THE COMBINATION TECHNIQUE FOR THE SPARSE GRID SOLUTION OF PDE'S ON MULTIPROCESSOR MACHINES

MICHAEL GRIEBEL*

Abstract. We present a new method for the solution of partial differential equations. In contrast to the usual approach which needs in the 2-D case $O(h_n^{-2})$ grid points, our combination technique works with only $O(h_n^{-1} ld(h_n^{-1}))$ grid points, where h_n denotes the employed grid size. The accuracy of the obtained solution deteriorates only slightly from $O(h_n^2)$ to $O(h_n^2 ld(h_n^{-1}))$ for a sufficiently smooth solution. Additionally, the new method is perfectly suited for parallelization. On a machine with $ld(h_n^{-1})$ processors we get in practice an overall parallel complexity of only $O(h_n^{-1})$. The method can be generalized to higher dimensions. Then, the gain is expected to be even more dramatic. For the 2-dimensional case, we report the results of numerical experiments obtained on a Transputer system and on the CRAY Y-MP.

1. The combination technique. We consider a partial differential equation

$$Lu = f$$

in the unit cube $\bar{\Omega} = [0,1]^d \subset \mathbb{R}^d$ with a linear, elliptic operator L of second order and appropriate boundary conditions. For reasons of simplicity, we first restrict ourselves to the case $d = 2$ and later consider the case $d = 3$.

The usual approach is to discretize the problem by a finite element, finite difference, or finite volume method on an equidistant grid $\Omega_{n,n}$ with grid size $h_n = 2^{-n}$ in x- and y-direction and to solve the arising linear system of equations $L_{n,n}u_{n,n} = f_{n,n}$. Then, we get a solution $u_{n,n}$ with error $e_{n,n} = u - u_{n,n} = O(h_n^2)$, if u is sufficiently smooth. Here, we assume that $u_{n,n}$ represents an appropriate interpolant defined by the values of the discrete solutions on grid $\Omega_{n,n}$.

For the solution of the discrete system, one of the most effective techniques is the multigrid method. Roughly speaking, the number of operations is of the order $O(h_n^{-2})$ and is therefore proportional to the number of grid points.

Extending this standard approach, we now study linear combinations of discrete solutions of the problem on different rectangular grids. Let $\Omega_{i,j}$ be the uniform grid on Ω with mesh sizes $h_1 = 2^{-i}$ and $h_2 = 2^{-j}$ in x- and y-direction, respectively.

To this end, we consider the so-called combination solution

$$u_{n,n}^c = \sum_{i+j=n+1} u_{i,j} - \sum_{i+j=n} u_{i,j} \tag{1}$$

that has been introduced in [5]. Here, i ranges from 1 to n. This method is illustrated in Fig. 1. Note that this extends the approach in [6] substantially.

Thus, we have to solve n different problems $L_{i,j}u_{i,j} = f_{i,j}, i + j = n + 1$, each with about 2^n unknowns, and $n - 1$ different problems $L_{i,j}u_{i,j} = f_{i,j}, i + j = n$, each with about 2^{n-1} unknowns, and combine their bilinearly interpolated solutions. This gives a solution defined on the so-called sparse grid $\Omega_{n,n}^s$, see Fig. 2. The sparse grid $\Omega_{n,n}^s$ is a subset of the associated full grid $\Omega_{n,n}$.

* Institut für Informatik, Technische Universität München, Arcisstraße 21, D-8000 München 2

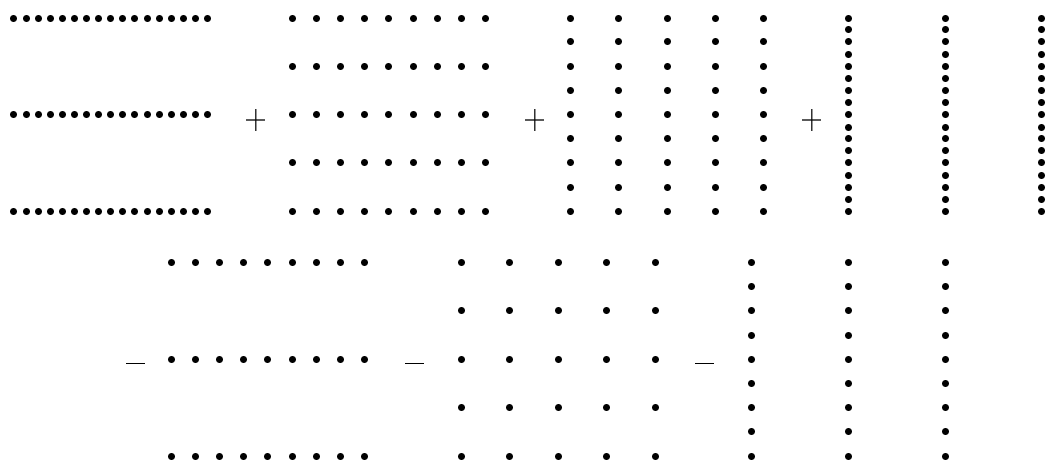

FIG. 1. *The linear combination of grids $\Omega_{i,j}$ and solutions $u_{i,j}$ with i+j=n+1 and i+j=n, n=4.*

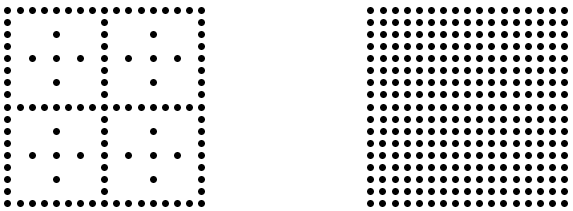

FIG. 2. *The sparse grid $\Omega^s_{4,4}$ and the associated full grid $\Omega_{4,4}$.*

2. Properties of the combination technique. Altogether, the combination technique involves $O(h_n^{-1}\mathrm{ld}(h_n^{-1}))$ unknowns in contrast to $O(h_n^{-2})$ unknowns for the conventional full grid approach. Additionally, the combination solution $u^c_{n,n}$ is nearly as accurate as the standard solution $u_{n,n}$. It can be proved (see [5]) that the error satisfies

$$e^c_{n,n} = u - u^c_{n,n} = O(h_n^2 \mathrm{ld}(h_n^{-1})).$$

This is only slightly worse than for the associated full grid, where the error is of the order $O(h_n^2)$. For the proof, we assume that u is sufficiently smooth, so that for $u_{i,j}$ (interpolated from grid $\Omega_{i,j}$ to the domain Ω) an error splitting of the form

$$u_{i,j} = u + h_i^2\omega_1(h_i) + h_j^2\omega_2(h_j) + h_i^2h_j^2\omega_3(h_i,h_j) \tag{2}$$

holds for any fixed point $(x,y) \in \Omega$ with ω_1, ω_2, ω_3 bounded by a constant C. If we insert the error splitting formula (2) into equation (1), we see that the leading error terms cancel. We get the estimation

$$|u - u^c_{n,n}| \leq C \cdot \; h_n^2 \cdot (1 + 5/4 \; \mathrm{ld}(h_n^{-1})) = O(h_n^2 \mathrm{ld}(h_n^{-1})).$$

Related sparse grid techniques have been studied in [1], where it is shown that the energy-norm of the sparse grid error is of the order $O(h_n^{-1})$, which is the same as for the full grid approach.

The combination technique can be extended to the three-dimensional case. We get

$$u^c_{n,n,n} = \sum_{i+j+k=n+2} u_{i,j,k} - 2 \sum_{i+j+k=n+1} u_{i,j,k} + \sum_{i+j+k=n} u_{i,j,k}. \tag{3}$$

This means that all problems *in total* only have $O(h_n^{-1}(\mathrm{ld}(h_n^{-1}))^2)$ instead of $O(h_n^{-3})$ unknowns. Analogous to the two-dimensional case, the accuracy deteriorates slightly both pointwise and with respect to the L_2- and L_∞-norm from $O(h_n^2)$ to $O(h_n^2(\mathrm{ld}(h_n^{-1}))^2)$. For details of the proof see [5]. Now, we have to solve $O((\mathrm{ld}(h_n^{-1})^2)$ different problems, where the size of each problem is of the order $O(h_n^{-1})$. The generalization to higher dimensions is straightforward.

The combination technique is not restricted to the unit square. We have successfully treated problems on distorted quadrilaterals, triangles, and more general domains with polygonal boundaries. Additionally, problems with a non-linear operator and PDE-systems like the Stokes and Navier-Stokes equations have been solved (see [2]).

To some extent, the combination technique even works in the case of non-smooth solutions. Now, h_i^2 and h_j^2 in (2) have to be replaced by h_i^α, h_j^β with appropriate α and β, but, because of the properties of the combination technique, the leading error terms still cancel. However, for problems with severe singularities, the appropriate combination of adaptively refined grids is recommended.

At last, we remark that all the different discrete problems whose solutions have to be combined are totally independent of each other and can be solved **fully in parallel**. This will be exploited in the following.

3. Parallel implementation on a multiprocessor system. We implemented the combination algorithm on a Transputer system with one T800 processor with 8 MB storage and 14 T800 processors with 1 MB storage. All Transputers run with 20 MH and need three wait states.

We have chosen a high level approach using HELIOS 1.2A, its TFM 2.04 and the CDL (see [7]). The different tasks have been written in C. The read- and write-routines are on posix-level. As topology of the Transputer network we selected a trinary tree. We experimented with other topologies (chain, ring, grid, binary tree), but we got nearly the same results. This might be due to the limited number of processors used.

The implementation of the combination technique is straight forward. Each problem can be assigned to one task. Therefore, a farm structure with one master and P workers was used. The master task was allocated to the Transputer with 8 MB. The whole algorithm reads as follows:

Additionally we implemented the combination technique on a **dedicated** CRAY Y-MP/8128 in Fortran using the autotasking facility cft77 with compiler version 4.0 (see [**?**]). Here, the solution of each problem is computed by a vectorized version of the multigrid algorithm. The parallel treatment of the different problems is indicated explicitly by the compiler directive CFPP$ CNCALL.

In contrast to the previous implementation in C and HELIOS we now solve the problems on the grids $\Omega_{i,j}$,$i+j=n+1$ in one first loop which is marked as concurrent. Then, we sum up the obtained solutions. The problems on the grids $\Omega_{i,j}$,$i+j=n$

```
cdl-script: master (, P <> worker)
master: anz = (2n − 1)/P; rest = (2n − 1) − anz ∗ P;
    for i = 0, i < P
    write to worker(i): number anz + (i < rest) of problems to be solved by worker;
    for k = 0, k < anz + (rest > 0)
        for i = 0, i < ((k ≠ anz) ∗ P + (k == anz) ∗ rest)
            prb = i + 1 + k ∗ P;
            write to worker(i): problem data ((prb − (prb > n) ∗ n),
                                              (n − prb + (prb ≤ n) + (prb > n) ∗ n));
        for i = 0, i < ((k ≠ anz) ∗ P + (k == anz) ∗ rest)
            read from worker(i): solution computed by worker(i);
            accumulate it to the sparse grid solution u^c_{n,n};
worker: read from master: number nrp of problems to be solved by worker;
     for k = 0, k < nrp
        read from master: problem data (i, j);
        solve the problem L_{i,j} u_{i,j} = f_{i,j} by a multigrid algorithm;
        write to master: solution u_{i,j};
```

FIG. 3. *Parallel combination algorithm.*

are solved in a second concurrent loop. At last, these solutions are subtracted. This approach helps to balance processor loads somewhat.

4. A model problem and results of numerical experiments. We consider the simple model problem

$$\Delta u = 0 \quad \text{in } \Omega = (0,1) \times (0,1) \tag{4}$$

with the exact solution $u = \sin(\pi \cdot y) \cdot \sinh(\pi \cdot (1-x))/\sinh(\pi)$ and Dirichlet boundary conditions on $\delta\Omega$.

First, we demonstrate that the accuracy of the solution that was obtained by the combination approach is nearly the same as for the full grid solution.

We see in figure 4 that the error of the solution $u_{n,n}$ that was computed on the full grid $\Omega_{n,n}$ and the error that was obtained by the new combination technique are of different types. In the first case the error is smooth while in the second case it is oscillating. Note that the errors in figure 4 are not equally scaled. The error of the combined solution is about a factor 2-3 worse.

Table 1 shows the pointwise behavior of the error of the solution that was achieved on the full grid and by the new combination technique. Here, we only consider the points P1=(0.5,0.5) and P2=(0.25,0.25). The same behavior results for other points and for the L_2- and L_∞-norms of the error.

For a smooth solution on a regular full grid we know from theory that the convergence rate of the discretization error is of the order $O(h_n^2)$. This is reflected in the left part of table 1. For the solution that was obtained by the combination technique we expect a discretization error of the order $O(h_n^2 ld(h_n^{-1}))$. The influence of the ld-term can be seen in the right part of table 1. Despite of that, the error is of nearly the same size as the error on the associated full grid. But remember that the number of grid points and thus the number of floating point operations that are necessary to solve the

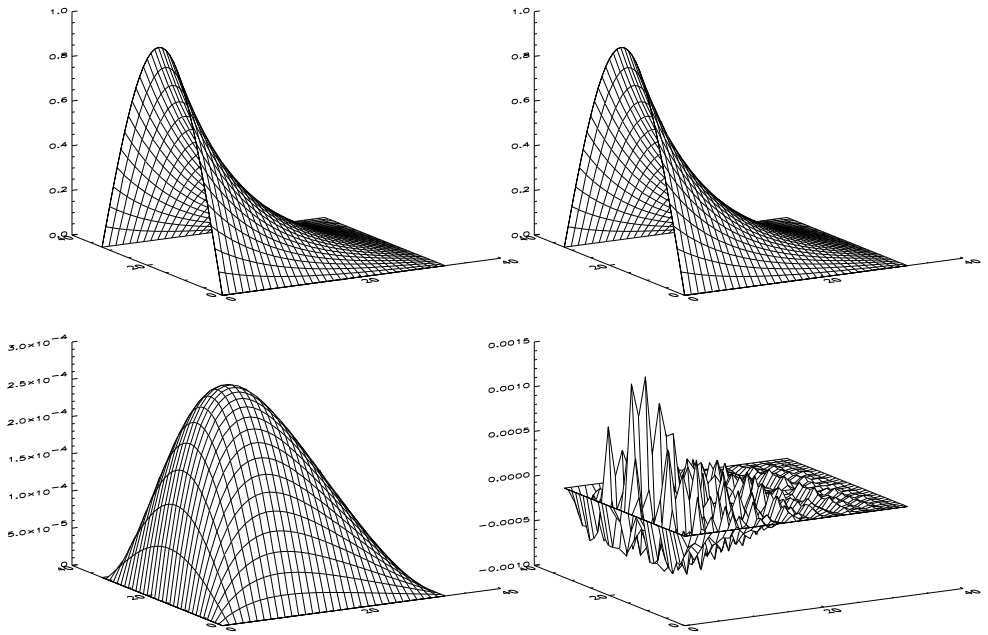

FIG. 4. *$u_{5,5}$ and $e_{5,5}$ (left), $u^c_{5,5}$ and $e^c_{5,5}$ interpolated to $\Omega_{5,5}$ (right) for our model problem.*

different problems e.g. by a multigrid method is of order $O(h_n^{-2})$ for the full grid and only of order $O(h_n^{-1} ld(h_n^{-1}))$ for the combination technique.

Now we turn to the results of our parallel algorithm which was implemented on the Transputer system. Note that we use a multigrid method (10 V-cycles with one pre- and one post-smoothing Gauss-Seidel step) for the solution of each single problem. Thus, the computational cost of the parallel tasks is of the same order as the cost of the communication of each computed solution to the master task. Therefore, the simple Laplace problem is a quite hard test example for the parallel performance of our algorithm. Further experiments with more difficult PDE's and PDE-systems like Stokes equations gave analogous results.

In the first row of table 2 ($P = 1$), we see that our algorithm shows in the sequential

TABLE 1
Pointwise error, full grid on the left, combined solution on the right.

n	$e_{n,n}(\frac{1}{2},\frac{1}{2})$	$\frac{\lvert e_{n-1,n-1}\rvert}{\lvert e_{n,n}\rvert}$	$e_{n,n}(\frac{1}{4},\frac{1}{4})$	$\frac{\lvert e_{n-1,n-1}\rvert}{\lvert e_{n,n}\rvert}$	$e^c_{n,n}(\frac{1}{2},\frac{1}{2})$	$\frac{\lvert e^c_{n-1,n-1}\rvert}{\lvert e^c_{n,n}\rvert}$	$e^c_{n,n}(\frac{1}{4},\frac{1}{4})$	$\frac{\lvert e^c_{n-1,n-1}\rvert}{\lvert e^c_{n,n}\rvert}$
1	7.4268E-2	-	-	-	7.4268E-2	-	-	-
2	1.5498E-2	4.792	1.3252E-2	-	9.9913E-3	7.433	-	-
3	3.7333E-3	4.151	3.1526E-3	4.204	1.2929E-3	7.727	6.1846E-3	-
4	9.2504E-4	4.036	7.7889E-4	4.048	5.5787E-5	23.176	1.2064E-3	5.126
5	2.3075E-4	4.009	1.9415E-4	4.012	-5.1448E-5	1.084	2.2655E-4	5.325
6	5.7655E-5	4.002	4.8503E-5	4.003	-2.9150E-5	1.764	3.8414E-5	5.897
7	1.4412E-5	4.001	1.2123E-5	4.001	-1.1357E-5	2.566	5.0772E-6	7.566
8	3.6028E-6	4.000	3.0307E-6	4.000	-3.8568E-6	2.944	1.3935E-7	3.643
9	9.0070E-7	4.000	7.5768E-7	4.000	-1.2186E-6	3.165	-2.4756E-7	0.562
10	2.2517E-7	4.000	1.8942E-7	4.000	-3.6824E-7	3.309	-1.3249E-7	1.868
11					-1.0796E-7	3.410	-5.0770E-8	2.609
12					-3.0964E-8	3.486	-1.7105E-8	2.968
13					-8.7346E-9	3.544	-5.3792E-9	3.179

TABLE 2
Times (in sec.) for combination algorithm on $\Omega^s_{n,n}$ on P Transputers

$P\backslash n$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.02	0.10	0.30	0.72	1.52	3.20	6.78	14.65	31.95	70.60	155.65	344.02	754.43	1652.04
2	0.03	0.08	0.22	0.45	0.90	1.75	3.79	7.75	17.30	36.77	83.32	178.26	397.85	849.14
3	0.06	0.11	0.22	0.45	0.74	1.62	3.72	7.18	16.25	35.81	74.74	167.20	383.03	796.53
4	0.09	0.12	0.26	0.40	0.76	1.33	3.04	5.97	13.35	27.63	63.34	133.57	296.65	623.43
5	0.11	0.15	0.24	0.38	0.64	1.29	2.42	4.92	11.00	22.81	52.01	110.98	237.15	534.63
6	0.13	0.17	0.25	0.43	0.67	1.04	2.24	4.49	9.34	21.31	44.90	91.95	210.49	450.66
7	0.14	0.19	0.28	0.37	0.62	1.07	1.77	4.09	8.54	18.45	37.69	87.21	184.71	374.45
8	0.16	0.20	0.30	0.38	0.65	1.09	1.74	3.22	7.48	16.24	36.70	76.26	171.85	366.33
9	0.18	0.23	0.32	0.41	0.56	0.99	1.82	3.25	6.13	14.48	32.15	73.44	154.46	315.88
10	0.21	0.25	0.34	0.43	0.58	1.03	1.86	3.29	6.16	11.93	28.63	64.43	147.35	312.00
11	0.22	0.27	0.37	0.45	0.60	0.84	1.67	3.31	6.19	11.95	23.62	57.37	129.43	296.87
12	0.26	0.29	0.39	0.48	0.62	0.86	1.70	3.31	6.23	12.00	23.67	47.44	115.36	261.10
13	0.28	0.32	0.41	0.50	0.64	0.89	1.38	3.00	6.23	12.03	23.70	47.50	95.67	232.60
14	0.30	0.34	0.44	0.53	0.67	0.91	1.41	3.01	6.14	12.09	23.76	47.57	95.72	193.06

version a $O(h_n^{-1} ld(h_n^{-1}))$-behavior.

For $P = 2n - 1$ we get the best overall times, as every problem runs on one single processor. However, as the problems on $\Omega_{i,j}$ have for $i + j = n + 1$ about twice as many unknowns as for $i + j = n$, nearly half of the processors are partly wasted. From the point of view of efficiency, the best choice is $P = n$. Figure 5 shows the achieved speed up and the efficiency.

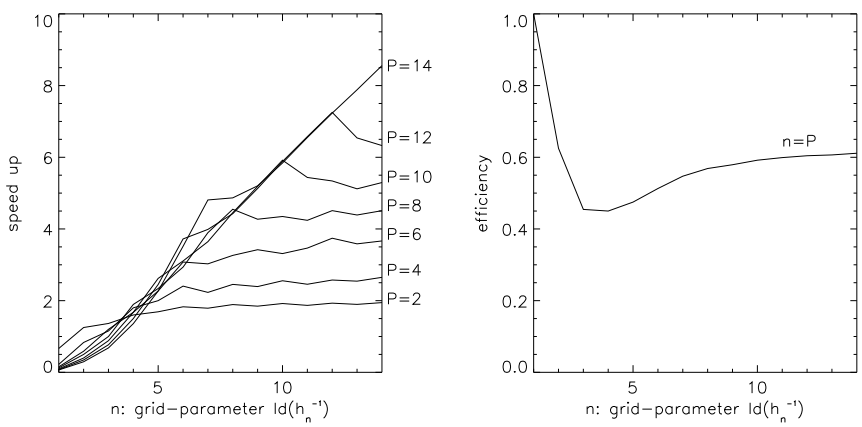

FIG. 5. *Speed up for different P and efficiency for n=P.*

Note that for our high level approach using HELIOS an efficiency of 50-61 percent is a quite good result. This reduction of efficiency is mainly due to the overhead caused by the operating system. A distribution of one problem on $\Omega_{i,j}$, $i + j = n + 1$, to one processor and two of the problems on $\Omega_{i,j}$, $i + j = n$, to one processor gave similar results. For a direct implementation in Occam a still higher efficiency can be expected.

Additionally, it can be seen from the diagonal entries of table 2 that our combination algorithm has, at least in practice, a parallel complexity of only $O(h_n^{-1})$ if we use $ld(h_n^{-1})$ processors.

These results have to be compared to the standard approach on the full grid. Table 3 shows the measured times for the solution by a multigrid algorithm on the full $O(h_n^{-2})$-

grid which was run sequentially on one single T800 with 8 MB memory. We used 10 V-cycles with one pre- and one post-smoothing step.

TABLE 3
Times (in sec.) for standard multigrid algorithm on $\Omega_{n,n}$ in sequential mode on one processor.

n	1	2	3	4	5	6	7	8	9	10
time	0.01	0.05	0.16	0.52	1.95	7.78	31.68	129.26	526.56	2125.41

The numbers of table 3 clearly show the $O(h_n^{-2})$-behavior of the conventional approach. If we compare this to the results in table 2, we see directly the superiority of the combination technique. For example, the solution of the problem on grid $\Omega_{10,10}$ with the sequential multigrid algorithm needs 2125.41 seconds. Using the combination technique we obtain at least for $n = 11$ the same accuracy. (The value $n = 11$ instead of $n = 10$ copes with the additional ld-term of the accuracy.) However, the solution with 11 processors (plus one processor for the master task) only needs 23.62 seconds, which is about 90 times faster. In general, the results for the parallel combination are a factor of the order $O(h_n^{-1})$ better than for the standard approach in its sequential version. Of course, the standard multigrid algorithm on $\Omega_{n,n}$ can be accelerated somewhat by a parallel implementation, but the good results of table 2 can never be reached.

Now we turn to the results that were obtained on the CRAY Y-MP/8128. For the parallel version of the combination algorithm (parallel treatment of the different problems by explicit compiler directives) which was run on a **dedicated** CRAY Y-MP/8128 with 8 processors we measured the elapsed times as shown in table 4. Here,

TABLE 4
Times (in sec.) for parallel combination algorithm on $\Omega^s_{n,n}$ on CRAY using P processors.

$P\backslash n$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\times$	10^{-4}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-1}	10^{-1}	10^{-1}
1	1.87	1.52	4.78	10.9	22.1	35.9	5.91	9.67	15.7	26.8	47.2	8.78	17.1	34.4
2	1.87	1.00	3.03	6.07	11.0	18.8	3.15	5.06	8.12	13.7	25.5	4.55	8.81	17.6
3	1.87	1.00	2.17	4.73	8.69	13.4	2.14	3.56	5.55	10.0	17.2	3.08	6.11	12.4
4	1.87	1.00	2.04	3.55	6.78	11.4	1.75	2.60	4.36	7.67	13.3	2.33	4.96	9.97
5	1.87	1.02	2.09	3.55	6.84	10.2	1.47	2.30	3.58	5.97	10.6	2.09	4.02	7.72
6	1.87	1.05	2.10	3.58	5.52	8.42	1.41	2.13	3.48	5.80	9.32	1.63	3.65	6.96
7	1.87	1.07	2.16	3.62	5.53	8.42	1.28	1.82	2.91	4.70	8.84	1.58	2.86	5.52
8	1.87	1.09	2.19	3.64	5.55	8.55	1.22	1.78	2.80	4.67	7.99	1.47	2.75	5.42

the entries of each column have to be multiplied by the associated factor in the second row. The measured Mflop per wall-second are shown in table 5.

For the parallel version of our algorithm we get the best times and Mflops rates on the CRAY in the case $n \leq 8$ (mostly) for $P \geq n$. In the case $n \geq 8$ we get the best times and Mflops rates of course for $P = 8$.

Furthermore, the speed up is good for $P \geq n$. Then, the efficiency is optimal for $P = n$. In table 6 we see the speed up and the efficiency for $P = n$ with $n < 8$ and for $P = 8$ with $n \geq 8$.

We achieve on the CRAY fast execution times with fairly good speed up (e.g. with 8 processors 5.43 for $n = 8$ and 6.35 for $n = 14$) and efficiency (e.g. with 8 processors

TABLE 5

Mflop per second for parallel combination algorithm on $\Omega^s_{n,n}$ on CRAY using P processors.

$P\backslash n$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.01	0.11	0.27	0.73	0.53	4.86	11.75	22.32	39.69	63.07	91.47	120.3	143.8	160.8
2	0.01	0.10	0.37	1.02	1.49	5.53	11.86	24.67	48.55	87.26	134.9	200.4	244.5	298.7
3	0.02	0.11	0.38	1.02	2.45	5.85	12.66	26.28	55.08	101.6	152.9	259.3	349.4	416.2
4	0.02	0.11	0.37	1.02	2.44	5.69	12.74	27.97	46.74	105.5	200.9	320.3	413.9	505.0
5	0.02	0.11	0.36	1.01	2.45	5.71	10.68	26.51	61.35	121.3	218.8	343.2	484.6	627.8
6	0.02	0.11	0.37	1.03	1.79	5.36	13.25	29.56	59.74	105.9	224.9	392.1	508.4	682.4
7	0.01	0.10	0.37	1.01	2.46	5.79	13.05	29.48	62.76	128.5	234.0	398.0	589.9	808.5
8	0.02	0.10	0.36	0.98	2.36	5.57	12.82	28.48	60.41	122.3	216.9	394.8	624.2	832.7

TABLE 6

Speed up and efficiency for parallel combination algorithm on $\Omega^s_{n,n}$ with P=n processors.

	$P = n$							$P = 8$						
n	1	2	3	4	5	6	7	8	9	10	11	12	13	14
speed up	1	1.52	2.24	3.07	3.23	4.26	4.62	5.43	5.61	5.74	5.91	5.97	6.22	6.35
efficiency	100	76.5	74.7	76.8	64.6	71.0	66 .0	67.9	70.1	71.8	73.9	74.6	77.8	79.4

67.9 for $n = 8$ and 79.4 for $n = 14$). Thus, for the CRAY there is not such a large reduction of speed up and efficiency caused by the operation system as for the HELIOS implementation. Additionally, we obtain for large n really high Mflops rates.

The gain which is possible by parallelization behaves in principle analogous to the results of the Transputer implementation, but on a much better level. This, of course, is due to the superior processors of the CRAY and their vectorization ability. For large values of n, however, the quite small number of available processors on the CRAY in contrast to a Transputer system is a certain drawback.

5. Concluding remarks. In this paper, we presented the combination technique as a method to solve partal differential equations on sparse grids. In contrast to the usual approach which needs $O(h_n^2)$ grid points in the 2D case, our combination technique works with only $O(h_n^{-1} ld(h_n))$ grid points, where h_n denotes the employed grid size. The accuracy of the solution is nearly the same as for the full grid solution, provided that the solution is sufficiently smooth.

We discussed the parallelization aspects of our method and presented performance figures for implementations on a Transputer system and on the CRAY Y-MP. There, excellent parallelization properties of the combination technique have been demonstrated. Furthermore, we showed that the combination technique performs on a machine with $ld(h_n^{-1})$ processors in practice with an overall parallel complexity of only $O(h_n^{-1})$.

Recently, the combination technique has been implemented and tested for a three-dimensional model problem on a workstation network (see [3]) and on different MIMD-computers (see [4]).

REFERENCES

[1] H. BUNGARTZ, *An adaptive Poisson solver using hierarchical bases and sparse grids*, SFB Bericht 342/21/91 A, Institut für Informatik, August 1991. Also to be published in: Proceedings of the

International Symposium on Iterative Methods in Linear Algebra, Bruxelles, April 2-4, 1991.

[2] M. Griebel, *Sparse grid multilevel methods, their parallelization and their application to cfd*, in Proceedings of the Parallel CFD'92 Conference, Mai 17-20, 1992, R. P. et al., ed., 1992. Implementations and Results Using Parallel Computers.

[3] M. Griebel, W. Huber, U. Rüde, and T. Stoertkuhl, *The combination technique for parallel sparse-grid-preconditioning or -solution of pde's on workstation networks*, in Proceedings of the CONPAR92 VAPP Conference, September 1-4, 1992, M. C. et al., ed., 1992.

[4] ——, *The combination technique for parallel sparse-grid-preconditioning and -solution of pde's on on multiprocessor machines and workstation networks*, SFB Bericht 342/11/92 A, Institut für Informatik, TU München, Mai 1992.

[5] M. Griebel, M. Schneider, and C. Zenger, *A combination technique for the solution of sparse grid problems*, SFB Bericht 342/19/90, Institut für Informatik, TU München, October 1990. Also to be published in: Proceedings of the International Symposium on Iterative Methods in Linear Algebra, Bruxelles, April 2-4, 1991.

[6] J. Hennart and E. Mund, *On the h- and p-versions of the extrapolated gordon's projector with applications to elliptic equations*, SIAM J. Sci. Stat Comput., 9 (1988), pp. 773–791.

[7] Perihelion, *The HELIOS operating system*, Prentice Hall, UK, 1989.