SOFTWARE CONCEPTS OF A SPARSE GRID FINITE DIFFERENCE CODE

T. Schiekofer, G. Zumbusch
Institut für Angewandte Mathematik
Abteilung Wissenschaftliches Rechnen und Numerische Simulation
Wegelerstraße 6
53115 Bonn

SUMMARY

Sparse grids provide an efficient representation of discrete solutions of PDEs and are mainly based on specific tensor products of one-dimensional hierarchical basis functions. They easily allow for adaptive refinement and compression. We present special finite difference operators on sparse grids that possess nearly the same properties as full grid operators. Using this approach, partial differential equations of second order can be discretized straightforwardly. We report on an adaptive finite difference research code implementing this on sparse grids. It is structured in an object oriented way. It is based on hash storage techniques as a new data structure for sparse grids. Due to the direct access of arbitrary data traditional tree like structures can be avoided. The above techniques are employed for the solution of parabolic problems. We present a simple space-time discretization. Furthermore a time-stepping procedure for the solution of the Navier Stokes equations in 3D is presented. Here we discretize by a projection method and obtain Poisson problems and convection-diffusion problems.

SPARSE GRIDS

A detailed introduction to the field of sparse grids can be found in [$BUN92$, $GRI91$, $ZEN91$]. Nevertheless, the most important aspects are described in what follows.

Let us consider the piecewise linear hierarchical basis in one dimension [$FAB09, YSE86$] as a starting point for sparse grids. The generalization from one dimension to the d-dimensional case can be done by a tensor product ansatz. Counting dimensions in the corresponding subspace splitting of the resulting full grid discretization space in two dimensions with piecewise bilinear hierarchical basis functions, it follows that the number of grid points in subspaces T_{i_1,i_2} with $i_1 + i_2 = c$ is 2^{c-2}. The contribution of subspaces T_{i_1,i_2} with $i_1 + i_2 = c$ to the interpolant of a function u is of the same order $\mathcal{O}(2^{-2c})$ as far as the L_2- and the maximum norm are concerned, and of order $\mathcal{O}(2^{-c})$ with respect

Table 1: comparison of sparse and full grids

property	regular sparse grid	full grid
accuracy in L_2	$\mathcal{O}(h^2 \cdot (\log(h^{-1})^{d-1}))$	$\mathcal{O}(h^2)$
accuracy in L_∞	$\mathcal{O}(h^2 \cdot (\log(h^{-1})^{d-1}))$	$\mathcal{O}(h^2)$
accuracy in $\|\cdot\|_A$	$\mathcal{O}(h^1)$	$\mathcal{O}(h^1)$
# grid points	$\mathcal{O}(N \cdot (\log(N)^{d-1}))$	$\mathcal{O}(N^d)$
$d = 3, N = 2^{10}$	$1.1 \cdot 10^5$	$1.1 \cdot 10^9$
$d = 3, N = 2^{20}$	$3.3 \cdot 10^8$	$1.2 \cdot 10^{18}$

to the energy norm (of the Laplacian) if u fulfills some smoothness requirements, which are $\frac{\partial^4 u}{\partial x_1^2 x_2^2} \in \mathcal{C}(\bar{\Omega})$ in a classical setting (Ω is the corresponding domain). Analogously, in d dimension u has to fulfill $\frac{\partial^{2d} u}{\partial x_1^2 \cdots x_d^2} \in \mathcal{C}(\bar{\Omega})$. For the proofs of this estimates see [$BUN92$]. Considering costs versus accuracy it seems more reasonable to take a triangular scheme of subspaces into account rather than a quadratic scheme. This triangular schema of subspaces which can be found in Figure 1 (left side) defines sparse grids. A sparse grid on level 5 can be found in Figure 1 (right side).

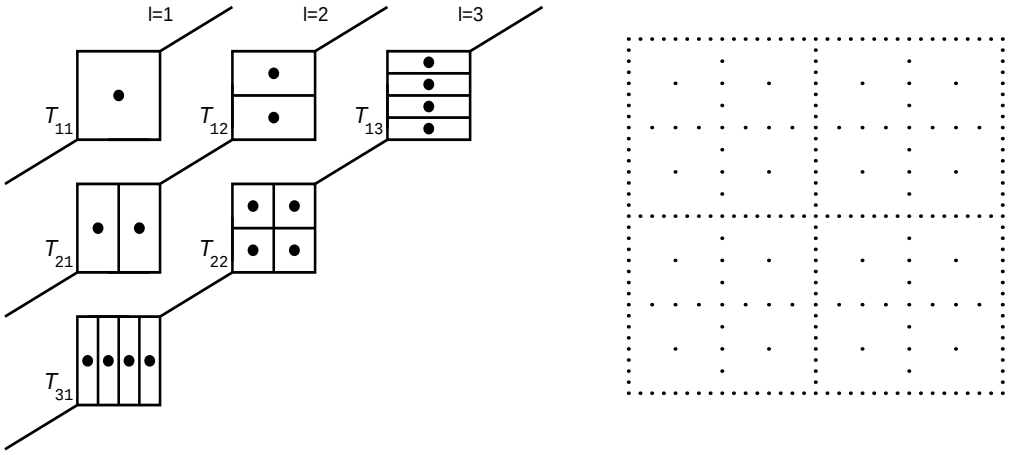

Figure 1: Triangular scheme of subspaces (left) and sparse grid on level 5 (right).

Due to the construction by subspaces, adaptivity is provided in a natural way. The great advantage of sparse grids compared to full grids is their substantially less number of grid points for comparable interpolation errors. A comparison can be found in Table 1.

For further information on sparse grids we refer to [$BUN92$, $BDZ96$, $BD97$, $GRI91$, $GRI97$, $SCH98$, $ZEN91$] and the references therein.

Amongst others, discretizations of partial differential equations can be obtained using finite elements and finite differences. The theory and application of finite element discretizations on sparse grids so far can be found in [$BUN92, BUN96, DOR97, ZEN91$] and the references therein. Due to the hierarchical basis the resulting matrices, i.e. the stiffness matrices and mass matrices, are not sparse. Hence, a naive application of the matrices in a matrix vector multiplication can not be performed in $\mathcal{O}(n)$ operations where n is the number of grid points. However, using a factorized representation of the finite element matrices gives optimal complexity $\mathcal{O}(n)$, just as in the full grid case. Additionally, multigrid methods have been studied, implemented and tested for finite elements on sparse grids. In the case of finite elements the resulting matrices for constant coefficients on sparse grids remain symmetric for symmetric operators. One drawback of the finite element method is the difficult implementation already in two dimensions, that is a major problem in three or even more dimensions for general differential operators involving non-constant coefficient functions.

FINITE DIFFERENCE OPERATORS

The easier implementation of finite differences in arbitrary dimensions compared to the implementation of finite elements is one motivation for the development of these operators. Finite difference operators on sparse grids for first, second or higher derivatives can all be obtained in the same way. As finite difference on full grids use nodal values, it seems natural that the starting point for the sparse grid operators are also nodal values rather than hierarchical values. If a discrete derivative in coordinate direction $j \in \{1, \ldots, d\}$ has to be computed, one has to perform 3 steps, cf. [$GRI97, SCH98$] ($I = \{1, 2, \ldots, d\}$):

1. apply a transformation $H_{I\{j\}}$ from the nodal basis to the hierarchical basis in each coordinate direction except j
2. apply the finite difference stencil $\tilde{D}_j$ in coordinate direction j (the stencils are just the same as in the well known full grid case) according to the local mesh parameter on the respective one-dimensional grid lines
3. apply a transformation $H^{-1}_{I\{j\}}$ from the hierarchical basis back to the nodal basis in each coordinate direction except j

Using this scheme, the d-dimensional discrete Laplace reads

$$\Delta = \sum_{j=1}^{d} H^{-1}_{I\{j\}} D_{jj} H_{I\{j\}}$$

and a convective part of an elliptic partial differential equation of second order can be written as

$$b^T \nabla = \sum_{j=1}^{d} b_j H^{-1}_{I\{j\}} D_j H_{I\{j\}}.$$

Here, D_{jj} denotes a one-dimensional stencil for second derivatives, e.g. $h_{li}^{-2}[1 \quad -2 \quad 1]$, and D_j represents a stencil for first derivatives, e.g. backward, forward, upwind or centered differences. All of these stencils have to be applied according to the local mesh parameters h_{li}. As in the finite element case, the corresponding matrices are not sparse. But in contrast to the finite element case, we obtain non-symmetric discretization matrices even for non-self adjoint operators. Due to the lack of symmetry we use BiCGSTAB as iterative solver for the arising linear systems of equations where we use the inverse of the diagonal of the matrices as preconditioner. Although the discretization matrices are not symmetric, the eigenvalues remain real and positive. Furthermore, the discretization matrices for the Laplace are P-matrices [$HJ91$], i.e. all k-by-k principal minors of the discretization matrices are positive for $1 \leq k \leq n$ where n denotes the number of grid points. Note that, in the case of full grids, this approach drops down to well known finite difference stencils [$SCH98$].

CONSISTENCY AND STABILITY

Finite differences on sparse grids possess the same order of consistency as on full grids. Here, the order of consistency in the sparse grid case in measured according to the finest occurring mesh parameter which is on the boundary. The proofs for these assertions are based on Taylor series expansions and can be found in [$SCH98$]. The assumptions that have to be made are $u \in C^{d+2,1}(\bar{\Omega})$.

Table 2: Order of consistency for sparse grid finite differences.

operator	order of consistency
$\partial^2/\partial \mathbf{x}^2$	$\mathcal{O}(h^2)$ (second order)
$\mathbf{\Delta}$	$\mathcal{O}(h^2)$ (second order)
$\partial^2/\partial \mathbf{x}\partial \mathbf{y}$	$\mathcal{O}(h)$ (first order)
$\partial/\partial \mathbf{x}$	$\mathcal{O}(h)$ (first order: forward/backward/upwind difference)
$\partial/\partial \mathbf{x}$	$\mathcal{O}(h^2)$ (second order: centered difference)

As consistency and stability lead to convergence, stability of the above approach is of major interest. Therefore, we computed the eigenvalues of the Laplace in two dimensions numerically on successive levels of discretization. As it can be seen in Table 3, the smallest eigenvalue tends to $2\pi^2$ which is the smallest eigenvalues of the continuous operator, and the largest eigenvalue as well as the norm of the matrices increases by a factor of 4 which indicates that the condition number is proportional to h^{-2}. Preconditioning with the diagonal of the discretization matrices leads to condition numbers that are proportional to h^{-1}. Note that, preconditioning the Laplace on full grids with the diagonal of the discretization matrix is useless as the diagonal is just a scaled identity.

Table 3: Extreme eigenvalues and condition numbers κ of the sparse grid finite difference Laplace operator in 2 dimensions.

Level	λ_{min}	λ_{max}	κ	quotient
1	18.387503	69.612496	3.79	—
2	19.273832	263.300596	13.66	3.61
3	19.587081	1031.815928	52.67	3.86
4	19.691734	4103.953343	208.43	3.96
5	19.724926	16391.988294	831.24	3.99
6	19.735029	65543.997071	3320.36	3.99

DESIGN OF A SPARSE GRID CODE

The goal for the development of our finite difference sparse grid code is to be able to test and to verify different types of discretizations on sparse grids and to tackle different types of partial differential equations. Hence a flexible and modular design is a must.

Techniques such as abstract data types and object oriented programming provide such a flexibility, see [$ABL97$]. However, they often lead to slow inefficient code due to an over-use of design features. For example, overloading the arithmetic operators '+' and '*' for vectors in C++ usually is less efficient than proving directly a `saxpy` operation for expressions of $\alpha\vec{v} + \vec{w}$ type. Performing tree traversal for the operations above is even slower. Hence, splitting a large code into many small functions and loops may lead to inefficiencies, which cannot be resolved by an optimizing compiler.

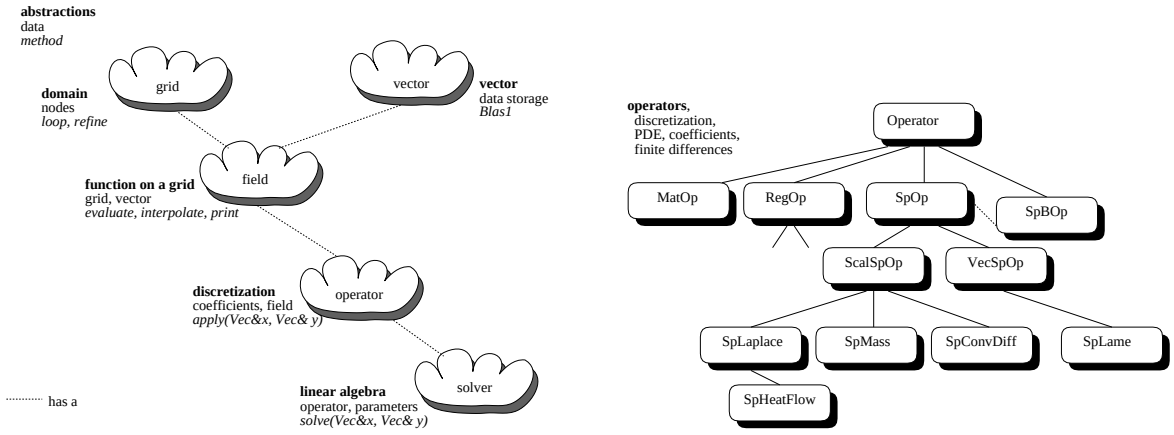

Figure 2: Hierarchy of abstractions of a sparse grid code. General abstractions (left) and differential operators (right).

We have based our code on several fundamental abstractions, which are well separated both in functionality and in implementation. This guarantees that we do not loose efficiency in a substantial way due to this separation. We have identified the following building blocks, see figure 2 (left). They are ordered from low level, computationally

expensive and efficient, to high level routines, where efficiency is achieved through call of some efficient subroutines. Similar abstractions can be found in other object oriented software packages for partial differetial equatins such as Diffpack, see [$BL97$].

- *grid*: geometric description of an adaptively refined sparse grid and provides i/o, refinement and addressing, or e.g. a full grid for debugging purposes
- *vector*: a large container for real numbers, including `BLAS` level 1
- *field*: a (solution) function on a sparse grid. It provides a mapping between a grid and a vector and interprets the data as (collocated) scalar or vector field
- *operator*: the finite difference operators and operates on grids, see figure 2 (right)
- *solver*: different iterative (Krylov) solvers, uses a differential operator

HASH STORAGE TECHNIQUES

Previous implementations of adaptive sparse grid were based on tree data structures, see [$ZEN91$]. Each node of the grid is represented by some data, such as a solution value and several auxiliary values, and several pointers to other nodes. At least two of the pointers point to hierarchical sons of the node in one direction. Other pointers link the node to sons in other directions and to the father node. All operations, including all numerical operations such as `saxpy` operations were implemented as tree traversals, which is complicated to develop, error prone, and slow compared to vector operations, see above.

We propose a different storage technique here, which discards trees and pointers and is based on hash addressing instead, see [$WS95, SCH98$]. Initially our motivation to substitute the tree was a shortage of development and coding time. The coordinates of a node often can be computed easily, they are coded in a specific way and form a key for a hash table. Hash storage is based on a mapping of the data to a vector (table) by a hash function, see figure 3. Given a cheap, deterministic and optimal hash function, each piece of data can be retrieved in constant time by its key. However, real hash functions lead to collisions in the vector, which can be resolved e.g. by chaining. In a statistical setting, the average access time in the hash table is still constant time $\mathcal{O}(1)$.

The coding and hash table lookup can be separated completely from the remaining code. The coordinates of any of the $2d$ son or d father nodes in all directions of a sparse grid can be computed by simple integer operations. A specific son node may or may not exist, a fact which is reported from the hash table lookup operation. We can discard the necessary pointers of the tree, which saves both memory and administration of the tree. In an adaptively refined sparse grid, the nearest neighbor nodes cannot be computed like for a regular sparse grid. The direct lookup is substituted by a small search procedure, looking for the neighbor in the hash table. This has no impact on the overall complexity of a finite difference operator, however.

We did not specify the hash function so far. Several functions based on pseudo-random numbers experimentally perform very well. A special choice can be used for the

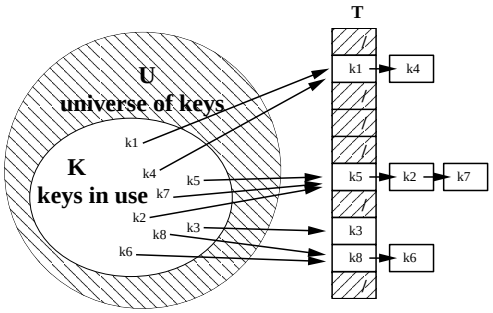

Figure 3: Hash table, collision resolution with chaining.

parallelization of a hash based code such as [$WS95, PB96$]. Space-filling curves can be used for dynamically partitioning the nodes onto a parallel computer and as a hash key on the local processor, see [$GZ98$].

APPLICATIONS TO SPACE-TIME DISCRETIZATIONS

Let us consider the numerical solution of a parabolic boundary-initial-value problem. Standard algorithms separate the problem into a time and a space discretization. A solution at a time slice $t = t_i$ is computed from one or several previous time steps t_j, $j < i$. An implicit Euler discretization in time leads to an elliptic boundary-value problem in space for each time step t_i.

We propose a different approach here. We rewrite the time dependent problem, e.g. a heat equation in the domain $\Omega \subset \mathbb{R}^d$

$$u_t \;=\; \Delta u \tag{1}$$

into a boundary value problem in the domain $[t_{\text{start}}, t_{\text{end}}] \times \Omega \subset \mathbb{R}^{d+1}$ of one dimension higher:

$$\Big(\frac{\partial}{\partial x_0} \;-\; \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2}\Big)\; u \;=\; 0 \tag{2}$$

A first order upwind discretization of $\frac{\partial}{\partial x_0}$ in space-time now is equivalent to the previous implicit Euler discretization in time. The numerical approximation is not affected. A space-time discretization (equation 2) implies the storage of the solution at all time slices at once, a $d+1$ dimensional manifold, which is more expensive than the storage of a small number of time slices in the time-stepping algorithm (equation 1). Hence Space-time discretizations are used for special purposes only, see [$VAN93$].

However, the storage of a $d+1$ dimensional space-time manifold in sparse grids can be almost as expensive as the storage of a d dimensional space manifold. The number of nodes of a regular space-time discretization is a factor of $|logh|$ higher than of a single space discretization. In the case of a specific time-periodic problem, where the complete

history in t has to be stored anyway, sparse grids have been applied successfully, see [$BAL94$].

The discretization of the space-time operator (equation 2) with finite differences on a sparse grid follows the three-step recipe given in the first chapter. We use a first order upwind discretization in x_0 direction and central differences in the remaining directions $x_1, \ldots, x_d$, see figure 4. The standard linear hierarchical basis transform H is used in space along with the implementation of boundary conditions. In time direction x_0 we use a hierarchical basis transform of piecewise constant functions centered towards t_{end}. This corresponds to the first order time discretization and is based on the stencil [0 1 1] compared to the linear hierarchical basis $[\frac{1}{2}\ 1\ \frac{1}{2}]$. The solution at $t = t_{\text{start}}$ is implemented as a Dirichlet condition, while at $t = t_{\text{end}}$ there is no boundary condition present. Higher order one-sided and central finite differences have been tested, too.

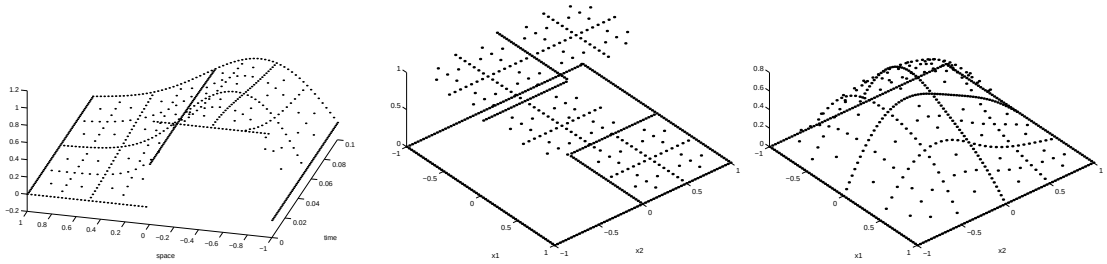

Figure 4: Transient heat equation with homogeneous Dirichlet conditions, discretized in space-time with sparse grids. Initial conditions piecewise constant. 2D (left) and clip planes through a 3D cube at $t_{\text{start}} = 0$ and $t_{\text{end}} = .1$ (middle and right).

As a consequence of the sparse grid space-time discretization it is easy to employ adaptive refinement for time dependent problems. A simple error indicator in $\mathbb{R}^{d+1}$ of scaled hierarchical surplus type gives rise to a local refinement in space and at the same time to a local time-step in the vicinity of the spatial refinement. Compared to standard procedures with separated space refinement and a global, adaptive time-step control, this is conceptually very simple. Furthermore a consistent local time-stepping is achieved.

APPLICATIONS TO NAVIER-STOKES EQUATIONS

We now want to apply the finite difference operators on sparse grids to the unsteady Navier Stokes equations that describe the laminar flow of incompressible and viscous fluids. Here, the equations of interest are

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \text{grad})\mathbf{u} + \text{grad } p = Re^{-1}\Delta \mathbf{u} + (1 - \beta T)\mathbf{g} \tag{3}$$

$$\text{div } \mathbf{u} = 0 \tag{4}$$

$$\frac{\partial T}{\partial t} + (\mathbf{u} \cdot \text{grad})\mathbf{T} = \frac{1}{RePr}\Delta T + q \tag{5}$$

together with adequate boundary conditions. Here, $\mathbf{u}, T, p, \mathbf{g}, Re$ and Pr denote the velocities, temperature, pressure, external forces, Reynolds number and the Prandtl number, respectively. The coupling of fluid flow and temperature is achieved by the Boussinesq approach. We discretize equations (3)-(5) using a pressure linked method comparable to the SIMPLE method. Here, we solve a Laplace equation with Dirichlet boundary conditions for the pressure where we take the tangential components on the boundary into account to obtain these boundary conditions. A detailed description of this method can be found in [$GRE87$].

As test cases for our method we consider problems arising in simulating CVD (chemical vapour deposition) phenomena. In this test case fluid is flowing through a CVD reactor while the apparatus is heated from below. The heat source is centered at $(\frac{1}{4}, \frac{1}{2}, 0)$ of the computational domain $\Omega = [0,1]^3$ and diminishes exponentially on the boundary. A numerical difficulty lies in the rotating susceptor at the bottom of the reactor. Obviously, the rotation has influence both on temperature and velocities. Hence, it is reasonable to employ adaptive refinement to capture the critical criteria (e.g. jumps in the boundary conditions for the velocities) of the quantities temperature and velocities. The approximate geometry of the reactor can be seen in Figure 5. The marked box that contains the susceptor (and also the heat source) is the domain of our simulations. On the right hand side of this figure an associated grid resolving susceptor and heat source can be obtained. Computations are carried out for the moderate Reynolds number $Re = 1$. Note that, small Reynolds numbers are realistic for the flow in such a reactor. The impact of the rotation can be obtained in Figure 6. On the left side of this figure a temperature distribution in the clip plane indicated in Figure 5 can be seen. In the middle and on the right side the figure shows the velocity in direction of the flow. Due to the rotation of the susceptor we obtain a non-symmetric temperature distribution. Furthermore, the velocity of the fluid above the susceptor reflects the rotation of the susceptor.

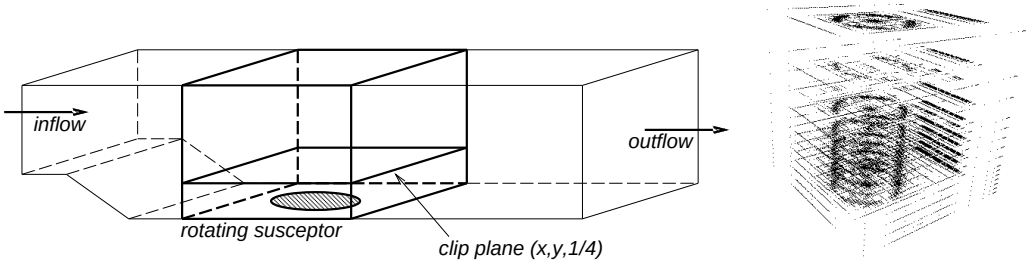

Figure 5: Approximate geometry of the reactor and domain of simulation (marked box).

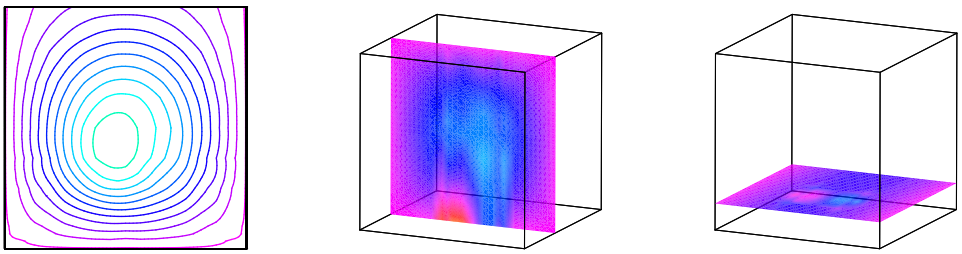

Figure 6: temperature distribution in the clip plane (left) and velocity component in direction of the flow (middle and right; fluid is flowing from front left to backward right)

CONCLUSIONS

In this paper we have presented several concepts for the solution of partial differential equations: For the discretization of PDEs we have discussed the well known sparse grids and a novel discretization scheme on sparse grids, based on the finite differences, including its properties. For the discretization of time dependent problems, we have introduced sparse grids discretizations in space-time domain. Furthermore a time stepping projection method for incompressible Navier Stokes equations has been presented.

For the implementation of such sparse grid methods, we have discussed some software concepts. This included an object oriented layout of the basic ingredients of such a code for a highly flexible and modular and still efficient code development. Furthermore an alternative storage technique based on hash tables for adaptive sparse grids has been discussed, which showed several advantages such as simplicity and low memory consumption.

REFERENCES

[*ABL*97] ARGE E., BRUASET A. M. and LANGTANGEN, H. P.: "Object-oriented Numerics", in Numerical Methods and Software Tools in Industrial Mathematics, eds.: M. Dæhlen and A. Tveito, Birkhäuser, Basel, 1997.

[*BAL*94] BALDER R.: "Adaptive Verfahren für elliptische und parabolische Differentialgleichungen auf dünnen Gittern", Ph.D. Thesis, TU München (1994)

[*BL*97] BRUASET A. M. and LANGTANGEN, H. P.: "A Comprehensive Set of Tools for Solving Partial Differential Equations; Diffpack", in Numerical Methods and Software Tools in Industrial Mathematics, eds.: M. Dæhlen and A. Tveito, Birkhäuser, Basel, 1997.

[*BUN*92] BUNGARTZ H.-J.: "Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung", Ph.D. Thesis, TU München (1992)

[*BDZ*96] BUNGARTZ H.-J., DORNSEIFER T. and ZENGER C.: "Tensor product approximation spaces for the efficient numerical solution of partial differential equations", to appear in Proc. Int. Workshop on Scientific Comput., Konya, 1996, Nova Science Publishers

[*BD*97] BUNGARTZ H.-J. and DORNSEIFER T.: "Sparse grids: Recent developments for elliptic partial differential equations", Institut für Informatik der TU München, SFB-Report 342/02/97 A, 1997.

[*DOR*97] DORNSEIFER, T.: "Diskretisierung allgemeiner elliptischer Differentialgleichungen in krummlinigen Koordinatensystemen auf dünnen Gittern", Ph.D. Thesis, TU München (1997)

[*FAB*09] FABER G.: "Über stetige Funktionen", Mathematische Annalen, Vol. 66, pp. 81–94, 1909

[*GRE*87] GRESHO P.M. and SANI R.L.: "On Pressure Boundary Conditions for the Incompressible Navier-Stokes-Equations", International Journal for Numerical Methods in Fluids, also: Lawrence Livermore National Laboratory, Preprint UCRL-96471

[*GRI*91] GRIEBEL M.: "A parallelizable and vectorizable multi-level algorithm on sparse grids in Parallel Algorithms for partial differential equations", Notes on Numerical Fluid Mechanics, Volume 31, pp. 94-100, W. Hackbusch, ed., Vieweg, Braunschweig, 1991

[*GRI*97] GRIEBEL M.: "Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences", Notes on Numerical Fluid Mechanics, Proceedings Large Scale Scientific Computations, 7. June - 11. June, 1997, Varna, Bulgaria, Vieweg-Verlag, Braunschweig, 1998, to appear

[*GDN*98] GRIEBEL M., DORNSEIFER T. and NEUNHOEFFER T.: "Numerical Simulation in Fluid Dynamics — a Practical Introduction", SIAM, Philadalphia, 1998.

[*GZ*98] GRIEBEL M. and ZUMBUSCH G.: "Hash-Storage Techniques for Adaptive Multilevel Solvers and their Domain Decomposition Parallelization", Proceedings of Domain Decomposition Methods 10, eds.: J. Mandel, C. Farhat, X.-C. Cai, Contemporary Mathematics 218, AMS, 1998, pp. 279-286

[*HAC*86] HACKBUSCH W.: "Theorie und Numerik elliptischer Differentialgleichungen", Teubner, Stuttgart 1986.

[*HJ*91] HORN R.A. and Johnson C.R.: "Topics in Matrix Analysis", Cambridge University Press, Cambridge/New York, 1991.

[*PB*96] PARASHAR M. and BROWNE J.C.: "On partitioning dynamic adaptive grid hierarchies", in Proceedings of the 29th Annual Hawaii International Conference on System Sciences 1996.

[*PT*90] PEYRET T. and TAYLOR T.D.: "Computational Methods for Fluid Flow", Springer Series in Computational Physics (corrected third printing), Stuttgart 1990.

[*SCH*98] SCHIEKOFER T.: "Die Methode der Finiten Differenzen auf Dünnen Gittern zur adaptiven Multilevel-Lösung partieller Differentialgleichungen", Dissertation Universität Bonn, Institut für Angewandte Mathematik, to appear 1998.

[*VAN*93] VANDEWALLE S.: "Parallel Multigrid Waveform Relaxation for Parabolic Problems", Teubner, Stuttgart 1993.

[*WS*95] WARREN, M. and SALMON J.: "A portable parallel particle program", Comput. Phys. Comm., 87, pp. 266–290

[*YSE*86] YSERENTANT H.: "On the multi-level splitting of finite element spaces", Numer. Math., 49, pp. 379–412

[*ZEN*91] ZENGER CHR.: "Sparse Grids", in Parallel Algorithms for PDEs; Notes on Numerical Fluid Mechanics Vol. 31, W. Hackbusch, ed., Vieweg, Braunschweig, 1991.