NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
*Numer. Linear Algebra Appl.* 2010; **17**:199–210
Published online 9 February 2010 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nla.697

# Optimizing the number of multigrid cycles in the full multigrid algorithm

A. Thekale[1,*,†], T. Gradl[2], K. Klamroth[3] and U. Rüde[2]

[1]*Department of Mathematics, Applied Mathematics II, Martensstr. 3, 91058 Erlangen, Germany*
[2]*Department of Computer Science, System Simulation, Cauerstr. 6, 91058 Erlangen, Germany*
[3]*Department of Mathematics, Optimization and Approximation, Gaußstr. 20, 42097 Wuppertal, Germany*

SUMMARY

Multigrid (MG) methods are among the most efficient and widespread methods for solving large linear systems of equations that arise, for example, from the discretization of partial differential equations. In this paper we introduce a new approach for optimizing the computational cost of the *full MG* method to achieve a given accuracy by determining the number of MG cycles on each level. To achieve this, a very efficient and flexible *Branch and Bound* algorithm is developed. The implementation in the parallel finite element solver *Hierarchical Hybrid Grids* leads to a significant reduction in CPU time. Copyright © 2010 John Wiley & Sons, Ltd.

Received 27 April 2009; Revised 1 December 2009; Accepted 2 December 2009

KEY WORDS: full multigrid; optimization; Branch and Bound; hierarchical hybrid grids

## 1. INTRODUCTION

Partial differential equations (PDEs) are used to describe many real life processes in various fields, e.g. engineering, natural sciences or economics. Their solution is thus significant for both science and industry. To solve such equations, they are usually discretized and lead to large linear systems of equations.

Multigrid (MG) methods [1, 2] belong to the group of iterative solvers and are among the most efficient solvers for PDEs. They respect the large- and small-scale features of the solution by using a hierarchy of grids with different mesh widths. On the finest grid only a few iterations of a local method, e.g. the Gauss–Seidel algorithm, are employed to produce a smooth error. The smooth error can then be eliminated cheaply by a recursive procedure on the coarser grids. Various strategies for combining smoothing and recursion have been developed, among the most popular ones are $V$-, $W$-, and F-cycles. They differ in their computational efficiency when applied to different types of PDEs.

*Correspondence to: A. Thekale, Department of Mathematics, Applied Mathematics II, Martensstr. 3, 91058 Erlangen, Germany.
†E-mail: alexander.thekale@am.uni-erlangen.de

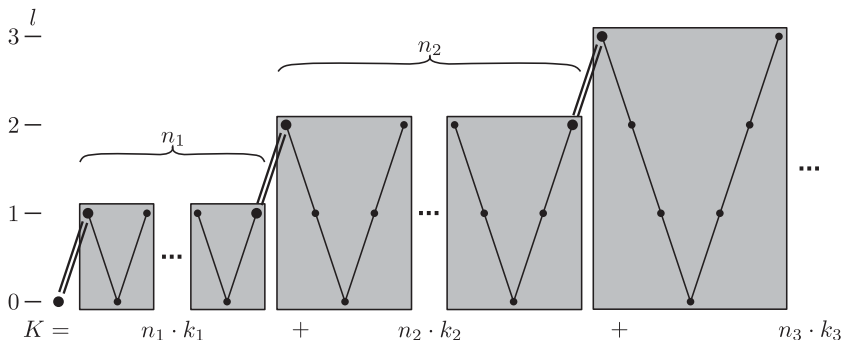


Figure 1. The full multigrid method. V-cycles (outlined by gray boxes) are used as basic building blocks. The total cost of the algorithm ($K$) is determined by the cost of an individual V-cycle ($k_l$) and the number of V-cycles performed on each level ($n_l$).

Several optimiziation strategies for MG methods have already been developed, for example, to improve the hardware utilization [3] and to optimize the mesh sizes [4] and the smoother properties [5].

The remainder of this paper does not assume a special cycle type, and we will therefore use the general term *MG cycle*. The MG cycles within the boxes shown in Figure 1 are V-cycles. These cycles are used as building blocks of the *full multigrid* (*FMG*) algorithm depicted in Figure 1. FMG starts with solving the discrete system of equations up to the *discretization error* on the coarsest level. The discretization error is defined as the difference between the exact continuous solution of the PDE and the solution obtained by solving the discrete system. The approximate solution is then interpolated to the next finer level, where one or more MG cycles are performed. In each MG cycle the error of the approximation is reduced relative to the discretization error by a factor which we call the *convergence rate*. After that, the solution is again interpolated to the next finer level, and so on. The FMG algorithm is asymptotically optimal, which means that the cost to solve the problem is proportional to the number of grid points on the finest grid. This makes FMG the predestined candidate for solving very large systems.

Reducing the CPU time of an FMG run is therefore important, especially when applied to high-end applications on supercomputers [6–8]. To minimize this computational cost while maintaining an acceptable accuracy, it is necessary to understand the error propagation in FMG. Thus, a basic error and cost model is derived in Section 2. This motivates the Branch and Bound optimization algorithm that is constructed in Section 3. Some model extensions, which are necessary to represent the quirks of real-world MG solvers, can easily be integrated in the Branch and Bound algorithm, stressing its flexibility. This is described in Section 4. Some examples demonstrating the practical value of the solver are shown in Section 5. The variables used in this paper are described in Table I.

## 2. BASIC ERROR AND COST MODEL

### *2.1. Assumptions*

The assumptions in this section do not describe realistic MG methods in enough detail, but are sufficient to motivate the optimization approach described in Section 3. Extensions necessary for modeling MG methods realistically are developed in Section 4.



DOI: 10.1002/nla

Table I. Notation used throughout the paper.

| | |
|---|---|
| $d$ | Spatial dimension |
| $l$ | Level |
| $l_c$ | Coarsest level. On this level the problem is solved up to $e^*_{l_c}$ |
| $l_f$ | Finest level |
| $e^*_l$ | Discretization error on level $l$ |
| $e^{\uparrow}_l$ | Interpolation error on level $l$ |
| $e_l$ | Total error on level $l$ |
| $c$ | Constant determining the error bound |
| $n_l$ | Number of MG cycles on level $l$ |
| $\rho_{l,i}$ | Convergence rate of the $i$th MG cycle on level $l$ |
| $k_l$ | Cost of one MG cycle on level $l$ |
| $K$ | Total cost of an FMG run |

While in practice the FMG algorithm can start at any refinement level, we first assume that it starts at level zero (i.e. $l_c = 0$) where the cost for the exact discrete solution can be neglected. Additionally, we assume that the discretization error $e^*_l$ on level $l$ follows perfectly an $O(h^2)$ behavior, i.e. it decreases by a factor of $2^2 = 4$ with each level of refinement. Imposing, without loss of generality, a normalization of the initial discretization error as $e^*_0 = 1$, the discretization error on any level is given by $e^*_l = 2^{-2l} = 4^{-l}$. Section 4.4 will show that the convergence rate $\rho_{l,i}$ may be different for each level $l$ and for each MG iteration $i$ at that level. For our basic cost and error model, we assume that the convergence rate is always the same ($\rho_{l,i} = \rho$ for all $l = 1, \ldots, l_f$ and $i \in \mathbb{N}$). The treatment of varying $\rho_{l,i}$ is described in Section 4.4.

The problem shall be discretized on a rectangular grid with $2^l$ grid points in each space dimension. Therefore, the total number of grid points on level $l$ is $2^{dl}$. V-cycles shall be used as MG cycles on every level. The cost of a V-cycle on level $l$ is the sum of the costs of its basic building blocks on level $l$ (smoothing, residual computation, restriction and prolongation) plus the cost of a V-cycle on level $l-1$. The costs of the building blocks are proportional to the number of grid points on that level. Without loss of generality, the proportionality constant is set to 1. To describe the cost of an MG cycle, we use abstract *work units* proportional to the number of grid point updates during that cycle. These work units would have to be translated to actual execution times regarding implementation and machine characteristics. With these assumptions, the cost $k_l$ of a V-cycle on level $l$ is given by $k_l = 2^{dl} + k_{l-1}$. The cost for solving the problem on the coarsest grid is set to 1: $k_0 = 1$. Therefore, $k_l = \sum_{i=0}^{l} 2^{di}$.

### 2.2. Construction of the basic model

When using an iterative solver, the algebraic error $e_l - e^*_l$ is reduced by a factor of $\rho$ (the convergence rate) in every iteration (see Figure 2). After $n_l$ iterations on level $l$, the total error $e_l$ is in dependence of the initial error $e_{l-1}$ propagated from the coarser grid:

$$e_l = e^*_l + \rho^{n_l}(e_{l-1} - e^*_l). \tag{1}$$



DOI: 10.1002/nla

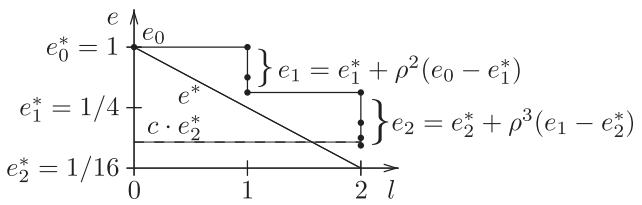


Figure 2. Error reduction, according to the basic error model, for an FMG run with $n_1 = 2$ and $n_2 = 3$.

Recursively using this equation for $e_{l-1}$ yields

$$e_l = e_l^* + \rho^{n_l}(e_{l-1}^* + \rho^{n_{l-1}}(e_{l-2} - e_{l-1}^*) - e_l^*) \tag{2}$$

$$= e_l^* + \rho^{n_l}(e_{l-1}^* - e_l^*) + \rho^{n_l + n_{l-1}}(e_{l-2} - e_{l-1}^*). \tag{3}$$

On level 0 the total error is equal to the discretization error ($e_0 = e_0^*$), because the discrete problem is solved exactly on this level. Using this terminal for the recursive Equation (2), and recalling that $e_l^* = 4^{-l}$, the error on the finest level can be written as

$$e_{l_f} = e_{l_f}^* + \sum_{l=1}^{l_f} (e_{l-1}^* - e_l^*)\rho^{n_{l_f} + \cdots + n_l} \tag{4}$$

$$= 4^{-l_f} + \sum_{l=1}^{l_f} 3 \cdot 4^{-l} \rho^{n_{l_f} + \cdots + n_l}. \tag{5}$$

Related error models can also be found in, for example, [9–11]. The total cost $K$ of the FMG algorithm is the sum of the costs of all V-cycles on all levels

$$K = \sum_{l=0}^{l_f} n_l k_l \quad \text{with } n_0 = 1 \quad \text{and} \quad k_0 = \text{const.} \tag{6}$$

## 3. BRANCH AND BOUND OPTIMIZATION

In this section we describe an optimization approach aiming at the minimization of the total cost $K$ as derived in (6) while guaranteeing that the final error is below a prespecified error level. As will be shown in Section 5, this leads in general to a significant reduction in the overall computational cost since the optimization routine can be implemented very efficiently.

In the following, the error bound will be specified with respect to the discretization error $e_{l_f}^*$ on the finest level, e.g. a constant $c > 1$ is defined by the user such that the final error $e_{l_f}$ after the FMG run is less or equal than $c \cdot e_{l_f}^*$. Altogether, we want to find the numbers of MG cycles $n_i$ in the levels $i = 1, \ldots, l_f$ such that the total cost $K$ in (6) is minimal and the error bound $e_{l_f} \leqslant c \cdot e_{l_f}^*$



DOI: 10.1002/nla

is maintained. More formally, we can thus formulate the optimization problem as the following nonlinear integer programming problem:

$$\begin{aligned} \min_{(n_1,\ldots,n_{l_f})^\top} \quad & \sum_{l=1}^{l_f} n_l k_l \\ \text{such that} \quad & e_{l_f} \leqslant c \cdot e^*_{l_f} \\ & n_1, \ldots, n_{l_f} \in \mathbb{N} = \{0, 1, 2, 3, \ldots\}. \end{aligned} \tag{7}$$

Note that for $l=0$ we have $n_0=1$ and thus the leading term of the sum in the objective function was omitted. Replacing $e_{l_f}$ by its expression in (5), problem (7) is equivalent to

$$\begin{aligned} \min_{(n_1,\ldots,n_{l_f})^\top} \quad & \sum_{l=1}^{l_f} n_l k_l \\ \text{such that} \quad & \sum_{l=1}^{l_f} 3 \cdot 4^{-l} \rho^{n_{l_f}+\cdots+n_l} \leqslant c e^*_{l_f} - 4^{-l_f} \\ & n_1, \ldots, n_{l_f} \in \mathbb{N}. \end{aligned}$$

We will mainly work with the shorter formulation (7) in the following.

As the number of integer variables in problem (7) is rather small, an exact solution method based on a Branch and Bound procedure appears to be appropriate. For a detailed discussion of Branch and Bound methods, we refer to [12]. In order to get good bounds as early as possible in the Branch and Bound tree, we will branch on the integer variables in a backwards procedure, i.e. beginning with variable $n_{l_f}$ and then continuing with variable $n_{l_f-1}$ and so forth. The error on level $l-1$ can be computed based on the current error on level $l$ and the number $n_l$ of MG cycles applied on level $l$. Equation (1) yields

$$e_{l-1} = e^*_l + \rho^{-n_l}(e_l - e^*_l). \tag{8}$$

If we thus start from the finest level $l_f$ with the highest acceptable error $e_{l_f} = c \cdot e^*_{l_f}$, we can easily determine a lower bound $n^{\text{low}}_{l_f}$ and an upper bound $n^{\text{up}}_{l_f}$ on the possible number of MG cycles on level $l_f$. For an intermediate level $l$ such bounds can be computed as follows: The lower bound $n^{\text{low}}_l$ must be selected such that the error $e_{l-1}$ is strictly larger than the discretization error on level $l-1$, i.e.

$$e_{l-1} = e^*_l + \rho^{-n^{\text{low}}_l}(e_l - e^*_l) > e^*_{l-1}.$$

Hence, $n^{\text{low}}_l \in \mathbb{N}$ is the smallest integer satisfying

$$n^{\text{low}}_l > \frac{\ln(e_l - e^*_l) - \ln(e^*_{l-1} - e^*_l)}{\ln \rho}. \tag{9}$$

Consequently, less than $n^{\text{low}}_l$ MG cycles on level $l$ are not admitted since otherwise the error after level $l-1$ would need to be smaller or equal to the discretization error, which is impossible.



DOI: 10.1002/nla

An upper bound $n_l^{\mathrm{up}}$, on the other hand, must be chosen such that the error $e_{l-1}$ is larger than the discretization error at level 0. Observe that a larger number of iterations on this level would not be useful since in this case, this error could be achieved by performing MG cycles exclusively on the current level. We thus obtain

$$e_l^* + \rho^{-n_l^{\mathrm{up}}}(e_l - e_l^*) \geqslant 1 = e_0^*$$

with $e_0^* = 1$. Then $n_l^{\mathrm{up}} \in \mathbb{N}$ is the smallest integer such that

$$n_l^{\mathrm{up}} \geqslant \frac{\ln(e_l - e_l^*) - \ln(1 - e_l^*)}{\ln \rho}. \tag{10}$$

The Branch and Bound algorithm is thus initiated by branching on the variable $n_{l_f}$, generating child nodes with values for $n_{l_f}$ in $\{n_{l_f}^{\mathrm{low}}, \ldots, n_{l_f}^{\mathrm{up}}\}$ and associating the costs $n_{l_f}^{\mathrm{low}} k_{l_f}, \ldots, n_{l_f}^{\mathrm{up}} k_{l_f}$ and the errors $e_{l_f - 1} = e_{l_f}^* + \rho^{-n_{Lf}}(e_{l_f} - e_{l_f}^*)$ with them, respectively.

This procedure is then iterated, wherein each iteration that active child node with the smallest cost is selected for further branching, until the coarsest level $l = 1$ is reached. In this process, the cost values and the error values are simply added over the different levels. The number of MG cycles on level 1 is then uniquely determined by $n_1 = n_1^{\mathrm{up}}$. The resulting search strategy may be interpreted as a depth first search.

As soon as level 1 is reached for the first time, a feasible solution $(n_1, \ldots, n_{l_f})^\top$ is obtained with the currently best known cost value $K_{\mathrm{best}}$. Note that $K_{\mathrm{best}}$ is an upper bound on the optimal solution value of problem (7) , which can now be used to prune all those active nodes of the Branch and Bound tree that already have an accumulated cost value that is larger than $K_{\mathrm{best}}$. Those nodes can never lead to a better solution than the already known one.

This procedure is then iterated until all active nodes have been investigated. The best known solution and the best known solution value are updated whenever another, better solution is found during the search. If several solutions achieve the same optimal cost value, the solution with the smallest total error $e_{l_f}$ according to (5) can be selected.

In Figure 3 an example for a Branch and Bound tree for $l_f = 3$ levels is shown. The node numbers show the order of appearance of the nodes in the tree, and the boxed numbers are the cumulative costs along the according branch. The optimal solution for this problem is $(1, 0, 2)^\top$ with optimal cost 18. As the assigned costs in nodes 9 and 3 are larger than $K_{\mathrm{best}}$ at the time of their consideration, the attached branches are pruned.

Note that due to the cost structure of the problem the cumulative costs are typically significantly higher at early levels of the Branch and Bound tree and become smaller at the lower levels. We can thus expect to detect suboptimal branches early in the tree, which explains the high numerical efficiency of the procedure, see Section 5.

## 4. MODEL EXTENSIONS

Having the optimization algorithm for the basic error model at hand, we now state some model extensions to reflect the 'real life' FMG method.



DOI: 10.1002/nla

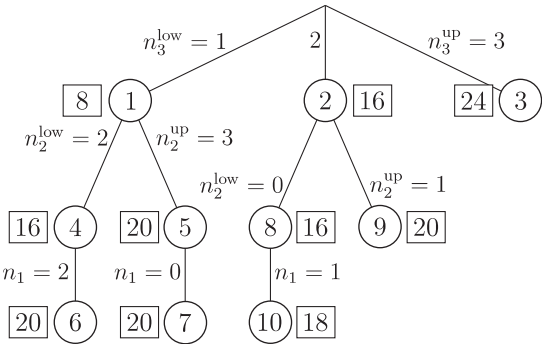


Figure 3. Branch and Bound tree.

### 4.1. Coarsest level is not level 0

We allow the user to specify some coarsest level $l_c \neq 0$ at which the FMG algorithm starts with solving the problem up to the discretization error. Then, (5) and (6) become

$$e_{l_f} = 4^{-l_f} + \sum_{l=l_c+1}^{l_f} 3 \cdot 4^{-l} \rho^{n_{l_f} + \cdots + n_l} \quad \text{and} \tag{11}$$

$$K = \sum_{l=l_c+1}^{l_f} n_l k_l. \tag{12}$$

This modification does not change the optimization algorithm in principle, but is just a shift of indices. As $l_c$ is fixed throughout the optimization and our aim is to find $n_l$ for $l = l_c+1, \ldots, l_f$, the cost on level $l_c$ is ignored in the optimization.

### 4.2. Arbitrary order of the discretization error

The discretization error depends on the selected discretization method, the smoothness of the solution and other factors. Usually only the *order D* of the discretization error is known. $e^*$ is said to be of order $D$, if there exists a constant $c_* > 0$ such that, for all levels $l = 1, \ldots, l_f$,

$$e_l^* \leqslant c_* 2^{-Dl}. \tag{13}$$

Using (13), we can reformulate (1) to

$$e_l = (1-\rho^{n_l}) e_l^* + \rho^{n_l} e_{l-1} \leqslant (1-\rho^{n_l}) c_* 2^{-Dl} + \rho^{n_l} e_{l-1} = c_* 2^{-Dl} + \rho^{n_l} (e_{l-1} - c_* 2^{-Dl}).$$

$D$ and $c_*$ are independent of the current level $l$ and the current iteration $i$ in level $l$. They are used in the optimizer as two additional parameters that are incorporated into the error calculation on each level.

### 4.3. Additional interpolation error

In practice, the prolongation of the solution from level $l-1$ to level $l$ is affected by an additional error $e_l^{\uparrow}$ depending on the selected interpolation method. This error is usually not known exactly,



DOI: 10.1002/nla

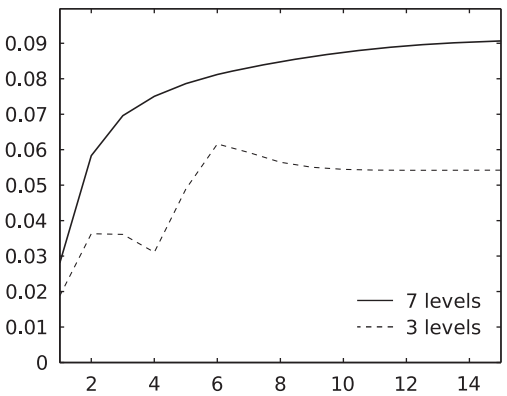


Figure 4. Convergence rates.

but only its order $P$ and a bounding constant $c_\uparrow$ can be given:

$$e_l^\uparrow \leqslant c_\uparrow 2^{-Pl}. \tag{14}$$

The interpolation error $e_l^\uparrow$ is added to the error $e_{l-1}$ from the coarser level, before the MG cycles are applied on level $l$. Therefore, (1) becomes

$$e_l = e_l^* + \rho^{n_l}(e_{l-1} + e_l^\uparrow - e_l^*), \tag{15}$$

and (4), the error on the finest level, can be written as

$$e_{l_f} = e_{l_f}^* + \sum_{l=1}^{l_f} (e_{l-1}^* - e_l^* + e_l^\uparrow)\rho^{n_{l_f}+\cdots+n_l}. \tag{16}$$

$P$ and $c_\uparrow$ are also used in the optimizer as two additional parameters that are incorporated into the error calculation on each level.

### *4.4. Level- and iteration-dependent convergence rates*

If an MG cycle is run repeatedly, in every iteration using the result of the previous iteration as initial guess, the convergence rates are usually very good in the beginning, but then they deteriorate and approach an asymptotic convergence rate [13]. This behavior is shown in Figure 4. The figure also points out that the level influences the value of the asymptotic convergence rate and the characteristics of approaching that value. On the coarser levels, the influence of the boundary is more prominent than on the fine levels, and the type of boundary conditions determines the convergence rates.

To account for varying convergence rates in the error model, we introduce a variable $\rho_{l,i}$ which denotes the convergence rate of the $i$th MG cycle on level $l$. Equation (1) has to be rewritten as

$$e_l = e_l^* + (e_{l-1} - e_l^*) \prod_{i=1}^{n_l} \rho_{l,i}, \tag{17}$$



DOI: 10.1002/nla

and (4) becomes

$$e_{l_f} = e^*_{l_f} + \sum_{l=1}^{l_f} \left( (e^*_{l-1} - e^*_l) \prod_{j=1}^{l_f} \prod_{i=1}^{n_l} \rho_{j,i} \right). \tag{18}$$

The convergence rates $\rho_{l,i}$ are often not known in as much detail as depicted in Figure 4 before the solver is actually started. Fortunately, though, the optimization algorithm is so fast that it can be started several times during the FMG run without noticeably increasing the total run time. Therefore, the optimizer should be run whenever new convergence rates are available.

### *4.5. Modifications of the Branch and Bound algorithm*

In Sections 4.1–4.4 several extensions of the basic error model derived in Section 2 have been discussed. All these extensions do not affect the Branch and Bound algorithm in general, the only modification is the determination of $n_l^{\text{low}}$ and $n_l^{\text{up}}$ for $l = l_c + 1, \ldots, l_f$. If all extensions from above are included, (8) becomes

$$e_{l-1} = c_* 2^{-Dl} - c_\uparrow 2^{-Pl} + \left( \prod_{i=1}^{n_l} \rho_{l,i} \right)^{-1} (e_l - c_* 2^{-Dl}). \tag{19}$$

Therefore, $n_l^{\text{low}}$ is the smallest $n_l \in \mathbb{N}$ such that

$$\prod_{i=1}^{n_l} \rho_{l,i} < \frac{e_l - c_* 2^{-Dl}}{c_* 2^{-D(l-1)} - c_* 2^{-Dl} + c_\uparrow 2^{-Pl}} \tag{20}$$

and $n_l^{\text{up}}$ is the smallest $n_l \in \mathbb{N}$ such that

$$\prod_{i=1}^{n_l} \rho_{l,i} \leqslant \frac{e_l - c_* 2^{-Dl}}{c_* 2^{-Dl_c} - c_* 2^{-Dl} + c_\uparrow \sum_{i=l_c+1}^{l} 2^{-Pi}} \tag{21}$$

for $l = l_c + 1, \ldots, l_f$. Note that (20) and (21) do not provide, in comparison to (9) and (10), a direct formula, but $n_l^{\text{low}}$ and $n_l^{\text{up}}$ can be determined without much computational effort by iteratively multiplying $\rho_{l,i}$ until the desired bounds are satisfied.

## 5. EXAMPLES

### *5.1. Interesting optimization results*

In the following, we show two special cases that can occur as a result of the optimization. In both examples we assume V-cycles as the basic MG cycles within the FMG algorithm.

The first example shows that there are cases in which it is optimal to perform no MG cycles on some of the levels. For the parameter set $\{d = 1, l = 10, \rho = 0.1, c = 1.06\}$ the optimal result is $(1, 0, 1, 2, 1, 0, 1, 1, 1, 2)^\top$ with a cost of $K = 6026$ and a final error of $e_{10} = 1.011 \times 10^{-6}$. We have not yet studied the feasibility of leaving out entire levels in FMG. It might, depending on the problem characteristics, lead to bad convergence rates on the level following the one with no MG cycles. If we add the constraint $n_l > 0$ for all $l = 1, \ldots, l_f$, then the optimal result for the above parameter set is $(1, 1, 1, 1, 1, 1, 1, 1, 1, 2)^\top$ with $K = 6129$ and $e_{10} = 1.0014 \times 10^{-6}$.



DOI: 10.1002/nla

Sometimes it is even cheapest to employ an additional level of refinement. For the parameters $\{d=1, l=9, \rho=0.3, c=1.1\}$ the best solution for $l=9$ is $(2,1,2,2,1,3,1,1,4)^\top$ with $K=5407$ and $e_9=4.1960\times10^{-6}$. The best solution for $l=10$, however, is $(3,2,1,2,1,2,1,2,1,1)^\top$ with $K=4764$ and $e_{10}=4.1954\times10^{-6}$. The reason is that the first solution requires many V-cycles on level 9 to reach the tight error bound of $e_9\leqslant1.1\cdot e_9^*$. With 10 levels the same error can be reached much easier because of the lower discretization error on the 10th level. Of course it is not always possible to go to the next finer level in practice, for example due to storage space restrictions on the computer.

### *5.2. The optimizer in practice*

The optimizer has been incorporated into the *Hierarchical Hybrid Grids (HHG)* MG solver [14, 15]. HHG is designed for very large finite element simulations on three-dimensional semi-structured grids. It uses an extremely memory-efficient storage scheme for the linear system of equations and MPI parallelization to exploit the capabilities of supercomputers. The largest simulation with HHG to date solved a finite element problem comprising $3\times10^{11}$ unknowns. On 9170 cores (4585 Intel Itanium2 Montecito dual-core CPUs, respectively) of HLRB II,[‡] the time to solution was about 1.5 minutes [16]. For problems of such dimensions, optimizing the number of MG cycles is invaluable, because that immediately translates to a reduction in expensive CPU hours on supercomputers, shorter waiting times in the job queues of these computers and a reduced risk of encountering hardware failures.

The following example will demonstrate the efficacy of the optimizer for HHG. To be able to document the feasibility of the optimization results, we choose a PDE for which the analytical solution is known. The Laplace equation $\Delta u=0$ in the domain $\Omega\subset\mathbb{R}^3$ with $u=g(x,y,z)=\sin(\pi x)\cdot\sinh(\pi y)$ on the Dirichlet boundary $\partial\Omega$ is satisfied by $u=g$ in $\Omega$.

The Laplace equation is solved by FMG on a hierarchy of seven levels ($l_f=6$). The coarsest level is set to $l_c=2$. Thus, optimal numbers of V-cycles per level have to be found for the levels 3–6. Comparing the discrete solution found by HHG with the analytical solution, the error bounding constants can be calculated as $c_*=91$ and $c_\uparrow=229$. The orders of the discretization and interpolation operators are $D=P=2$. The Euclidean norm was used to measure the errors throughout this example. In practice, these constants can, for the lack of an analytical solution, of course not be calculated that exactly, but they can be estimated, e.g. by evaluating the curvature of the solution. The constant bounding the final error is set to $c=3.5$, i.e. $e_6\leqslant0.078$ must be satisfied at the end of the FMG run.

Figure 5 compares three FMG runs achieving that error goal. The first run used a 'rule of thumb' setup with two V-cycles on every level. Using only one cycle per level would be infeasible with a final error of $e_6=0.12$. Two cycles per level yield an acceptable solution with $e_6=0.039$ after a run-time of 1.3 seconds. The figure shows how the error is reduced by the FMG algorithm over time. First, the problem is solved on level 2 exactly, which leaves an error of 4.8 (the discretization error on level 2) at time $t=0$ s. The prolongation to level 3 increases the error even more, to 7.2, before two V-cycles on level 3 reduce the error to 1.3. Then, the error is increased again by the prolongation to level 4, and so on. All four prolongations are visible as peaks in the error curve.

The pattern for the second run was determined by our Branch and Bound optimization, assuming a constant convergence rate of $\rho=0.3$. The optimization finds $(3,2,2,1)^\top$ on levels 3–6 to be the

[‡]http://www.lrz-muenchen.de/services/compute/hlrb/.



DOI: 10.1002/nla

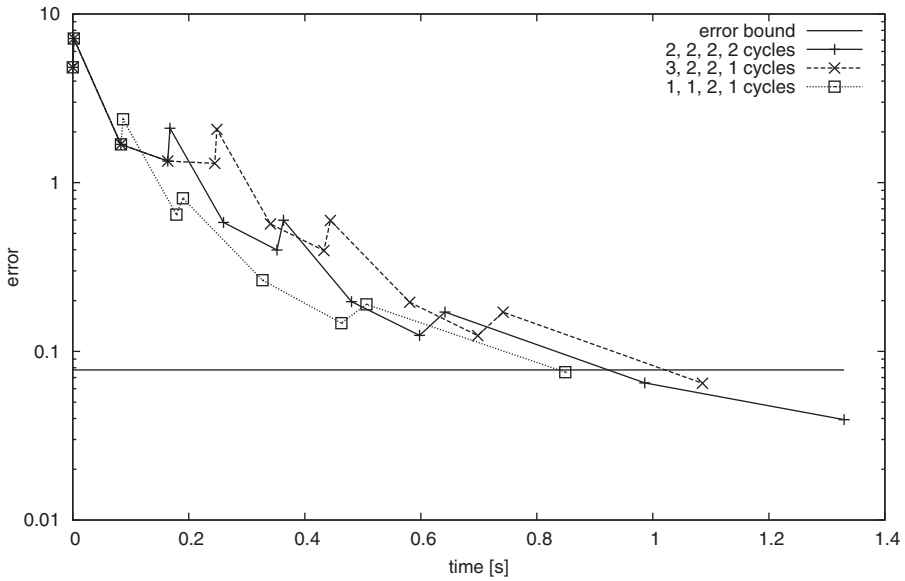


Figure 5. Practical optimization example.

cheapest pattern satisfying the error bound. The error bound is marked with a horizontal line in Figure 5. An FMG run with this cycle pattern takes 1.1 s, which is an improvement of 15% over the 'rule of thumb' setup. Note that in this specific example it would also be sufficient to just stop the $(2, 2, 2, 2)^\top$ run after the first cycle on level 6. The pattern $(2, 2, 2, 1)^\top$ would yield an acceptable error in less time than the pattern $(3, 2, 2, 1)^\top$. The optimizer did not find this pattern, because it did not know the exact convergence rates. However, without the optimizer, using only the 'rule of thumb', this pattern, though obvious in the figure, would not be found, either.

If we apply the extension of the Branch and Bound algorithm, which was introduced in Section 4.4 where the optimizer was provided with the convergence rates measured during the previous runs, the optimizer finds the pattern $(1, 1, 2, 1)^\top$. With this more realistic estimate of the convergence rates at hand the optimizer is able to shave off another 0.24 s of the FMG run-time. The third setup takes 0.85 s, 35% less than the initial setup.

## 6. CONCLUSIONS AND OUTLOOK

A fast and flexible Branch and Bound algorithm for optimizing the computational cost of the FMG algorithm has been developed. The cost-optimal solutions achieved with this algorithm are much cheaper than 'rule of thumb' strategies like performing two cycles on every level. Thus, the presented optimizer can be a valuable tool in MG practice. Implemented in the HHG solver, it drastically reduces the required CPU time. There is still room for improvements in the details, e.g. in the accurate prediction of the convergence rates.

Other groups have already put efforts into research on similar problems. Optimal local refinement with respect to cost and accuracy has been studied by Bai and Brandt [4] and De Sterck *et al.* [17]. Wienands *et al.* have optimized the recursive structure of MG cycles, i.e. how many recursive



DOI: 10.1002/nla

V-cycles to perform on each level of a V-cycle. In a future project, Wienands' algorithm will be combined with the work presented in this paper.

ACKNOWLEDGEMENTS

A. Thekale and T. Gradl are Ph.D. students of the International Doctorate Program *Identification, Optimization and Control with Applications in Modern Technologies* within the Elite Network of Bavaria. K. Klamroth and U. Rüde are the corresponding supervisors.

REFERENCES

1. Brandt A. Multigrid techniques: 1984 guide with applications to fluid dynamics. *GMD–Studien Nr. 85*. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1984.
2. Hackbusch W. Multigrid methods and applications. *Computational Mathematics*, vol. 4. Springer: Berlin, 1985.
3. Douglas CC, Hu J, Kowarschik M, Rüde U, Weiß C. Cache optimization for structured and unstructured grid multigrid. *Electronic Transactions on Numerical Analysis* 2000; **10**:21–40.
4. Bai D, Brandt A. Local mesh refinement multilevel techniques. *SIAM Journal on Scientific and Statistical Computing* 1987; **8**(2):109–134.
5. McCormick SF. Multigrid methods for variational problems: general theory for the V-cycle. *SIAM Journal on Numerical Analysis* 1985; **22**:634–643.
6. Freundl C, Gradl T, Rüde U, Bergen B, Towards petascale multilevel finite element solvers. In *Petascale Computing*: *Algorithms and Applications*, Chapter 18, Bader D (ed.). Chapman & Hall/CRC: London, Boca Raton, FL, 2007; 375–389.
7. Bergen B, Hülsemann F, Rüde U. Is $1.7 \cdot 10^{10}$ unknowns the largest finite element system that can be solved today? *SC '05*: *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*. IEEE Computer Society: Washington, DC, U.S.A., 2005.
8. Hülsemann F, Kowarschik M, Mohr M, Rüde U. Parallel geometric multigrid. *Numerical Solution of Partial Differential Equations on Parallel Computers*, Chapter 5. Lecture Notes for Computational Science and Engineering, Bruaset AM, Tveito A (eds), vol. 51. Springer: Berlin, 2005; 165–208.
9. Douglas CC, Douglas J. A unified convergence theory for abstract multigrid or multilevel algorithms serial and parallel. *SIAM Journal on Numerical Analysis* 1993; **30**:136–158.
10. Douglas CC, Douglas J, Fyfe DE. A multigrid unified theory for non-nested grids and/or quadrature. *East–West Journal of Numerical Mathematics* 1994; **2**:285–294.
11. Reusken A. Convergence of the multigrid full approximation scheme including the V-cycle. *Numerische Mathematik* 1988; **53**:663–686.
12. Nemhauser GL, Wolsey LA. *Integer and Combinatorial Optimization*. Wiley-Interscience: New York, NY, U.S.A., 1988.
13. Trottenberg U, Oosterlee C, Schüller A. *Multigrid*. Academic Press: New York, 2001.
14. Bergen B. *Hierarchical Hybrid Grids*: *Data Structures and Core Algorithms for Efficient Finite Element Simulations on Supercomputers*. Advances in Simulation, vol. 14. SCS Europe, July 2006.
15. Bergen B, Gradl T, Hülsemann F, Rüde U. A massively parallel multigrid method for finite elements. *Computing in Science and Engineering* 2006; **8**(6):56–62.
16. Gradl T, Freundl C, Köstler H, Rüde U. Scalable multigrid. In *High Performance Computing in Science and Engineering. Garching/Munich 2007*, Wagner S, Steinmetz M, Bode A, Brehm M (eds). Springer: Berlin, LRZ KONWIHR, 2008; 475–483.
17. De Sterck H, Manteuffel T, McCormick S, Nolting J, Ruge J, Tang L. Efficiency-based h- and hp-refinement strategies for finite element methods. *Numerical Linear Algebra with Applications* 2008; **15**:89–114.



DOI: 10.1002/nla