INT. J. NUM. METH. HEAT FLUID FLOW, VOL. 5, 251–269 (1995)

THE EFFICIENT SOLUTION OF FLUID DYNAMICS PROBLEMS BY THE COMBINATION TECHNIQUE

MICHAEL GRIEBEL AND VERONIKA THURNER

Bayerischer Forschungsverbund für Technisch-Wissenschaftliches Hochleistungsrechnen, Institut für Informatik, Technische Universität München, Arcisstr. 21, Postfach 20 24 20, D-80290 München, Germany

ABSTRACT

We study the sparse grid combination technique as an efficient method for the solution of fluid dynamics problems. The combination technique needs only $O(h_n^{-1}(\log(h_n^{-1}))^{d-1})$ grid points for d-dimensional problems, instead of $O(h_n^{-d})$ grid points used by the full grid method. Here, $h_n = 2^{-n}$ denotes the mesh width of the grids. Furthermore, provided that the solution is sufficiently smooth, the accuracy (with respect to the L_2- and the L_∞-norm) of the sparse grid combination solution is $O(h_n^\alpha(\log(h_n^{-1}))^{d-1})$, which is only slightly worse than $O(h_n^\alpha)$ obtained by the full grid solution. Here, α includes the order of the underlying discretization scheme, as well as the influence of singularities. Thus, the combination technique is very economic on both storage requirements and computing time, but achieves almost the same accuracy as the usual full grid solution. Another advantage of the combination technique is that only simple data structures are necessary. Where other sparse grid methods need hierarchical data structures and thus specially designed solvers, the combination method handles merely d-dimensional arrays. Thus, the implementation of the combination technique can be based on any 'black box solver'. However, for reasons of efficiency, an appropriate multigrid solver should be used. Often, fluid dynamics problems have to be solved on rather complex domains. A common approach is to divide the domain into blocks, in order to facilitate the handling of the problem. We show that the combination technique works on such blockstructured grids as well. When dealing with complicated domains, it is often desirable to grade a grid around a singularity. Graded grids are also supported by the combination technique. Finally, we present the first results of numerical experiments for the application of the combination method to CFD problems. There, we consider two-dimensional laminar flow problems with moderate Reynolds numbers, and discuss the advantages of the combination method.

KEY WORDS Sparse grids Combination technique Multi-level Hierarchical basis Block structured grids CFD applications

THE COMBINATION METHOD

A frequent problem in engineering sciences is the analysis of flows by numerical simulation. The simulation of complex real life experiments usually needs a great deal of computing time and returns vast amounts of data. Thus, in order to obtain sufficiently accurate simulation results, it is necessary to find algorithms which economize on both computing time and storage space.

For some time, computer storage became available faster than computational speed increased, a development which proved the need for algorithms of lower computational effort. With the introduction of multigrid techniques, a jump in computational speed occurred. Consequently, research work in numerical fluid mechanics now focuses on storage again. Here, the sparse grid methods provide an approach which is highly economical on storage requirements, yet yields a fairly accurate solution on a fine, sparse grid that is computed from coarse grid solutions.

0961–5539/95/030251–17$2.00
© 1995 Pineridge Press Ltd

Received August 1993
Revised February 1994

Usually, computational fluid dynamics solvers compute solutions on *full* grids, i.e. grids with $O(h^{-d})$ grid points, where h denotes the mesh size of the grid, and d the dimension of the problem (cf. References 4, 10, or 11 for examples of such solvers). An advantage of this approach is that simple d-dimensional arrays can be used as data structures, which facilitates the implementation. However, a disadvantage is that storage requirements and computing time increase polynomially as the mesh size decreases, and soon get out of range.

A different approach involves *sparse* grids[13]. *Figure 1* shows a two-dimensional, regular quadratic full grid with 33 grid points in each direction, and its corresponding sparse grid. Sparse grids only need $O(h^{-1}(\log(h^{-1}))^{d-1})$ grid points. The accuracy deteriorates both pointwise and with respect to the L_2- and L_∞-norm only slightly from $O(h^\alpha)$ to $O(h^\alpha(\log(h^{-1}))^{d-1})$, provided that the solution is sufficiently smooth. Here, α denotes the order of the underlying discretization scheme as well as the influence of singularities. With respect to the energy norm, even the same order as on full grids is obtained[1].

Thus, the efficient usage of sparse grids for the solution of computational fluid dynamics problems greatly reduces storage requirements and computing time, but still yields results which are only slightly worse than the solution obtained on a full grid. For a recent overview on sparse grid methods for computational fluid dynamics problems, see Reference 7.

One way of using sparse grids efficiently involves hierarchical, tree-like data structures and special algorithms for both the discretization and the solution. Since conventional solvers usually do not provide means for dealing with hierarchical data structures, they cannot be employed for solving problems on sparse grids. Thus, new algorithms and new codes have to be developed in order to compute solutions on sparse grids efficiently.

Obviously, an algorithm is needed which combines the advantages of both methods: low storage requirements, a low number of operations involved, but still simple data structures. In the following, we present an algorithm that fulfills these requirements.

Full and sparse grids

As an introductory example, consider the differential equation:

$$Lu = f \tag{1}$$

with the linear operator L of second order in the unit cube $\Omega = [0,1]^d \subset \mathbb{R}^d$, and with appropriate boundary conditions. For reasons of simplicity, the case $d = 2$ will be considered first.

Furthermore, let $\Omega_{i,j}$ be the rectangular grid on the unit square Ω, with mesh width $h_j = 2^{-j}$ in x- and $h_i = 2^{-i}$ in y-direction. Thus, any grid can be represented uniquely by an index pair (i,j), with $i,j \in \mathbb{N}$. *Figure 2* shows an index diagram and its corresponding grids.

The usual approach for solving a partial differential equation is to discretize (1) on an equidistant grid $\Omega_{n,n}$, and to solve the arising linear system of equations:

$$L_{n,n}u_{n,n} = f_{n,n} \tag{2}$$

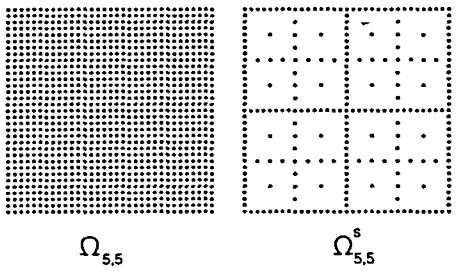

Figure 1 The 33 × 33 full grid and its corresponding sparse grid

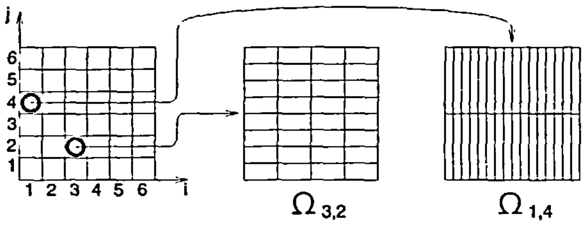

Figure 2 Index diagram and corresponding grids

by an appropriate method. Then, the obtained solution, $u_{n,n}$ has the error:

$$e_{n,n} = u - u_{n,n} = O(h^{\alpha})$$

where α reflects the underlying discretization scheme as well as the influence of singularities. Here, $u_{n,n}$ is assumed to be the interpolant of the discrete solution on grid $\Omega_{n,n}$.

An efficient way of solving the discrete system (2) is the multigrid method. The number of operations needed by a multigrid solver is proportional to the number of grid points, and therefore of order $O(h^{-2})$.

An approach that reduces the number of grid points, and thus the problem size, to order $O(h^{-1} \cdot \log(h^{-1}))$ for the two-dimensional case is the sparse grid method. However, the structure of a sparse grid is more complicated than that of a full grid. Usually, partial differential equation solvers only feature full grid solutions, but do not possess means for dealing with the more economical, but also more complicated sparse grid method. Note that sparse grid methods involving finite elements do exist[1]. However, they use hierarchical, tree-like data structures. Consequently, completely new code has to be programmed. An approach that computes the economic sparse grid solution with an $O(h^{-1} \cdot \log(h^{-1}))$ algorithm but merely uses simple full grid data structures is the *combination technique.*

Calculation of a sparse grid solution using the combination method

A sparse grid solution may be calculated as a *linear combination* of full grid solutions $u_{i,j}$ according to the *combination technique* that was introduced in Reference 5. For the two-dimensional case,

$$u^c_{n,n} = \sum_{i+j=n+1} u_{i,j} - \sum_{i+j=n} u_{i,j}, \qquad \text{with} \qquad i = 1, \dots, n \tag{3}$$

is used as combination formula. As illustrated in the index diagram in *Figure 3*, this linear combination is equivalent to the following algorithm: 'Add up the bilinearly interpolated solutions of all grids $\Omega_{i,j}$ with index pairs (i,j) on the diagram's diagonal, and subtract the bilinearly interpolated solutions of all grids with index pairs on the diagram's subdiagonal'. The full grid

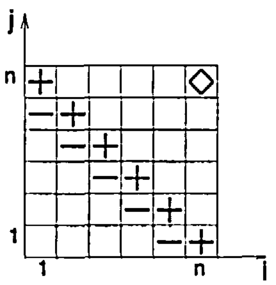

Figure 3 Index diagram of linear combination for the two-dimensional case

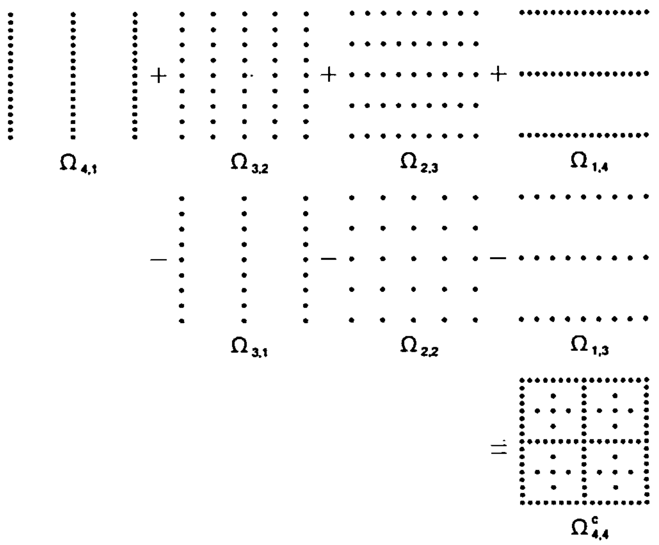

Figure 4 Linear combination of grids $\Omega_{i,j}$ and solutions $u_{i,j}$ according to the combination formula with $n = 4$

solution $u_{n,n}$ that corresponds to the sparse grid solution $u^c_{n,n}$ computed in this diagram is denoted in the upper right corner of the diagram. *Figure 4* illustrates the combination of grids and corresponding solutions.

According to the combination technique, different problems of type $L_{i,j}u_{i,j} = f_{i,j}$ occur, in general on non-equidistant grids. More precisely, n problems with about 2^n unknowns and $n - 1$ problems with about 2^{n-1} unknowns have to be solved. All in all, this results in a total number of $O(h^{-1} \cdot \log(h^{-1}))$ unknowns that must be dealt with in the two-dimensional case.

The accuracy of the sparse grid solution is $O(h^\alpha \cdot \log(h^{-1}))$ (with respect to the L_2- and L_∞-norm), provided that the solution is sufficiently smooth[5]. Thus, the sparse grid solution is only slightly worse than the order $O(h^\alpha)$ that is obtained by the full grid solution. More precisely, for the proof we assume that for each solution $u_{i,j}$ (interpolated from grid $\Omega_{i,j}$ to grid $\Omega_{n,n}$), an error splitting of the form:

$$u_{i,j} = u + h_i^\alpha \omega_1(h_i) + h_j^\alpha \omega_2(h_j) + h_i^\alpha h_j^\alpha \omega_3(h_i, h_j) \tag{4}$$

holds for any fixed point $(x, y) \in \Omega$. ω_1, ω_2 and ω_3 are appropriate functions of x and y depending on the parameters h_i or/and h_j, where $\forall i, j$

$$\begin{aligned} |\omega_1(h_i)| &\leqslant c_1 \\ |\omega_2(h_j)| &\leqslant c_2 \\ |\omega_3(h_i, h_j)| &\leqslant c_3 \end{aligned} \tag{5}$$

holds for some constants c_1, c_2 and c_3 smaller than a constant C (cf. Reference 2). We see that if we insert the error splitting formula (4) into (3), the leading error terms cancel. Thus, we get the estimate:

$$|u - u^c_{n,n}| \leqslant C \cdot h_n^\alpha \cdot (1 + 5/4 \cdot \log(h_n^{-1})) = O(h_n^\alpha \cdot \log(h_n^{-1})) \tag{6}$$

Related techniques have been studied in Reference 1, where it is shown that the energy-norm of the error of the combination technique is of the same order as for the full grid approach[8].

Thus, although the combination technique is an algorithm involving substantially less operations than the regular full grid method, the obtained sparse grid solution is only slightly worse than the corresponding full grid solution.

As mentioned above, the combination technique can be extended to the three-dimensional case. This leads to:

$$u^c_{n,n,n} = \sum_{i+j+k=n+2} u_{i,j,k} - 2 \cdot \sum_{i+j+k=n+1} u_{i,j,k} + \sum_{i+j+k=n} u_{i,j,k}, \qquad \text{with} \qquad i = 1, \ldots, n$$

Another property of the combination technique is that it provides a natural basis for *parallelization*. Each of the full grid problems to be calculated according to the linear combination formula (3) may be computed independently on a different workstation. Communication has to take place only in the end, where the summation and subtraction of the different solutions is performed. Thus, already on a relatively coarse grain level with very low communication and setup requirements, we obtain a simple parallel algorithm[8].

After the problems on grids $\Omega_{i,j}$, $i + j \in \{n, n + 1\}$ were solved, their bilinearly interpolated solutions have to be combined according to (3). This leads to another problem: Interpolating each solution $u_{i,j}$ to grid $\Omega_{n,n}$, and combining the interpolated solutions results in an algorithm of order $O(h^{-2})$ operations for the two-dimensional case. This is the same order an ordinary full grid solver would have required. Therefore, an interpolation algorithm of lower order is needed: a technique using *hierarchical bases* satisfies this requirement.

Hierarchical bases

For reasons of simplicity, the interpolation algorithm will be explained for the one-dimensional case first. Let u_1 and u_2 be discrete values of the solution on the one-dimensional grids Ω_1 and

Ω_2, as illustrated in *Figure 5*. Following the conventional approach, one would first interpolate u_1 at x_1 and x_3, and then add both solutions at the five calculated grid points.

A more economical algorithm is obtained by the following approach: both solutions u_1 and u_2 are first transformed from their standard basis representation into their representation in hierarchical basis. For a one-dimensional grid with $N+1$ grid points, where $N = 2^n$, this transformation is achieved by the following algorithm:

```
integer $l, h, x$
real $u[0 : N]$
for $l = 1$ to $n$ step 1
     $h = 2^{l-1}$
     for $x = h$ to $N - h$ step $2h$
          $u[x] = u[x] - (u[x - h] + u[x + h])/2$
     endfor
endfor
```

For the two-dimensional case, the transformation algorithm works similarly. Let $u_{i,j}$ be the solution on grid $\Omega_{i,j}$, and let $u_{i,j}$ be stored in an array $u[0{:}N,\ 0{:}M]$, with $N = 2^i$ and $M = 2^j$. Then, the representation of $u_{i,j}$ with respect to hierarchical basis is obtained by the following algorithm:

```
integer $lx, ly, hx, hy, x, y$
real $u[0 : N, 0 : M]$
for $ly = 1$ to $j$ step 1
     $hy = 2^{ly-1}$
     for $x = 0$ to $N$ step 1
          for $y = hy$ to $M - hy$ step $2hy$
               $u[x, y] = u[x, y] - (u[x, y - hy] + u[x, y + hy])/2$
          endfor
     endfor
endfor
for $lx = 1$ to $i$ step 1
     $hx = 2^{lx-1}$
     for $y = 0$ to $M$ step 1
          for $x = hx$ to $N - hx$ step $2hx$
               $u[x, y] = u[x, y] - (u[x - hx, y] + u[x + hx, y])/2$
          endfor
     endfor
endfor
```

The extension of this transformation algorithm to higher dimensions is straightforward. For the hierarchization of a d-dimensional function, we start computations on the fine full grid, and step by step turn to coarser full grids until we reach the coarsest grid with merely one grid point in the interior of the domain. At each of these grid-levels, for every grid point that is *not* present in the coarser full grid of the next level, we compute and save the difference between the value of u at this grid point and the d-linear interpolant of the function values at the neighbouring grid points which belong to the next coarse level.

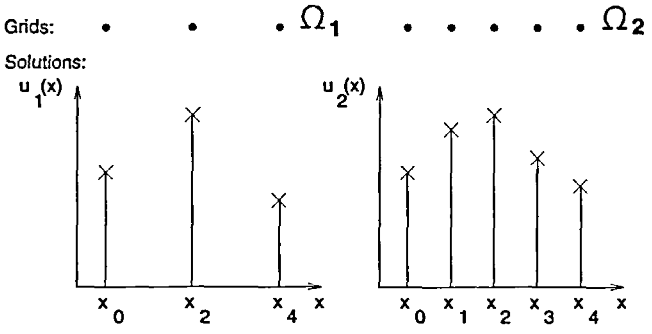

Figure 5 Grids and discrete solutions of a small example problem

The effect of this transformation on u_1 and u_2 is illustrated in *Figure 6*. The bold broken lines indicate the transformed values; thin lines indicate values in standard basis representation. (Note that in *Figure 6*, the thin lines are partly covered by the broken bold lines.) The resulting hierarchical representations of u_1 and u_2 are shown in *Figure 7*.

The advantage of using the hierarchical basis representation during the combination process is that no interpolation to additional grid points has to be performed *explicitly*: For each 'missing' grid point, the interpolated value would be zero in hierarchical basis representation, and thus can be omitted during the combination of solutions. This is illustrated in *Figure 6*. The interpolated values $u_1(x_1)$ and $u_1(x_3)$ are zero in their hierarchical basis representation. Consequently, when accumulating solutions in their hierarchical basis representation according to (3) in order to obtain the combined solution $u^c_{n,n}$, less operations are needed than before: for each solution $u_{i,j}$ that has to be added to $u^c_{n,n}$, no operations are needed in any point that exists in $\Omega^s_{n,n}$, but not in $\Omega_{i,j}$, since the value of $u_{i,j}$ in hierarchical basis representation is zero in those points. Thus, in terms of the hierarchical basis, the interpolation from $\Omega_{i,j}$ to $\Omega^s_{n,n}$ does not involve any additional computational efforts.

All in all, the transformation of the full grid solutions $u_{i,j}$ to hierarchical basis representation, the combination of solutions according to (3) (where 0 values are omitted), and the retransformation into standard basis representation on the *sparse* grid $\Omega^s_{n,n}$ results in an algorithm with an order $O(h^{-1} \cdot \log(h^{-1}))$ of operations. This is of the same order as the number of unknowns involved in the combination technique. Thus, the overall complexity of the combination algorithm is also of order $O(h^{-1} \cdot \log(h^{-1}))$.

Altogether, the combination of the different solutions can be performed efficiently according to the following algorithm (here for the two-dimensional case):

- Set the values of the combined solution $u^c_{n,n}$ on grid $\Omega^s_{n,n}$ to zero: $u^c_{n,n} = 0$.
- For all (i,j) with $i + j \in \{n, n+1\}$ do:
  - transform the solution $u_{i,j}$ obtained on grid $\Omega_{i,j}$ into its hierarchical representation. This needs $O(2^{i+j})$ operations.

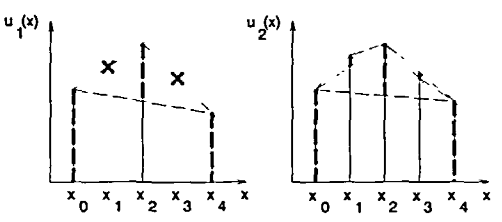

Figure 6 Transformation of u_1 and u_2 into hierarchical basis representation

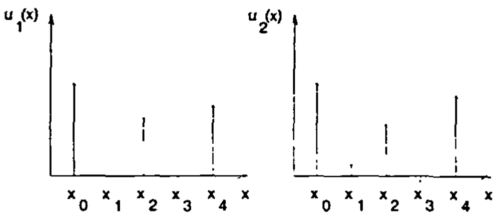

Figure 7 u_1 and u_2 in hierarchical basis representation

— add (if $i+j=n+1$) or subtract (if $i+j=n$) the solution $u_{i,j}$ on grid $\Omega_{i,j}$ in hierarchical representation to the hierarchical representation of the combined solution $u^c_{i,j}$ on grid $\Omega^s_{n,n}$. This can be done with $O(2^{i+j})$ operations as well.

Thus, we computed the combined solution $u^c_{n,n}$ on the sparse grid $\Omega^s_{n,n}$ in hierarchical basis representation, with $O(h^{-1}\cdot\log(h^{-1}))$ operations all in all.

Note that the sparse grid solution is stored on a *sparse* grid in hierarchical basis representation. The backward transformation of $u^c_{n,n}$ into standard basis representation together with the interpolation to the *full* grid $\Omega_{n,n}$ would be of order $O(h^{-2})$. Furthermore, the storage requirements of the retransformed solution on the full grid would be of order $O(h^{-2})$ as well.

Depending on the kind of application for which the combination technique is used, it has to be decided which type of representation of the solution is required. For example, if the combination technique is to be inserted into an existing code as a black box solver, and other parts of the existing program are to be applied to the combined solution (e.g. graphics), then the original data structure has to be re-established (which usually will be a two-dimensional array). This, however, leads to order $O(h^{-2})$ of both operations and storage requirements.

Nevertheless, even in this case it may be profitable to employ the combination technique, in order to exploit its advantages mentioned above with respect to computational cost and parallelization properties. Again, the combination technique can be used together with *any* suitable existing solver, so that computational and parallelization advantages can be made available no matter which code is used. Experience on the usage of the combination technique with several different CFD-solvers is described elsewhere[4,7,8].

If the two-dimensional array is *not* required as data structure of the result, i.e. if we settle for the representation in hierarchical basis only, the combined solution can be stored in a hierarchical, tree-like data structure. In this case, both storage requirements and number of operations are merely of order $O(h^{-1}\cdot\log(h^{-1}))$. However, if graphics or post-processing of any other kind is desired, those programs have to be able to handle hierarchical data structures.

A third method which is also of order $O(h^{-1}\cdot\log(h^{-1}))$ uses 'distributed' grids: the combination of the $u_{i,j}$ according to (3) is not carried out. Instead, the 'operands' of the combination formula (3), i.e. the small full grid solutions $u_{i,j}$, are stored. When a linear operator F is to be applied to the combined solution, it is applied to the distributed solutions $u_{i,j}$ instead:

$$F(u^c_{n,n}) = \sum_{i+j=n+1} F(u_{i,j}) - \sum_{i+j=n} F(u_{i,j}), \qquad \text{with} \qquad i=1,\ldots,n \tag{7}$$

Thus, an *implicit* working with the sparse grid solution is possible[9].

Combination method for block-structured grids

In many engineering problems, grids occur on rather complicated domains. In order to facilitate the handling of such problems, the domain is decomposed into several smaller, simpler domains, usually called 'blocks'. This approach is very advantageous.

Firstly, the block structure of a grid reduces main memory requirements. In an inner iteration step, the problem is solved one block at a time. An outer iteration establishes the overall solution. Secondly, the block structure of a grid is a natural basis for the parallelization of the solver. Each processor solves the problem on one of the blocks, and communication is necessary merely along block interfaces, in order to achieve a smooth solution on the whole domain.

The combination technique works with block-structured grids, too. However, care has to be taken that for each index pair (i,j), the respective block grids are orientated such that grid lines are continuous across block interfaces. This turns out to be a problem for domains with holes. In this case, it is possible that block orientation changes within the domain in such a way that grid lines are no longer continuous across block interfaces. A simple example with $\Omega_{1,2}$ grids for each block is shown in *Figure 8*. Therefore, in the following studies we restrict ourselves to domains without holes.

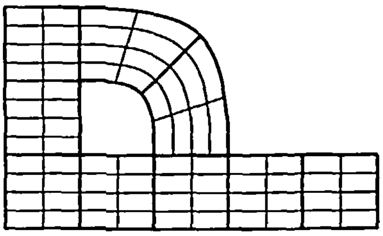
Figure 8 Domain with a hole: grid lines are discontinuous at block interfaces

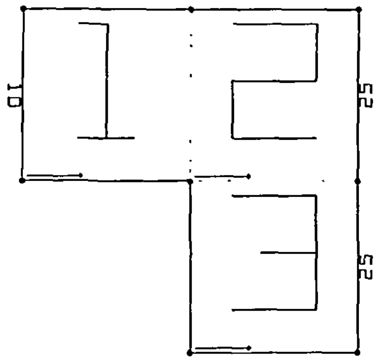
Figure 9 *L*-shaped area, 3-block domain

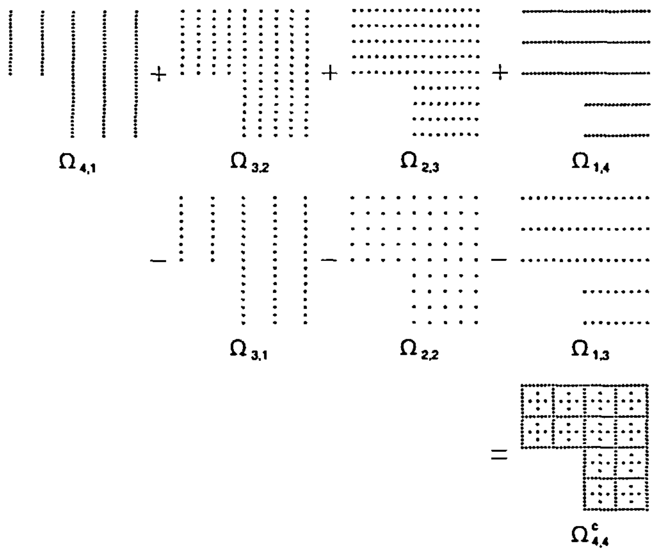

Figure 10 Combination of grids of the *L*-shaped domain

A simple example of a multi-block domain is the L-shaped area in *Figure 9* which consists of 3 blocks.

The extension of the combination technique to multi-block domains is straightforward. Again, (3) is used as combination formula, where the index pairs indicate the number of grid points *per block*. For the three-block L-shaped area, the combination technique with $n = 4$ is illustrated in *Figure 10.*

Combination method for graded grids

In many application problems, domains with singularities occur, e.g. caused by re-entrant corners. In order to obtain a sufficiently good approximation of the solution around the singularity, mesh widths have to be held rather small. Thus, when using a regular grid, many grid points are needed. However, most of these grid points cover an area of the domain where the solution is rather smooth, and a lot less points would suffice to yield a fairly good result.

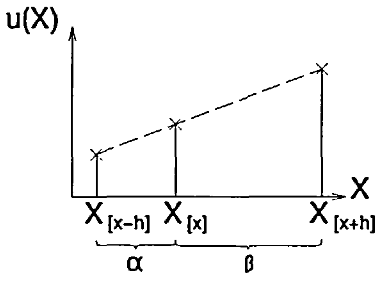

Figure 11 Transformation for graded grid

Therefore it is desirable to allow for the grading of a grid: around a singularity, mesh widths are kept small, whereas in those parts of the domain where the solution is expected to be rather smooth, mesh widths are chosen to be larger.

The combination technique works well with graded grids. However, a slight modification has to be made in the algorithms which transform a solution from standard basis into hierarchical basis representation. So far, the hierarchical basis functions were symmetrical. On a graded grid, however, the different scaling of the grid in x- and y-direction has to be considered for the interpolation (cf. *Figure 11*).

This leads to slight changes in the transformation algorithms. For the one-dimensional case, the interior part $u[x] = u[x] - (u[x-h] + u[x+h])/2$ of the loop has to be replaced by:

$$\alpha = X[x] - X[x-h]$$
$$\beta = X[x+h] - X[x]$$
$$u[x] = u[x] - \frac{\alpha \cdot u[x+h] + \beta \cdot u[x-h]}{\alpha + \beta}$$

Here, X denotes the X-coordinates of the grid points, whereas x is the *index* in the x-direction of the current grid point. Note that the hierarchical basis functions are not symmetrical in this case. Of course, this does not complicate their handling: as before, the support of a basis function ranges from $X[x-h]$ to $X[x+h]$. Note that here h denotes not the mesh width, but the difference in the index corresponding to the current basis function. However, α and β now differ in their size. For the two-dimensional case, the necessary changes in the algorithm are analogous. The combination of the solutions $\tilde{u}_{i,j}$ on the graded grids $\tilde{\Omega}_{i,j}$ according to (3) for a one block domain is shown in *Figure 12*.

COMBINATION TECHNIQUE AND THE NAVIER–STOKES EQUATIONS

Now we want to apply the combination technique for solving the Navier–Stokes equations.

So far, we have merely studied the combination technique for the linear elliptic differential equation $Lu = f$. For this problem, the error of the combination technique is of order $O(h^{\alpha} \cdot \log(h^{-1}))$, where α depends on the order of the underlying discretization scheme as well as on the strength of the singularities, if present[5].

Now, we study the Navier–Stokes equations:

$$\begin{aligned} -\Delta u + Re(uu_x + vu_y) + p_x &= 0 \\ -\Delta v + Re(uv_x + vv_y) + p_y &= 0 \\ u_x + v_y &= 0 \end{aligned} \qquad (8)$$

on different domains.

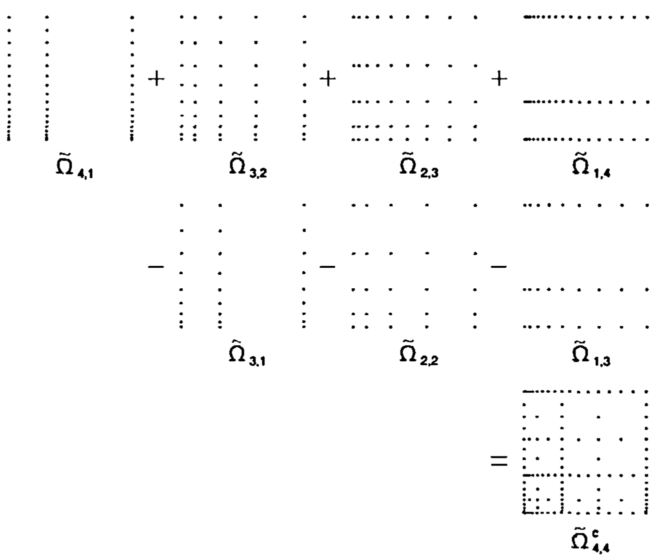

Figure 12 Linear combination of graded grids $\tilde{\Omega}_{i,j}$ and solutions $\tilde{u}_{i,j}$ according to the combination formula with $n = 4$

Even for simple linear model problems, it is fairly difficult to show the existence of an error splitting such as (4). For the Navier–Stokes equations, in general not even an analytical solution is known, and only numerical approximations can be gained. Therefore, it is usually not possible to prove the existence of an error splitting like (4) theoretically. However, even for those cases numerical experiments using the combination technique produced satisfactory results. This indicates that some cancellation of certain error-terms analogous to (4) and (6) still takes place.

A program package that can be used to numerically solve Navier–Stokes flow problems is the *L_iSS* package, which was developed at the GMD (Gesellschaft für Mathematik und Datenverarbeitung mbH). We use it in the following to solve the different discretized Navier–Stokes problems arising in the combination technique. Other packages that are equally suitable for the efficient solution of flow problems since they involve multigrid-type solvers, are e.g. COMET/FASTEST, which were developed at the Department of Fluid Dynamics of the University of Erlangen[4].

The L_iSS program package was designed for simulating two-dimensional, incompressible, laminar flows in general domains. Note that in addition to (8), the L_iSS package can also handle time-dependent Navier–Stokes equations and the Stokes equations (cf. Reference 11, pp. 82–83 for further details). However, in this paper, we restrict outselves to (8) on general domains with appropriate boundary conditions.

As discretization, flux-difference splitting is used[3]. In general, this only leads to an accuracy of order $O(h^{\alpha})$, $1 \leqslant \alpha \leqslant 2$. Note that for higher Reynolds numbers, $\alpha \approx 1$ holds. In addition, L_iSS supports three other discretizations (cf. Reference 11, pp. 204–206).

For a given grid, the L_iSS Navier–Stokes solver calculates the velocity u in horizontal direction, the velocity v in vertical direction, and the pressure p in each grid point. Note that the L_iSS package uses non-staggered grids, and is therefore well suited to the combination technique.

The L_iSS package incorporates a multigrid solver for Navier–Stokes problems. The user can choose between *F*-, *V*- or *W*-type cycles, with *W*-cycles being the default. However, only standard coarsening is possible. The grid levels to be used in the cycles may be specified by the user.

Many flow problems that are studied using a numerical solver are applied to domains that are generally rather complex. The L_iSS package supports the partitioning of a domain into blocks, each of which has to be *topologically* rectangular. As it was mentioned before, this feature not only simplifies the handling of complex geometries; it also provides a natural basis for the parallelization of solvers of partial differential equations.

Boundary fitted grids are generated by means of transformed Poisson equations (cf. Reference 11, pp. 62–70 for further details). If the domain consists of several adjacent blocks, each grid block is computed separately. Nevertheless, grid lines are generated continuous across block interfaces.

In the following studies, full multigrid cycles were used, with smoothing iterations on all available levels. For the successive, differently sized *V*-cycles of which the full multigrid *F*-cycle consists, the maximum number of multigrid iterations to be carried out may be explicitly set by the user. However, if the residual reduction is less than a certain (user defined) threshold *eps*, iteration is finished earlier. In our experiments, the following values were used for the different *V*-cycles which make up the *F*-cycle:

- for the 'smallest' *V*-cycle (starting on a rather coarse grid): 20 iterations, or $eps = 10^{-8}$;
- for the intermediate *V*-cycles: 20 iterations, or $eps = 10^{-5}$;
- for the 'largest' *V*-cycle (starting on the finest grid level): 20 iterations, or $eps = 10^{-12}$.

NUMERICAL EXPERIMENTS

To demonstrate the quality of the combination technique, several two-dimensional flow problems were studied. Since the exact solution of most of the following model problems is not known in general, an accurate approximation obtained by higher order τ-extrapolation computed on the finest grid our computers could handle is used as a referential solution u_{ref}.

Thus, the error of the full grid solution is defined as:

$$e_{n,n} = u_{ref}^{I_{n,n}^{f}} - u_{n,n}$$

where $u_{ref}^{I_{n,n}^{f}}$ denotes the interpolation of U_{ref} to grid $\Omega_{n,n}$, which is much coarser than Ω_{ref}. Analogously, the error of the solution obtained by the combination technique is defined as:

$$e_{n,n}^{c} = u_{ref}^{I_{n,n}^{s}} - u_{n,n}^{c}$$

where $uI_{n,n}^{s}$/ref denotes the solution u_{ref} interpolated to the sparse grid $\Omega_{n,n}^{s}$.

For the analysis of the development of the errors, the discrete L_2-norm is defined as:

$$\|e_{n,n}\|_2 := \sqrt{\frac{1}{|\Omega_{n,n}^{i}|}\left(\sum_{\forall(x,y)\in\Omega_{n,n}^{i}} (e_{n,n}(x,y))^2\right)}$$

where $\Omega_{n,n}^{i}$ denotes the interior of the grid $\Omega_{n,n}$. Correspondingly, the discrete L_∞-norm is defined as:

$$\|e_{n,n}\|_\infty := \max_{\forall(x,y)\in\Omega_{n,n}^{i}} (|e_{n,n}(x,y)|)$$

Furthermore, we consider the development of the error in certain special grid points.

Flow through a channel

The first model problem is the flow through a channel. Calculations were made with a one-block domain and an equidistant grid, as shown in *Figure 13*. The boundary conditions consist of a parabolic inflow at the left side of the domain, and no-slip wall conditions at top and bottom sides. For the outflow at the right side of the domain, outflow conditions for u and v are derived

Table 1 Channel, Reynolds number 500: development of the L_2- and L_∞-norms of the error in u

	error on full grid				error of combined solution			
n	$\|e_{n,n}\|_\infty$	$\frac{\|e_{n-1,n-1}\|_\infty}{\|e_{n,n}\|_\infty}$	$\|e_{n,n}\|_2$	$\frac{\|e_{n-1,n-1}\|_2}{\|e_{n,n}\|_2}$	$\|e^c_{n,n}\|_\infty$	$\frac{\|e^c_{n-1,n-1}\|_\infty}{\|e^c_{n,n}\|_\infty}$	$\|e^c_{n,n}\|_2$	$\frac{\|e^c_{n-1,n-1}\|_2}{\|e^c_{n,n}\|_2}$
2	0.0008587	*	0.0007785	*	0.0010419	*	0.0008690	*
3	0.0002877	2.9845776	0.0001832	4.2504045	0.0010508	0.9915220	0.0004785	1.8161912
4	0.0001157	2.4877094	0.0000424	4.3245916	0.0009682	1.0853564	0.0005458	0.8765887
5	0.0000494	2.3413637	0.0000097	4.3704026	0.0005820	1.6634895	0.0004205	1.2981862
6	0.0000209	2.3689157	0.0000023	4.2384821	0.0003703	1.5717994	0.0002769	1.5186081
7	0.0000084	2.4733055	0.0000005	4.2030537	0.0002602	1.4228486	0.0001699	1.6293010

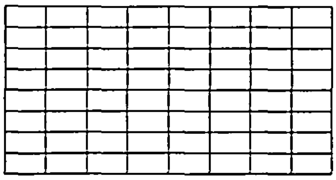
Figure 13 Channel: grid $\Omega_{3,3}$ of one-block domain

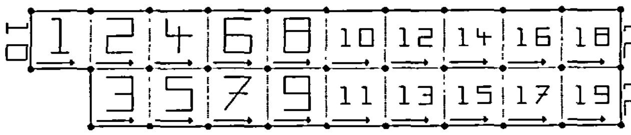

Figure 14 Backward facing step: block structure

from flux-difference splitting. Pressure p was set to 0 at the outflow region. In our experiment, a Reynolds number of 500 was chosen. *Table 1* shows the development of the L_2- and L_∞-norms of the error for the velocity u in x-direction for both the full grid solution and the combination technique. Since the exact solution is known for this simple model problem, we used it here as the referential solution.

We see that for the L_∞-norm, the error quotient of the full grid solution approaches the value 2. From this result we infer that the accuracy obtained by the full grid method is $O(h)$. For the combination method, however, the achieved accuracy seems to be of order $O(h \cdot \log(h^{-1}))$.

Furthermore, for the L_2-norm, we observe an $O(h^2)$-accuracy for the velocity u in x-direction for the full grid solution. This might be due to the symmetry and simplicity of the solution: in this very special case with a constant solution u, flux-difference splitting discretization on a regular grid results in a cancellation of the leading $O(h)$ error term in most of the grid points because of the constant flow profile and the equidistant grid, and produces an $O(h^2)$ accuracy. A more detailed analysis of the respective accuracies will be given for the next model problem. Since the combination method involves the solution of problems on grids with in general different mesh size in the x- and y-direction, the flux-difference splitting discretization results only in first order accurate solutions.

Domain with a backward facing step

Now we turn to a more interesting model problem: we consider the flow over a backward facing step that involves the L-shaped domain shown in *Figure 14*. The domain is substructured in 19 quadratic blocks. Furthermore, in order to cope with the singularity situated at the reentrant corner, we use a graded grid where the mesh width is refined towards the singularity, as well as towards the bounding walls. As an example for a graded grid, the block-structured grid $\Omega_{3,3}$ is shown in *Figure 15*.

As boundary conditions for our numerical experiments, we chose parabolic inflow at the leftmost side of the domain, parabolic outflow at the rightmost side of the domain, and no-slip conditions at the remaining walls. Moreover, we first selected a moderate value of 50 for the Reynolds number.

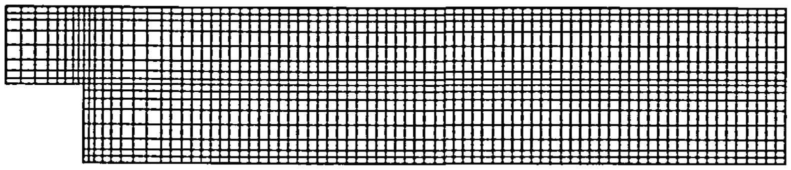

Figure 15 Backward facing step: graded full grid $\Omega_{3,3}$

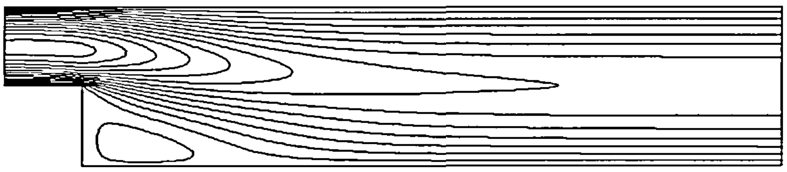

Figure 16 Backward facing step, graded grid, Reynolds number 50: contour lines of velocity u of combined solution on grid $\Omega_{5,5}$

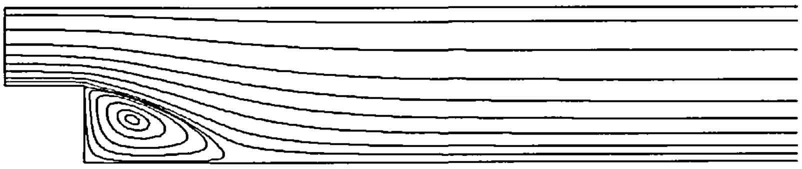

Figure 17 Backward facing step, graded grid, Reynolds number 50: contour lines of stream function Ψ of combined solution on grid $\Omega_{5,5}$

Table 2 Backward facing step, graded grid, Reynolds number 50: development of the L_2- and L_∞-norms of the error in u

	error on full grid				error of combined solution			
n	$\|e_{n,n}\|_\infty$	$\frac{\|e_{n-1,n-1}\|_\infty}{\|e_{n,n}\|_\infty}$	$\|e_{n,n}\|_2$	$\frac{\|e_{n-1,n-1}\|_2}{\|e_{n,n}\|_2}$	$\|e^c_{n,n}\|_\infty$	$\frac{\|e^c_{n-1,n-1}\|_\infty}{\|e^c_{n,n}\|_\infty}$	$\|e^c_{n,n}\|_2$	$\frac{\|e^c_{n-1,n-1}\|_2}{\|e^c_{n,n}\|_2}$
2	0.1482879	*	0.0591591	*	0.1574975	*	0.0642801	*
3	0.0760391	1.9501529	0.0282950	2.0907969	0.1096267	1.4366712	0.0372796	1.7242695
4	0.0414564	1.8341958	0.0145896	1.9393918	0.0803236	1.3648128	0.0246288	1.5136564
5	0.0194566	2.1307106	0.0068300	2.1360960	0.0509586	1.5762525	0.0161749	1.5226617
6	0.0070550	2.7578352	0.0024826	2.7511673	0.0315403	1.6156674	0.0101769	1.5893720

Figures 16 and *17* illustrate the solution we obtained with the combination method. They show contour lines of the velocity u in x-direction, and the stream function Ψ respectively. The computed full grid solutions practically look identical.

Computations on regular, non graded grids showed basically the same behaviour of the solutions in large parts of the domain. However, near the re-entrant corner, the errors were larger due to the pollution effect of the singularity situated there. Therefore, the use of a properly graded grid is recommended. Nevertheless the choice of a correctly graded grid depends not only on the shape of the domain, but also on the Reynolds number. Thus, the proper choice of the grading generally is a crucial task.

For the velocity u in x-direction, *Table 2* shows the development of the L_2- and L_∞-norms of the error of both the full grid solution and the combined solution. The error of the velocity v

in y-direction behaves principally the same. The results for the pressure p are similar as well. However, since the pressure is only determined up to a constant, the pressure-values obtained on different grids have to be properly modified before the combination can take place. This can be achieved by subtracting the value of the pressure at e.g. the upper right corner of the domain from the pressure-values in every point for each specific grid.

From *Table 2*, one may assume that the accuracy obtained on the full grid is of order $O(h)$, whereas the accuracy of the combination method is of order $O(h \cdot \log(h^{-1}))$. This can be shown in more detail by dividing both numerator and denominator of the error quotient by the corresponding logarithmic terms. Thus, we now compute:

$$\frac{\|e^c_{n-1,n-1}\|_\infty / \log(2^{n-1})}{\|e^c_{n,n}\|_\infty / \log(2^n)} \quad \text{and} \quad \frac{\|e^c_{n-1,n-1}\|_2 / \log(2^{n-1})}{\|e^c_{n,n}\|_2 / \log(2^n)}.$$

The resulting values are shown in *Table 3*.

We see that for both norms, the quotient approaches the value 2. Therefore, for this problem the accuracy obtained by the combination technique is of the order $O(h \cdot \log(h^{-1}))$. In addition, *Table 4* shows the error in the velocity u at the centrepoints of block 3 and 5, respectively.

A comparison of the size of the error reveals that both methods achieve *equal accuracy* for

- 225 inner grid points per block for the full grid solution, and
- 129 inner grid points per block for the combined solution.

This demonstrates the efficiency of the combination technique. Thus, the combination technique involves less unknowns and consequently less operations.

Now, we turn to larger Reynolds numbers and choose the value 500. For the velocity u in x-direction, *Table 5* shows the development of the L_2- and L_∞-norms of the error of both the full grid solution and the combined solution. The error of the velocity v in y-direction shows the same behaviour.

Once again, we observe an accuracy of roughly $O(h)$ for the full grid case. Thus, the accuracy of the full grid solution is not affected by the choice of a higher Reynolds number.

Table 3 Backward facing step, graded grid, Reynolds number 50: accuracy of combined solution

n	$\frac{\|e^c_{n-1,n-1}\|_\infty/\log(2^{n-1})}{\|e^c_{n,n}\|_\infty/\log(2^n)}$	$\frac{\|e^c_{n-1,n-1}\|_2/\log(2^{n-1})}{\|e^c_{n,n}\|_2/\log(2^n)}$
3	2.1550068	2.5864052
4	1.8197504	2.0182117
5	1.9703156	1.9033194
6	1.9388009	1.9072488

Table 4 Backward facing step, graded grid, Reynolds number 50: error development of u at the centrepoints of blocks 3 and 5

	error on full grid				error of combined solution			
n	$\|e3_{n,n}\|$	$\frac{\|e3_{n-1,n-1}\|}{\|e3_{n,n}\|}$	$\|e5_{n,n}\|$	$\frac{\|e5_{n-1,n-1}\|}{\|e5_{n,n}\|}$	$\|e3^c_{n,n}\|$	$\frac{\|e3^c_{n-1,n-1}\|}{\|e3^c_{n,n}\|}$	$\|e5^c_{n,n}\|$	$\frac{\|e5^c_{n-1,n-1}\|}{\|e5^c_{n,n}\|}$
2	0.0301999	*	0.0172789	*	0.0299728	*	0.0026696	*
3	0.0166723	1.8113821	0.0268919	0.6425334	0.0156895	1.9092863	0.0113660	0.2348789
4	0.0077501	2.1512368	0.0213154	1.2616164	0.0092650	1.6943816	0.0140938	0.8064542
5	0.0031529	2.4580706	0.0120622	1.7671252	0.0039801	2.3278053	0.0134984	1.0441121
6	0.0010405	3.0303067	0.0047621	2.5329464	0.0000433	91.9236174	0.0104715	1.2890626

However, the accuracy of the solution obtained by the combination method is somewhat more sensitive to the higher Reynolds number. This is due to the extremely distorted grids $\Omega_{1,n}$, $\Omega_{2,n-1}$, $\Omega_{n-1,2}$ and $\Omega_{n,1}$ that are involved in the combination process. The solution that is produced on these grids by the flux-difference discretization is not well suited to result in a sufficient cancellation of leading error terms in the combination process. Consequently, the accuracy drops. Nevertheless, for sufficiently large values of n, we expect the combination method to regain its $O(h \cdot \log h^{-1})$-behaviour.

Channel with a constriction

Now we consider the flow through a channel with a constriction. The shape of the domain and the block structure we chose are illustrated in *Figure 18*. Since we want to study the accuracy of the combination technique itself and not the additional problems posed on the combination method by the influence of curved boundaries, we restrict ourselves to polygonal boundaries for the resolution of the shape at the constriction.

The domain is substructured into 28 quadrilateral blocks. We chose the graded grid in *Figure 19* where the mesh width decreases towards the bounding walls. As boundary conditions for our numerical experiments, we chose parabolic inflow at the leftmost side of the domain, and

Table 5 Backward facing step, graded grid, Reynolds number 500: development of the L_2- and L_∞-norms of the error in u

	error on full grid				error of combined solution			
n	$\|e_{n,n}\|_\infty$	$\frac{\|e_{n-1,n-1}\|_\infty}{\|e_{n,n}\|_\infty}$	$\|e_{n,n}\|_2$	$\frac{\|e_{n-1,n-1}\|_2}{\|e_{n,n}\|_2}$	$\|e^c_{n,n}\|_\infty$	$\frac{\|e^c_{n-1,n-1}\|_\infty}{\|e^c_{n,n}\|_\infty}$	$\|e^c_{n,n}\|_2$	$\frac{\|e^c_{n-1,n-1}\|_2}{\|e^c_{n,n}\|_2}$
2	0.1482879	*	0.0591591	*	0.4816670	*	0.1198857	*
3	0.0760391	1.9501529	0.0282950	2.0907969	0.4090551	1.1775111	0.0924246	1.2971189
4	0.0414564	1.8341958	0.0145896	1.9393918	0.3653634	1.1195843	0.0778842	1.1866919
5	0.0194566	2.1307106	0.0068300	2.1360960	0.3291922	1.1098786	0.0654446	1.1900788
6	0.0070550	2.7578352	0.0024826	2.7511673	0.2929877	1.1235701	0.0541855	1.2077881

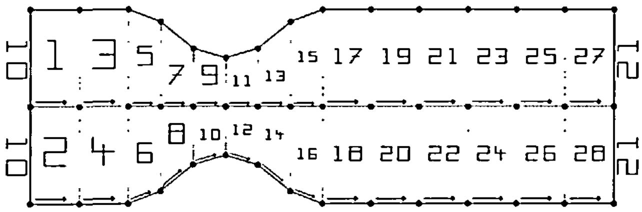

Figure 18 Channel with a constriction: domain and block-structure

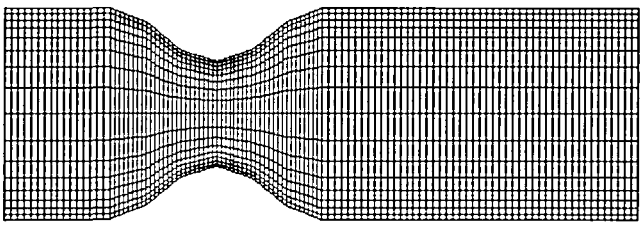
Figure 19 Channel with a constriction: graded grid $\Omega_{3,3}$

no-slip conditions at its top and bottom. The outflow conditions for the rightmost side of the domain are derived from the flux-difference discretization. Again, a moderate value of 50 was chosen for the Reynolds number.

Figures 20 and *21* illustrate the solution we obtained with the combination method. They show contour lines of the velocity u in x-direction, and the stream function Ψ respectively. The full grid solution behaves just like the combined solution.

For the velocity u in x-direction, *Table 6* shows the development of the L_2- and L_∞-norms of the error of both the full grid solution and the combined solution. The error of the velocity v in y-direction behaves more or less in the same way.

Once again it can be seen that the accuracy of the combination method is of the order $O(h \cdot \log(h^{-1}))$. This is more clearly shown in *Table 7*, where we present the quotients of the errors divided by the respective logarithmic terms. The resulting values are close to 2.

Furthermore, *Table 8* shows the error in the velocity u at the centrepoints of block 12 and 14, respectively.

Once more, a comparison of the size of the errors reveals that both methods achieve equal

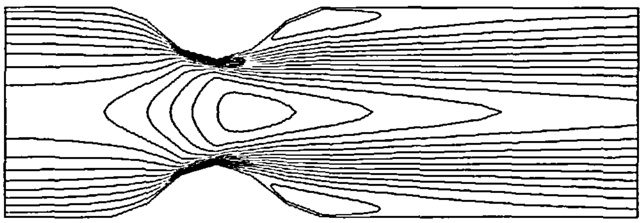
Figure 20 Channel with constriction, graded grid, Reynolds number 50: contour lines of velocity u of combined solution on grid $\Omega_{5,5}$

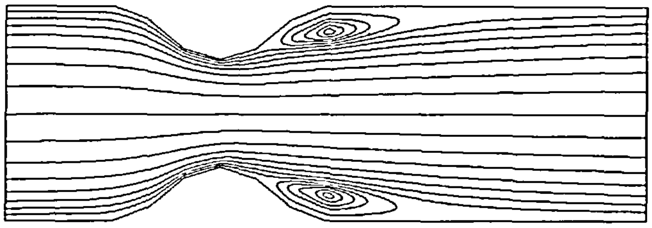
Figure 21 Channel with constriction, graded grid, Reynolds number 50: contour lines of stream function Ψ of combined solution on grid $\Omega_{5,5}$

Table 6 Channel with a constriction, graded grid, Reynolds number 50: development of the L_2- and L_∞-norms of the error in u

	error on full grid				error of combined solution			
n	$\|e_{n,n}\|_\infty$	$\frac{\|e_{n-1,n-1}\|_\infty}{\|e_{n,n}\|_\infty}$	$\|e_{n,n}\|_2$	$\frac{\|e_{n-1,n-1}\|_2}{\|e_{n,n}\|_2}$	$\|e^c_{n,n}\|_\infty$	$\frac{\|e^c_{n-1,n-1}\|_\infty}{\|e^c_{n,n}\|_\infty}$	$\|e^c_{n,n}\|_2$	$\frac{\|e^c_{n-1,n-1}\|_2}{\|e^c_{n,n}\|_2}$
2	0.3262069	*	0.1217483	*	0.3559262	*	0.1285810	*
3	0.2186938	1.4916148	0.0765705	1.5900159	0.2424631	1.4679600	0.0874394	1.4705155
4	0.1257605	1.7389699	0.0426588	1.7949518	0.1692680	1.4324217	0.0563446	1.5518691
5	0.0628579	2.0007107	0.0205897	2.0718556	0.1131389	1.4961076	0.0348364	1.6174026
6	0.0238464	2.6359538	0.0076063	2.7069327	0.0698580	1.6195564	0.0202508	1.7202486

Table 7 Channel with a constriction, graded grid, Reynolds number 50: accuracy of combination technique

n	$\dfrac{\Vert e^c_{n-1,n-1}\Vert_\infty/\log(2^{n-1})}{\Vert e^c_{n,n}\Vert_\infty/\log(2^n)}$	$\dfrac{\Vert e^c_{n-1,n-1}\Vert_2/\log(2^{n-1})}{\Vert e^c_{n,n}\Vert_2/\log(2^n)}$
3	2.2019404	2.2057733
4	1.9098956	2.0691588
5	1.8701349	2.0217532
6	1.9434665	2.0642983

Table 8 Channel with a constriction, graded grid, Reynolds number 50: error development of u at the centrepoints of blocks 12 and 14

	error on full grid				error of combined solution			
n	$\Vert e12_{n,n}\Vert$	$\dfrac{\vert e12_{n-1,n-1}\vert}{\vert e12_{n,n}\vert}$	$\vert e14_{n,n}\vert$	$\dfrac{\vert e14_{n-1,n-1}\vert}{\vert e14_{n,n}\vert}$	$\vert e12^c_{n,n}\vert$	$\dfrac{\vert e12^c_{n-1,n-1}\vert}{\vert e12^c_{n,n}\vert}$	$\vert e14^c_{n,n}\vert$	$\dfrac{\vert e14^c_{n-1,n-1}\vert}{\vert e14^c_{n,n}\vert}$
2	0.0355327	*	0.2783096	*	0.0365755	*	0.2860317	*
3	0.0133459	2.6624335	0.1630949	1.7064276	0.0003691	99.1045624	0.1892374	1.5114965
4	0.0026419	5.0516531	0.0882897	1.8472697	0.0182089	0.0202681	0.1191162	1.5886790
5	0.0003289	8.0337330	0.0423866	2.0829645	0.0223609	0.8143171	0.0709612	1.6786101
6	0.0004789	0.6867427	0.0156717	2.7046603	0.0200569	1.1148740	0.0391186	1.8140029

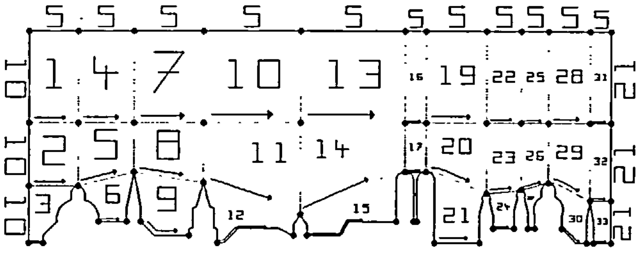

Figure 22 Skyline of Munich: domain and block-structure

accuracy for

- 225 inner grid points per block for the full grid solution, and
- 129 inner grid points per block for the combined solution.

Again, this demonstrates the efficiency of the combination technique.

Skyline of Munich

Finally, we present the result of calculations involving a problem with a rather complicated domain and block structure. We consider the flow over the skyline of Munich as shown in *Figure 22.* Of course, there is no practical relevance for this problem. Instead of a 3-dimensional model of Munich, merely the skyline is used, which is a 2-dimensional projection. Furthermore, no turbulence model was applied. In addition, only a Reynolds number of 500 was chosen, which is not sufficient at all to model the flow of air realistically.

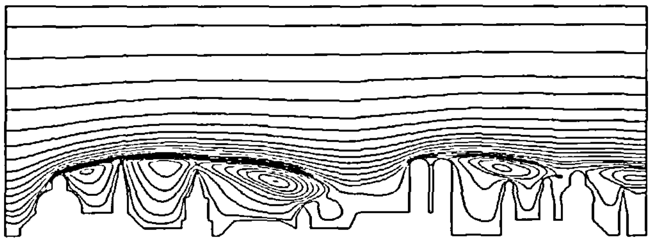

Figure 23 Skyline of Munich, Reynolds number 500: contour lines of stream function Ψ of combined solution on grid $\Omega_{5,5}$

Thus, this model problem just serves as an example to demonstrate that with the aid of block structuring, quite complicated domains can be handled by the combination technique as well.

As boundary conditions for our numerical experiments, we chose a half-parabolic inflow with its vertex at the domain's upper left for the leftmost side of the domain, a constant velocity in x-direction for the topmost side of the domain, and no-slip conditions at its bottom side. The outflow conditions for the rightmost side of the domain are derived from flux-difference equations. *Figure 23* illustrates the solution obtained with the combination method for $n = 5$.

CONCLUDING REMARKS

In this paper, we studied the combination technique for the solution of flow problems. So far, we have seen that with moderate Reynolds numbers, the quality of the solutions obtained by the combination method is of the order $O(h \cdot \log(h^{-1}))$, whereas the conventional full grid technique produces solutions with an $O(h)$-accuracy. Since the combination method involves substantially less unknowns than the full grid technique, it is far more efficient.

Furthermore, the combination method also works well for graded grids, which are necessary to resolve singularities and regions with highly varying solution gradients. In addition, we demonstrated that the combination technique can be applied successfully together with block structuring and domain decomposition techniques.

Finally, we shortly want to mention further advantages of the combination method. Firstly, it leads to a natural basis for parallelization. Secondly, it can be added rather easily to any existing solver.

However, for a real practical application of the combination method within a CFD solver, certain further problems have to be studied. So far, we considered only flow problems with rather low Reynolds numbers. For less moderate Reynolds numbers, the combination method is somewhat sensitive to the use of extremely distorted grids, such as $\Omega_{1,n}$, $\Omega_{n,1}$. In this case, the corresponding solutions should not be taken into account. This results in a smaller range of solutions involved in the combination process and results in a slight modification of the combination formula (3):

$$u_{n,n} = \sum_{i+j=n+1} u_{i,j} - \sum_{i+j=n} u_{i,j}, \qquad \text{with} \qquad i = n_0, \ldots, n - n_0 + 1, \qquad 1 \leqslant n_0 \leqslant n \tag{9}$$

So far, the relation between n_0 and the accuracy of the solution is still a not cleared up question, and thus subject of further research. Furthermore, future work has to be done studying problems with higher Reynolds numbers and higher order discretization schemes.

Finally, the solvers applied to the subproblems on the different grids that appear in the combination method should be robust in the sense that they also converge rapidly on distorted grids.

The application of the combination method for the prediction of a three dimensional flow wull be subject of future work.

ACKNOWLEDGEMENT

We are indebted to Guy Lonsdale, Klaus Stüben and Hubert Ritzdorf of the GMD for continuous support concerning the L_iSS-package.

REFERENCES

1 Bungartz, H. Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung, *Dissertation*, Institut für Informatik, Technische Universität München (1992)

2 Bungartz, H., Griebel, M., Röschke, D. and Zenger, C. A proof of convergence for the combination technique for the Laplace equation using tools of symbolic computations, *SFB Bericht 342/4/93 A*, Institut für Informatik, Technische Universität München (1993)

3 Dick, E. and Linden, J. A multigrid method for solving the incompressible Navier-Stokes equations based on flux-difference splitting, in *Numerical Methods in Laminar and Turbulent Flow* (Eds Taylor, C. *et al.*), Pineridge Press, Swansea, pp. 1579–1590 (1989)

4 Durst, B. Numerische Simulation einer periodischen Nischenströmung unter Verwendung dünner Gitter, *Diplomarbeit*, Institut für Informatik, Technische Universität München (1991)

5 Griebel, M., Schneider, M. and Zenger, C. A combination technique for the solution of sparse grid problems, in *Iterative Methods in Linear Algebra* (Eds de Groen, P. and Beauwens, R.), North Holland, Amsterdam, pp. 163–281 (1992)

6 Griebel, M. The combination technique for the sparse grid solution of PDEs on multiprocessor machines, *Parallel Process. Lett.*, 61–70 (1992)

7 Griebel, M. Sparse grid methods, their parallelization, and their application to CFD, in *Parallel Computational Fluid Dynamics* (Ed. Häuser, J.), North-Holland, Amsterdam (1993)

8 Griebel, M., Huber, W., Rüde, U. and Störtkuhl, T. The combination technique for parallel sparse-grid-preconditioning and -solution of PDEs on multiprocessor machines and workstation networks, in *Lecture Notes in Computer Science 634, Parallel Processing, CONPAR92-VAPP* (Eds Bouge, L. *et al.*), Springer-Verlag, Heidelberg (1992)

9 Griebel, M., Huber, W., Störtkuhl, T. and Zenger, C. On the parallel solution of 3D PDE's on a network of workstations and on vector computers, *Lecture Notes in Computer Science*, Springer, Heidelberg (1993)

10 Hirt, C. W., Nichols, B. D. and Romero, N. C. SOLA – A numerical solution algorithm for transient fluid flows, *Technical reports UC-34 and UC-79d*, Los Alamos Scientific Laboratory of the University of California, Los Alamos (1975)

11 Lonsdale, G. and Stüben, K. The L_iSS package, *Arbeitspap. GMD 524*, Gesellschaft für Mathematik und Datenverarbeitung mbH, Sankt Augustin (1991)

12 Lonsdale, G. and Schüller, A. Parallel and vector aspects of a multigrid Navier-Stokes solver, *Arbeitspap. GMD 550*, Gesellschaft für Mathematik und Datenverarbeitung mbH, Sankt Augustin (1991)

13 Zenger, C. Sparse grids, *Techn. Rep. TUM-I-9037, (SFB-Bericht Nr. 324/18/90 A)*, Institut für Informatik, Technische Universität München (1990)