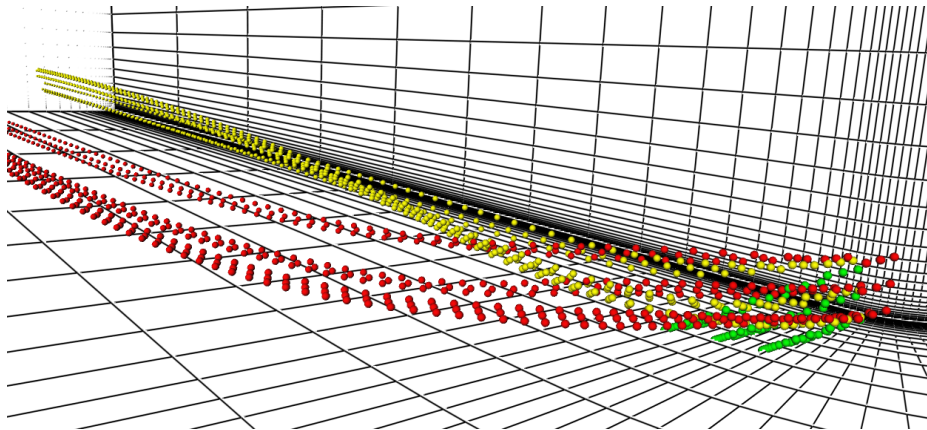


Fig. 4. Streak balls display the flow in the blunt fin data set; the red balls are computed on a curvilinear sparse grid of level 4, the yellow ones on a grid of level 3, and the green ones on a grid of level 2.

- Environments and Scientific Visualization '96*, Heidelberg, 1996. Springer-Verlag. Proceedings of the Eurographics Workshop in Prague, Czech Republic.
9. N. Heußner and M. Rumpf. Efficient Visualization of Data on Sparse Grids. In H.-C. Hege and K. Polthier, editors, *Mathematical Visualization*, pages 31–44, Heidelberg, 1998. Springer-Verlag.
 10. D. N. Kenwright and D. A. Lane. Optimization of Time-Dependent Particle Tracing Using Tetrahedral decomposition. In G. M. Nielson and Silver D., editors, *Visualization '95*, pages 321–328, Los Alamitos, CA, 1995. IEEE Computer Society, IEEE Computer Society Press.
 11. A. Sadarjoeen, T. van Walsum, A. J. S. Hin, and F. H. Post. Particle Tracing Algorithms for 3D Curvilinear Grids. In *Fifth Eurographics Workshop on Visualization in Scientific Computing*, 1994.
 12. S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Soviet Mathematics*, 4:240–243, 1963. Translation of Doklady Akademii Nauk SSSR.
 13. C. Teitzel, R. Grosso, and T. Ertl. Efficient and Reliable Integration Methods for Particle Tracing in Unsteady Flows on Discrete Meshes. In W. Lefer and M. Grave, editors, *Visualization in Scientific Computing '97*, pages 31–41, Wien, April 1997. Springer-Verlag. Proceedings of the Eurographics Workshop in Boulogne-sur-Mer, France.
 14. C. Teitzel, R. Grosso, and T. Ertl. Particle Tracing on Sparse Grids. In D. Bartz, editor, *Visualization in Scientific Computing '98*, pages 81–90, Wien, April 1998. Springer-Verlag. Proceedings of the Eurographics Workshop in Blaubeuren, Germany.
 15. C. Zenger. Sparse grids. In *Parallel Algorithms for Partial Differential Equations: Proceedings of the Sixth GAMM-Seminar*, Kiel, 1990.