A MULTIGRID APPROACH FOR MULTI-DIMENSIONAL SCALING*

MICHAEL M. BRONSTEIN , ALEXANDER M. BRONSTEIN , RON KIMMEL , AND IRAD YAVNEH†

Abstract.
A multigrid approach for the efficient solution of large-scale multidimensional scaling (MDS) problems is presented. The main motivation is a recent application of MDS to isometry-invariant representation of surfaces, in particular, for expression-invariant recognition of human faces. Simulation results show that the proposed approach significantly outperforms conventional MDS algorithms.

Key words. multigrid, multiresolution, isometric embedding, multidimensional scaling, SMACOF, face recognition, bending-invariant canonical form, dimensionality reduction.

AMS subject classifications. 91C15

1. Introduction. *Multidimensional scaling* (MDS) is a generic name for a family of algorithms that construct a configuration of points in a target metric space from information about inter-point distances (*dissimilarities*), measured in some other metric space. The range of MDS applications is very broad and ranges from stock market analysis [18] to computational chemistry [1] and breast cancer diagnosis [23]. MDS is widely used in dimensionality reduction, data analysis and visualization applications, when, for example, one wishes to understand complicated high-dimensional data structures and represent them by low-dimensional ones [3, 24].

From the point of view of the underlying optimization problem, MDS is known to be hard, as it involves a nonlinear non-convex objective function requiring heavy computation, whose gradient and Hessian are also heavy to compute, and moreover, the Hessian is full. Current MDS algorithms are notoriously slow, and limited to small data sets. Efficiently solving large-scale MDS problems arising in numerous applications has been a challenge for long time.

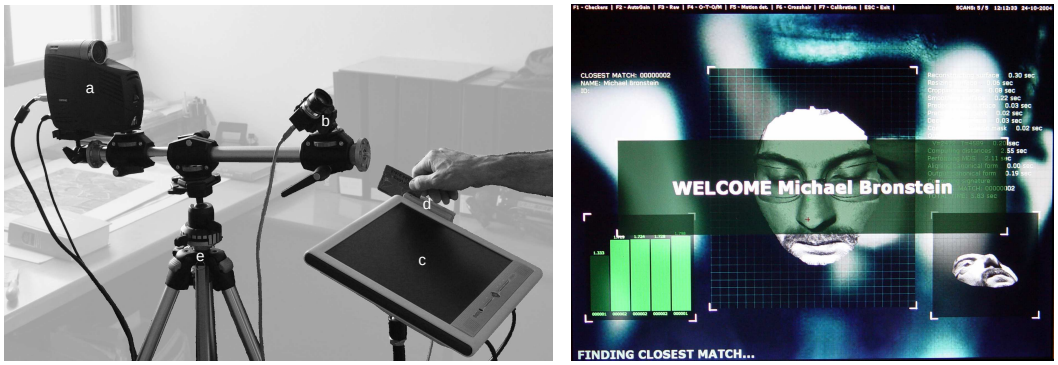
In many cases, the dissimilarities can be thought of as geodesic distances measured on a smooth Riemannian manifold, and the underlying geometry can be used to advantage. This particular setting of the MDS problem is known as the *isometric embedding* problem, and from the geometric point of view, it is the problem of representing the intrinsic structure of a Riemannian manifold in some other metric space. The isometric embedding problem was found to be important in numerous applications. Schwartz et al. [26] embedded a convoluted 3D surface of the brain cortex into a plane to study its structure as a 2D image. A similar approach was used in [28, 19] for texture mapping. Elad and Kimmel [14, 16] proposed embedding surfaces into a higher dimensional Euclidean space in order to compute their isometry-invariant signatures, which were called *canonical forms*. This approach was applied in [7, 8] to the problem of 3D face recognition.

The results presented here are motivated mainly by the 3D face recognition application. One of the greatest challenges in human face recognition is the ability to recognize people with varying facial expressions, as facial expressions can change the facial appearance dramatically. The 3DFACE prototype 3D face recognition system developed at the Department of Computer Science, Technion (Figure 1.1) uses MDS to compute an expression-invariant representation (canonical form) of the facial surface [8]. Constructing such a representation requires an efficient solution to a large-scale isometric embedding problem. One of the most crucial considerations in the face recognition application is the computation time. In the current implementation, near real-time computation of canonical forms is achieved mainly by CPU power and "technically" rather than algorithmically fast implementation in C, optimized for AMD Opteron 64-bit processor with SSE extensions [10].

In this paper, we propose the multigrid framework to boost the performance of standard MDS algorithms. The scope of the paper is the following: In Section 2, we define the MDS problem and standard methods used for its solution. Section 3 describes the particular instance of the MDS problem

*This work was supported in part by the Israel Science Foundation (ISF), Grant No. 738/04 and Bar Nir Bergreen Software Technology Center of Excellence (STL).

†Department of Computer Science, Technion – Israel Institute of Technology, Haifa 32000, Israel (bronstein@ieee.org, alexbron@ieee.org, ron@cs.technion.ac.il, irad@cs.technion.ac.il).



(a) Face recognition system (b) User interface

FIG. 1.1. *A prototype 3D face recognition system developed in the Technion (a) and a screenshot from its user interface (b) showing correct recognition of one of the authors.*

arising in isometric embedding of Riemannian manifolds. In Section 4 we introduce a multigrid MDS algorithm. Section 5 shows experimental results on the problem of facial surface embedding, and Section 6 concludes the paper.

2. Multidimensional scaling. Let $\mathbf{\Delta}$ be a symmetric $N \times N$ dissimilarity matrix with non-negative elements δ_{ij} and zero diagonal. The goal of multidimensional scaling is to find a set of points $\mathbf{x}_1, ..., \mathbf{x}_N$ in an m-dimensional metric space with the metric $d_{ij} \equiv d(\mathbf{x}_i, \mathbf{x}_j)$, such that $d_{ij} \approx \delta_{ij}$ for all $i, j = 1, ...N$. In the majority of cases, the target space is an m-dimensional Euclidean space $\mathbb{R}^m$ (which is also the assumption here), though other choices are also possible [15, 9, 27]. Practically, this influences only the way in which the distances d_{ij} are computed.

One of the common methods in MDS is to obtain the configuration of points by the minimization of the *stress*

$$s(\mathbf{X}; \mathbf{\Delta}, \mathbf{W}) = \sum_{i>j} w_{ij}(d_{ij}(\mathbf{X}) - \delta_{ij})^2, \tag{2.1}$$

where $\mathbf{X} = (x_{ij})$ is the $N \times m$ matrix of coordinates of the resulting points in an m-dimensional space, called the *configuration matrix* [21]. The symmetric $N \times N$ matrix of weights $\mathbf{W} = (w_{ij})$ determines the relative contribution of distances to the error criterion. We will distinguish between the *weighted stress*, where some values of w_{ij} are specified, and the *non-weighted stress*, where $w_{ij} = 1$.

2.1. First-order methods. Optimization of $s(\mathbf{X})$ can be performed by first-order, gradient descent-type methods, in which the direction at the $k+1$st iteration is $\mathbf{X}^{(k+1)} = -\nabla s(\mathbf{X}^{(k)})$. The gradient of $s(\mathbf{X})$ with respect to $\mathbf{X}$ is given by

$$\frac{\partial}{\partial x_{kl}} s(\mathbf{X}; \mathbf{\Delta}, \mathbf{W}) = 2 \sum_{j \neq k} w_{kj} \frac{(d_{kj} - \delta_{kj})}{d_{kj}} (x_{kl} - x_{jl}), \quad k = 1, ..., N; l = 1, ..., m; \tag{2.2}$$

and can be written as

$$\nabla s(\mathbf{X}; \mathbf{\Delta}, \mathbf{W}) = 2\mathbf{V}\mathbf{X} - 2\mathbf{B}(\mathbf{X}; \mathbf{\Delta}, \mathbf{W})\mathbf{X}, \tag{2.3}$$

where $\mathbf{V}$ is a matrix with elements

$$v_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j \\ \sum_{j=1}^{N} w_{ij} & \text{if } i = j \end{cases}, \tag{2.4}$$

and $\mathbf{B}$ is a matrix with elements

$$b_{ij} = \begin{cases} -w_{ij}\delta_{ij}d_{ij}^{-1}(\mathbf{X}) & \text{if } i \neq j \text{ and } d_{ij}(\mathbf{X}) \neq 0 \\ 0 & \text{if } i \neq j \text{ and } d_{ij}(\mathbf{X}) = 0 \\ -\sum_{j\neq i} b_{ij} & \text{if } i = j \end{cases} . \tag{2.5}$$

It was observed [20] that the first-order optimality condition, $\nabla s(\mathbf{X}) = 0$, can be written as $\mathbf{X} = \mathbf{V}^\dagger \mathbf{B}(\mathbf{X})\mathbf{X}$, and that the sequence

$$\mathbf{X}^{(k+1)} = \mathbf{V}^\dagger \mathbf{B}(\mathbf{X}^{(k)})\mathbf{X}^{(k)}, \tag{2.6}$$

converges to the local minimum of $s(\mathbf{X})$ (here $\dagger$ denotes matrix pseudoinverse). The algorithm using this multiplicative update is called SMACOF [12, 13, 3]. It can be easily shown to be equivalent to weighted gradient descent with constant step size $\mathbf{X}^{(k+1)} = -\frac{1}{2}\mathbf{V}^\dagger \nabla s(\mathbf{X}^{(k)})$, and if a non-weighted stress is used, it is essentially a gradient descent with constant step size $\mathbf{X}^{(k+1)} = -\frac{1}{2N}\nabla s(\mathbf{X}^{(k)})$.

SMACOF is widely used for large-scale MDS problems. Its disadvantage is slow convergence in the proximity of the minimum, which is inherent to all first-order methods.

2.2. Second-order methods. Some recent works (e.g. [21]) propose using second-order (Newton-type) algorithms for stress minimization. A basic Newton iteration has the form $\mathbf{X}^{(k+1)} = -\mathcal{H}^{-1}\nabla s(\mathbf{X}^{(k)})$, where $\mathcal{H}$ represents the Hessian, which is a fourth-order tensor in this notation. If $\mathbf{X}$ is column-stacked into a $Nm \times 1$ vector, the Hessian can be represented by a $Nm \times Nm$ matrix, consisting of m^2 blocks of size $N \times N$. Each block is given by

$$\mathbf{H}^{kl} = \begin{cases} \tilde{\mathbf{H}}^{kl} & \text{if } k \neq l \\ \tilde{\mathbf{H}}^{kl} + 2(\mathbf{V} - \mathbf{B}) & \text{if } k = l \end{cases} ; \quad 1 \leq k, l \leq m, \tag{2.7}$$

where

$$\tilde{h}_{ij}^{kl} = \begin{cases} -2w_{ij}(x_{ik} - x_{jk})(x_{il} - x_{jl})\delta_{ij}d_{ij}^{-3} & \text{if } i \neq j \\ 2\sum_j w_{ij}(x_{ik} - x_{jk})(x_{il} - x_{jl})\delta_{ij}d_{ij}^{-3} & \text{if } i = j \end{cases} . \tag{2.8}$$

The Newton method has quadratic convergence, as, in the proximity of the minimum, a function is accurately approximated by a quadratic function. The disadvantage of the Newton method is the relatively high computational complexity required for the Hessian construction and inversion.

2.3. Remarks. The MDS problem is in all respects a hard optimization problem. The stress is a nonlinear non-convex function w.r.t. $\mathbf{X}$, therefore convex optimization algorithms do not guarantee convergence to a global minimizer of $s(\mathbf{X})$ and can converge to local minima. Practice shows, however, that often a good initialization can prevent convergence to local minima; we will show a way to choose such an initialization in the particular case of isometric embedding.

Another characteristic of the MDS problem is the high computational complexity of the stress function and its derivatives; this stems from the fact that the matrix $\mathbf{\Delta}$ is full. Analyzing the computational complexity of the stress $s(\mathbf{X})$ and its derivatives $\nabla s(\mathbf{X}), \nabla^2 s(\mathbf{X})$ (see Table 2.1), two conclusions can be made. First, the stress and the gradient computation complexity is roughly the same. It therefore follows that the complexity of line search, which is necessary in first-order algorithms like conjugate gradients, becomes excessive. In many cases, gradient descent with constant step size (or, equivalently, the SMACOF algorithm with multiplicative update) is advantageous. Secondly, the Newton algorithm is efficient only for small-scale problems (practically, hundreds of points), as the cost of Hessian construction and inversion (requiring $\mathcal{O}(N^3m^3)$ operations) becomes prohibitive with large N.

Unfortunately, though the Hessian is structured, it is *not* sparse. The diagonals of each block are N times larger than the rest of the elements, which leads to a structure with dominant $2m - 1$ diagonals. The Hessian can be approximated by such a multidiagonal matrix, yet, since the diagonals in each block consist of sums of all the block elements, the construction cost of the sparse Hessian

TABLE 2.1

Approximate computational complexity (in terms of multiplication operations) of the stress function $s(\mathbf{X})$ and its derivatives $\nabla s(\mathbf{X}), \nabla^2 s(\mathbf{X})$. N denotes the number of points, m denotes the embedding space dimension. C_s denotes the complexity of square root computation in terms of multiplication operations; typically $C_s \approx 25$.

	Weighted	Non-weighted
s	$\frac{1}{2}N(N-1)(3+C_s)$	$\frac{1}{2}N(N-1)(2+C_s)$
∇s	$\frac{1}{2}N(N-1)(3+C_s)+2Nm$	$\frac{1}{2}N(N-1)(2+C_s)+Nm$
$\nabla^2 s$	$\frac{1}{2}N(N-1)(8+C_s)m^2$	$\frac{1}{2}N(N-1)(7+C_s)m^2$

approximation is similar to the construction cost of the full Hessian. The only advantage that can be achieved by this is the reduction of the Hessian inversion complexity for each Newton step. However, the convergence of the Newton method using such an approximation is usually inferior in comparison to other alternatives.

3. Isometric embedding. In a general MDS problem, the only data given is the dissimilarity matrix $\mathbf{\Delta}$, and it can arise from any finite metric space. Usually, there is no additional information about the "origin" of these dissimilarities. Yet, in many cases it can be assumed that the dissimilarities are geodesic distances on a smooth Riemannian manifold. We call such cases the isometric embedding problem. In isometric embedding, we represent the intrinsic geometry of a smooth Riemannian manifold using the intrinsic geometry of some other manifold (in our case, $\mathbb{R}^m$).

As an illustration, think of the problem of Earth mapping (also known as the "map-maker problem"). We wish to map the spherical surface of the Earth into a plane, preserving in the best possible way the distances between geographic objects (Figure 3.1). In other words, we embed a sphere into an Euclidean space. Since a "pure" isometric embedding does not exist in this case, we are looking for a near-isometric embedding that minimally distorts the original geodesic distances.

In the discrete setting, we have an m'-dimensional manifold $\mathcal{S}$, which is sampled at N points $\xi_1, ..., \xi_N$, and the geodesic distances $\delta_{ij} \equiv \delta(\xi_i, \xi_j)$ between all the points are computed. Isometric embedding is a mapping between two finite metric spaces,

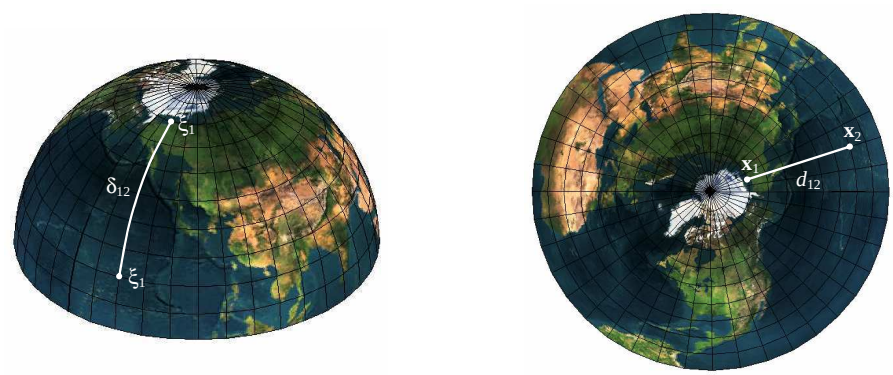
$$\varphi : (\{\xi_1, ...\xi_N\} \subset \mathcal{S}, \mathbf{\Delta}) \to (\{\mathbf{x}_1, ...\mathbf{x}_N\} \subset \mathbb{R}^m, \mathbf{D}), \tag{3.1}$$

trying to achieve $d_{ij} \approx \delta_{ij}$. This can be seen as a particular case of the general MDS problem, in which we assume that the dissimilarities δ_{ij} arise from sampling the geodesic distances on a smooth Riemannian manifold. What is more important, is that the geometric relations between the original points are known. They can either be given explicitly, as a grid or triangulation, or inferred from $\mathbf{\Delta}$ itself.[1]

In the 3D face recognition application, which was our main motivation, isometric embedding is used to create expression-invariant representations of facial surfaces [7]. The embedding allows to get rid of the deformations of the facial surface introduced by the facial expressions (which can be approximated as isometries of the facial surface [11]) and creates a representation which can be treated as a rigid object (see example in Figure 3.2). The facial surfaces (two-dimensional manifolds) are represented by $N \approx 3000$ points and N^2 corresponding inter-point geodesic distances, which are measured numerically using the Fast Marching Algorithm [22]. Then, 40 iterations of the SMACOF algorithm are performed to embed the facial surface into $\mathbb{R}^3$; as the initialization, the original 3D coordinates of the points are used (this allows to avoid convergence to local minima in most cases [10]).

4. Multigrid isometric embedding. Our previous results [10] demonstrated that substantial performance improvement can be achieved by using a multiresolution initialization to the embedding problem. Here, we extend those attempts into a genuine multigrid approach.

[1] One possibility is to define adjacent points as nearest neighbors in the sense of the dissimilarity δ_{ij}. For example, starting with a point 1, we find its nearest neighbor $i_1 = \text{argmin}_j \delta_{1j}$. These two points will be merged at a coarser resolution level, with the dissimilarities δ_{ij} involved in the computation of the interpolation coefficients (e.g. interpolation coefficients inversely proportional to δ_{ij} or δ_{ij}^2). The process is repeated for the remaining points in a similar manner. This idea is general and not limited to smooth Riemannian manifolds.



(a) Original surface (b) Embedding into $\mathbb{R}^2$

FIG. 3.1. *Illustration of the isometric embedding problem arising in cartography. A planar map (b) of the Earth (a) is created by embedding the spherical surface into $\mathbb{R}^3$, such that the geodesic distances are replaced with Euclidean ones.*

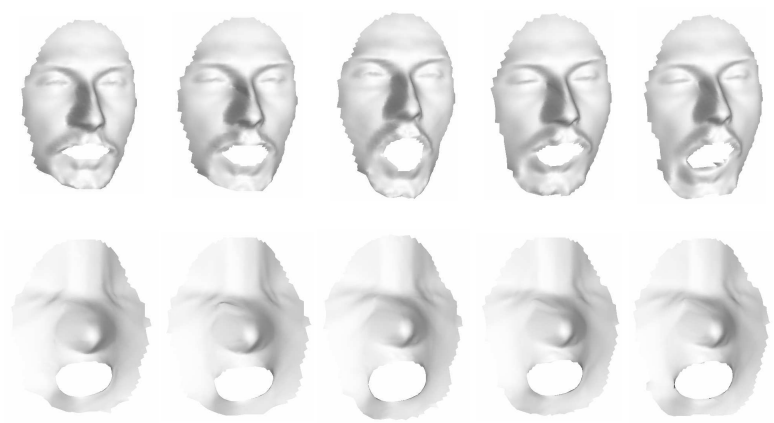


FIG. 3.2. *Example of canonical forms (second row) of facial surfaces (first row) and their insensitivity to facial expressions.*

Originally, multigrid methods were introduced in the context of differential equations [17, 2, 4, 5]. More recently, this framework was adapted to non-linear discrete optimization problems (see for example [25]). The optimization problem $\min_{\mathbf{X}} s(\mathbf{X})$ is equivalent to the solution of the non-linear equation $\nabla s(\mathbf{X}) = 0$, arising from the first-order optimality conditions. The spirit of multigrid is to solve the nonlinear problem $\nabla s(\mathbf{X}) = 0$ using a sequence of approximate solutions to nonlinear problems of the form $\nabla s(\mathbf{X}) = \mathbf{T}$, solved on coarse grids. The term $\mathbf{T}$ arises from the residual transferred from previous levels. In the optimization problem formulation, we need to minimize functions of the form $s(\mathbf{X}) - \mathrm{trace}(\mathbf{X}^{\mathrm{T}}\mathbf{T})$, whose gradient equals $\nabla s(\mathbf{X}) - \mathbf{T}$.

4.1. Modified stress. The second (linear) term makes the function $s(\mathbf{X}) - \mathrm{trace}(\mathbf{X}^{\mathrm{T}}\mathbf{T})$ unbounded. To overcome this problem, we define the *modified stress*

$$\hat{s}(\mathbf{X}; \mathbf{\Delta}, \mathbf{W}) = \sum_{i>j} w_{ij}(d_{ij}(\mathbf{X}) - \delta_{ij})^2 + \lambda \sum_{j=1}^{m} \left(\sum_{i=1}^{N} x_{ij} \right)^2 , \tag{4.1}$$

and use it instead of $s(\mathbf{X})$ hereinafter. For $0 < \lambda < \infty$ such a modification does not change the solution of the original problem, but restricts the center of mass of the resulting set of points to be at the origin. Since the second term is quadratic, the function $\hat{s}(\mathbf{X}) - \mathrm{trace}(\mathbf{X}^{\mathrm{T}}\mathbf{T})$ is bounded. This modification results in an increment by 2λ of every element of the diagonal blocks, $\mathbf{H}^{kk}$.

4.2. Coarsening strategy. We use a hierarchy of grids $\Omega_1 \supset ... \supset \Omega_R$, constructed in a multiresolution manner, where R is the coarsest resolution level, and denote by N_r the number of grid points at the r-th level. The transfer from resolution level r to the coarser level $r+1$ or the finer level $r-1$ is performed using an $N_{r+1} \times N_r$ matrix $\mathbf{P}_r^{r+1}$ (referred to as *restriction* operator in the multigrid literature) and an $N_{r-1} \times N_r$ matrix $\mathbf{P}_r^{r-1}$ (*interpolation* operator), respectively. These matrices are sparse (typically, every row contains from 1 up to 4 non-zero elements) and are often chosen to satisfy

$$\mathbf{P}_r^{r+1} = (\mathbf{P}_{r+1}^{r})^{\mathrm{T}}. \tag{4.2}$$

The optimization problem is transferred to the next coarser level by applying a restriction operator $\tilde{\mathbf{P}}_r^{r+1}$ (not necessarily equal to $\mathbf{P}_r^{r+1}$) to the matrices $\mathbf{\Delta}$ and $\mathbf{W}$. Consequently, we have a hierarchy of problems of the form

$$s_r(\mathbf{X}_r, \mathbf{T}_r) \equiv \hat{s}(\mathbf{X}_r; \mathbf{\Delta}_r, \mathbf{W}_r) - \mathrm{trace}(\mathbf{X}_r^{\mathrm{T}}\mathbf{T}_r), \tag{4.3}$$

that need to be solved at each level.

4.3. V-cycle. The simplest multigrid algorithm is the V-cycle. The complete nonlinear multigrid optimization algorithm can be defined recursively in the following manner:

`MG_Vcycle`$(\mathbf{X}_r, \mathbf{T}_r, \mathbf{\Delta}_r, \mathbf{W}_r, K_r, K'_r)$

- If $r = R$ (coarsest level) solve

$$\min_{\mathbf{X}_R} s_R(\mathbf{X}_R, \mathbf{T}_R)$$

  and return.
- Otherwise:
  - Relaxation: Apply K_r iterations of an unconstrained optimization algorithm to $s_r(\mathbf{X}_r, \mathbf{T}_r)$ initialized with $\mathbf{X}_r$ and obtain $\mathbf{X}'_r$.
  - Compute

$$\begin{aligned} \mathbf{G}'_r &= \nabla s_r(\mathbf{X}'_r); \\ \mathbf{X}'_{r+1} &= \mathbf{P}_r^{r+1}\mathbf{X}'_r; \\ \mathbf{G}'_{r+1} &= \nabla s_{r+1}(\mathbf{X}'_{r+1}); \\ \mathbf{T}_{r+1} &= \mathbf{G}_{r+1} - \mathbf{P}_r^{r+1}\mathbf{G}_r. \end{aligned}$$

  - Apply the multigrid method on the coarser level,

$$\mathbf{X}''_{r+1} \leftarrow \texttt{MG_Vcycle}(\mathbf{X}'_{r+1}, \mathbf{T}_{r+1}, \mathbf{\Delta}_{r+1}, \mathbf{W}_{r+1}, K_{r+1}, K'_{r+1})$$

  - Correction:

$$\begin{aligned} \mathbf{E}_r &= \mathbf{P}_{r+1}^{r}(\mathbf{X}''_{r+1} - \mathbf{X}'_{r+1}); \\ \mathbf{X}''_r &\leftarrow \mathbf{X}'_r + \mathbf{E}_r. \end{aligned}$$

  - Relaxation: Apply K'_r iterations of an unconstrained optimization algorithm to $s_r(\mathbf{X}, \mathbf{T})$ initialized with $\mathbf{X}''_r$ and obtain $\mathbf{X}'''_r$.

The procedure `MG_Vcycle`$(\mathbf{X}, \mathbf{T}, \mathbf{\Delta}, \mathbf{W}, K, K')$ is initialized with some $\mathbf{X}$ and $\mathbf{T} = 0$ on the finest grid. The whole algorithm is repeated for a few iterations. When multigrid works well, a small number of V-cycles is required to obtain an accurate solution.

From the point of view of computation complexity it may be advantageous to use SMACOF iterations as the relaxation procedure at higher-resolution levels and Newton iterations at coarser resolution levels.

(a) V-cycle (b) F-cycle

FIG. 4.1. *Schematic representation of a V-cycle (a) and a F-cycle (b).*

4.4. F-cycle. Another popular MG cycle is the F-cycle [6]. In the recursive definition, the F-cycle procedure $\texttt{MG_Fcycle}(\mathbf{X}_r, \mathbf{T}_r, \mathbf{\Delta}_r, \mathbf{W}_r, K_r, K'_r)$ is very similar to V-cycle, except the recursive call, which has the form

$$\mathbf{X}''_{r+1} \leftarrow \texttt{MG_Fcycle}(\mathbf{X}'_{r+1}, \mathbf{T}_{r+1}, \mathbf{\Delta}_{r+1}, \mathbf{W}_{r+1}, K_{r+1}, K'_{r+1}),$$
$$\mathbf{X}''_{r+1} \leftarrow \texttt{MG_Vcycle}(\mathbf{X}''_{r+1}, \mathbf{T}_{r+1}, \mathbf{\Delta}_{r+1}, \mathbf{W}_{r+1}, K_{r+1}, K'_{r+1}).$$

The difference between V-cycle and F-cycle is shown in Figure 4.1.

5. Results. The simulations in this paper focus on the face recognition application. We performed three experiments. The goal of the first two experiments was comparing the standard SMACOF algorithm with our multigrid implementation (V-cycle) on problems of different size. Random initialization (Experiment I) and initialization using the original point coordinates (Experiment II) were used. A facial surface patch was sampled several times by a different number of points ($N = 225, 625, 1425$ and 3249) for each test. The geodesic distances between all the points were computed using the Fast Marching algorithm [22]. The embedding space dimensionality was $m = 3$ in all the experiments.

Three resolution levels were used in the multigrid algorithm; each subsequent level containing $\frac{1}{4}$ of the points of the previous one. The grids were formed in a dyadic manner. The relaxation of each level was based on gradient descent SMACOF-like iterations described in Section 4. The optimization was terminated when the stress function at subsequent iterations decreased by less than 1% (this stopping criterion is common in MDS literature, see e.g. [3]). For comparison, a conventional SMACOF algorithm was executed with the same initialization and was terminated when reaching the same stress achieved by the multigrid algorithm.

The algorithm performance was evaluated by the stress value versus the execution time and the computational complexity in terms of multiplication operations (FLOPs). The square root operation was estimated as $C_s = 25$ FLOPs. All tests were performed on a PC with a 2 GHz Mobile Intel Pentium 4 processor and 1 GB RAM. The algorithms were implemented in MATLAB under Windows XP.

Tables 5.1 and 5.2 and Figure 5.1a-b show a comparison between a conventional SMACOF algorithm and the multigrid version in terms of convergence time and complexity in Experiments I and II, respectively. Figure 5.1c shows the boosting obtained by the multigrid implementation.

Several conclusions can be drawn from these results. First, multigrid implementation demonstrates significant performance improvement (over 6 times in Experiment II). The improvements contributed by the multigrid algorithm, when compared to standard SMACOF, become more pronounced as the size of the problem increases. That is, for large-scale problems the improvement is dramatic. Secondly, the multigrid algorithm converges approximately in the same number of iterations independent of N (in both experiments), while the number of standard SMACOF iterations grows with N. Thirdly, in Experiment I, the multigrid algorithm appears to be less sensitive to initialization (the standard deviation of the total convergence time divided by the mean convergence time is by an order of

Table 5.1
Experiment I: Comparison of multigrid and standard SMACOF algorithm on isometric embedding problems of different size, random initialization, results averaged over 10 *runs. Overall execution time shown in seconds, overall complexity shown in MFLOPs.*

N	SMACOF			MG		
	Iterations	Time	Complexity	Iterations	Time	Complexity
225	71.3	7.6 ± 1.78	113.26 ± 26.2	13.6	5.28 ± 0.36	65.83 ± 4.7
625	73	52.94 ± 10.16	894.53 ± 172.77	13.2	30.63 ± 1	480.86 ± 15.36
1425	111.5	421.76 ± 110.5	$(7.07 \pm 1.86) \times 10^3$	14.2	164.52 ± 5.53	$(2.67 \pm 0.079) \times 10^3$
3249	156	$(2.79 \pm 1.17) \times 10^3$	$(5.13 \pm 2.11) \times 10^4$	15.2	854.41 ± 29.67	$(1.49 \pm 0.04) \times 10^4$

magnitude smaller for the multigrid algorithm than for the standard SMACOF). Finally, the multigrid algorithm appears to be less likely to converge to local minima if random initialization is used; this is a similar phenomenon to what was observed when using multiresolution initialization [10].

In Experiment III, different variations of the multigrid algorithm were compared on a large problem ($N = 3249$). Specifically, we compared V-cycle with 3 and 4 resolution levels and F-cycle with 4 resolution levels. The results are shown in Table 5.3 and Figure 5.2. F-cycle with four resolution levels shows the best results; it outperforms the standard SMACOF algorithm by 8.12 times in sense of execution time and by 10.17 times in sense of overall complexity.

Table 5.2
Experiment II: Comparison of multigrid and standard SMACOF algorithm on isometric embedding problems of different size, initialization with the original points in $\mathbb{R}^3$*. Overall execution time shown in seconds, overall complexity shown in MFLOPs.*

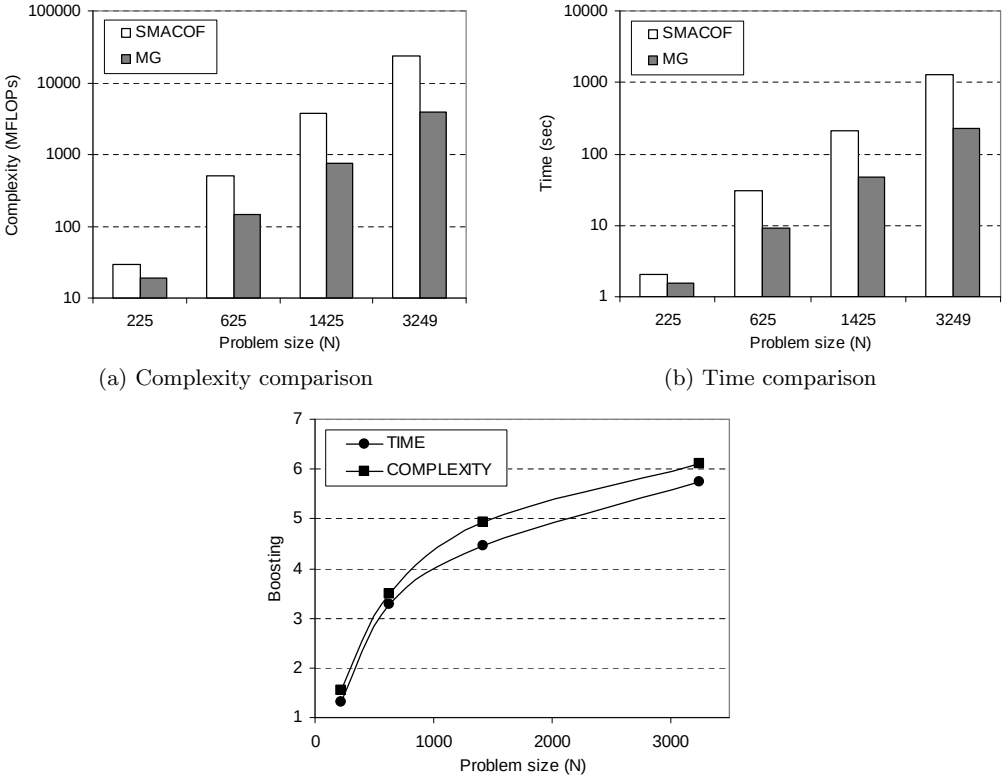
N	SMACOF			MG		
	Iterations	Time	Complexity	Iterations	Time	Complexity
225	18	2.03	29.62	4	1.53	19.22
625	41	30.41	507.03	4	9.26	145.72
1425	58	212.80	3.71×10^3	4	47.75	753.5
3249	72	1.29×10^3	2.38×10^4	4	224.37	3.90×10^3

Table 5.3
Experiment III: Comparison of different multigrid algorithms, initialization with the original points in $\mathbb{R}^3$*. Overall execution time shown in seconds, overall complexity shown in MFLOPs.*

Algorithm	Iterations	Time	Complexity
SMACOF	93	1.68×10^3	3.03×10^4
V-cycle ($R = 3$)	4	276.15	3.90×10^3
V-cycle ($R = 4$)	4	220.42	3.89×10^3
Full MG ($R = 4$)	3	205.15	2.99×10^3

6. Conclusions. We propose a multigrid framework for efficient solution of multidimensional scaling problems. Tested on on large-scale embedding problems arising in 3D face recognition, our multigrid implementation demonstrates an order of magnitude better performance compared to the conventional SMACOF algorithm.

Though presented exclusively on isometric embedding problems, the proposed framework is applicable to general MDS problems. The range of applications in which our multigrid MDS algorithm can be used is very wide, and includes, to mention a few, problems in data visualization, machine learning, computational chemistry, etc. Among immediate applications currently under research is the general problem of isometry-invariant shape recognition and real-time texture mapping on complicated 3D objects.



(a) Complexity comparison

(b) Time comparison

(c) Boosting

FIG. 5.1. *Experiment II: Comparison of multigrid and standard SMACOF algorithm on isometric embedding problems of different size, initialization with the original points in* $\mathbb{R}^3$*. (a) Complexity comparison, (b) execution time comparison, (c) boosting factor.*

REFERENCES

[1] D. K. Agrafiotis, D. N. Rassokhin, and V. S. Lobanov, *Multidimensional scaling and visualization of large molecular similarity tables*, Journal of Computational Chemistry **22** (2001), no. 5, 488–500.

[2] N. S. Bakhvalov, *On the convergence of a relaxation method under natural constraints on an elliptic operator*, USSR Comput. Math. and Math. Phys. **6** (1966), 861–883.

[3] I. Borg and P. Groenen, *Modern multidimensional scaling - theory and applications*, Springer-Verlag, Berlin Heidelberg New York, 1997.

[4] A. Brandt, *Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems*, Proc. 3rd Int'l Conf. Numerical Methods in Fluid Mechanics (Berlin), Springer Verlag, 1973, pp. 82–89.

[5] ———, *Multi-level adaptive solutions to boundary-value problem*, Math. Comp. **37** (1977), 333–390.

[6] ———, *Multigrid techniques: 1984 guide with applications to fluid dynamics*, GMD-Studie 85, Sankt Augustin, West Germany, 1984.

[7] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, *Expression-invariant 3D face recognition*, Proc. Audio and Video-based Biometric Person Authentication, 2003, pp. 62–69.

[8] ———, *Three-dimensional face recognition*, Tech. Report CIS-2004-04, Dept. of Computer Science, Technion, Israel, 2004.

[9] ———, *On isometric embedding of facial surfaces into* $\mathbb{S}^3$, Scale Space, 2005, to appear.

[10] M. M. Bronstein, *Three-dimensional face recognition*, Master's thesis, Department of Computer Science, Technion - Israel Institute of Technology, 2004.

[11] M. M. Bronstein, A. M. Bronstein, and R. Kimmel, *Expression-invariant representation of faces*, IEEE Trans. PAMI (2004), submitted.

[12] J. De Leeuw, *Recent developments in statistics*, ch. Applications of convex analysis to multidimensional scaling, pp. 133–145, North-Holland, Amsterdam, 1977.

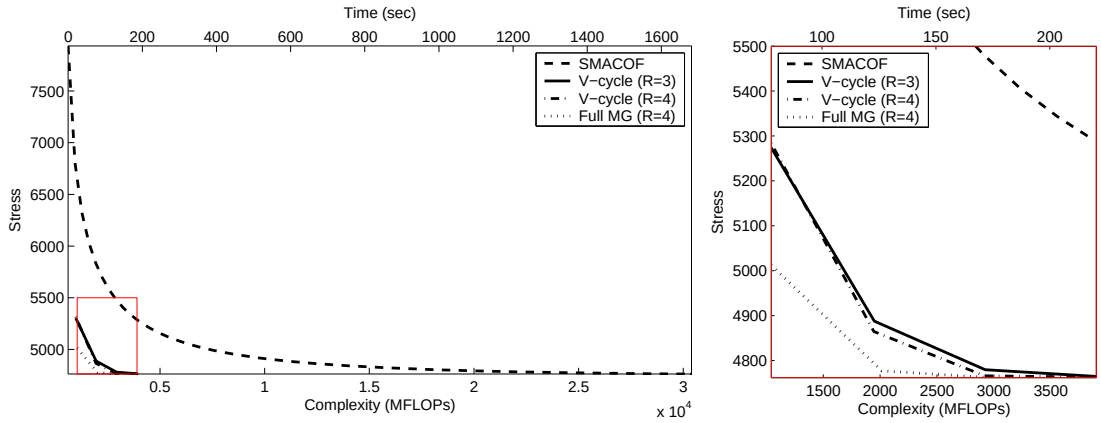


FIG. 5.2. *Experiment III: Comparison of different algorithms on problem with $N = 3249$ points, initialization with the original points in $\mathbb{R}^3$. Shown is the stress starting from the first iteration, as a function of computational complexity (bottom scale). The corresponding approximate CPU time is given on the upper scale. A zoom into a plot region marked with red is shown on the right.*

[13] J. De Leeuw and I. Stoop, *Upper bounds on Kruskal's stress*, Psychometrika **49** (1984), 391–402.

[14] A. Elad and R. Kimmel, *Bending invariant representations for surfaces*, Proc. CVPR, 2001, pp. 168–174.

[15] ———, *Geometric methods in bio-medical image processing*, vol. 2191, ch. Spherical flattening of the cortex surface, pp. 77–89, Springer-Verlag, Berlin Heidelberg New York, 2002.

[16] ———, *On bending invariant signatures for surfaces*, IEEE Trans. PAMI **25** (2003), no. 10, 1285–1295.

[17] R. P. Fedorenko, *On the speed of convergence of an iterative process*, USSR Comput. Math. and Math. Phys. **4** (1964), 559–564.

[18] P. Groenen and P. H. Franses, *Visualizing time-varying correlations across stock markets*, Journal of Empirical Finance **7** (2000), 155–172.

[19] R. Grossman, N. Kiryati, and R. Kimmel, *Computational surface flattening: a voxel-based approach*, IEEE Trans. PAMI **24** (2002), no. 4, 433–441.

[20] L. Guttman, *A general nonmetric technique for finding the smallest coordinate space for a configuration of points*, Psychometrika (1968), 469–506.

[21] A. Kearsley, R. Tapia, and M. Trosset, *The solution of the metric STRESS and SSTRESS problems in multidimensional scaling using Newton's method*, Computational Statistics **13** (1998), no. 3, 369–396.

[22] R. Kimmel and J. A. Sethian, *Computing geodesic on manifolds*, Proc. US National Academy of Science, vol. 95, 1998, pp. 8431–8435.

[23] H. Lahdesmaki, O. Yli-Harja, I. Shmulevich, and W. Zhang, *Distinguishing key biological pathways by knowledge-based multidimensional scaling analysis: application to discriminate between primary breast cancers and their lymph node metastases*, Proc. Bioinformatics, 2003.

[24] N. Linial, E. London, and Y. Rabinovich, *The geometry of graphs and some its algorithmic applications*, Combinatorica **15** (1995), 333–344.

[25] S. Nash, *A multigrid approach to discretized optimization problems*, Journal of Optimization Methods and Software **14** (2000), 99–116.

[26] E. L. Schwartz, A. Shaw, and E. Wolfson, *A numerical solution to the generalized mapmaker's problem: flattening nonconvex polyhedral surfaces*, IEEE Trans. PAMI **11** (1989), 1005–1008.

[27] J. Walter and H. Ritter, *On interactive visualization of high-dimensional data using the hyperbolic plane*, Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2002, pp. 123–131.

[28] G. Zigelman, R. Kimmel, and N. Kiryati, *Texture mapping using surface flattening via multi-dimensional scaling*, IEEE Trans. Visualization and computer graphics **9** (2002), no. 2, 198–207.

A NONLINEAR ENERGY-BASED MULTILEVEL QUANTIZATION SCHEME *

MARIA EMELIANENKO†

Abstract

Quantization has diverse applications in many areas of science and engineering. In this paper, we present a new multilevel algorithm that accelerates existing numerical approaches to finding optimal quantizers. A framework for theoretical analysis of a new algorithm and results of computational experiments are provided.

1. Introduction. A vector quantizer maps N-dimensional vectors in the domain $\Omega \in \mathbb{R}^N$ into a finite set of vectors $\{\mathbf{z}_i\}_{i=1}^k$. Each vector $\mathbf{z}_i$ is called a code vector or a *codeword*, and the set of all the codewords is called a codebook. Associated with each codeword, $\mathbf{z}_i$, also called a *generator*, is a nearest neighbor region called Voronoi region $\{V_i\}_{i=1}^k$. For each i, V_i consists of all points in the domain Ω that are closer to $\mathbf{z}_i$ than to all the other generating points. A Voronoi tessellation refers to a tessellation of a given domain $\Omega \in \mathbb{R}^N$ by the Voronoi regions $\{V_i\}_{i=1}^k$ associated with a set of given generating points $\{\mathbf{z}_i\}_{i=1}^k \subset \Omega$.

For a given density function ρ defined on Ω, we may define the centroids, or mass centers, of regions $\{V_i\}_{i=1}^k$ by

$$\mathbf{z}_i^* = \frac{\displaystyle\int_{V_i} \mathbf{y}\rho(\mathbf{y})\, d\mathbf{y}}{\displaystyle\int_{V_i} \rho(\mathbf{y})\, d\mathbf{y}}\,. \tag{1.1}$$

Then, a *centroidal Voronoi tessellation* (CVT), also called an *optimal quantizer*, refers to a Voronoi tessellation for which the generators themselves are the centroids of their respective Voronoi regions, that is, $\mathbf{z}_i = \mathbf{z}_i^*$ for all i. The concept of CVT has diverse applications in many areas of science and engineering, such as image and data analysis, vector quantization, resource optimization and numerical partial differential equations [1], [2], [8], [9], [12], [14]. In particular, the CVTs have been widely used in the design of optimal vector quantizers in electrical engineering [6], [7], [16]. We refer to [1] for a more comprehensive review of the mathematical theory and diverse applications of CVTs.

In the seminal work of Lloyd on the least square quantization [13], one of the algorithms proposed for computing optimal quantizers is an iterative algorithm consisting of the following simple steps: starting from an initial Voronoi tessellation corresponding to an old set of generators, a new set of generators is defined by the mass centers of the Voronoi regions. This process is continued until certain stopping criterion is met.

Given a set of points $\{\mathbf{z}_i\}_{i=1}^k$ and a tessellation $\{V_i\}_{i=1}^k$ of the domain, we may

*This research was supported in part by grants from US NSF DMS-0409297 and NSF ITR-0205232.

†Department of Mathematics, Pennsylvania State University, University Park, PA 16802. (emeliane@math.psu.edu).

define the *energy functional* or the *distortion value* for the pair $(\{\mathbf{z}_i\}_{i=1}^k, \{V_i\}_{i=1}^k)$ by:

$$\mathcal{H}\Big(\{\mathbf{z}_i\}_{i=1}^k, \{V_i\}_{i=1}^k\Big) = \sum_{i=1}^k \int_{V_i} \rho(\mathbf{y})|\mathbf{y} - \mathbf{z}_i|^2 \, d\mathbf{y} \, .$$

The minimizer of $\mathcal{H}$ necessarily forms a CVT which illustrates the optimization property of the CVT [1]. Meanwhile, it is easy to see that the Lloyd algorithm is an energy descent iteration, which gives strong indications to its practical convergence.

Lloyd's algorithms sparked enormous research efforts in later years [7] − [9] and their variants have been proposed and studied in many contexts for different applications. An attempt has been made recently in [10], [11] to introduce a multigrid approach to quantization. For modern applications of the CVT concept in large scale scientific and engineering problems such as data communication and mesh generation, efficient algorithms for computing the CVTs play crucial roles.

2. Convergence properties of the Lloyd's iteration. Even with their great success in practice, only limited rigorous results on the convergence properties of Lloyd's iteration have been obtained and many important computational issues remain to be explored [1]. Some important characterizations of convergence for the Lloyd's scheme have been obtained recently in [3]. Here are the results that demonstrate global convergence properties of the Lloyd iteration and its relationship to the critical points of the energy functional.

THEOREM 2.1. *Any limit point* $\mathbf{Z}$ *of Lloyd algorithm is a fixed point of the Lloyd map, and thus,* $(\mathbf{Z}, \mathbf{Z})$ *is a stationary point of* $\mathcal{H}$*. The set of limit points share the same distortion value* $\mathcal{H}$ *for a given iteration.*

THEOREM 2.2. *If the iterations in the Lloyd algorithm stay in a compact set where the Lloyd map* T *is continuous, then the algorithm is globally convergent to a critical point of* $\mathcal{H}$*.*

We refer to [3] for the proofs and further discussions of these and related results.

Beyond the study on the global convergence, the characterization of the convergence rate is often also important in practice. For instance, one may inquire if a geometric convergence rate can be established. This is indeed verified in [1] for the constant density function and later in [3] under strong type of log-concavity conditions, where the established geometric convergence rate r is shown to be of the order of $1 - ck^{-2}$, so that the Lloyd method slows down for large values of k, the total number of generators. Even in the one-dimensional case, both our theoretical estimates and the experiments indicate that the convergence of Lloyd iterations is at most linear.

3. The new energy-based nonlinear multigrid algorithm. The evidence of slow convergence of the Lloyd iteration and its descent properties motivated our search for a Lloyd iteration based numerical scheme with superior convergence properties.

One possible approach to the problem of speeding up convergence for Lloyd's method is to use a domain decomposition or multigrid strategies. There are many ways one could implement such an algorithm in the context of CVTs. However, the problem of constructing a CVT is nonlinear in nature and hence cannot be analyzed using standard linear multigrid theory. We hope to overcome the difficulties of this approach by essentially relying on the energy minimization.

3.1. Description of the algorithm. Our motivation in using the energy minimization approach was the optimality property of the CVTs mentioned above. The

optimality property implies that at the optimal quantizer $\mathbf{z}^*$,

$$\mathcal{H}(\mathbf{z}^*) = \min_{\mathbf{z}\in\Omega} \mathcal{H}(\mathbf{z}) \Rightarrow \nabla\mathcal{H}(\mathbf{z}*) = 0$$

Since the energy functional is in general non-convex, we can use a dynamic nonlinear preconditioner to relate our problem to a convex optimization problem. More precisely, denote $R = \mathrm{diag}\{R_i\}, i = 1, \ldots, k+1$ with $R_i = \int_{V_i} \rho(\mathbf{y})\, d\mathbf{y}$ being the masses of the corresponding Voronoi cells. Then we arrive at an equivalent formulation of the energy minimization problem:

$$R\nabla\mathcal{H}(\mathbf{z}*) = 0 \Longleftrightarrow ||R\nabla\mathcal{H}||^2 \to \min \tag{3.1}$$

As we are going to see later, this formulation helps to make the energy functional convex and hence has theoretical advantages over the original one. In other words, we will base our new multigrid algorithm on the following nonlinear optimization problem

$$\min_{\mathbf{z}\in\Omega} \tilde{\mathcal{H}}(\mathbf{z}), \text{ where } \tilde{\mathcal{H}}\Big(\{\mathbf{z}_i\}_{i=1}^k, \{V_i\}_{i=1}^k\Big) = ||R\nabla\mathcal{H}||^2 \tag{3.2}$$

We decompose the space into a sum of closed nested subspaces $\Omega = \sum_{i=1}^H V_i$. On each subspace V_i we fix basis functions $\{\psi_j^i\}_{j=1}^{n_i}$ and define corresponding one-dimentional subspaces $M_{i,j} = \mathrm{span}(\psi_j^i)$. Then our decomposition can be regarded as $\Omega = \displaystyle\sum_{i=1}^{H}\sum_{j=1}^{n_i} M_{i,j}$. In general, basis functions can be chosen arbitrary with the only restriction that the system they form is orthogonal. For simplicity, theoretical estimates given below are carried out for the "hat" basis. It has to be noted that convergence results will remain true as long as the orthogonality condition is met, while exact values of the constants might change. With the use of the above notations, we design a succesive subspace correction algorithm as follows:

ALGORITHM 3.1 (**Successive correction** $V(\nu_1, \nu_2)$ **scheme**).

Input:
Ω*, the domain of interest;* $\Omega = \sum_{i=1}^H V_i$*,* V_i*-closed subspaces,*
$\{\psi_j^i\}_{j=1}^{n_i}$ *- basis for the space* V_i*,* ρ*, a probability distribution on* Ω*;*
k*, number of generators;* $\mathbf{Z} = \{z_i\}_1^k$*, the initial set of generators.*
Output:
$\{V_i\}_1^k$*, a CVT with* k *generators* $\{z_i\}_1^k$ *in* Ω
Method:
1. For i=H:-1:2
Repeat ν_1 *times:*
given $\mathbf{Z}$*, find* $\mathbf{Z} = \mathbf{Z} + \alpha_j^0\psi_j^i$ *sequentially for each* $j = 1, \ldots, n_i$
such that $\tilde{\mathcal{H}}(\mathbf{Z} + \alpha_j^0\psi_j^i) = \min_{\alpha_j} \tilde{\mathcal{H}}(\mathbf{Z} + \alpha_j\psi_j^i)$
endfor
2. $\mathbf{Z} \leftarrow$*CoarseGridSolve(*$\mathbf{Z}$*)*
3. For i=2:1:H
Repeat ν_2 *times:*
given $\mathbf{Z}$*, find* $\mathbf{Z} = \mathbf{Z} + \alpha_j^0\psi_j^i$ *sequentially for each* $j = 1, \ldots, n_i$
such that $\tilde{\mathcal{H}}(\mathbf{Z} + \alpha_j^0\psi_j^i) = \min_{\alpha_j} \tilde{\mathcal{H}}(\mathbf{Z} + \alpha_j\psi_j^i)$
endfor
4. Repeat the procedure 1 to 3 until some stopping criterion is met.

Each step of the procedure outlined above involves solving a system of nonlinear equations which plays the role of relaxation. Solution at current iterate is updated after each nonlinear solve by the Gauss-Seidel type procedure, hence the resulting scheme is successive in nature. It is worth noting that in the 1-dimensional case the set of basis functions

$$Q_i = [\psi_1^i, \ldots, \psi_{n_i}^i]^T \in R^{n_i \times k}$$

used at each iteration can be pregenerated using the following procedure:

$$\begin{array}{l} Q_1 = I_{k\times k} \\ Q_s = (\Pi_{i=1}^s P_i) Q_1 \end{array}$$

Here P_i is the basis transformation from space V_{i+1} to V_i which plays a role of a restriction operator. For the "'hat"' basis, for instance, transformation matrix has the structure

$$P_i = \begin{pmatrix} 1 & 0.5 & 0 & 0 & \ldots \\ 0 & 0.5 & 1 & 0.5 & \\ 0 & 0 & 0 & 0.5 & \\ & & \ldots & & \ddots \end{pmatrix} \in R^{\frac{(n_{i+1}+1)}{2} \times n_{i+1}}$$

The basis transformation in higher dimensions is specific to geometry, so it is difficult in general to hardwire it into the code. In this case at each step of the algorithm one can build a layered structure based on the cell incidence and construct a basis supported on each of these layers. Efficient ways of building this structure are discussed in [5] .

3.2. Multigrid convergence analysis. Since well-studied linear multigrid techniques are not applicable in this case, we resort to nonlinear approach based on convexity. In the analysis to follow we partially adopt the framework of [15] , at the same time exploiting the CVT properties that come with original formulation. In this paper we will present only the one-dimensional analysis and will omit some of the technical proofs for the sake of brevity. More details and discussion on the higher dimensional case can be found in [4], [5] .

3.2.1. Energy estimates. As shown below, multigrid scheme proposed in the previous section is uniformly convergent, provided that $\tilde{\mathcal{H}}$ satisties the following two conditions in some space V :

Condition 1. 'Convexity'
There exist constants $K > 0, L > 0, p \geq q > 1$ s.t.

$$(\tilde{\mathcal{H}}'(w) - \tilde{\mathcal{H}}'(v), w - v) \geq K||w - v||_V^p, \forall w, v \in V$$

Condition 2. 'Continuity'

$$(\tilde{\mathcal{H}}'(w) - \tilde{\mathcal{H}}'(v), w - v) \leq L||w - v||_V^q, \forall w, v \in V$$

Consider the set of iteration points $\mathbf{W}$ defined as follows:

$$\mathbf{W} = \{(w_i)|_{i=0}^{k+1}| \quad w_0 = a, w_{k+1} = b, a \leq w_i \leq b \forall 0 \leq i \leq k+1\}$$

Supply $\mathbf{W}$ with the following norm:

$$||y||^2_{1,\mathbf{W}} = \frac{1}{k}\sum_{i=1}^{k+1}(y_i - y_{i-1})^2$$

This definition of the norm on $\mathbf{W}$ is justified by the fact that if $\sum_{i=1}^{k+1}(y_i - y_{i-1})^2 = 0$, we have $y_i - y_{i-1} = 0 \quad \forall i$, which contradicts the condition that $y_0 = a, \quad y_{k+1} = b$ for all points in $\mathbf{W}$, unless $y_i = a, \forall 0 \le i \le k+1$.
Let us show that for some classes of densities the energy functional defined in (3.2) satisfies both of the conditions 1 and 2 in $\mathbf{W}$ with induced norm. The two lemmas that follow give some useful estimates for energy derivatives.

LEMMA 3.1. *Let $\rho(x) = 1$ be the density function on $[a, b]$. Then the following relation holds:*

$$(\tilde{\mathcal{H}}'(u) - \tilde{\mathcal{H}}'(w), u - w) = \frac{1}{4}\sum_{i=1}^{k+1}(a_i - b_i)^2$$

where $a_i = u_i - u_{i-1}, b_i = w_i - w_{i-1}, i = 1, \ldots, k$, $u_0 = w_0 = a, u_{k+1} = w_{k+1} = b$.

Proof. can be found in [4]. □

This lemma justifies the fact that for constant densities the energy functional $\tilde{\mathcal{H}}$ satisfies continuity and convexity conditions for all points in $\mathbf{W}$.

It is possible to extend this result to a broader class of density functions. Indeed, consider a small enough perturbation of a density in the form $\rho(x) = 1 + \epsilon g(x)$, where $g(\cdot)$ is any smooth function on $[a, b]$. We can derive the following

LEMMA 3.2. *For any $\rho(x) = 1 + \epsilon g(x)$, there exist constants $C_l(\epsilon, k)$ and $C_u(\epsilon, k))$ such that*

$$C_l(\epsilon, k)\sum_{i=1}^{k+1}(a_i - b_i)^2 \le (\tilde{\mathcal{H}}'(u) - \tilde{\mathcal{H}}'(w), u - w) \le C_u(\epsilon, k)\sum_{i=1}^{k+1}(a_i - b_i)^2$$

where $a_i = u_i - u_{i-1}, b_i = w_i - w_{i-1}, i = 1, \ldots, k$, $u_0 = w_0 = a, u_{k+1} = w_{k+1} = b$ and

Proof. Let T_i be the centroid of the i-th cell. Then for any density function of the form $\rho(x) = 1 + \epsilon g(x)$, we have

$$\frac{\partial \tilde{\mathcal{H}}}{\partial u_i} = 2(u_i - T_i) = \frac{2}{M_i(u)}\Big(\int (u_i - u)du + \epsilon\int (u_i - u)g(u)du\Big) =$$

$$= \frac{(u_i^+ - u_i^-)(2u_i - u_i^+ - u_i^-) + 2\epsilon\int (u_i - u)g(u)du}{(u_i^+ - u_i^-) + \epsilon\int g(u)du} =$$

$$= 2u_i - u_i^+ - u_i^- + \frac{2\epsilon\int (u_i - u)g(u)du - \epsilon(u_i^+ - u_i^-)\int g(u)du}{(u_i^+ - u_i^-) + \epsilon\int g(u)du} =$$

$$= \frac{1}{2}\Big(a_i - a_{i+1}\Big) - \epsilon\Big(\frac{\int (2u - (u_i^+ + u_i^-))g(u)du}{(u_i^+ - u_i^-) + \epsilon\int g(u)du}\Big)$$

Denote $Q = \{Q_i\}$, where $Q_i(u) = \dfrac{\int (2u - (u_i^+ + u_i^-))g(u)du}{(u_i^+ - u_i^-) + \epsilon \int g(u)du} = \dfrac{N_i(u)}{D_i(u)}$. Then

$$(\tilde{\mathcal{H}}'(u) - \tilde{\mathcal{H}}'(w), u - w) = \frac{1}{2}\sum_{i=1}^{k}(a_i - b_i)^2 - \epsilon(Q(u) - Q(w), u - w) \tag{3.3}$$

The first term in (3.3) comes from the constant part of the density and hence complies with the results of the previous theorem. It remains to get a similar estimation for the second term. From Cauchy inequality,

$$|(Q(u) - Q(w), u - w)| \le \sum_{i=1}^{k}|Q_i(u) - Q_i(w)| \cdot |u_i - w_i| \tag{3.4}$$

Put $g(x) = x^n$. Let us get exact expression for Q_i for this specific type of density first. For the numerator we have:

$$N_i(u) = \int (2u - (u_i^+ + u_i^-))g(u)du =$$

$$= 2\int u^{n+1}\,du - (u_i^+ + u_i^-)\int u^n\,du = \frac{2u^{n+2}}{n+2}\Big|_{u_i^-}^{u_i^+} - \frac{(u_i^+ + u_i^-)u^{n+1}\big|_{u_i^-}^{u_i^+}}{n+1} =$$

$$= \Big(u_i^+ - u_i^-\Big)\Big(\frac{2}{n+2}\sum_{k+l=n+1}(u_i^+)^k(u_i^-)^l - \frac{(u_i^+ + u_i^-)}{n+1}\sum_{k+l=n}(u_i^+)^k(u_i^-)^l\Big)$$

while the denominator is equal to

$$D_i(u) = (u_i^+ - u_i^-)\Big(1 + \frac{\epsilon}{n+1}\sum_{k+l=n}(u_i^+)^k(u_i^-)^l\Big)$$

Finally, for the ratio $Q_i = \dfrac{N_i}{D_i}$ we have

$$\frac{N_i(u)}{D_i(u)} = \Big(\frac{2}{n+2}\sum_{k+l=n+1}(u_i^+)^k(u_i^-)^l - \frac{(u_i^+ + u_i^-)}{n+1}\sum_{k+l=n}(u_i^+)^k(u_i^-)^l\Big)/$$
$$\Big(1 + \frac{\epsilon}{n+1}\sum_{k+l=n}(u_i^+)^k(u_i^-)^l\Big) =$$
$$= \Big(\frac{2}{n+2}S_{n+1}(u_i^+, u_i^-) - \frac{(u_i^+ + u_i^-)}{n+1}S_n(u_i^+, u_i^-)\Big)/\Big(1 + \frac{\epsilon}{n+1}S_n(u_i^+, u_i^-)\Big)$$

where

$$S_n(u_i^+, u_i^-) = \sum_{k+l=n}(u_i^+)^k(u_i^-)^l$$

For simplicity let us redefine the modified numerator as $\tilde{N}_i(u) = \dfrac{2}{n+2}S_{n+1}(u_i^+, u_i^-) - \dfrac{(u_i^+ + u_i^-)}{n+1}S_n(u_i^+, u_i^-)$ and denominator as $\tilde{D}_i(u) = 1 + \dfrac{\epsilon}{n+1}S_n(u_i^+, u_i^-) \ge 1$. Then

$$|Q_i(u) - Q_i(w)| \le \frac{1}{\tilde{D}_i(u)\tilde{D}_i(w)}\Big|\tilde{N}_i(u)\tilde{D}_i(w) - \tilde{N}_i(w)\tilde{D}_i(u)\Big| \le$$
$$\le \frac{1}{2}\Big|\tilde{N}_i(u) - \tilde{N}_i(w)\Big|\Big(\tilde{D}_i(u) + \tilde{D}_i(w)\Big) + \frac{1}{2}\Big(\tilde{N}_i(u) + \tilde{N}_i(w)\Big)\Big|\tilde{D}_i(u) - \tilde{D}_i(w)\Big|$$

Notice further that

$$\begin{aligned}\left|\tilde{D}_i(u)-\tilde{D}_i(w)\right| &= \frac{\epsilon}{n+1}\left|S_n(u_i^+,u_i^-)-S_n(w_i^+,w_i^-)\right| \\ \left|\tilde{N}_i(u)-\tilde{N}_i(w)\right| &= \Big|\frac{2}{n+2}\Big(S_{n+1}(u_i^+,u_i^-)-S_{n+1}(w_i^+,w_i^-)\Big)- \\ &- \frac{1}{n+1}\Big((u_i^++u_i^-)S_n(u_i^+,u_i^-)-(w_i^++w_i^-)S_n(w_i^+,w_i^-)\Big)\Big|\end{aligned}$$

In general, if $S_n(a,b)=\sum_{k+l=n}a^kb^l$, then it is not hard to show that $\Big|S_n(a,b)-S_n(c,d)\Big|\le\frac{n(n+3)}{2}\Big(|a-c|+|b-d|\Big)$ and $\Big|(a+b)S_n(a,b)-(c+d)S_n(c,d)\Big|\le(n+4)(n+1)\Big(|a-c|+|b-d|\Big)$. Hence we immediately get the following inequalities:

$$\begin{aligned}|\tilde{D}_i(u)-\tilde{D}_i(w)| &\le \frac{\epsilon(n+3)}{2}(|u_i^+-w_i^+|+|u_i^--w_i^-|) \\ |\tilde{N}_i(u)-\tilde{N}_i(w)| &\le (n+4)(|u_i^+-w_i^+|+|u_i^--w_i^-|)\end{aligned}$$

Finally, since $\tilde{D}_i\le 1+\epsilon$, and $\tilde{N}_i\le 2$,

$$\begin{aligned}|Q_i(u)-Q_i(w)| \le (1+\epsilon)|\tilde{N}_i(u)-\tilde{N}_i(w)|+2|\tilde{D}_i(u)-\tilde{D}_i(w)| \le \\ (1+\epsilon)(2n+7)(|u_i^+-w_i^+|+|u_i^--w_i^-|)\end{aligned}$$

so that

$$|Q_i(u)-Q_i(w)|\le(1+\epsilon)(2n+7)(|u_i^+-w_i^+|+|u_i^--w_i^-|) \tag{3.5}$$

Combining (3.3), (3.4) and (3.5), we get

$$\begin{aligned}&(\tilde{\mathcal{H}}'(u)-\tilde{\mathcal{H}}'(w),u-w)\le \\ \le\ & \tfrac{1}{2}\sum_{i=1}^N(a_i-b_i)^2+2\epsilon(1+\epsilon)(2n+7)\sum|u_i^+-w_i^+||u_i-w_i| \\ \le\ & \tfrac{1}{2}\sum_{i=1}^N(a_i-b_i)^2+2\epsilon(1+\epsilon)(2n+7)\sum|u_i-w_i|^2\end{aligned}$$

Since $x_i=x_i-x_0=\sum_{l=1}^i(x_l-x_{l-1})$, L_2 -norm can be dominated by H_1 -norm as follows: $\sum x_i^2\le 2k\sum(x_i-x_{i-1})^2$. Hence

$$(\tilde{\mathcal{H}}'(u)-\tilde{\mathcal{H}}'(w),u-w)\le(\frac{1}{2}+4\epsilon(1+\epsilon)k(2n+7))\sum_{i=1}^k(a_i-b_i)^2=C_u(\epsilon,k,n)||u-w||_1^2$$

Same arguments applied to the lower bound yield

$$(\tilde{\mathcal{H}}'(u)-\tilde{\mathcal{H}}'(w),u-w)\ge(\frac{1}{2}-4\epsilon(1+\epsilon)k(2n+7))\sum_{i=1}^k(a_i-b_i)^2=C_l(\epsilon,k,n)||u-w||_1^2$$

By using Taylor expansion, this result can be extended to all smooth function $g(\cdot)$ on $[a,b]$. The statement of the lemma follows. □

Note that it follows from lemma 3.2 that in order to preserve convexity the perturbation has to be of the order of $\epsilon=O(k^{-1})$.

In addition to conditions on the energy functional, we also need the following assumptions on the space decomposition:

Condition 3.
$\forall v \in \mathbf{W}, \exists v_i \in \mathbf{W}_i s.t. \sum_{i=1}^m v_i = v$, and

$$(\sum_{i=1}^m ||v_i||^2_{1,\mathbf{W}})^{1/2} \leq C_1 ||v||_{1,\mathbf{W}}$$

Condition 4. 'Strengthened Cauchy-Schwartz'
$\forall w_{ij} \in \mathbf{W}, u_i \in \mathbf{W}_i, v_j \in \mathbf{W}_j \Rightarrow$

$$\sum_{i,j=1}^m (\tilde{\mathcal{H}}'(w_{ij}+u_i) - \tilde{\mathcal{H}}'(w_{ij}), v_j) \leq C_2 (\sum_{i=1}^m ||u_i||^2_{1,\mathbf{W}_i})^{1/2} (\sum_{j=1}^m ||v_j||^2_{1,\mathbf{W}_j})^{1/2}$$

3.2.2. Uniform convergence results. Finally, putting together Conditions 3,4 and using convexity and continuity of $\tilde{\mathcal{H}}$ in $\mathbf{W}$, we're equipped to prove the following uniform convergence result:

THEOREM 3.3. *Under assumptions 3,4 on space decomposition, Algorithm 3.1 converges uniformly in* $\mathbf{W}$ *for any density of the type* $\rho(x) = 1 + \epsilon g(x)$ *with* $d_n = \tilde{\mathcal{H}}(u_n) - \tilde{\mathcal{H}}(u)$ *satisfying*

$$d_n \leq r d_{n-1}, \quad \rho \in (0,1)$$

for some constant $r = \frac{C}{1+C}$ *, where* $C = C_1^2 C_2^2 L / K^3$ *.*

The proof of this result is similar to the one given in [15] and is omitted here.

COROLLARY 3.4. *In the case of a "'hat"' basis, the constants* C_1 *and* C_2 *can be estimated as* $C_1 = 1$ *and* $C_2 = 2L$ *, so for example when* $\rho(x) = 1$ *,* $C = 4$ *.*

It is possible to obtain similar estimates for a nonlinear density case (see [4]).

It follows that for the suitable choice of decomposition in 1D the asymptotic factor is independent of the size of the problem and the number of grid levels, which gives a significant speedup comparing to other methods, like the traditional Lloyd iteration. We justify this in the numerical examples that follow.

3.3. Numerical results. Below are the computational results obtained for the V(1,1) multigrid implementation of the new algorithm in comparison with the regular Gauss-Seidel performance. We plot the convergence factor $\rho \approx \dfrac{z_{n+1} - z_n}{z_n - z_{n-1}}$ for each V(1,1) cycle with respect to the total number of generators (grid points) involved.

Table I shows the number of multigrid cycles $V(\nu_1, \nu_2, \mu)$ needed to reduce the error to $\epsilon = 10^{-12}$.

$k/V(\nu_1,\nu_2)$	V(1,0)	V(0,1)	V(1,1)	V(2,0)	V(0,2)	V(2,2)
3	7	8	6	6	7	4
5	11	11	8	8	8	6
9	13	14	9	9	9	7
17	18	18	12	12	12	8
33	21	20	13	12	13	8
65	21	22	12	12	12	8
129	21	21	12	12	12	8
257	20	23	12	12	13	7
513	20	22	12	11	13	7
1025	19	22	11	11	13	7

Table I. Number of $V(\nu_1, \nu_2)$ cycles needed to reduce the error to machine zero vs. number of generators.

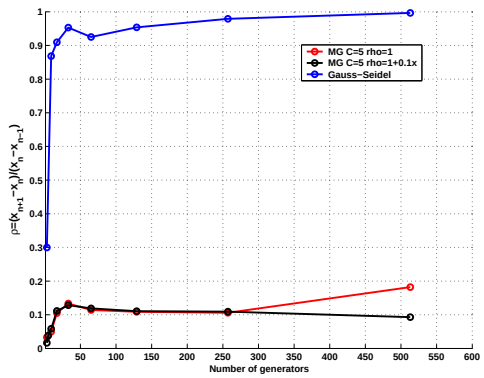


FIG. 3.1. *Plot of the convergence factor over the number of generators for the multigrid method vs. regular Gauss-Seidel method for* $\rho = 1$ *and* $\rho = 1 + 0.1x$

Figure 3.1 justifies the fact that the speed of convergence for proposed scheme does not grow with the number of generators.
The geometric rate of energy and error reduction asserted by the Theorem 3.3 is confirmed by the experiments. Indeed, Figure 3.2 shows convergence history of a $V(1,1)$-cycle vs. total number of relaxations for the $k = 64$ case.

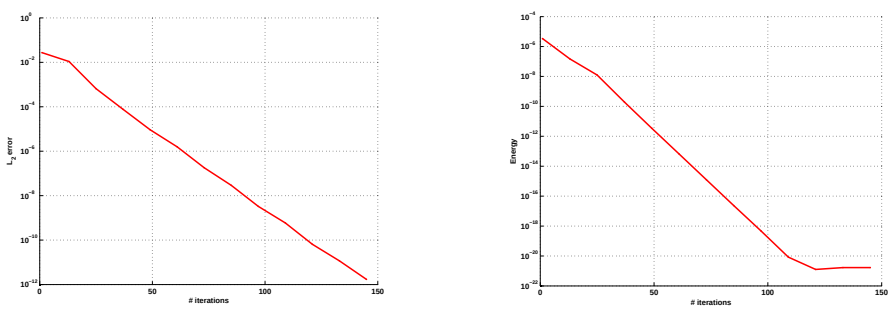


FIG. 3.2. *(a) Convergence history for* $k = 64$ *generators (log-normal scale); (b)Energy reduction for* $k = 64$ *generators (log-normal scale)*

The results for other nonlinear densities , though not shown here, comply with the theoretical conclusions reached above (see [4]). Multidimensional extensions are discussed in [5] .

4. Applications. CVTs have a rich field of applications in various areas of mathematics as well as engineering. Here we provide several examples to give a flavor of the kind of problems that benefit from the study of this concept.

Figure 4.1 shows tesselations of the sphere for different density functions and an example of mesh generated by means of CVT. The point distributions generated this way can be used for vector quantization, optimal resource allocation, image compression, optimal mesh generation and in many other applications. In many of these applications efficiency of the numerical scheme plays essential, often crucial role, so new approaches in accelerating existing numerical methods are very important.

5. Conclusion. We introduced a new energy-based multigrid method for quantization where a dynamic nonlinear preconditioning helps to take advantage of a nonlinear convex optimization setting. This is the first multigrid quantization scheme

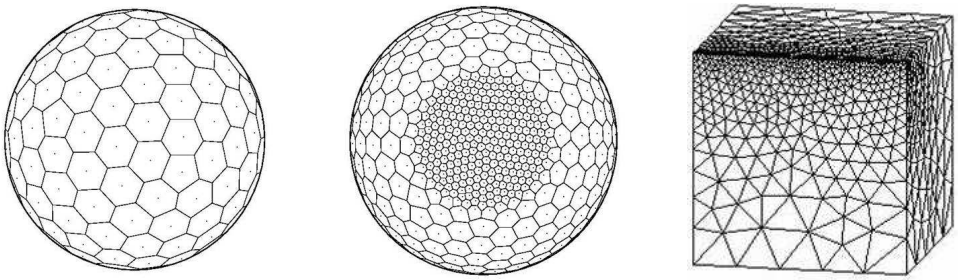


FIG. 4.1.

in literature with a rigorous proof of uniform convergence with respect to the grid size and the number of grid levels. The scheme demonstrated significant speedup comparing to the traditional Lloyd's method. More work is under way for the analysis of the multigrid scheme in higher dimensions.

6. Acknowledgement. The author would like to thank Professors Qiang Du, Lili Ju, Jinchao Xu and Ludmil Zikatanov for their suggestions and collaborations on this subject.

REFERENCES

[1] Q. Du, V. Faber and M. Gunzburger, *Centroidal Voronoi tessellations: applications and algorithms*, SIAM Review, 41, 1999, pp. 637–676.

[2] Q. Du and M. Gunzburger, *Grid Generation and Optimization Based on Centroidal Voronoi Tessellations*, to appear.

[3] Q. Du, M. Emelianenko and L. Ju, *Convergence Properties of the Lloyd Algorithm for Computing the Centroidal Voronoi Tessellations*, submitted to SIAM J. Num. An., 2004.

[4] Q. Du and M. Emelianenko, *Uniform Convergence of an Energy-based Multigrid Quantization Scheme*, preprint, 2004.

[5] Q. Du, M. Emelianenko and L. Zikatanov, *An Energy-based Multigrid Quantization Scheme in Multidimension*, in preparation, 2005.

[6] A. Gersho; *Asymptotically optimal block quantization,* IEEE Trans. Inform. Theory 25, 1979, pp. 373–380.

[7] R. Gray and D. Neuhoff; *Quantization,* IEEE Trans. Inform. Theory 44, 1998, pp. 2325–2383.

[8] L. Ju, Q. Du, M. Gunzburger; *Probablistic methods for centroidal Voronoi tessellations and their parallel implementations*, **Parallel Computing**, 28, pp.1477-1500, 2002

[9] L. Ju, M. Gunzburger and Q. Du, *Meshfree, Probabilistic Determination of Points, Support Spheres, and Connectivities for Meshless Computing*, **Computer Methods in Applied Mechanics and Engineering**, 191, pp.1349-1366, 2002

[10] Y. Koren, I. Yavneh and A. Spira, *A Multigrid Approach to the 1-D Quantization Problem*, preprint, 2003

[11] Y. Koren, I. Yavneh, *Adaptive Multiscale Redistribution for Vector Quantization*, accepted to SIAM J. Sci. Comp., 2004

[12] Y. Linde, A. Buzo, and R. Gray; *An algorithms for vector quantizer design,* IEEE Trans. Comm. 28, 1980, pp. 84–95.

[13] S. Lloyd, Least square quantization in PCM, *IEEE Trans. Infor. Theory*, **28**, 1982, pp. 129–137.

[14] J. MacQueen; *Some methods for classification and analysis of multivariate observations,* Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. I, Ed. by L. Le Cam and J. Neyman, University of California, 1967, pp. 281–297.

[15] X.-C. Tai and J. Xu, *Global and Uniform Convergence of Subspace Correction Methods for Some Convex Optimization Problems*, Math. Comp., 1998

[16] A. Trushkin; *On the design of an optimal quantizer,* IEEE Trans. Infor. Theory 39, 1993, pp. 1180–1194.

Multigrid-based preconditioner for heterogeneous, high wavenumber Helmholtz problems*†

Yogi A. Erlangga‡

Abstract

A multigrid preconditioned Krylov subspace iterative method is explained and used to solve the Helmholtz equation with high wavenumbers in a heterogeneous media. The preconditioner is based on the Helmholtz operator with additional imaginary term. This preconditioner is handled by using one multigrid iteration with standard multigrid components. Fourier analysis is used to evaluate the multigrid performance. Tests based on high wavenumber problems have been made to show the robustness of the method.

1 Introduction

For the Helmholtz equation at high wavenumbers multigrid is known to have convergence problems [5]. These problems arise from both smoothing and coarse grid correction, the two important ingredients of multigrid.

We first consider the 2D Helmholtz equation

$$\mathcal{A}x := -\left(\partial_{xx} + \partial_{yy} + (1-\alpha i)k^2\right)\phi = g(x,y), \text{ in } \Omega \subset \mathbb{R}^2 \tag{1}$$

with a constant wavenumber k, and $i = \sqrt{-1}$ the complex identity. Assume the homogeneous Dirichlet boundary conditions on $\Gamma = \partial\Omega$. In general, $0 \leq \alpha \ll 1$. For the time being, we set $\alpha = 0$, i.e., damping is not present in Ω. Using the $O(h^2)$ 5-point discretization stencil, we have

$$A_h \hat{=} \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4-(kh)^2 & -1 \\ & -1 & \end{bmatrix}, \tag{2}$$

and the eigenvalues are given as

$$\lambda_h^{l,m} = \widetilde{\lambda}_h^{l,m} - k^2 \equiv \frac{2}{h^2}(2-\cos(l\pi h) - \cos(m\pi h) - k^2. \tag{3}$$

*The research is financially supported by the Dutch Ministry of Economic Affairs under the Project BTS01044

†This manuscript is an abridged version of the author paper available online at http://ta.twi.tudelft.nl

‡PhD Student, Delft Univ. of Technology, Dept. Appl. Math. Anal., Mekelweg 8, 2628 CD Delft, The Netherlands. Correspondence: y.a.erlangga@ewi.tudelft.nl

With $\widetilde{\lambda}_h^{1,1}$ the smallest eigenvalue of the Laplacian term, the matrix A_h has all eigenvalues on the right half plane as long as $k^2 < \widetilde{\lambda}_h^{1,1}$. If $k^2 > \widetilde{\lambda}_h^{1,1}$, the eigenvalues now have positive and negative signs. In this situation, point-wise Jacobi iteration with underrelaxation does not converge. Its smoothing properties are, however, satisfactory. Therefore, the multigrid convergence deteriorates gradually with increasing k^2. As k^2 gets closer to the 6th eigenvalue, standard multigrid diverges. The Jacobi iteration now also diverges for smooth eigenfrequencies. Multigrid will only converge as long as the coarsest level is fine enough to sufficiently represent these smooth eigenfrequencies.

Another problem appears when eigenvalues undergo a sign change if grids are coarsened. This may particularly occur to eigenvalues close to the origin. If a sign change occurs the coarse grid solution does not give convergence acceleration to the finer grid problems. In [5] multigrid combined with Krylov subspace iteration methods has been used as a remedy. For multigrid, GMRES is used as a smoother and as a cure for problematic coarse grid correction.

In this paper, we discuss another way to efficiently solve the Helmholtz equation. In our method, standard multigrid components can still be used. The most distinguished ingredient is the use of a properly chosen operator as a preconditioner favorable for multigrid and Krylov subspace acceleration. Similar operators have been proposed in [2] and [8], and [7], where we introduced a complex shift to the Laplace operator. The operator introduced here in §2 can be regarded as a further generalization of such operators. For Krylov subspace iteration method, we use Bi-CGSTAB [12]. From our experience, Bi-CGSTAB is found to be more efficient for Helmholtz problems than any other Krylov subspace methods (e.q. GMRES or QMR) in combination with a complex Helmholtz preconditioner.

2 Shifted Laplace preconditioner

To solve (1) iteratively we consider an equivalent linear system

$$AM^{-1}\psi = g, \ \psi = M\phi, \tag{4}$$

where $M \in \mathbb{C}^{N\times N}$ is a preconditioning matrix. The matrix M is obtained from the operator

$$\mathcal{M} := -\partial_{xx} - \partial_{yy} - (\beta_1 - \beta_2 i)k^2, \tag{5}$$

with $\beta_1, \beta_2 \in \mathbb{R}$ free parameters to be chosen. The operator $\mathcal{M}$ can be regarded as a shifted Laplace operator. In [7] we optimize β_1, β_2 such that the condition number of AM^{-1} is minimal, with M positive definite. We find that $(\beta_1, \beta_2) = (0, 1)$ satisfies this condition. In this paper we remove such a constraint and evaluate more general options for (β_1, β_2). In particular, we investigate $(\beta_1, \beta_2) = (1, 1)$ and $(\beta_1, \beta_2) = (1, 0.5)$.

In the construction of M we set the same boundary conditions for (5) as for (1). Furthermore, we are flexible in choosing discretization method for (5). In the present paper, we implement the $O(h^2)$ 5-point discretization stencil, the same as for (1). (First tests with 4th order schemes give similar convergence.)

One way to understand the effect of preconditioning to the convergence of an iterative method is by investigating the spectral properties of the preconditioned system. This is

discussed in §3. To simplify the analysis, we consider 2D Helmholtz problems with Dirichlet boundary conditions in a square domain.

We denote $M_{0,1}$ for a preconditioning matrix obtained from operator $\mathcal{M}$ with $(\beta_1, \beta_2) = (0, 1)$. Using this convention, we denote $M_{1,1}$ and $M_{1,0.5}$ for the two other options.

3 Multigrid for the preconditioner

We consider a geometric multigrid with standard multigrid coarsening. On the finest grid, the 5-point discretization is used. We denote the preconditioning equation on the finest grid as $M_h \phi_h = \psi_h$. The relevant coarse grid operators are defined by $M_{2h}, M_{4h}, \ldots,$ and are constructed using the Galerkin coarse grid approximation, namely,

$$M_{2h} := R_h^{2h} M_h P_{2h}^h, \;\; M_{4h} := R_{2h}^{4h} M_{2h} P_{4h}^{2h}, \;\; \ldots, \tag{6}$$

with R and P, the restriction and prolongation operator, respectively. For the prolongation we use an operator-dependent interpolation, proposed in [14]. This prolongation operator is suitable for problems with inhomogeneity in the domain. We slightly modify this operator to make it suitable for complex-valued matrices.

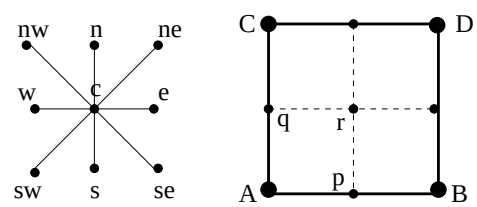


Figure 1: Left: Nine-point stencil with numbering, Right: coarse grid cell (upper case) and four fine grid cells (lower case)

Consider the nine-point cell as depicted in Figure 1. The coarse grids are shown with upper case letters while the finer grids are with the lower case. An operator element m_p^w, for example, denotes the west element of operator M_h at point p on the fine grid. An interpolation weight is denoted bys w. Using these notations, we define the fine grid correction quantity e as follows.

- for fine grid point p: $e_{h,p} = w_A e_{H,A} + w_B e_{H,B}$

$$w_A = \min(1, \max(0, w_w)); w_B = \min(1, \max(0, w_e)), \tag{7}$$

where

$$d_w = \max(|m_p^{sw} + m_p^w + m_p^{nw}|, |m_p^{sw}|, |m_p^{nw}|) \tag{8}$$

$$d_e = \max(|m_p^{se} + m_p^s + m_p^{se}|, |m_p^{se}|, |m_p^{ne}|) \tag{9}$$

$$w_w = \frac{d_w}{d_w + d_e}, \; w_e = \frac{d_e}{d_w + d_e} \tag{10}$$

- for fine grid point q: $e_{h,q} = w_A e_{H,A} + w_C e_{H,C}$

$$w_A = \min(1, \max(0, w_s)); w_C = \min(1, \max(0, w_n)), \tag{11}$$

where

$$\begin{aligned} d_n &= \max(|m_q^{nw} + m_q^n + m_q^{ne}|, |m_q^{nw}|, |m_q^{ne}|) && (12)\\ d_s &= \max(|m_q^{sw} + m_q^s + m_q^{se}|, |m_q^{sw}|, |m_q^{se}|) && (13)\\ w_s &= \frac{d_s}{d_s + d_n},\ w_n = \frac{d_n}{d_s + d_n} && (14) \end{aligned}$$

- for fine grid points that are identical with coarse grid points:

$$e_h(A) = e_{2h}(A) \tag{15}$$

- for fine grid points r: $e_h(r)$ is determined such that

$$M_h P_{2h}^h e_{2h} = 0. \tag{16}$$

For the restriction we use the standard full weighting operator. This is somewhat uncommon as one usually takes the adjoint of the prolongation operator for the restriction operator. The combination we favor here, however, leads to a robust multigrid method for Helmholtz problems with heterogeneities and high wavenumbers, as will be shown in §4. See also [1] and [4] for this strategy.

For multigrid as a preconditioner for Krylov subspace method, we analyze the performance using spectral properties obtained by Fourier analysis [10, 13]. We consider a regular splitting of M_h, representing one multigrid iteration,

$$C_h \phi_h^{l+1} = (C_h - M_h)\phi_h^l + \psi_h, \tag{17}$$

and hence,

$$\phi_h^{l+1} = (I_h - C_h^{-1} M_h)\phi_h^l + C_h^{-1}\psi_h, \tag{18}$$

with C_h^{-1} an approximation of M_h^{-1}.

Consider the case ϕ in (4) is solved exactly, denoted by ϕ_h. The error after the j-th iteration is determined as $\epsilon_h^j = \phi_h^j - \phi_h$. In the two-grid formulation, this error can be written as

$$\epsilon_h^{j+1} = T_h^{2h} e_h^j, \text{ with } T_h^{2h} = S^{\nu_1} K_h^{2h} S^{\nu_2} \text{ and } K_h^{2h} = I_h - P_{2h}^h M_{2h}^{-1} R_h^{2h} M_h. \tag{19}$$

Thus, considering (18), T_h^{2h} in (19) can also be defined as

$$T_h^{2h} = (I_h - C_h^{-1} M_h). \tag{20}$$

The matrix T_h^{2h} can be formed into a block matrix version $\widetilde{T}_h^{2h}$ using the relation

$$\widetilde{T}_h^{2h} = U_h T_h^{2h} U_h^{-1}, \tag{21}$$

where U_h a unitary matrix with four consecutive rows defined by orthogonal eigenvectors related to

$$v_h^{l,m} = \sin l\pi x \sin m\pi y, \tag{22}$$

with l, m corresponding with $\lambda_h^{l,m} = 0$. Substituting (20) into (21) we have

$$\begin{aligned} \widetilde{T}_h^{2h} &= I_h - U_h C_h^{-1} M_h U_h^{-1} \\ &= I_h - U_h C_h^{-1} U_h^{-1} U_h M_h U_h^{-1} \\ &= I_h - \widetilde{C}_h^{-1} \widetilde{M}_h. \end{aligned} \tag{23}$$

where $\widetilde{C}_h^{-1} \equiv U_h C_h^{-1} U_h^{-1}$ and $\widetilde{M}_h \equiv U_h M_h U_h^{-1}$. From equation (23),

$$\widetilde{C}_h^{-1} = (I_h - \widetilde{T}_h^{2h}) \widetilde{M}_h^{-1}. \tag{24}$$

Thus,

$$\widetilde{A}_h \widetilde{C}_h^{-1} = \widetilde{A}_h (I_h - \widetilde{T}_h^{2h}) \widetilde{M}_h^{-1}. \tag{25}$$

We present the spectra of $A_h C_h^{-1}$ where C_h^{-1} is a multigrid approximation of M_h^{-1}, as shown in Figure 2 and Figure 3. These spectra are compared with spectra where M_h^{-1} is computed exactly (solid line). The wavenumber k equals 40 with $h = 1/64$. In case $M_{1,1}$ the spectrum is shown in Figure 2. The smoother is point-wise Jacobi with underrelaxation (ω-JAC) with $\omega = 0.7$, denoted by 0.7-JAC. The left figure shows the spectra for which only one 0.7-JAC smoothing applied. The spectra of $A_h M_h^{-1}$ is not well approximated by $A_h C_h^{-1}$. A different situation appears if one pre- and one postsmoothing are performed; the spectrum of $A_h C^{-1}$ is now well within the circle. For the two cases, the two-grid convergence factor is 0.7 and 0.45, respectively. It appears that one multigrid iteration with one pre- and one postsmoothing is sufficient to approximate M_h^{-1}.

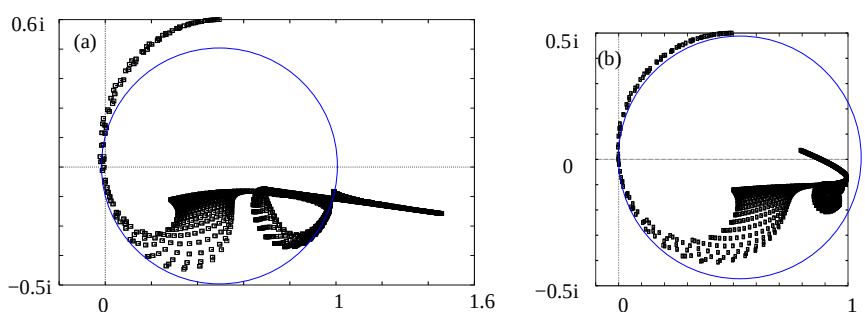


Figure 2: Spectra of preconditioned system with one two-grid iteration to approximate $M_{1,1}$. (a) one 0.7-JAC, (b) two 0.7-JAC. Spectra for exact inversion of $M_{1,1}$ is drawn with solid line.

A similar result is also observed for $M_{1,0.5}$, as shown in Figure 3. In this case we have used 0.5-JAC as the smoother while the remaining multigrid components are the same. The spectrum of $A_h C_h^{-1}$ is a good approximation to $A_h M_h^{-1}$ if $\nu_1 + \nu_2 = 2$ (one pre- and one postsmoothing).

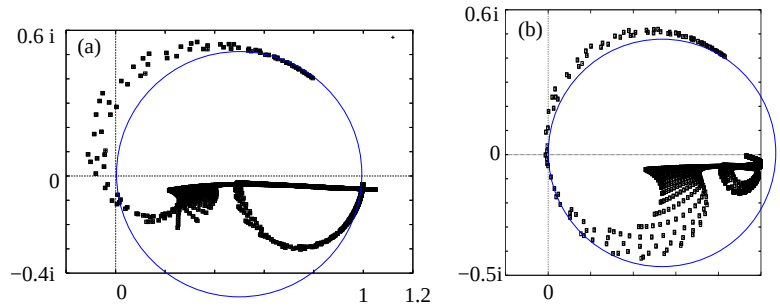


Figure 3: Spectra of preconditioned system with one two-grid iteration to approximate $M_{1,0.5}$. (a) one 0.5-JAC, (b) two 0.5-JAC. Spectra for exact inversion of $M_{1,0.5}$ is drawn with solid line.

4 Numerical result

In this section, various Helmholtz problems are presented with an increasing difficulty level. The problems are solved using preconditioned Bi-CGSTAB with (5) as the preconditioner. For the numerical tests, we use the realistic boundary condition based the second order absorbing boundary condition [6]:

$$\mathcal{A}_\Gamma \phi := \frac{\partial \phi}{\partial \nu} - ik\phi - \frac{i}{2k}\frac{\partial^2 \phi}{\partial \tau^2} = 0 \text{ on } \Gamma, \tag{26}$$

with μ and τ the outward normal and tangential direction, respectively.

We show the numerical performance for three choices: $M_{0,1}$, $M_{1,1}$, and $M_{1,0.5}$. One multigrid iteration with one pre- and one postsmoothing is used to approximately computed M^{-1}. In this context, F(1,1)-cycle is used.

For prolongation, the matrix-dependent interpolation as in (7)-(16) is used, with full weighting as the restriction operator. The coarse grid matrix is constructed based on Galerkin coarse grid discretization. Different values of ω are used for different M: 0.8-JAC for $M_{0,1}$, 0.7-JAC for $M_{1,1}$ and 0.5-JAC for $M_{1,0.5}$. These choices resulted in robust convergence of the set of Helmholtz problems tested.

4.1 Constant wavenumber

We first consider a relatively easy problem: the constant k Helmholtz problem. We solve the problem in a rectangular domain $\Omega = (0,1)^2$. A point source is located at the center of the domain. The convergence in terms of number of iterations to reduce the initial residual by 7 orders plus corresponding CPU time is presented in Table 1.

From these results it appears that the fastest convergence is obtained for $M_{1,0.5}$. Compared to $M_{0,1}$ proposed in [7], the convergence using $M_{1,0.5}$ is improved by factor of two for low wavenumbers. The difference becomes more pronounced for high wavenumbers.

In all wavenumbers computed, multigrid has been used without any grid level limitation. The coarsest grid level used contains only 3 grid points in the interior.

Table 1: Number of preconditioned Bi-CGSTAB iteration and CPU time (in brackets) to reduce the initial residual by 7 orders

	k :					
(β_1, β_2)	40	50	100	150	200	500
$M_{0,1}$	57 (0.44)	73 (0.92)	126 (7.7)	188 (28.5)	–	–
$M_{1,1}$	36 (0.30)	39 (0.51)	74 (4.5)	90 (13.9)	114 (30.8)	291 (515)
$M_{1,0.5}$	26 (0.21)	31 (0.40)	52 (3.3)	73 (10.8)	92 (25.4)	250 (425)

4.2 The wedge model

A problem of intermediate difficulty is the wedge problem adopted from [9]. The domain is defined to be rectangular of dimension $600 \times 1000\ m^2$. In this case we use a source at the center just below the upper surface with frequency, $f = kc/(2\pi)L$, varying from 10 to 60 Hz. In a unit domain, this frequency corresponds to a scaled wavenumber $\bar{k} = 20$ for $f = 10$ Hz, and $\bar{k} = 240$ for $f = 60$ Hz. Figure 4 shows the geometry and examples of the solution for $f = 30$ Hz and 50 Hz. The convergence performance in terms of number of iterations to reduce the initial residual by 7 orders and corresponding CPU time is presented in Table 2.

From Table 2 we can also conclude that with $M_{1,0.5}$ a faster convergence has been achieved as compared to $M_{0,1}$. The speed up is also more pronounced for higher frequencies (with almost by factor of four for $f = 60$ Hz).

4.3 The Marmousi problem

This problem is part of the full Marmousi problem which mimics subsurface geology [3]. The domain is a rectangle of dimension $6000 \times 1600\ m^2$. A source is located at the center just

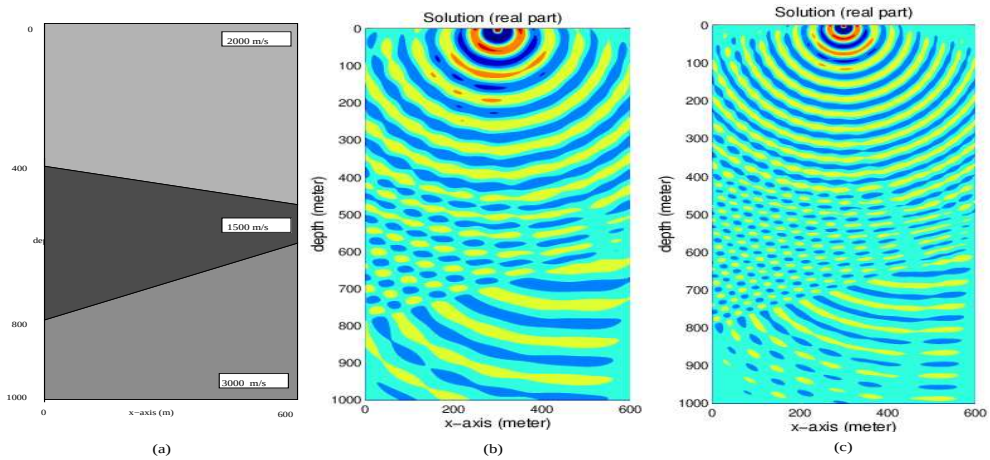


Figure 4: Wedge problem: (a) Problem geometry with velocity profile indicated, (b) Real part of numerical solution at 30 Hz, (c) Real part of numerical solution at 50 Hz.

Table 2: Bi-CGSTAB convergence for the wedge problem for $M_{0,1}$, $M_{1,1}$, and $M_{1,0.5}$. The convergence data is shown for number of Bi-CGSTAB iterations and CPU time in seconds (in brackets).

f (Hz)	Grid	$M_{0,1}$	$M_{1,1}$	$M_{1,0.5}$
10	75 × 125	52 (1.2)	30 (0.67)	19 (0.42)
20	149 × 249	91 (8.8)	45 (4.5)	27 (2.8)
30	232 × 386	128 (30.6)	64 (15.8)	37 (9.4)
40	301 × 501	161 (66.1)	80 (33.5)	49 (20.8)
50	376 × 626	205 (134.5)	98 (65.5)	58 (38.7)
60	481 × 801	232 (247.3)	118 (127.6)	66 (71.9)

below the upper surface. The frequency varies from 1 to 30 Hz. The model and solutions (with damping, will be explained in the next subsection) are shown in Figure 5 (with Figure 5c is for a case with $\alpha \neq 0$ considered in §4.4). The convergence results are given in Table 3, again in terms of number of iterations to reduce the initial residual by 7 orders and corresponding CPU time

The same conclusion as for the previous test cases can be drawn from the Marmousi problem: fastest convergence is obtained with $M_{1,0.5}$ for any frequency. Furthermore, even though the theoretical analysis is based on constant wavenumber cases, the results from heterogeneous problem are consistent with the analysis.

4.4 Problem with damping

As a final problem in this paper, we consider the previous three problems with small damping, i.e. $\alpha \neq 0$. Damping is physically present in real life problems and may vary in space. In our presentation, we introduce a constant, small damping upto 5% of the local wavenumber.

The results are shown in Tables 4, 5 and 6 for the constant k, the wedge and the Marmousi

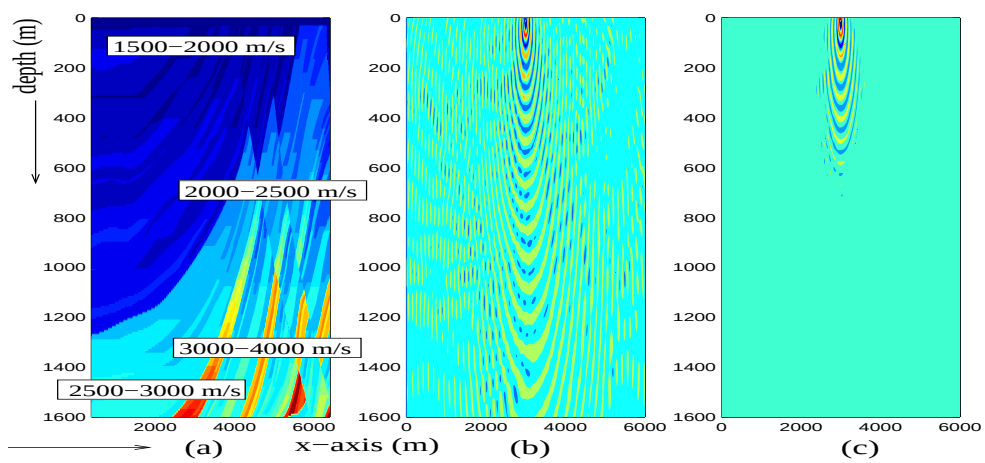


Figure 5: Marmousi problem (not to scale). a) Velocity distribution in meter/s, b) real part of the solution for $f = 20$ Hz, no damping, c) real part of the solution for $f = 20$ Hz, 2.5 % damping.

Table 3: Bi-CGSTAB convergence for the Marmousi problem with multigrid based on $M_{0,1}$, $M_{1,1}$ and $M_{1,0.5}$. The convergence data are shown for number of Bi-CGSTAB iterations and CPU time in seconds (in brackets).

f (Hz)	Grid	$M_{0,1}$	$M_{1,1}$	$M_{1,0.5}$
1	751 × 201	74 (37.5)	54 (27.6)	38 (19.7)
10	751 × 201	180 (89.2)	84 (42.4)	47 (24.2)
20	1501 × 401	414 (832.3)	168 (308.7)	104 (212.1)
30	2001 × 534	458 (1724.8)	211 (799.4)	136 (519.4)

problem, respectively. In general the presence of (small) damping further accelerates the convergence. With 5% damping, for example, a convergence improvement by factor of 3 can be achieved for the Marmousi problem. Interestingly, with damping the convergence is also nearly independent of k as seen for the Marmousi problem. Our analysis (not presented in this paper) shows that for an increasing value of the damping parameter the convergence of the Krylov subspace methods is independent of k (and hence, of the frequency).

Table 4: Constant wave number (problem in §4.1) with damping. Number of prec. Bi-CGSTAB iterations and CPU time in seconds (in brackets) to reduce the initial residual by 7 orders are shown. Damping parameter α is varied in the Helmholtz problem.

		k :					
	α	40	50	100	150	200	500
$M_{0,1}$	2.5%	48 (0.38)	61 (0.77)	93 (5.6)	121 (18.5)	–	–
	5.0%	45 (0.35)	55 (0.70)	75 (4.7)	97 (14.9)	–	–
$M_{1,1}$	2.5%	33 (0.27)	37 (0.48)	51 (3.2)	61 (9.6)	74 (20.2)	125 (227)
	5.0%	28 (0.24)	30 (0.39)	41 (2.6)	49 (7.5)	56 (15.5)	95 (174)
$M_{1,0.5}$	2.5%	24 (0.20)	26 (0.35)	39 (2.5)	47 (7.3)	57 (15.2)	91 (164)
	5.0%	21 (0.18)	23 (0.32)	32 (2.1)	37 (5.8)	44 (11.9)	64 (115)

A uniform damping may not be realistic in practice. Extension to local damping variation is, however, easily done and the same behavior of convergence can also be expected. This is part of future investigations.

5 Conclusion

In this paper a preconditioner for the Helmhotz equation has been proposed and discussed. The preconditioner is based on the Helmholtz operator plus an additional imaginary term. Standard multigrid is used to approximately solve the preconditioner. With one multigrid iteration, one pre- and one postsmoothing are sufficient to approximate the inversion of the preconditioner operator. Even though the analysis is based on a homogeneous problem, the outcome of the analysis can be observed also for heteregeneous problems. Employing the point-wise Jacobi smoother, the method is parallelizable algorithm.

Table 5: The wedge problem (problem in §4.2) with damping. Bi-CGSTAB convergence for the three multigrid based M-preconditioners are compared. Number of Bi-CGSTAB iterations and CPU time in seconds (in brackets).

f (Hz)	Grid	damping	$M_{0,1}$	$M_{1,1}$	$M_{1,0.5}$
10	75 × 125	2.5%	48 (1.1)	27 (0.62)	17 (0.39)
		5.0%	42 (0.91)	25 (0.57)	16 (0.38)
20	149 × 249	2.5%	75 (7.2)	39 (4.0)	23 (2.4)
		5.0%	65 (6.3)	35 (3.5)	20 (2.1)
30	232 × 386	2.5%	94 (22.8)	49 (12.3)	29 (7.5)
		5.0%	86 (21.0)	42 (10.7)	25 (6.6)
40	301 × 501	2.5%	116 (48.0)	60 (25.4)	35 (15.2)
		5.0%	91 (37.9)	46 (19.8)	28 (12.4)
50	376 × 626	2.5%	135 (89.0)	67 (45.5)	37 (24.8)
		5.0%	99 (66.5)	54 (37.1)	32 (22.0)
60	481 × 801	2.5%	147 (159.1)	74 (81.1)	42 (47.1)
		5.0%	110 (119.6)	58 (64.5)	32 (36.7)

Table 6: The Marmousi problem (problem in §4.3) with damping. Bi-CGSTAB convergence for the three multigrid based M-preconditioners are shown. Number of Bi-CGSTAB iterations and CPU time in seconds (in brackets).

f (Hz)	Grid	damping	$M_{0,1}$	$M_{1,1}$	$M_{1,0.5}$
1	751 × 201	2.5%	67 (34.1)	55 (28.2)	32 (17.0)
		5.0%	64 (32.5)	53 (27.3)	31 (16.5)
10	751 × 201	2.5%	119 (59.4)	59 (30.1)	33 (17.5)
		5.0%	96 (48.3)	48 (24.8)	28 (15.0)
20	1501 × 401	2.5%	203 (410.8)	88 (179.9)	55 (115.3)
		5.0%	145 (294.7)	64 (133.4)	37 (79.3)
30	2001 × 534	2.5%	197 (745.6)	94 (361.1)	58 (226.8)
		5.0%	119 (455.3)	61 (238.4)	38 (151.9)

References

[1] R.E. Alcouffe, A. Brandt, J.E. Dendy Jr. and J.W. Painter, The multi-grid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Comput.* 2: 430-454, 1981.

[2] A. Bayliss, C.I. Goldstein and E. Turkel, An iterative method for Helmholtz equation, *J. Comput. Phys.,* 49: 443-457, 1983.

[3] A. Bourgeois, M. Bourget, P. Lailly, M. Poulet, P. Ricarte and R. Versteeg, Marmousi, model and data. *Marmousi Experience EAEG,* 5-16, 1991.

[4] J.E. Dendy Jr., Blackbox multigrid for nonsymmetric problems, *Appl. Math. Comput.* 13: 261-283, 1983.

[5] H. R. Elman, O. G. Ernst and D. P. O'Leary , A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations, *SIAM J. Sci. Comput.* 23: 1291-1315, 2001.

[6] B. Engquist and A. Majda, Absorbing boundary conditions for the numerical simulation of waves, *Math. Comp.,* 31: 629-651, 1977.

[7] Y.A. Erlangga, C. Vuik and C.W. Oosterlee, On a class of preconditioners for the Helmholtz equation, *Applied Num. Math.* 50: 409-425, 2004.

[8] A.L. Laird, M.B. Giles, *Preconditioned iterative solution of the 2D Helmholtz equation,* Report, no 02/12, Oxford Comp. Lab. Oxford, UK, 2002.

[9] R.E. Plessix, W.A. Mulder, Separation-of-variables as a preconditioner for an iterative Helmholtz solver, *Appl. Num. Math.,* 44: 385–400, 2003.

[10] U. Trottenberg, C.W. Oosterlee and A. Schüller, *Multigrid*, Academic Press, London, 2001.

[11] E. Turkel, Numerical Difficulties Solving Time Harmonic Equations, *In:* Multiscale Computational Methods in Chemistry and Physics, A. Brandt, et. al. (eds), IOS Press, Ohmsha, 319-337, 2001.

[12] H.A. van der Vorst, Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Comput.* 13: 631-645, 1992.

[13] R. Wienands, C.W. Oosterlee and T. Washio, Fourier Analysis of GMRES(m) Preconditioned by Multigrid, *SIAM J. Sci. Comput.* 22: 582-603, 2000.

[14] P.M. de Zeeuw, Matrix-dependent prolongations and restrictions in a blackbox multigrid solver, *J. Comp. Appl. Math.* 33: 1 - 27, 1990.

A MULTIGRID METHOD FOR ANISOTROPIC PDE'S IN ELASTIC IMAGE REGISTRATION

LARS HÖMKE*

Abstract. This paper deals with the solution of a nonlinear ill-conditioned inverse problem arising in digital image registration. In the first part of the paper, we define the problem as the minimization of a regularized non-linear least-squares functional, which measures the image difference and smoothness of the transformation. We study inexact Newton methods for solving this problem, i.e. we linearize the functional around a current approximation and replace the Hessian by a suitable operator in order to obtain well-posed subproblems in each step of the iteration.
These anisotropic subproblems are solved using a multigrid solver. Due to the anisotropy in the coefficients of the underlying equations, standard multigrid solvers suffer from poor convergence rates. We discuss modifications to the multigrid components, specifically to the smoothing procedure, the interpolation and the coarse grid correction. Numerical results that demonstrate the improvements obtained with these new components are given.

1. Introduction. In the past few years PDE-based methods have become very popular in image processing. This is mainly due to the fact that these methods are very flexible and powerful, and that with the increase in computational power the solution of the resulting large systems of equations has become feasible even on "normal" PC's.

Image registration is a very active field of research, with applications in medical image analysis, target tracking, data fusion, recognition and satellite imagery. In the field of modern brain research image registration allows for, e.g. the combination of different imaging modalities, such as (f)MRI, PET and histological sections and the comparison of individual brains in a common reference space.

Let us first state the image registration problem in general terms. Given are two images, a template T and a reference R. We assume that T and R can be represented by compactly supported functions

$$T, R : \Omega \subset \mathbb{R}^m \to \mathbb{R}. \tag{1.1}$$

Furthermore we assume that T is distorted by an invertible transformation Φ^{-1}. We search for the transformation In image processing one mostly finds $m = 2$ (images) or $m = 3$ (volume data sets).

$$\Phi(u)(\cdot) : \ \mathbb{R}^m \ \to \ \mathbb{R}^m, \ \Phi(u)(x) : \ x \mapsto x - u(x), \ x \in \Omega. \tag{1.2}$$

Problem 1.1 (Image registration problem). *Find $u(x)$ such that the energy functional*

$$E_\alpha(u) := \mathcal{D}[T, R, \Omega; u(x)] + \alpha\mathcal{R}[\Omega; u(x)] : \mathbb{R}^{2m} \to \mathbb{R} \tag{1.3}$$

is minimal with $\alpha \in \mathbb{R}^+$. $\mathcal{D}$ measures the disparity between $T(x - u(x))$ and $R(x)$ on the image domain Ω, and $\mathcal{R}$ imposes (smoothness) constraints on the solution.

Generally this is an ill-posed problem, yet it becomes well-posed for some $\alpha > 0$. Several choices exist for $\mathcal{D}$ as well as for $\mathcal{R}$. Here we use the integral of squared differences (SSD)

$$\mathcal{D}^{\text{SSD}} := \frac{1}{2}\int_\Omega (T(x - u(x)) - R(x))^2 \mathrm{d}x \tag{1.4}$$

*Research Center Jülich, Germany, Institute of Medicine, hoemke@am.uni-duesseldorf.de

to measure the dissimilarity. The SSD can be used when identical structures are encoded by the same values in both images. If that is not the case other distance measures that relate the two images have to be used, e.g. mutual information [7]. For the regularizer $\mathcal{R}$ we chose $\mathcal{R}(u) = l[u,u]$ with the bilinear form

$$\ell[u,v] = \int_\Omega \Big(\lambda \mathrm{div}(u)\mathrm{div}(v) + 2\mu \sum_{i,j=1}^{m} e_{i,j}(u)e_{i,j}(v)\Big)\mathrm{d}x,$$

where

$$e_{i,j}(u) = \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}),$$

which is lend from the theory of linear elasticity for homogeneous and isotropic materials. It measures the elastic potential of the deformation u. The parameters $\lambda \geq 0$ and $\mu > 0$, the so-called Lamé constants, define the elastic properties of the underlying material. The bilinear form ℓ can be written as $\ell[u,v] = \langle Lu, v\rangle$, where the canonical linear mapping L is given by the so-called *Navier-Lamé operator*

$$Lu = -\mu\Delta u - (\lambda+\mu)\nabla(\nabla u).$$

For our minimization problem we end up with a regularized nonlinear least squares functional that we tackle with inexact Newton type methods, specifically a Gauss-Newton method with a trust region approach. The functional is linearized around a current approximation which defines an iteration, the so called outer iteration. The normal equations that correspond to the linear anisotropic subproblems that arise in every step of the outer iteration are solved iteratively with a fast and efficient multigrid solver. This iteration is called inner iteration. At some points we restrict ourselves to presentation of the 2d-case ($m = 2$) due to space restrictions, but all techniques can be easily extended and are also used in 3d.

The rest of the paper is organized as follows. The equations for the Newton type method are derived in section 2. The discretization of the images and the equations is discussed in section 3. In section 4 a fast and efficient multigrid solver for the linear subproblems is proposed. The paper concludes with some numerical results in section 5 and a final discussion.

2. A Newton type method for image registration. In this section we present a Gauss-Newton method for solving the minimization problem. Since the functional is highly nonlinear a sequence of subproblems has to be solved. Each of this subproblems should have the following important features:

- it is well-posed and
- can be solved efficiently.

To this end E_α is linearized around a current approximation. The Taylor expansion of $E_\alpha(u)$ around $u^{(k)}$ is given by:

$$E_\alpha(u^{(k)}+v) \approx E_\alpha(u^{(k)}) + \langle J_{E_\alpha}(u^{(k)}), v\rangle + \frac{1}{2}\langle Hess_{E_\alpha}(u^{(k)})v, v\rangle,$$

with $v = (u - u^{(k)})$ and $\langle\cdot,\cdot\rangle$ denoting the Euclidian scalar product. Using the abbreviation

$$T^{(k)}(x) = T(x - u^{(k)}(x))$$

for the transformed template image, the Jacobian and the Hessian of E_α at $u^{(k)}$ are given by

$$\begin{aligned} J_{E_\alpha}(u^{(k)}) &= J_\theta^t(u^{(k)})\theta(u^{(k)}) + \alpha L u^{(k)}, \\ Hess_{E_\alpha}(u^{(k)}) &= J_\theta^t(u^{(k)})J_\theta(u^{(k)}) + S(u^{(k)}) + \alpha L, \end{aligned}$$

where

$$\theta(u^{(k)}) = T^{(k)}(x) - R(x)$$

is the difference between reference and transformed template. Since $J_\phi(u) = -I_m$, the $m \times m$ identity, we have

$$J_\theta^t(u^{(k)}) = J_T(\phi(u^{(k)})) \cdot J_\phi(u^{(k)}) = -\begin{pmatrix} \frac{\partial T^{(k)}(x)}{\partial x_1} \\ \vdots \\ \frac{\partial T^{(k)}(x)}{\partial x_m} \end{pmatrix}$$

and

$$J_\theta^t(u^{(k)})J_\theta(u^{(k)}) = \begin{pmatrix} \frac{\partial^2 T^{(k)}(x)}{\partial x_1^2} & \cdots & \frac{\partial^2 T^{(k)}(x)}{\partial x_1 \partial x_m} \\ \vdots & & \vdots \\ \frac{\partial^2 T^{(k)}(x)}{\partial x_1 \partial x_m} & \cdots & \frac{\partial^2 T^{(k)}(x)}{\partial x_m^2} \end{pmatrix}.$$

Here,

$$S(u^{(k)}) = \theta(u^{(k)})\theta''(u^{(k)}) = \theta(u^{(k)})\nabla^2 T^{(k)}(x_1, x_2)$$

constitutes the nonlinear part of $Hess_{E_\alpha}(u^{(k)})$, which becomes small for small θ, i.e. well matched images.
When we replace $S(u^{(k)})$ by a spd operator βB, $\beta \in \mathbb{R}_0^+$, we obtain the unconstrained minimization problem

$$\min_v \Big\{ \big\langle J_{E_\alpha}(u^{(k)}), v \big\rangle + \frac{1}{2}\big\langle (J_\theta^t(u^{(k)})J_\theta(u^{(k)}) + \alpha L + \beta B)v, v \big\rangle \Big\}. \tag{2.1}$$

Using the method of Lagrange multipliers it follows, that the constrained minimization problem

$$\min_v \Big\{ \big\langle J_{E_\alpha}(u^{(k)}), v \big\rangle + \frac{1}{2}\big\langle (J_\theta^t(u^{(k)})J_\theta(u^{(k)}) + \alpha L)v, v \big\rangle \Big\}, \quad \text{s.t.} ||v||_B \leq \eta_k, \tag{2.2}$$

is equivalent to the unconstrained minimization problem (2.1), where β is a Lagrange multiplier. This can be interpreted as a Gauss-Newton method with an additional trust region strategy. This combination is often called a Levenberg-Marquardt method. Solutions v are only accepted if they lie in the so-called trust region with radius η_k. The parameter η_k is determined with the Armijo-Golstein rule (c.f. [3],[4]). Here $B := L$ is chosen and the normal equation that has to be solved in each step of the iteration

$$u^{(k+1)} = u^{(k)} + v \tag{2.3}$$

is

$$\langle \underbrace{J_{E_\alpha}(u^{(k)})}_{:=-f(u^{(k)})}, v\rangle + \frac{1}{2}\langle (\underbrace{J_\theta^t(u^{(k)})J_\theta(u^{(k)}) + (\overbrace{\alpha+\beta}^{:=\tilde{\alpha}})L}_{:=M})v, v\rangle, \quad \forall v \in \mathcal{X}. \tag{2.4}$$

With $\mathcal{X} := \overbrace{H(\Omega)_0^1 \times \ldots \times H(\Omega)_0^1}^{n\text{-times}}$ the elliptic operator L, and consequently also M, is spd and a unique minimizer u^* of (2.1) exists. Here the space $H^1(\Omega)$ is the Sobolev space of functions in $L^2(\Omega)$ with weak derivatives in $L^2(\Omega)$. $H_0^1(\Omega)$ is the subspace of $H^1(\Omega)$ consisting of the functions with zero boundary values.

3. Discretization. We regard the images as given on a domain $\Omega \subset \mathbb{R}^m$. For $m = 2$ that is usually the unit square $[0,1]^2$ and for $m = 3$ the unit cube. The image data is given on a regular equidistant grid,

$$\mathcal{G}_h^m := \left\{(x_{1,i_1}, \ldots, x_{m,i_m}) |\, x_{l,i_j} = i_j \cdot h,\ i_j \in \mathbb{N},\ 0 \le i_j \le 1/h\right\} \tag{3.1}$$

with one pixel/voxel of a given value at each grid point. Hence the discretization of Ω is given by

$$\begin{aligned} \Omega_h^m &= \mathcal{G}_h^m \cap \Omega^m, \\ \partial\Omega_h^m &= \mathcal{G}_h^m \cap \partial\Omega^m. \end{aligned}$$

The grid $\mathcal{G}_h^m$ is cell centered. For the sake of simplicity of the presentation we assume that we have $i_j = 0, \ldots, 2^l$, $j = 1, \ldots, m$, $l \in \mathbb{N}$. The discretization step h is then given by $1/2^l$ and we can easily define a series of grids

$$(\Omega_{h_j}^m)_{j=1}^l = \left\{\Omega_{h_1}^m, \ldots, \Omega_{h_l}^m\right\}, \quad h_i = 1/2^i$$

where $\Omega_{h_1}^m$ corresponds to the coarsest grid with only one inner point.

The partial derivatives in L are approximated by central finite differences. The derivatives of the template image T are computed likewise. For $m = 2$ that results in the following discrete operator M_h:

$$\left.\begin{aligned} &\left(\frac{\tilde{\alpha}}{h^2}\begin{bmatrix} & -\mu & \\ -2(\mu+\lambda) & (6\mu+2\lambda) & -(2\mu+\lambda) \\ & -\mu & \end{bmatrix} + T_xT_x\right)u_1 \\ &+ \left(\frac{\tilde{\alpha}(\mu+\lambda)}{4h^2}\begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix} + T_xT_y\right)u_2 \\ &\left(\frac{\tilde{\alpha}(\mu+\lambda)}{4h^2}\begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix} + T_yT_x\right)u_1 \\ &+ \left(\frac{\tilde{\alpha}}{h^2}\begin{bmatrix} & -(2\mu+\lambda) & \\ -\mu & (6\mu+2\lambda) & -\mu \\ & -(2\mu+\lambda) & \end{bmatrix} + T_yT_y\right)u_2 \end{aligned}\right\} = (\tilde{\alpha}L_h + J_{\theta_h}^t J_{\theta_h})u \tag{3.2}$$

Note that $J_{\theta_h}^t J_{\theta_h} \neq 0$ only in regions where T changes locally, e.g. at edges. This leads to considerable jumps in the coefficients at these location, especially when $\tilde{\alpha}/h^2 \ll T_x, T_y$.

Algorithm 1 multigrid algorithm

```
function MULTIGRID(L_h, f_h, u_h)
        ▷ L_h differential operator, f_h right hand side, u_h start approximation / solution
    if h = 1/2 then
        u_h ← (L_h f_h)^{-1}
    else
        u_h ← S_h^{ν_1}(L_h, f_h, u_h)        ▷ S_h Gauss-Seidel relaxation, ν_1 presmoothing steps
        d_h ← f_h − L_h u_h                   ▷ compute defect
        H ← 2h
        d_H ← I_h^H d_h                       ▷ restrict defect
        v_H ← MULTIGRID (L_H, d_H, 0)         ▷ recursive call to multigrid
        v_h ← I_H^h v_H                       ▷ interpolate coarse grid correction
        u_h ← u_h + v_h                       ▷ update solution
        u_h ← S_h^{ν_2}(L_h, f_h, u_h)        ▷ ν_2 postsmoothing step
    end if
end function
```

4. Multigrid solver. The main aim of this paper is to propose a multigrid solver that solves the normal equation (2.4) efficiently. In this section we describe the components of this solver. Subsequently the following notation is used. The discretized differential operators are written as uppercase Roman letters with a subscript indicating the discretization step, e.g. A_h, L_h. Data is denoted by lowercase Roman letters with a subscript indicating the discretization step, e.g. u_h, v_h. The expression d_h is reserved for the defect and f_h for the right hand side of the equation. The operator $I_{h_1}^{h_2}$ denotes the restriction operator if $h_1 < h_2$ and the interpolation operator if $h_1 > h_2$. A pseudo-code implementation of the multigrid subroutine is given in algorithm 1.

4.1. Multigrid components. As a starting point we use standard multigrid components. Since we deal with linear subproblems the correction scheme can be used. We use standard coarsening, i.e. $H = 2h$, and the coarse grid operator L_H is the compatible operator L_{2h}. Data is restricted by fullweighting and interpolated by bi/trilinear interpolation. The smoother is a pointwise coupled Gauss-Seidel relaxation method. We perform V-,F-, or W-cycles.

In general good convergence rates will not be obtained with this setup, because of the jumping coefficients(anisotropies) in the equations [6]. In section 5 numerical results that underline this fact are given. Next we introduce several changes in the standard components that improve the convergence of the multigrid.

4.2. Coarse grid operator. Remember that the operator M_h consist of two parts: L_h and $A_h = J_{\theta_h}^t J_{\theta_h}$. In case of L_h we use the compatible coarse grid operator $L_H = L_{2h}$. For A_h we have two options:

- restriction of the coefficients
- restriction of the template T, and computation of the coefficients on the coarse grid.

The latter option can be interpreted as a direct discretization of A on the coarse grid Ω_H. For the numerical experiments at the end of this paper this strategy is used, since it proofed to give better results.
A third alternative would be the Galerkin operator $I_h^H M_h I_H^h$. Yet, we avoided this choice here for the sake of computational complexity.

4.3. Smoother. It is a well known fact that pointwise relaxation methods fail to smooth the error well in case of highly anisotropic problems. Generally strongly

coupled components should be updated collectively [6]. Since in our problem the anisotropy depends on the input data and thus, location and direction of strongly coupled components are not known a priori, we use alternating Gauss-Seidel line relaxations, instead of pointwise relaxations. In one smoothing step all lines along all dimensions are relaxed successively. Linewise relaxation results in a linear system of equations for each line. Due to the structure of the operator M_h the coefficient matrix of that system has m bands of width 3 and mn equations,where n is the number of grid points in a line. The coefficients on the bands for the x-direction are underlaid by gray boxes in equation (3.2). The bands can be stored as vectors and the system is solved with a band-gauss solver. The solver needs $\mathcal{O}(n)$ time and $\mathcal{O}(n)$ space where n is the number of grid points in a line. Hence each point is relaxed in $\mathcal{O}(1)$ time, as for the pointwise relaxation.

4.4. Operator dependent interpolation. The adjoint operator to the restriction by fullweighting is bilinear interpolation for $m = 2$, respectively trilinear interpolation for $m = 3$. The interpolated correction $v_h = I_H^h v_H$ solely depends on v_H. It does not use any information about the jumping coefficients in M_h. Depending on the discretization and knowledge about the underlying problem operator dependent interpolation can be derived [2][1][6]. In the following we introduce a interpolation operator that is induced by the discretization of the differential operator M_h.

On the coarse grid we have solved $M_H v_H = f_H$. Hence the interpolated correction v_h should be an approximate solution to the fine grid equation $M_h v_h = d_h$. In the following we replace h by a vector $\vec{h} = (h_1, h_2, \dots h_d)$. We interpolate v_H by successively refining the grid along each dimension, i.e. in each refinement step the amount of grid points doubles. The missing grid points can then be computed by solving for all points on the new lines simultaneously. As in the case of the line relaxation that leads to band matrix that can be solved efficiently with a band matrix solver in $\mathcal{O}(n)$ steps where n is the number of grid points on the line. The corresponding equations are

$$M_{\vec{h}^i} v_{\vec{h}^i} = d_{\vec{h}^i}, \qquad \vec{h}^i = (h_1, \dots, h_i, H_{i+1}, \dots, H_d), \quad i = 1, \dots, d \tag{4.1}$$

The missing values for $M_{\vec{h}^i}$ and $d_{\vec{h}^i}$ can be obtained by applying restriction to the

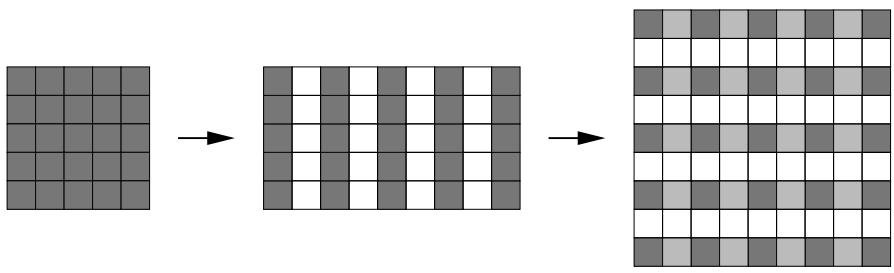


FIGURE 4.1. *Illustration of the interpolation procedure for the 2d-case. First the grid is refined in one dimension. Then the unknown values [white boxes] are computed by solving a system of linear equations for each line. Afterwards the grid is refined in the other dimension and the missing values in the new grid are computed.*

fine grid equivalents along all necessary directions. Figure 4.1 illustrates the procedure for the 2d-case.

Note that the computational efficiency of the presented technique mainly stems from the fact that we have a simple coarsening and refinement strategy on a regular

grid. The operator $M_{\vec{h}^i}$ and and the right side $d_{\vec{h}^i}$ can be easily computed. Furthermore the coefficient matrix of the resulting linear systems has a only m bands, where the width of the bands depends on the stencil width. Hence, we can find fast solvers that preserve the asymptotic complexity of the multigrid.

4.5. Operator dependend correction step. In this section we introduce a modification to the correction step. In the standard multigrid one computes $u_h \leftarrow u_h + I_H^h v_H = u_h + v_h$. Instead we multiply v_h by a factor τ that is optimal in a certain sense, i.e. line 12 in algorithm 1 is replaced by

$$u_h \leftarrow u_h + \tau v_h. \tag{4.2}$$

Here the parameter τ is chosen such that the underlying energy (2.1) becomes minimal, i.e. we solve the following minimization problem:

$$\min_{\tau\in\mathbb{R};\tau\geq 0} \Psi(\tau) = \min_{\tau\in\mathbb{R};\tau\geq 0} \Big\{ \big\langle J_{E_\alpha}, (u+\tau v)\big\rangle + \frac{1}{2}\big\langle (J_\theta^t J_\theta + \tilde{\alpha} L)(u+\tau v), (u+\tau v)\big\rangle \Big\}.$$

Ψ is a quadratic function and therefore it is differentiable. The necessary condition for a minimum is given by

$$\frac{\partial \Psi}{\partial \tau} = \langle J_{E_\alpha}, v\rangle + \langle (J_\theta^t J_\theta + \tilde{\alpha} L)(u+\tau v), v\rangle \stackrel{!}{=} 0.$$

With $A = J_\theta^t J_\theta$ and $f = -J_{E_\alpha}$ this leads to

$$\langle \underbrace{(A+\tilde{\alpha}L)u - f}_{=d(u)}, v\rangle + \tau\langle (A+\tilde{\alpha}L)v, v\rangle = 0.$$

Thus

$$\tau = -\frac{\langle d_h(u_h), v_h\rangle}{\langle (A_h + \tilde{\alpha} L_h)v_h, v_h\rangle} \tag{4.3}$$

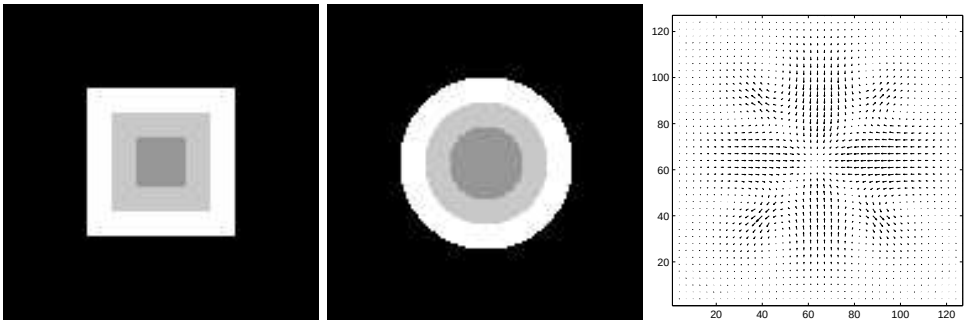
fulfills the necessary condition. This minimization is analog to minimizing $||u^* - (u^{(k)} + \tau v)||_1$ where u^* is the exact solution.

5. Numerical results. In this section we mainly focus on convergence rates for the multigrid, the inner iteration. For practical reasons the gray values in the images are scaled to $[0,1]$, and all norms are divided by the number of data points. If not indicated otherwise tests were performed with $\nu_1 = 2$, $\nu_2 = 1$, $\mu = 1$, and $\lambda = 1$. We have done F-cycles in all computations since that proved to be the best trade-off between performance and convergence.

5.1. Convergence rates. Convergence factors $\hat{q}^{(m)}$ where determined from the L_2-Norm of the defect using the formulas

$$q^{(m)} := \frac{||d_h^m||_2}{||d_h^{m-1}||_2}, \quad \hat{q}^{(m)} := \sqrt[m-m_0+1]{q^{(m)} q^{(m-1)} \dots q^{(m_0)}}, \tag{5.1}$$

where d_h^m is the defect after the m-th multigrid cycle [6]. Here we use $m_0 = 4$ and $m = 15$, i.e. the first three iterations are not included in the computation of the convergence rates.



(a) Template (b) Reference (c) Transformation

FIGURE 5.1. *Model problem: The template (a) is comprised of three nested squares with different gray values. In the reference (b) the squares are replaced by circles. The resulting transformation is shown in (c).*

$\tilde{\alpha}$	P/N/N	L/N/N	L/Y/N	L/Y/Y
0.01	0.638140	0.519372	0.194113	0.189293
0.005	0.648635	0.522911	0.160224	0.151399
0.001	0.672952	0.534282	0.140629	0.126447
0.0005	0.686041	0.541510	0.233798	0.161013

TABLE 5.1

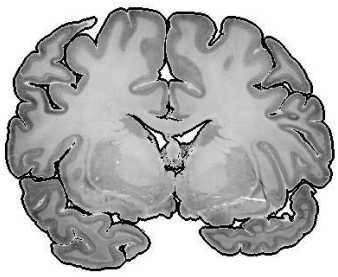
Convergence rates for different combinations of multigrid components, for a given starting $\tilde{\alpha}$. The first letter indicates whether pointwise (P) or linewise (L) relaxation has been used. The second letter indicates whether the operator dependent correction has been used (Y) or not (N). The third letter indicates whether operator dependent interpolation has been used.

Computations are based on a simple model problem (Figure 5.1). A square with three gray values is to be registered to a circle, with the same three gray values. The transformation for a complete registration is shown in figure 5.1(c).

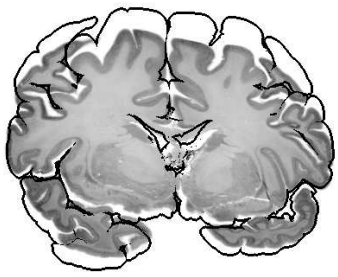
Note that for the computation of the convergence only one step is performed in the outer iteration . Thus, we have $f(u^{(0)}) = -(J_\theta^t \theta + \alpha L u^{(0)})$, with $u^{(0)} \equiv 0$. Hence, only the value of $\tilde{\alpha}$ is important and not how it is composed. For subsequent outer iterations that is not the case due to the fact that the regularization on the right hand side then depends on the choice of α, i.e. one might observe different behaviour for identical $\tilde{\alpha} = \alpha + \beta$ because the smoothness of the right hand side varies. The value $\tilde{\alpha} = 0.01$ is chosen so that $||A||_\infty \approx \tilde{\alpha}||L||_\infty$. As $\tilde{\alpha}$ decreases the anisotropy increases and we generally expect worse convergence rates.

We tested many parameters for different combinations of multigrid components. We have selected some to illustrate the most important points. Pointwise relaxations is compared to linewise relaxation. For the linewise relaxations we also added results for operator dependent correction step (see section 4.5) and operator dependent interpolation (see section 4.4). Convergence rates for the pointwise relaxation are very bad. Though linewise relaxations bring a considerable improvement of about 20%, rates are still not satisfactory. When the operator dependent factor in the correction step is employed convergence rates improve dramatically. Operator dependent interpolation yields only small improvements for larger $\tilde{\alpha}$, but the degradation in convergence rates is slower as this parameter is decreased. We do not report results for the operator dependent interpolation without the modification in the correction step, because the

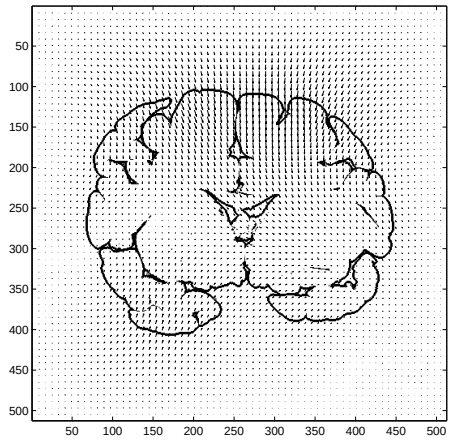
improvement achieved is not as big as for the operator dependent correction step. Note that convergence the absolute rates might differ for different input data, but the relative statements made above still hold.



(a) Reference



(b) Template



(c) Transformation

FIGURE 5.2. *histological section*

5.2. Remarks on the outer iteration. The number of multigrid cycles that has to be performed can be assessed by the rate of change in the functional E_α. When E_α does only change very little from i to $i+1$ multigrid cycles one can stop. In this context it is worthwhile to note that the defect reduction with operator dependent interpolation is greater in the first few cycles, than without. Furthermore we observed that with operator dependent interpolation the convergence are very similar in all steps of the outer iteration, whereas they vary without it.

For large data sets and deformations the outer iteration is embedded in a multi-resolution framework, i.e. solutions computed at coarser image resolutions are used as an initial approximation on finer resolutions.

Figure 5.2 shows an example of the registration of histological sections (512×512 pixels). The reference image has been distorted by an artificial transformation

to obtain the template image. The outer contour of the reference is indicated by a black line. Registration has been performed at four resolutions. The image difference after registration is below 0.1%. The corresponding deformation field is shown in figure 5.2(c).

6. Conclusion. In this paper, we have proposed a Newton-type algorithm for the minimization of a nonlinear least squares functional which occurs in digital image registration. The Newton system has the form of an anisotropic partial differential equation. Newton's direction is calculated by solving an approximate (positive definite) Newton system using a multigrid method. The numerical results indicate fast convergence and robustness of the multigrid scheme, also for highly jumping coefficients.

Note that the proposed solver can also deal with other anisotropies in the derived PDE's, e.g. local regularization parameters or local material constants. Furthermore the equations are related to the optical flow problem [5].

Further research directions may include the use of a Galerkin coarse grid approximation instead of the coarse grid operator proposed in section 4.2 [5]. The adjoint operator to the interpolation operator in section 4.4 could be used for restriction of the solution. In cases where $\lambda \gg \mu$, which is not the case in our applications, staggered grid discretizations are known to give better convergence.

REFERENCES

[1] R. Alcouffe, A. Brandt, J. Dendy, and J. Painter, *The multigrid method for the diffusion equation with strongly discontinuous coefficients*, SIAM J. Sci. Comput., 2 (1981), pp. 430–454.

[2] J. Dendy, *Black box multigrid*, J. Comp. Phys., 48 (1982), pp. 366–386.

[3] M. Hanke-Bourgeois, *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*, Teubner, Stuttgart · Leipzig · Wiesbaden, 2002.

[4] S. Henn, *A Levenberg-Marquardt scheme for nonlinear image registration*, BIT Numerical Mathematics, 43(4) (2003), pp. 743–759.

[5] E. M. Kalmoun and U. Rüde, *A variational multigrid for computing the optical flow*, tech. rep., Friedrich-Alexander-Universitt Erlangen-Nürnberg, Erlangen, 2003. (Lehrstuhlbericht 03-4).

[6] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, 2001.

[7] W. Wells and P. Viola, *Alignement by maximization of mutual information*, International Journal of Computer Vision, 24(2) (1997), pp. 137–154.

AN ACCURATE MULTIGRID SOLVER FOR COMPUTING SINGULAR SOLUTIONS OF ELLIPTIC PROBLEMS

H. KÖSTLER*

Abstract.
In this article we present a method to solve partial differential equations (PDE) containing generalized functions as source terms. Typical applications are point sources and sinks in porous media flow that are described by Dirac δ-functions or point loads and dipoles as source terms for electrostatic potentials.

For analyzing the accuracy of such computations, standard techniques cannot be used, since they rely on global smoothness. This is both true for Sobolev space arguments for finite element based methods, and for continuity and differentiability arguments in finite difference analysis. At the singularity, the solution and its derivatives tend to infinity and therefore standard error norms will not even converge.

We will demonstrate that these difficulties can be overcome by using other metrics to measure accuracy and convergence of the numerical solution. In the standard case our technique is equivalent to representing singular distributions by properly scaled finite element basis functions. Only minor modifications to the discretization and solver are necessary to obtain the same asymptotic accuracy and efficiency as for regular and smooth solutions.

1. Introduction. Typical error estimates for the numerical solution of boundary value problems depend on the smoothness of the true solution which is not given in many practical applications. Reasons for such singular solutions can for example be reentrant corners, discontinuous coefficients, singular functions in the boundary conditions or source terms with singularities. In this article we consider the last case. As application we choose for simplicity, although our method can be extended to more general situations, electrostatic potentials of point loads and dipoles, physically modeled by the Maxwell equations in the vacuum related to the source localization in the human brain (cf. [9]). This leads to the *Poisson equation* with Dirichlet boundary conditions in the unit cube $\Omega = (0,1)^3 \subset \mathbb{R}^3$

$$\begin{array}{rcll} -\Delta u & = & f & \text{in } \Omega \\ u & = & g & \text{on } \partial\Omega \end{array} , \tag{1.1}$$

where Δ is the Laplace operator, $\partial\Omega$ denotes the boundary of Ω and g is a smooth function. The source term f contains one or more singularities, because a point load is modeled by the Dirac δ-function. As an extension we will consider sources containing dipoles, that can be formally introduced as a directional derivative of a point load. Note that the Dirac δ-function is not a function in the classical sense, but a *distribution* (cf. [16]). Throughout the paper we will focus on the case of a single, isolated right-hand-side (RHS) singularity. The case of several singularities can be studied easily by using the superposition principle. For the discretization of (1.1) we use finite differences on equidistant grids $\Omega_h = \{(ih, jh, kh), 0 < i < N, 0 < j < N, 0 < k < N \; i, j, k, N \in \mathbb{N}\}$ with mesh size $h = \frac{1}{N}$. The discrete Laplace operator Δ_h is given by the usual 7-point stencil.

One has now several possibilities to take the singular RHS f into account. E.g. adaptive refinement could be used or one could make use of the analytic knowledge of the singularity (cf. [15]).

In contrast to these the *Zenger Correction Method* (cf. [14], [19]) discussed in Section 2 relies simply on a mesh-size dependent representation of the singular sources constructed by appropriate smoothing. It will be proved in Section 3 that the pointwise accuracy is of the same order as in the regular case. The error coefficient depends on the location and will deteriorate when approaching the singularity where the error estimate breaks down. Our approach is therefore useful for accurately computing the global solution, except in a small neighborhood of the singular points. In other words our methods overcomes the *pollution effect* of the singularity what means that using

*University of Erlangen-Nuremberg, Germany, Harald.Koestler@informatik.uni-erlangen.de

no special treatment for it, the accuracy of the solution would be destroyed in the whole domain. The great advantage of our method is that we do not need to know the exact singular component of the solution of the problem to approximate the singularity. Therefore it can be used for a variety of problems. We have integrated these techniques into a multigrid solver exploiting additional techniques for improving the accuracy, such as Richardson and τ-Extrapolation that are briefly summarized in Section 4.

In the 5th section the exemplary experimental results for the numerical solution of the Poisson equation with Zenger Correction and extrapolation are shown.

2. Grid dependent numerical approximations of singular distributions. The *Zenger Correction Method* uses the following distributions in order to approximate the physical singularities.

DEFINITION 2.1. *Let* $H_0 : \mathbb{R} \to \mathbb{R}$ *be the* Heaviside-function

$$H_0(x) := \begin{cases} 0 & : \quad x \leq 0 \\ 1 & : \quad x > 0 \end{cases} , \tag{2.1}$$

and the distributions H_i, $i \in \mathbb{Z}$, *be recursively defined by*

$$H_i(x) := \begin{cases} \dfrac{d}{dx} H_{i-1}(x) & : \quad i > 0 \\ \displaystyle\int_{-\infty}^{x} H_{i+1}(\xi) d\xi & : \quad i < 0 \end{cases} . \tag{2.2}$$

The family of functions H_i enables us to represent arbitrary physical multipoles. For example H_1 corresponds to the Dirac δ-function resp. a point load. For higher dimensions we use tensor products of these functions. In 3D for example a point load at $\mathbf{x}_0 = (x_0, y_0, z_0)^T$ is described by

$$H_1(x - x_0) H_1(y - y_0) H_1(z - z_0) := \mathcal{H}^{1,1,1}_{x_0,y_0,z_0}(\mathbf{x}) \ . \tag{2.3}$$

Generally we use multi indices $\mathbf{x}_0 = (x_{0,1}, x_{0,2}, \ldots, x_{0,n}) \in \mathbb{R}^n$ to indicate the location of the singularity and $\mathbf{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{Z}^n$ to specify the order and use the notation

$$\mathcal{H}^{\mathbf{a}}_{\mathbf{x}_0}(\mathbf{x}) := \prod_{i=1}^{n} H_{a_i}(x_i - x_{0,i}) \ . \tag{2.4}$$

The idea is now to *integrate* the right hand side f *analytically* a number of times, until the result is a product of (classical) functions H_i with $i < 0$. Then we *differentiate* this function *numerically* as often as it had been integrated. This results in a smooth approximation to the distribution f that depends on the meshsize h and becomes more accurate with smaller h. For the numerical differentiation we use finite differences with the notation

$$\delta^{\mathbf{i}} = \delta^{i_1}_{x_1} \circ \delta^{i_2}_{x_2} \circ \cdots \circ \delta^{i_n}_{x_n} \ , \tag{2.5}$$

for the multi index $\mathbf{i} \in \mathbb{Z}^n$ that indicates how often we differentiate in each direction. Dependent on the number of integrations resp. differentiation steps n we call this procedure the *Zenger Correction of n-th order.*

THEOREM 2.2. *In general for even n with $k - n < 0$ we obtain*

$$\delta^n_x H_k(x) = \begin{cases} 0 & : \quad |x| \geq \dfrac{n}{2} h \\ \dfrac{1}{h^n} \displaystyle\sum_{i=0}^{n} (-1)^i \binom{n}{i} H_{k-n}(x + (i - \tfrac{n}{2})h) & : \quad |x| < \dfrac{n}{2} h \end{cases} . \tag{2.6}$$

In [7] it is shown that formula (2.6) in fact is the nth numerical derivative of $H_k(x)$ and that it is equal to a B-Spline.

3. Error estimates. In this section we prove that if the singularity in the source term is replaced by the above approximation, we obtain an $O(h^2)$ discretization error as in the smooth case, except in a small neighborhood of the singularity. Therefore we first consider the solution u^* defined on $\overline{\Omega}$ of the boundary value problem (1.1) with homogeneous right hand side

$$\begin{array}{rcll} \Delta u & = & 0 & \text{in } \Omega \\ u & = & g & \text{on } \partial\Omega \end{array} \quad , \tag{3.1}$$

where g is bounded and piecewise continuous on the boundary of $\Omega = (0,1)^3$ and u_h^* is a grid function defined on Ω_h solving the discrete analogue of (3.1).

THEOREM 3.1. *The discretization error of problem (3.1) has an asymptotic expansion (cf. [3])*

$$u_h^* - u^* = \sum_{l=1}^{p} c_l h^{2l} + \mathcal{O}(h^{2p+2}) \ , \tag{3.2}$$

where the right term is restricted to Ω_h and $c_1, c_2, \ldots, c_p$ are bounded on $\overline{\Omega}$.

The proof for the 2D case is found in [6] and will be extended here for the 3D case. Before we can detail the proof we must collect some preliminaries which will be formulated in Lemma 3.2–3.5. Because of the linearity of (3.1) and its discrete analogon we can use the superposition principle. Without loss of generality it is sufficient to consider instead of $g(x,y,z)$ only boundary conditions $g_1(x,y,0)$ that vanish at all but one of the faces of Ω. In the following we set $g(x,y) = g_1(x,y,0)$ and assume that g is bounded on $\overline{\Omega}$ with $|g(x,y)| \le M$. Thus we have with $\mathbf{x} = (x,y,z)$

$$\begin{array}{rcll} \Delta u(\mathbf{x}) & = & 0 & \text{in } \Omega \\ u(\mathbf{x}) & = & 0 & \text{on } \partial\Omega \ , \quad z > 0 \\ u(x,y,0) & = & g(x,y) & \text{on } \partial\Omega \ , \quad z = 0 \end{array} \quad . \tag{3.3}$$

The analytical solution u^* of (3.3) can then be constructed by a separation ansatz and Fourier series expansion as

$$\begin{array}{rcll} u^*(\mathbf{x}) & = & \displaystyle\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} a_{mn} \sin(m\pi x)\sin(n\pi y) T(t_{mn}, z) & , \ 0 \le x, y \le 1, \ z > 0 \\ u^*(\mathbf{x}) & = & g(x,y) & , \ 0 \le x, y \le 1, \ z = 0 \end{array} \quad , \tag{3.4}$$

where $t_{mn} = \sqrt{m^2+n^2}$, $T(t_{mn}, z) = \dfrac{\sinh(\sqrt{m^2+n^2}\pi(1-z))}{\sinh(\sqrt{m^2+n^2}\pi)} \le e^{-t_{mn}\pi z}$ and

$$a_{mn} = 4\int_0^1\int_0^1 g(x,y)\sin(m\pi x)\sin(n\pi y)dxdy \le 4M \ . \tag{3.5}$$

The discrete analog of (3.3) is

$$\begin{array}{rcll} \Delta_h u_h(\mathbf{x}) & = & 0 & \text{in } \Omega_h \\ u_h(\mathbf{x}) & = & 0 & \text{on } \partial\Omega_h \ , \quad z > 0 \\ u_h(x,y,0) & = & g_h(x,y) & \text{on } \partial\Omega_h \ , \quad z = 0 \end{array} \tag{3.6}$$

and its solution u_h^* is given by

$$\begin{aligned} u_h^*(\mathbf{x}) &= \sum_{m=1}^{N-1}\sum_{n=1}^{N-1} b_{mn}(N)\sin(m\pi x)\sin(n\pi y)T(\mu_{mn},z) \quad , \ 0\le x,y\le 1,\ z>0 \\ u_h^*(\mathbf{x}) &= g(x,y) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad , \ 0\le x,y\le 1,\ z=0 \end{aligned} \tag{3.7}$$

with

$$b_{mn}(N) = \frac{4}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} g\left(\frac{i}{N},\frac{j}{N}\right)\sin\left(\frac{m\pi i}{N}\right)\sin\left(\frac{m\pi j}{N}\right) \quad . \tag{3.8}$$

This expansion for the discrete solution is analogous to the continuous Fourier expansion (3.4). In the next lemmas all proved in [12] we will show the existence of an asymptotic expansion for the real parameters μ_{mn} and the difference $T(t_{mn},z)-T(\mu_{mn},z)$.

LEMMA 3.2. *The real number μ_{mn} is implicitly defined by*

$$\sinh^2\left(\frac{\mu_{mn}\pi}{2N}\right) = \sin^2\left(\frac{m\pi}{2N}\right) + \sin^2\left(\frac{n\pi}{2N}\right),\ 0\le m\le N-1,\ 0\le n\le N-1 \ . \tag{3.9}$$

LEMMA 3.3. *μ_{mn} satisfies the estimate $\alpha(m+n)\le\mu_{mn}\le m+n$ with $\alpha=\frac{1}{\pi}\operatorname{arsinh}(\sqrt{2})$.*

LEMMA 3.4. *μ_{mn} has an asymptotic expansion of the form*

$$\mu_{mn} = \sqrt{m^2+n^2} + \sum_{l=1}^{p} c_{lmn}\left(m^2+n^2\right)^{2l} h^{2l} + \mathcal{O}(h^{2p+2}) \ , \tag{3.10}$$

where c_{lmn} is a bounded polynomial in m,n.

LEMMA 3.5. *$T(\mu_{mn},z)-T(t_{mn},z)$ has an asymptotic expansion*

$$T(\mu_{mn},z)-T(t_{mn},z) = T(t_{mn},z)\sum_{l=1}^{p} c_{lmn}\left(m^2+n^2\right)^{2l} h^{2l} + \mathcal{O}(h^{2p+2}) \ . \tag{3.11}$$

We are now ready to present the main proof of Theorem 3.1 for the 3D case and justify the existence of the representation of u_h^* by (3.7).

Proof. [Theorem 3.1]

We consider the discretization error that is the difference between (3.4) and (3.7)

$$\begin{aligned} u^*(\mathbf{x}) - u_h^*(\mathbf{x}) &= \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} a_{mn}\sin(m\pi x)\sin(n\pi y)T(t_{mn},z) \\ &\quad - \sum_{m=1}^{N-1}\sum_{n=1}^{N-1} b_{mn}(N)\sin(m\pi x)\sin(n\pi y)T(\mu_{mn},z) \\ &= I_1(\mathbf{x},h) + I_2(\mathbf{x},h) + I_3(\mathbf{x},h) \ , \end{aligned}$$

where

$$
\begin{aligned}
I_1(\mathbf{x},h) &= \sum_{m=N}^{\infty}\sum_{n=N}^{\infty} a_{mn}\sin(m\pi x)\sin(n\pi y)T(t_{mn},z)\\
&+ \sum_{m=1}^{N-1}\sum_{n=N}^{\infty} a_{mn}\sin(m\pi x)\sin(n\pi y)T(t_{mn},z)\\
&+ \sum_{m=N}^{\infty}\sum_{n=1}^{N-1} a_{mn}\sin(m\pi x)\sin(n\pi y)T(t_{mn},z)\\
I_2(\mathbf{x},h) &= \sum_{m=1}^{N-1}\sum_{n=1}^{N-1} a_{mn}\sin(m\pi x)\sin(n\pi y)\left\{T(t_{mn},z)-T(\mu_{mn},z)\right\}\\
I_3(\mathbf{x},h) &= \sum_{m=1}^{N-1}\sum_{n=1}^{N-1} \{a_{mn}-b_{mn}(N)\}\sin(m\pi x)\sin(n\pi y)T(\mu_{mn},z)\ ,
\end{aligned}
$$

and try to find an asymptotic expansion in powers of h for these three terms. First we take a closer look at $I_1(\mathbf{x},h)$ using

$$
e^{-x} \le \frac{k!}{x^k} \quad \forall k>1, \quad x>0 \tag{3.12}
$$

and $\dfrac{m+n}{2} \le \sqrt{\dfrac{m^2+n^2}{2}} \quad \forall m,n \ge 1$ to get

$$
\begin{aligned}
|N^s I_1(\mathbf{x},h)| &\le 4MN^s \sum_{m=N}^{\infty}\sum_{n=N}^{\infty} e^{-\sqrt{2}\sqrt{\frac{m^2+n^2}{2}}\pi z} + 4MN^s \sum_{m=1}^{N-1}\sum_{n=N}^{\infty} e^{-\sqrt{2}\sqrt{\frac{m^2+n^2}{2}}\pi z}\\
&+ 4MN^s \sum_{m=N}^{\infty}\sum_{n=1}^{N-1} e^{-\sqrt{2}\sqrt{\frac{m^2+n^2}{2}}\pi z}\\
&\le 4MN^s \left(\sum_{m=N}^{\infty}\sum_{n=N}^{\infty} e^{-\frac{n\pi z}{\sqrt{2}}} e^{-\frac{m\pi z}{\sqrt{2}}} + \sum_{m=1}^{N-1}\sum_{n=N}^{\infty} e^{-\frac{n\pi z}{\sqrt{2}}} e^{-\frac{m\pi z}{\sqrt{2}}} + \sum_{m=N}^{\infty}\sum_{n=1}^{N-1} e^{-\frac{n\pi z}{\sqrt{2}}} e^{-\frac{m\pi z}{\sqrt{2}}}\right)\\
&= 4MN^s \left(\frac{e^{-\frac{N\pi z}{\sqrt{2}}}}{1-e^{-\frac{\pi z}{\sqrt{2}}}}\frac{e^{-\frac{N\pi z}{\sqrt{2}}}}{1-e^{-\frac{\pi z}{\sqrt{2}}}} + \frac{1-e^{-\frac{N\pi z}{\sqrt{2}}}}{1-e^{-\frac{\pi z}{\sqrt{2}}}}\frac{e^{-\frac{N\pi z}{\sqrt{2}}}}{1-e^{-\frac{\pi z}{\sqrt{2}}}} + \frac{e^{-\frac{N\pi z}{\sqrt{2}}}}{1-e^{-\frac{\pi z}{\sqrt{2}}}}\frac{1-e^{-\frac{N\pi z}{\sqrt{2}}}}{1-e^{-\frac{\pi z}{\sqrt{2}}}}\right)\\
&\le \frac{4Mk!(\sqrt{2})^k}{(\pi z)^k}(1-e^{-\frac{\pi z}{\sqrt{2}}})^{-2}\left(2-e^{-\frac{N\pi z}{\sqrt{2}}}\right)\frac{1}{N^{k-s}}\\
&\le \frac{8Mk!(\sqrt{2})^k}{(\pi z)^k}(1-e^{-\frac{\pi z}{\sqrt{2}}})^{-2}\frac{1}{N^{k-s}}\ .
\end{aligned}
$$

For every s we can choose a $k>s$ such that $\lim_{N\to\infty}|N^s I_1(\mathbf{x},h)|=0$. Therefore $I_1(\mathbf{x},h)$ has the trivial asymptotic expansion

$$
I_1(\mathbf{x},h) = 0h + 0h^2 + 0h^3 + \dots \tag{3.13}
$$

Next we derive an asymptotic expansion for $I_2(\mathbf{x}, h)$. With (3.10) and (3.11) we have

$$\begin{aligned}
I_2(\mathbf{x}, h) &= \sum_{m=1}^{N-1}\sum_{n=1}^{N-1} a_{mn}\sin(m\pi x)\sin(n\pi y) \\
&\quad \cdot \left(T(t_{mn}, z)\sum_{l=1}^{p} c_{lmn}\left(m^2+n^2\right)^{2l} h^{2l} + \mathcal{O}(h^{2p+2}) \right) \\
&\leq 4M\sum_{l=1}^{p}\left\{\sum_{m=1}^{N-1}\sum_{n=1}^{N-1} e^{-(m+n)\pi z} c_{lmn}\left(m^2+n^2\right)^{2l}\right\} h^{2l} + \mathcal{O}(h^{2p+2}) \ .
\end{aligned}$$

The series $\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} e^{-(m+n)\pi z} c_{lmn}\left(m^2+n^2\right)^{2l}$ converges towards λ_l and therefore we are able to split up the above term

$$\begin{aligned}
I_2(\mathbf{x}, h) &\leq 4M\sum_{l=1}^{p}\lambda_l h^{2l} + \mathcal{O}(h^{2p+2}) \\
&\quad + \left| 4M\sum_{l=1}^{p}\sum_{m=N}^{\infty}\sum_{n=1}^{\infty} e^{-(m+n)\pi z} c_{lmn}\left(m^2+n^2\right)^{2l} h^{2l} + \mathcal{O}(h^{2p+2}) \right| \\
&\quad + \left| 4M\sum_{l=1}^{p}\sum_{m=1}^{\infty}\sum_{n=N}^{\infty} e^{-(m+n)\pi z} c_{lmn}\left(m^2+n^2\right)^{2l} h^{2l} + \mathcal{O}(h^{2p+2}) \right| \\
&= 4M\sum_{l=1}^{p}\lambda_l h^{2l} + \mathcal{O}(h^{2p+2}) \\
&\quad + \left| 4M\sum_{l=1}^{p}\sum_{m=0}^{\infty}\sum_{n=1}^{\infty} e^{-(m+n+N)\pi z} c_{lmn}\left((m+N)^2+n^2\right)^{2l} h^{2l} + \mathcal{O}(h^{2p+2}) \right| \\
&\quad + \left| 4M\sum_{l=1}^{p}\sum_{m=1}^{\infty}\sum_{n=0}^{\infty} e^{-(m+n+N)\pi z} c_{lmn}\left(m^2+(n+N)^2\right)^{2l} h^{2l} + \mathcal{O}(h^{2p+2}) \right| \\
&\leq 4M\sum_{l=1}^{p}\lambda_l h^{2l} + \mathcal{O}(h^{2p+2}) \\
&\quad + \left| 4M\sum_{l=1}^{p} e^{-N\pi z} N^{2l+2}\sum_{m=0}^{\infty}\sum_{n=1}^{\infty} e^{-(m+n)\pi z} c_{lmn}\left((m+1)^2+n^2\right)^{2l} h^{2l} + \mathcal{O}(h^{2p+2}) \right| \\
&\quad + \left| 4M\sum_{l=1}^{p} e^{-N\pi z} N^{2l+2}\sum_{m=1}^{\infty}\sum_{n=0}^{\infty} e^{-(m+n)\pi z} c_{lmn}\left(m^2+(n+1)^2\right)^{2l} h^{2l} + \mathcal{O}(h^{2p+2}) \right| \ .
\end{aligned}$$

By using (3.12) and choosing the parameter k high enough we can show that the last two terms are of order $\mathcal{O}(h^{2p+2})$ and therefore $I_2(\mathbf{x}, h)$ has the asymptotic expansion

$$I_2(\mathbf{x}, h) = \sum_{l=1}^{p}\lambda_l h^{2l} + \mathcal{O}(h^{2p+2}) \ . \tag{3.14}$$

An important fact is that the asymptotic expansion of $I_2(\mathbf{x}, h)$ does not depend on the properties

of g. Finally we look at the third term $I_3(\mathbf{x}, h)$.

$$\begin{aligned} I_3(\mathbf{x}, h) &= \sum_{m=1}^{N-1} \sum_{n=1}^{N-1} \{a_{mn} - b_{mn}(N)\} \sin(m\pi x) \sin(n\pi x) T(\mu_{mn}, z) \\ &\leq \sum_{m=1}^{N-1} \sum_{n=1}^{N-1} \{a_{mn} - b_{mn}(N)\} \left(T(t_{mn}, z) \sum_{l=0}^{p} c_{lmn} \left(m^2 + n^2\right)^{2l} h^{2l} + \mathcal{O}(h^{2p+2}) \right) \end{aligned}$$

Using the Euler-Maclaurin sum formula (cf. [10]) we can prove in an analog way as for $I_2(\mathbf{x}, h)$ that $I_3(\mathbf{x}, h)$ has an asymptotic expansion of the form

$$I_3(\mathbf{x}, h) = \sum_{l=1}^{p} \theta_l h^{2l} + \mathcal{O}(h^{2p+2}) \ , \tag{3.15}$$

where θ_l is bounded on Ω. This completes the proof. □

Definition 3.6 (h-bounded). *A family of functions $u_h(\mathbf{x})$ is called* h-bounded *on the domain $\Omega \subset \mathbb{R}^n$, if there exists a real valued, continuous function $r(\mathbf{x})$ on Ω, which is not necessarily bounded on Ω, so that for every $\mathbf{x} \in \Omega$ there exists a number $h_0 > 0$ with $|u_h(\mathbf{x})| \leq r(\mathbf{x})$ for all $h = \frac{1}{N} < h_0, N \in \mathbb{N}$, $\mathbf{x} \in \Omega_h$. If $r(\mathbf{x})$ is bounded on Ω, $u_h(\mathbf{x})$ is called* strictly h-bounded *(cf. [4], p.6).*

An h-bounded family of grid functions u_h may be unbounded on Ω for $h \to 0$, but because of the continuity of r it will be bounded for all $h > 0$ on every compact subset of Ω.

Theorem 3.7. *Let the solution of*

$$\Delta_h u_h \quad = \quad f_h \quad \text{in } \Omega_h \tag{3.16}$$

be bounded on $\overline{\Omega}_h$. If $\delta^{2\mathbf{i}} f_h$ is h-bounded on Ω and $\forall i_l, m_l$ with $0 < l \leq n$ is $0 \leq i_l \leq m_l$, then $\delta^{2\mathbf{m}} u_h$ is h-bounded on Ω.

Proof. Can be found in [12]. □

Now we are prepared for the central theorem. It states that the global pointwise discretization error after the Zenger Correction is $\mathcal{O}(h^2)$ as in the smooth case.

Theorem 3.8. *Let u^* be the (weak) solution of the boundary value problem*

$$\begin{array}{rcll} \Delta u & = & \mathcal{H}^{\mathbf{a}}_{\mathbf{x}_0}(\mathbf{x}) & \text{in } \Omega \\ u & = & 0 & \text{on } \partial\Omega \end{array} \ , \tag{3.17}$$

where $\mathcal{H}^{\mathbf{a}}_{\mathbf{x}_0}(\mathbf{x})$ is a distribution, whose singularity is located at $\mathbf{x}_0 \in \Omega =]0, 1[^n$. Let u_h^* the solution of

$$\begin{array}{rcll} \Delta_h u_h & = & f_h & \text{in } \Omega_h \\ u_h & = & 0 & \text{on } \partial\Omega_h \end{array} \ , \tag{3.18}$$

where $f_h = \delta^{2\mathbf{m}} \mathcal{H}^{\mathbf{a}-2\mathbf{m}}_{\mathbf{x}_0}(\mathbf{x})$ and where $\mathbf{m}$ is chosen such that $2m_l > a_l, 1 \leq l \leq d$. $d \in \{2, 3\}$ is the dimension of the problem. Then $\mathcal{H}^{\mathbf{a}-2\mathbf{m}}_{\mathbf{x}}$ are continuous functions and $u_h^* = u^* + h^2 r$, where r is h-bounded on $\Omega \backslash \{\mathbf{x}_0\}$.

Proof. Using the preparatory work from the preceding theorems the extension from the 2D case (cf. [14], pp.15) to the 3D case is straightforward. We know from Theorem 3.1 that smooth boundary values cause (in our simple domain) errors with h^2 expansions. Now we consider the boundary value problem (3.17) with boundary conditions $u(\mathbf{x}) = u_S^*(\mathbf{x})$ on $\partial\Omega_h$, where u_S^* is the analytical solution of the singular problem. The solution is given by $u^*(\mathbf{x}) = u_S^*(\mathbf{x})$.

Now we construct an auxiliary problem by "integrating" the original problem

$$\begin{array}{rcll} \Delta U & = & \mathcal{H}_{\mathbf{x}_0}^{\mathbf{a}-2\mathbf{m}}(\mathbf{x}) & \text{in } \Omega_h \\ U & = & F(\mathbf{x}) & \text{on } \partial\Omega_h \end{array} , \tag{3.19}$$

where $D^{2\mathbf{m}}F(\mathbf{x}) = u_S^*(\mathbf{x})$ with $D^{\mathbf{m}} = \dfrac{\partial^{|\mathbf{m}|}}{\partial x_1^{m_1}\partial x_2^{m_2}\cdots\partial x_d^{m_d}}$ and $|\mathbf{m}| = \sum_{i=1}^n m_i$. (3.19) is numerically approximated by

$$\begin{array}{rcll} \Delta_h U_h & = & \mathcal{H}_{\mathbf{x}_0}^{\mathbf{a}-2\mathbf{m}}(\mathbf{x}) & \text{in } \Omega_h \\ U_h & = & F(\mathbf{x}) & \text{on } \partial\Omega_h \end{array} . \tag{3.20}$$

We define R as the discretization error for the auxiliary problem (3.19) and (3.20), respectively. $U_h^* = U^* + h^2 R$, where U^* is the analytical solution of (3.19) and U_h^* is the numerical solution of (3.20). Note that $D^{2\mathbf{m}}U^*(\mathbf{x}) = u^*(\mathbf{x})$ on $\Omega \setminus \{\mathbf{x}_0\}$. The remainder R satisfies

$$\Delta_h R = Q := \frac{1}{h^2}(\mathcal{H}_{\mathbf{x}_0}^{\mathbf{a}-2\mathbf{m}}(\mathbf{x}) - \Delta_h U^*) \ .$$

Using Taylor-expansions and smoothness properties of U^* for estimating the remainder term we can show that

1. Q is bounded on Ω_h, since there exists a number $R_T \in \mathbb{R}$ with

$$|Q(\mathbf{x})| = |\frac{1}{h^2}(\Delta U^* - \Delta_h U^*)| \leq R_T \ .$$

2. $\delta^{2\mathbf{i}}Q$ is h-bounded on $\Omega\backslash\{\mathbf{x}_0\}$ for all i_l, $0 < l \leq n$.

Because of these two properties, R is bounded on Ω and using Theorem 3.7 even $\delta^{2\mathbf{n}}R$ is h-bounded on Ω. Now we define $\widetilde{u}_h$ as a auxiliary grid function that is obtained by finite differences applied to U_h $\widetilde{u}_h := \delta^{2\mathbf{m}}U_h$ in Ω_h, where the necessary values U_h outside Ω_h are chosen such that $\Delta_h\widetilde{U}_h = F(\mathbf{x})$ on the boundary of Ω_h (and sufficiently many points outside Ω_h), too. By construction $\widetilde{u}_h$ satisfies

$$\begin{array}{rcll} \Delta_h\widetilde{u}_h & = & f_h & \text{in } \Omega_h \\ \widetilde{u}_h & = & u_S^*(\mathbf{x}) + h^2\widetilde{r} & \text{on } \partial\Omega_h \end{array} , \tag{3.21}$$

where because of the smoothness of u along the boundary $\widetilde{r}$ is h-bounded. Thus u_h and $\widetilde{u}_h$ are solutions to almost the same numerical boundary value problem. Since 3.21 differs from (3.18) only by an $\mathcal{O}(h^2)$ perturbation in the boundary values, we can use the discrete maximum principle and show that u_h and $\widetilde{u}_h$ cannot differ by more than $\mathcal{O}(h^2)$. Furthermore $\widetilde{u}_h = \delta^{2\mathbf{m}}U^* + h^2\delta^{2\mathbf{m}}R = u^* + (\delta^{2\mathbf{m}}U^* - u^*) + h^2\delta^{2\mathbf{m}}R$. Now $\delta^{2\mathbf{n}}U^* - u^*$ and $\delta^{2\mathbf{n}}R$ are h-bounded in $\Omega\backslash\{\mathbf{x}_0\}$. This completes the proof. □

The Zenger Correction Method eliminates the pollution effect. Since the exact solution is still unbounded at the singularity the method cannot provide good accuracy at the singularity itself and therefore in standard global error norms on Ω. This is impossible with standard norms since the true solution tends to infinity where the singularity is located. Therefore we cut out a region $R \subset \Omega$ of fixed size, which contains the singularity when measuring the error. This results in a (pointwise) $\mathcal{O}(h^2)$ accuracy at any fixed distance from the singular point. Note that in a standard global norm (like L^∞) the method may not even converge, or may converge only slowly. In the literature (see e.g. [1]) special weighted norms are constructed to deal with singularities. The construction of such norms is dependent on the singularity. In contrast, our approach works for any singularity that is represented by a distribution and R can be chosen arbitrarily small. This will not affect the order of convergence, but only the constants in the estimate. Quantifying this dependence is possible but outside the scope of the present paper.

4. Extrapolation. In this section we present two extrapolation methods in order to improve the discretization error from $\mathcal{O}(h^2)$ to $\mathcal{O}(h^4)$.

4.1. Richardson Extrapolation. Richardson Extrapolation can be used if there exist asymptotic expansions of the discretization error (cf. [11]). In this case the solutions of different mesh sizes can be combined to eliminate the lower order terms. For our problem we use the mesh sizes h und $H = 2h$. In order to get the higher accuracy on the coarse grid we change the values there by

$$\widehat{\mathbf{u}}_H^* = \frac{4}{3}\mathcal{I}_h^H \mathbf{u}_h^* - \frac{1}{3}\mathbf{u}_H^* \ , \tag{4.1}$$

where $\mathcal{I}_h^H$ is an injection operator. With iterative solvers, such as multigrid $\mathbf{u}_h^*$ and $\mathbf{u}_H^*$ have to be computed with sufficient accuracy which may be nontrivial in a FMG setting.

4.2. τ-Extrapolation. τ-Extrapolation is a multigrid specific technique that in contrast to Richardson extrapolation works only on a single grid. It is based on the principle of *defect correction* and has been first mentioned by Brandt (cf. [2], see also Hackbusch [5], pp.278).

In the CS(correction scheme)-Multigrid algorithm two different iterations are used alternately, the smoother and the coarse grid correction (cf. [17]). These two iterations have a common fixed point described by $\mathbf{f}_h - \mathbf{A}_h\mathbf{u}_h = 0$ (cf. [14], p. 17f). The smoother converges fast for certain (usually the high frequency) solution components, but converges only slowly for the remaining (low frequency) modes. The coarse grid correction behaves vice versa. If these complementary properties are combined the typical multigrid efficiency is obtained.

Now we follow the idea of *double discretization*, i.e. in the coarse grid correction process higher order discretizations are used. Using a correction of the form

$$\mathbf{u}_h^{(k+1)} = \mathbf{u}_h^{(k)} + \mathbf{e}_h^{(k)} \ , \tag{4.2}$$

where $\mathbf{e}_h^{(k)}$ is computed as a coarse grid correction

$$\mathbf{e}_h^{(k)} = \mathcal{I}_H^h \mathbf{A}_H^{-1} \widehat{\mathcal{I}}_h^H (\mathbf{f}_h - \mathbf{A}_h \mathbf{u}_h^{(k)}) \ , \tag{4.3}$$

would lead to a standard multigrid method. τ-extrapolation consists in using a linear combination of fine and coarse grid residual to construct an extrapolated correction

$$\widehat{\mathbf{u}}_h^{(k+1)} = \mathbf{u}_h^{(k)} + \mathcal{I}_H^h \mathbf{A}_H^{-1} (\frac{4}{3}\widehat{\mathcal{I}}_h^H (\mathbf{f}_h - \mathbf{A}_h \mathbf{u}_h^{(k)}) - \frac{1}{3}(\widehat{\mathcal{I}}_h^H \mathbf{f}_h - \mathbf{A}_H \mathcal{I}_h^H \mathbf{u}_h^{(k)})) \ . \tag{4.4}$$

It can be shown that this modification of the coarse grid correction leads to a numerical error of order $\mathcal{O}(h^4)$ (cf. [13]). The modified coarse grid correction is only applied on the finest grid once per V-cycle. Additionally we have to take care when choosing the restriction and the interpolation operators. Normally trilinear interpolation for $\mathcal{I}_H^h$, full weighting for $\widehat{\mathcal{I}}_h^H$ and injection for $\mathcal{I}_h^H$ is used, but this can vary from problem to problem. One has also to pay attention not do do too many post smoothing steps, because this can destroy the higher accuracy. For the Poisson equation with singular source term we have to discretize the right hand side on each grid due to the fact that the restriction of the B-spline cannot approximate the right hand side well enough on the coarse grid. A concise analysis of the τ-extrapolation is e.g. found in [13].

5. Experimental results for a dipole in 3D. For the experiments we use CS-Multigrid as solver, e.g $CS(2, 2, 15)$ means that we do 2 presmoothing and 2 post-smoothing steps and a maximum of 15 V-cycles. As example we choose a dipole located at $\mathbf{x}_0 = (0.2501, 0.2501, 0.2501)^T$

with direction $\mathbf{v} = (0.5, 0.5, 0.71)^T$, $\|\mathbf{v}\|_2 = 1$ in the domain $\Omega = [0,1]^3$. To evaluate the accuracy away from the singularity we consider $\Omega \setminus R$, where $R = [0.125, 0.375]^3$ is a fixed neighborhood of $\mathbf{x}_0$. The analytical solution for the Poisson equation is in that case $u_d^*(\mathbf{x}) = \dfrac{\mathbf{v}\cdot\mathbf{x}}{4\pi|\mathbf{x}|^3}$. Therefore we solve the boundary value problem

$$\begin{aligned} -\Delta u(\mathbf{x}) &= \mathbf{v}\cdot\begin{pmatrix} \mathcal{H}_{\mathbf{x}_0}^{(2,1,1)}(\mathbf{x}) \\ \mathcal{H}_{\mathbf{x}_0}^{(1,2,1)}(\mathbf{x}) \\ \mathcal{H}_{\mathbf{x}_0}^{(1,1,2)}(\mathbf{x}) \end{pmatrix} && \text{in } \Omega \\ u(\mathbf{x}) &= u_d^*(\mathbf{x}) + g(\mathbf{x}) && \text{on } \partial\Omega \end{aligned} \tag{5.1}$$

with $g = \sin(x\pi)\sin(y\pi)\sinh(\sqrt{2}z\pi)$ and its discretization (using Zenger Correction of 4th order here)

$$\begin{aligned} -\Delta_h u_h(\mathbf{x}) &= \mathbf{v}\cdot\begin{pmatrix} \delta^{(4,4,4)}\mathcal{H}_{\mathbf{x}_0}^{(-2,-3,-3)}(\mathbf{x}) \\ \delta^{(4,4,4)}\mathcal{H}_{\mathbf{x}_0}^{(-3,-2,-3)}(\mathbf{x}) \\ \delta^{(4,4,4)}\mathcal{H}_{\mathbf{x}_0}^{(-3,-3,-2)}(\mathbf{x}) \end{pmatrix} && \text{in } \Omega_h \\ u_h(\mathbf{x}) &= u_{d,h}^*(\mathbf{x}) + g_h(\mathbf{x}) && \text{on } \partial\Omega_h \end{aligned} \quad . \tag{5.2}$$

The three tables show the discretization errors in different norms with and without additional extrapolation. The small numbers are the experimentally determined convergence rates α. An interesting fact is the high convergence rate of $\alpha = 3$ in the L_1-Norm in the domain Ω. It results from the closeness of the singularity to the next mesh point. The exact solution reaches a very high value there that causes a high numerical error. The maximum norm of the discretization error therefore stays approximately the same at the point next to the singularity and is higher than the sum of the errors of all the other points so the L_1-norm is determined by this one value.

TABLE 5.1
Convergence rates of the discretization error of problem (5.1) using Zenger Correction of 4th order and a $CS(2,2,12)$ solver.

h	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$		L_∞	L_1 Ω		L_2 Ω		L_1 $\Omega\backslash R$		L_2 $\Omega\backslash R$	
$\frac{1}{16}$	4.30e-02	2.0	2.62e+06	7.76e+02	3.1	4.51e+04	1.6	2.11e-02	2.1	2.80e-02	1.9
$\frac{1}{32}$	1.07e-02	2.0	2.62e+06	8.79e+01	3.1	1.52e+04	1.5	5.05e-03	2.0	7.38e-03	1.9
$\frac{1}{64}$	2.68e-03	2.0	2.62e+06	1.05e+01	3.0	5.24e+03	1.5	1.24e-03	2.0	1.93e-03	2.0
$\frac{1}{128}$	6.70e-04	2.0	2.62e+06	1.28e+00	3.0	1.83e+03	1.5	3.06e-04	2.0	4.95e-04	2.0
$\frac{1}{256}$	1.68e-04		2.62e+06	1.59e-01		6.43e+02		7.60e-05		1.25e-04	

TABLE 5.2
Convergence rates of the discretization error of problem (5.1) using Zenger Correction of 6th order with additional Richardson extrapolation and a $CS(2,2,14)$ solver.

h	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$		L_∞	L_1 Ω		L_2 Ω		L_1 $\Omega\backslash R$		L_2 $\Omega\backslash R$	
$\frac{1}{16}$	7.81e-05	3.9	2.62e+06	7.76e+02	3.1	4.51e+04	1.6	2.46e-04	3.4	1.15e-03	3.2
$\frac{1}{32}$	5.06e-06	4.0	2.62e+06	8.79e+01	3.1	1.52e+04	1.5	2.35e-05	3.7	1.22e-04	3.5
$\frac{1}{64}$	3.17e-07	4.0	2.62e+06	1.05e+01	3.0	5.24e+03	1.5	1.79e-06	3.9	1.04e-05	3.8
$\frac{1}{128}$	1.98e-08		2.62e+06	1.28e+00		1.83e+03		1.23e-07		7.55e-07	

6. Conclusion and Outlook. We have presented here the basic idea of the Zenger Correction Method including some simple examples. More examples, e.g. problems with point loads and

Table 5.3

Convergence rates of the discretization error of problem (5.1) using Zenger Correction of 6th order with additional τ-extrapolation and a $CS(3,0,13)$ solver.

h	$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$		L_∞	L_1 Ω		L_2 Ω		L_1 $\Omega\backslash R$		L_2 $\Omega\backslash R$	
$\frac{1}{16}$	1.14e-03	4.1	2.62e+06	7.76e+02	3.1	4.51e+04	1.6	3.51e-03	3.3	1.61e-02	3.0
$\frac{1}{32}$	6.55e-05	4.0	2.62e+06	8.79e+01	3.1	1.52e+04	1.5	3.69e-04	3.5	2.08e-03	3.2
$\frac{1}{64}$	4.05e-06	3.4	2.62e+06	1.05e+01	3.0	5.24e+03	1.5	3.20e-05	3.8	2.25e-04	3.6
$\frac{1}{128}$	3.88e-07	2.4	2.62e+06	1.28e+00	3.0	1.83e+03	1.5	2.38e-06	3.7	1.82e-05	3.8
$\frac{1}{256}$	7.38e-08		2.62e+06	1.59e-01		6.43e+02		1.86e-07		1.29e-06	

quadrupoles can be found in [7]. Currently we are working an application of our method as a dipole model for the biolelectric field problem. The method has been integrated into the NeuroFEM software package (cf. [18], [8]). In a forthcoming article we will also give a more detailed description of our method in the finite element context.

REFERENCES

[1] H. Blum and R. Rannacher, *Extrapolation techniques for reduzing the pollution effect of reentrant corners in the finite element method*, Numer. Math., 52 (1988), pp. 539–564.

[2] A. Brandt, *On the accurate computation of singular solutions of Laplace's and Poisson's equation*, Multigrid Methods: 1984 guide with applications to fluid dynamics, GMD Studie Nr. 85, St. Augustin, (1984).

[3] R. Estrada and R. P. Kanwal, *Asymptotic Analysis: A Distributional Approach*, Birkhäuser, Basel, 1994.

[4] R. Fössmeier, *Differenzenverfahren hoher Ordnung für elliptische Randwertprobleme mit gekrümmten Rändern*, PhD thesis, Technische Universität München, 1984.

[5] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer Verlag, 1985.

[6] P. Hofmann, *Asymptotic expansions of the discretization error of boundary value problems of the Laplace equation in rectangular domains*, Numerische Mathematik, 9 (1967), pp. 302–322.

[7] H. Köstler, *Akkurate Behandlung von Singularitäten bei partiellen Differentialgleichungen*, diplomarbeit, Friedrich-Alexander Universität Erlangen-Nürnberg, 2003.

[8] Max Planck Institute for Human Cognitive and Brain Sciences, *Neurofem web page*, 2004. http://www.neurofem.com.

[9] M. Mohr, *Simulation of Bioelectric Fields: The Forward and Inverse Problem of Electro-encephalographic Source Analysis*, PhD thesis, Friedrich-Alexander Universität Erlangen-Nürnberg, 2004.

[10] F. Olver, *Asymptotics and Special Functions*, Academic Press, New York, 1974.

[11] L.F. Richardson, *The deferred approach to the limit. I. Single lattice*, Phil. Trans. Roy. Soc. London A, 226 (1927), pp. 229–349.

[12] U. Rüde and H. Köstler, *Accurate techniques for computing singular solutions of elliptic problems*, Tech. Report 04-3, Friedrich-Alexander Universität Erlangen-Nürnberg, 2004.

[13] U. Rüde, *Multiple τ-extrapolation for multigrid methods*, Tech. Report I-8701, Technische Universität München, 1987.

[14] ———, *On the accurate computation of singular solutions of Laplace's and Poisson's equation*, Multigrid Methods: Theory, Applications, Supercomputing: Proceedings of the Third Copper Mountain Conference on Multigrid Methods, April 5-10, (1987).

[15] P. Schimpf, J. Haueisen, and C. Ramon, *Numerical approximations of ideal dipoles*, Proceedings of the 13th International Conference on Biomagnetism, F. S. University, Jena, Germany, Aug 10-14, (2002).

[16] I. Stackgold, *Green's Functions and Boundary Value Problems*, John Wiley & Sons, 2nd ed., 1998.

[17] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, 2001.

[18] C. H. Wolters, M. Kuhn, A. Anwander, and S. Reitzinger, *A parallel algebraic multigrid solver for finite element method based source localization in the human brain*, Computing and visualization in science, 5 (2002), pp. 165–177.

[19] C. Zenger and H. Gietl, *Improved schemes for the Dirichtlet problem of Poisson's equation in the neighbourhood of corners*, Numerische Mathematik, 30 (1978), pp. 315–332.

COARSE GRID CLASSIFICATION: A PARALLEL COARSENING SCHEME FOR ALGEBRAIC MULTIGRID METHODS

BRAM METSCH*

Abstract. In this paper we present a new approach to the parallelization of AMG, i.e., to the parallel coarse grid selection in AMG. Our approach does not involve any special treatment of processor subdomain boundaries and hence avoids a number of drawbacks of other AMG parallelization techniques. The key idea is to select a coarse grid on each processor from multiple possible grids such that the composed coarse grid forms a suitable coarse grid for the whole domain, i.e. there is no need for any boundary treatment. First, we construct multiple equivalent coarse grids on each processor subdomain. Then, we select one per processor by graph clustering techniques. The results of our numerical experiments clearly indicate that this approach results in high quality coarse grids which are very close to those obtained in sequential AMG. This leads to a significant better speed-up of our algorithm compared with other parallel coarsening techniques. Furthermore, the operator and grid complexities of our parallel AMG are often smaller than those obtained by other parallel AMG methods.

1. Introduction. The solution of large sparse linear systems $Au = f$ is an essential ingredient in most scientific computations. This solution step must involve only a similar amount of operations and computational storage as the discretization to allow for an efficient and scalable numerical simulation. Such a sparse linear solver which requires only $O(N)$ operations and memory for N degrees of freedom is called optimal. However, there exists no general (direct) algebraic solution technique yielding this complexity.

For linear systems arising from a finite difference or finite element discretization of an elliptic partial differential equation (PDE) on a structured grid, geometric multigrid methods [10] give such an optimal (iterative) solver. But for many applications it is difficult to construct a sequence of (nested) discretizations or meshes needed for geometric multigrid. Furthermore, geometric multigrid methods are in general not robust with respect to a deterioration of the coefficients of the operator or singular perturbations. In the 1980s algebraic multigrid (AMG) methods were developed [4, 5, 6, 14] to cope with these problems by extending the main ideas of geometric multigrid methods to a purely algebraic setting. This approach is based only on information available from the linear system to be solved. However, this flexibility has a price: A *setup phase*, in which the multigrid hierarchy, i.e., the sequence of grids, transfer operators and coarse grid operators, is constructed, has to be carried out before the well-known multigrid cycle (the *solution phase*) can be started. While the parallelization of the solution phase is straightforward, e.g. [15], the techniques for constructing the coarse grids in AMG are inherently sequential which makes the parallelization of the setup phase a challenging task.

Many different approaches to the parallelization of this step have been proposed over the years, e.g. [11, 12]. Most of them follow a data decomposition approach and need to employ a special treatment of the processor subdomain boundaries to cope with non-matching coarse grids. While the interior of each processor subdomain is still coarsened in the classical way, the coarse grid structure on the subdomain boundaries does not respect the classical coarsening principles. However, with the fine grid discretization fixed, the distribution of the domain among an increasing number of processors leads to smaller subdomain interiors and therefore smaller parts which are actually coarsened by classical AMG. This leads to a significant deterioration of the speed-up of these parallel AMG strategies.

In this paper we present a method for the construction of high quality coarse grids in parallel without an explicit boundary treatment, however, we still use the classical coarsening scheme [14]. We first construct multiple coarse grids on each processor domain individually, from these coarse grids, we then choose exactly one per processor domain by graph clustering techniques such that the composed coarse grid shows no inconsistencies near the processor subdomain boundaries. Hence,

*Institut für Numerische Simulation, Universität Bonn, Wegelerstraße 6, D-53115 Bonn

no explicit boundary treatment is required which leads to operator and grid complexities similar to those obtained in the sequential case. Moreover, as no additional coarse grid points are added on or near the processor subdomain boundary, a growing number of processors —and thus a growing number of subdomain boundary points— has only little impact on the coarse grid. Hence, the speed-up of proposed CGC method is much better than the speed-up of the algorithms employing an explicit boundary treatment and the scale-up of our algorithm is optimal.
The remainder of the paper is organized as follows: In Section 2 we give a short review of the (sequential) AMG algorithm and the respective heuristics, which led to its development. Then, we present the main challenges which classical AMG can encounter in the parallel case. We then summarize briefly, in Section 3, the previous parallel coarsening methods and their shortcomings. In Section 4 we present our coarse grid classification (CGC) algorithm. We discuss the results of our numerical experiments in Section 5 where we compare the quality of the CGC algorithm with the other methods considering discretizations of PDE's in two and three dimensions with several millions of unknowns on a massively parallel computer with up to 256 processors. The results of these experiments clearly show the smaller memory and compute time requirements of our algorithm with a multigrid convergence rate similar to that of sequential AMG. As expected, we achieve a significantly better speed-up and an optimal scale-up for our CGC algorithm compared with other parallelization techniques. Finally, we conclude with some remarks in Section 6.

2. Algebraic multigrid (AMG). In this section we give a short review of the algebraic multigrid method. We consider a linear system $Au = f$, which comes from a discretization of a PDE on a grid $\Omega = \{1, \ldots, N\}$, with $A = (a_{ij})_{i,j=1}^N$ being a large sparse real matrix, $u = (u_i)_{i=1}^N$ and $f = (f_i)_{i=1}^N$ vectors of length N. We assume that A is a symmetric, positive definite M-matrix or an essentially positive matrix. Algebraic multigrid methods were first introduced by A. Brandt in the early 1980s [4, 5, 6]. Here, we use information from the fine grid operator $A = A^1$ only to construct a sequence of coarse grids $\{\Omega^l\}_{l=1}^L$, transfer operators $\{P^l\}_{l=1}^{L-1}$ and coarse grid operators $\{A^l\}_{l=1}^L$.[1] This construction is carried out during the *setup phase*, as displayed in Program 1. In

Program 1 *AmgSetup*$(\Omega, A, N_{min}, L_{max}, L, \{A^l\}_{l=1}^L, \{P^l\}_{l=1}^{L-1}\{R^l\}_{l=1}^{L-1})$

```
begin
  Ω^1 ← Ω;
  A^1 ← A;
  for l ← 1 to L_max − 1 do
      partition Ω^l into C^l and F^l;                    coarsening step
      Ω^{l+1} ← C^l;
      compute interpolation P^l;                         interpolation step
      R^l ← (P^l)^T;
      A^{l+1} ← R^l A^l P^l;                             Galerkin step
      if |Ω^{l+1}| ≤ N_min then break; fi;
  od;
  L ← l + 1;
end.
```

this paper, we focus on the coarsening step, i.e. the splitting of the grid Ω^l into two disjoint sets C^l and F^l, i.e. $\Omega^l = C^l \dot\cup F^l$. The set of *coarse grid points* C^l is taken as the grid on the next level, $\Omega^{l+1} := C^l$, while the remaining points $j \in F^l$ are denoted as *fine grid points*. For the sake of simplicity, we will omit the level index l in the following. The conditions for the coarse grid selection are imposed by the requirement of a stable interpolation formula. Recall from multigrid theory [10, 14] that the low-frequency error components (also called smooth error components) have to

[1] In classical geometric MG, Ω^L denotes the finest grid and Ω^1 the coarsest grid. Here, as usual in AMG, the levels are numbered starting with Ω^1 as the finest grid.

be eliminated by the coarse grid correction step. Hence the interpolation has to approximate the smooth error components accurately. With such an interpolation P^l, we then define the restriction $R^l = (P^l)^T$ and the coarse grid operator $A^{l+1} = R^l A^l P^l$ by the Galerkin identity. The setup phase is then restarted recursively with A^{l+1} as input.

For M-matrices and essentially positive matrices $A = (a_{ij})_{i,j=1}^N$, a smooth error varies slowly in the direction of large negative couplings [14], e.g. $a_{ij} \ll 0$. This observation leads to the definition of the *strong couplings* S_i for a grid point i, which are the set of grid points

$$S_i := \{j \neq i : \ |a_{ij}| \geq \alpha \max_{k \neq i} |a_{ik}|\}, \quad S_i^T := \{j \neq i : \ i \in S_j\}$$

with a typical value of $\alpha = 0.25$. A stable interpolation formula for a fine grid point $i \in F$ should respect the strong couplings S_i to both coarse grid points and other fine grid points. Such a formula is given by the *standard interpolation scheme* [14]. However, for each fine grid point $j \in S_i \cap F$, the value at j should also be influenced by the coarse grid points $m \in S_i \cap C$, which influence i directly, to a reasonable amount, i.e. the inequality

$$\frac{\sum_{k \in S_i \cap C} a_{jk}}{\max_k |a_{jk}|} > \beta \cdot \frac{a_{ij}}{\max_k |a_{ik}|}, \tag{2.1}$$

with $\beta = 0.35$ typically, should hold. This condition ensures that for each fine grid point i with strong connections to other points j, a sufficient amount of these couplings lead to coarse grid points $k \in S_i \cap C$, especially, that every fine grid point with strong couplings has at least one strong connection to a coarse grid point.

We now formulate three conditions on the coarse grid.

C1 Let $i \in F$. Then each $j \in S_i$ is either in C or condition (2.1) holds

C2 For $i \in C$ and $j \in S_i \cup S_i^T$ should $j \notin C$. ($C \subset \Omega$ is an *independent* or *stable* set.)

C3 C is a maximal set with these properties.

In general, it is not possible to satisfy all these conditions. For the stability of interpolation, condition C1 is important and will be enforced strictly, while the second condition is intended to limit the size of the grid and thus to reduce the memory and compute time requirements. On the other hand, a great amount of coarse grid variables yields a high quality of smooth error interpolation which motivates C3.

The Ruge-Stüben coarsening algorithm [14] first assigns a weight $\lambda_i = |S_i^T|$ to each $i \in \Omega$. Then, in each iteration, a point with maximal weight is determined and inserted into the coarse grid C while all $j \in S_i^T$ are assigned to the fine grid F. Then, for each such j, the weights of all still unassigned points $k \in S_j$ are increased and the weights of all unassigned $j \in S_i$ are decreased. If the maximal weight is zero, the iteration is stopped. We now have constructed an independent set of coarse grid points C. Note that there is some freedom in choosing the first point of the coarse grid, a fact we will utilize in the development of our parallel coarsening algorithm.

After applying the classical coarsening algorithm, the condition (2.1) may be violated for a small number of strongly connected fine grid points $i, j \in F$ (the so-called $F - F$-couplings). This is fixed by the "second pass", which checks all such pairs and inserts i or j into the coarse grid if necessary.

Note that each point i chosen for the coarse grid C results in a change of the weights of all points within two layers around the grid point i. This states the sequential character of the coarsening algorithm, as the weight updates propagate throughout the whole domain during the iteration.

3. Parallel coarsening. In this section we review some previously developed approaches to the construction of coarse grids in parallel. Most of them are based on the classical Ruge-Stüben coarsening scheme presented in Section 2 and employ a special boundary treatment. The CLJP algorithm [7] implements a different coarsening scheme.

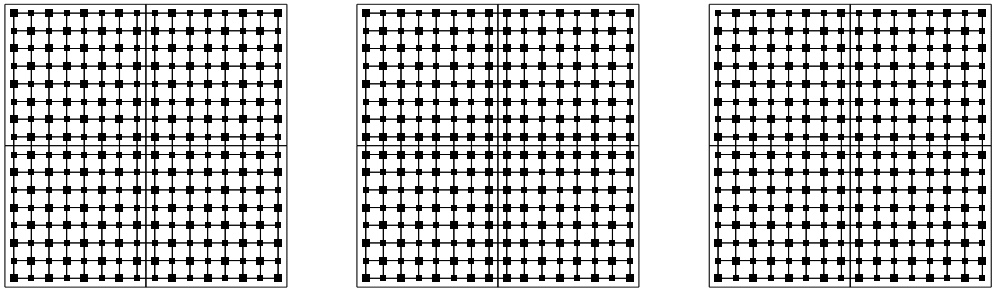


FIG. 3.1. *Application of the various parallelization techniques (left: NONE, center: RS3, right: CGC) to the 5-point discretization of the Laplace operator, distributed among 4 processors. Depicted are the C/F-splittings on the finest level, where the small squares indicate the fine grid points $i \in F$ and the large squares the coarse grid points $j \in C$.*

Note that if we simply apply the sequential AMG coarsening algorithm to each processor domain separately (i.e. without regarding points on neighboring domains), inconsistencies along the processor boundaries can occur as shown in Figure 3.1. (We refer to this scheme in the following as NONE.) We see that F-variables are coupled strongly across the subdomain boundaries, furthermore both involved F-points are not strongly coupled to a common C-point which is a massive violation of condition (2.1). In the following, we describe which measures have been taken to overcome this problem.
We denote by Ω_p the part of Ω that is being handled by processor $p \in \{1, \ldots, P\}$. The boundary $\partial\Omega_p$ is defined as the points $i \in \Omega_p$ which are coupled to points $j \notin \Omega_p$ on other processors, i.e.

$$\partial\Omega_p := \{i \in \Omega_p : \ a_{ij} \neq 0 \text{ for some } j \notin \Omega_p\}.$$

Minimum Subdomain Blocking. A first approach to parallel AMG was given by Krechel and Stüben in [12]. This so-called *minimum subdomain blocking* (MSB) decouples the coarsening on the individual processor domains by first coarsening the processor boundaries $\partial\Omega_p$ by means of the classical coarsening scheme considering only couplings *inside* of $\partial\Omega_p$. This ensures that each fine grid point $i \in \partial\Omega_p$ has a strong connection to a coarse grid point $j \in \partial\Omega_p$. After the boundaries are coarsened, the interior of each subdomain is coarsened using the classical Ruge-Stüben algorithm. However, strong couplings across the processor boundaries are ignored in this approach, i.e. neither condition (2.1) nor C2 is checked, and the grid layout inside the boundary may not respect anisotropies of the underlying operator.

Third pass (RS3) coarsening. Another method for processor boundary treatment is the third pass coarsening method [11]. This method fixes strong $F-F$-connections across processor boundaries by applying the second phase of the classical Ruge-Stüben algorithm to all pairs (i, j), with $i \in \partial\Omega_p \cap F$, $j \in (S_i \cap F) \setminus \Omega_p$, of strongly coupled fine grid points on different processors in an additional *third phase*, after the first and second phase of the classical coarsening algorithm have been carried out on each processor subdomain individually. Figure 3.1 shows the coarse grids obtained by the RS3 algorithm for a discretization of the Laplacian with the 5-point stencil. In this case, *every* boundary point $i \in \partial\Omega_p$ was added to the coarse grid by the third pass to obtain a stable interpolation.

CLJP coarsening. A completely different way of coarsening is given by the CLJP algorithm [7, 11]. It is based on parallel graph coloring methods of M. Luby, M. T. Jones and P. E. Plassman. The coarsening is done during multiple iterations, each of them including a communication phase. In each step, an independent set is chosen from the vertices of the directed graph defined by the

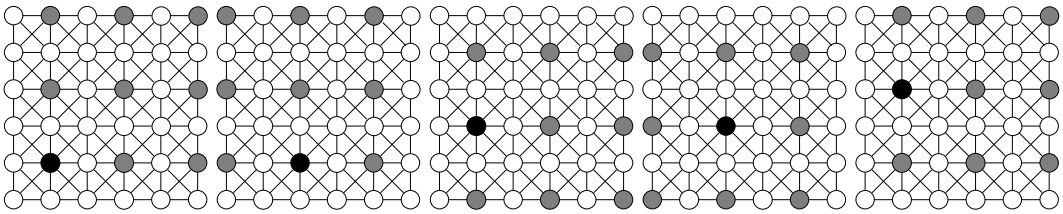


FIG. 4.1. *Resulting coarse grids for a 9-point discretization of the Laplace operator constructed by five different initial choices. The gray points indicate the respective coarse grid points, the black point indicates the first coarse grid point chosen.*

strong couplings. The points inside this set are added to the set of coarse grid points C and all incident edges are removed from the graph. A point is inserted into F if all its incident edges have been removed and it is not already member of C. The iteration stops if all edges of the graph have been removed. As strongly coupled neighbors $j \in S_i$ of coarse grid points i are not immediately determined to be fine grid points, this algorithm constructs a coarse grid of significantly greater magnitude than the classical coarsening scheme, which in turn increases the operator complexity and the number of floating-point operations during the solution phase.

Falgout coarsening. The so-called Falgout coarsening [11] is a combination of the classical Ruge-Stüben coarsening and the CLJP coarsening. First, each processor domain is being coarsened using the classical algorithm. Then, the coarse grid points in the *interior* of each processor domain, $i \in C \cap (\Omega_p \setminus \partial\Omega_p)$, are taken as the first independent set for the CLJP algorithm which proceeds until all points have been assigned. This way, the interior of a domain is coarsened as in the classical method, while the boundaries are coarsened "CLJP-like". This also means that many coarse grid points are chosen near (and not only on) the subdomain boundary, which in turn increases the operator and grid complexity.

4. Coarse Grid Classification. In the previous section we discussed several methods for generating coarse grids in parallel. Most of these algorithms include some kind of special boundary treatment and thus construct a different coarse grid structure for the interior and the boundary of a processor subdomain. However, this boundary treatment can lead to a large number of additional coarse grid points near the boundary (or even throughout the whole domain when using the CLJP method), thus requiring a higher amount of memory and compute time.

In our approach, we will not change the classical point selection scheme but keep the original Ruge–Stüben scheme on each processor, which has been shown to produce good results in the sequential case. At first sight this might seem contradictory to the demand for a true parallel algorithm since the MSB, RS3 and Falgout algorithms all need a special boundary treatment. But in fact another property of the classical Ruge-Stüben scheme will allow us to avoid an explicit boundary treatment despite its sequential nature.

Since the points are chosen sequentially, it is clear that by changing the initial choice for the first coarse grid points the Ruge-Stüben algorithm may give a different coarse grid, see Figure 4.1. Note that the quality of the resulting coarse grids with respect to multigrid convergence and memory requirements is very much similar. Hence, there is no special advantage of using either one of these coarse grids in a sequential computation. In parallel computations, however, this gives us a degree of freedom to *match* the coarse grids obtained on each processor individually by the Ruge–Stüben scheme at processor subdomain boundaries consistently. This observation is the starting point for our parallel *coarse grid classification* (CGC) algorithm.

More precisely, our approach is as follows. First, we construct multiple coarse grids on each processor domain independently by running the classical algorithm multiple times with different

equivalent starting points. Note that this is computationally efficient, since the classical Ruge–Stüben algorithm requires only a very small amount of computational time compared with the construction of the transfer and coarse grid operators while a well-constructed grid can save a great amount of time during the operator construction and in the multigrid cycle.
After the construction of these coarse grids on all processors, we need to select exactly one grid for each processor domain such that the union of these coarse grids forms a suitable coarse grid for the whole domain. We achieve this by defining a weighted graph whose *vertices* represent the *grids* constructed by the multiple coarsening runs. *Edges* are defined between vertices representing grids on neighboring processor domains and each edge weight measures the quality of the boundary constellation if these two grids are chosen to be part of the composed grid. Finally, we use this graph to choose one coarse grid for each processor domain which automatically matches with most of its neighbors.
In our implementation, see Program 2, each processor p first determines the maximal weight $\lambda_{\max}$

Program 2 CGC algorithm $CGC(S, S^T, ng, \{C_i\}_{i=1}^{ng}, \{F_i\}_{i=1}^{ng})$

```
for j ← 1 to |Ω| do λ_j ← |S_j^T|; od;
C_0 ← ∅;  λ_max ← arg max_{k∈Ω} λ_k;
do
    U ← Ω \ ⋃_{i≤it} C_i;
    if max_{k∈U} λ_k < λ_max then break; fi;
    it ← it + 1;  F_it ← ∅;  C_it ← ∅;
    do
        j ← arg max_{k∈U} λ_k;
        if λ_j = 0 then  break; fi;
        C_it ← C_it ∪ {j};  λ_j ← 0;
        for k ∈ S_j^T ∩ U do
            F_it ← F_it ∪ {k};  λ_k ← 0;
            for l ∈ S_k ∩ U do λ_l ← λ_l + 1; od;
        od;
        for k ∈ S_j ∩ U do λ_k ← λ_k − 1; od;
    od;
od
ng ← it;
```

of all points $i \in \Omega_p$. Every point with this weight can be chosen as a starting point for the classical coarsening algorithm. We choose one and construct a coarse grid $C_{(p),1}$. Now we select another point with weight $\lambda_{\max}$ which is not already a member of $C_{(p),1}$ and construct a second coarse grid $C_{(p),2}$ starting with this point. Note that we construct disjoint coarse grids $C_{(p),1} \cap C_{(p),2}$ only. We repeat these steps as long as there is a point with weight $\lambda_{\max}$ that is not already a member of a coarse grid $C_{(p),it}$. Note that the number of iterations is bounded by the maximal number of strong couplings $|S_i|$ over all points $i \in \Omega_p$, which in turn is bounded by the maximal stencil width. Hence, the number of constructed grids per processor ng_P is independent of the number of unknowns N and the number of processors P.
We now have obtained ng_p valid coarse grids $\{C_{(p),i}\}_{i=1}^{ng_p}$ on each processor p. To determine which grid to choose on each processor, we construct a directed, weighted graph $G = (V, E)$ whose vertices represent the created coarse grids,

$$V_p := \{C_{(p),i}\}_{i=1,\ldots,ng_p}, \quad V := \cup_{p=1}^{P} V_p.$$

The set of edges E consists of all pairs (v, u), $v \in V_p$, $u \in V_q$ such that $q \in \mathcal{S}_p$ is a neighboring processor of p,

$$E_p := \{\cup_{q\in\mathcal{S}_p} \cup_{v\in V_p,\ u\in V_q} (v, u)\}, \quad E := \cup_{p=1}^{P} E_p,$$

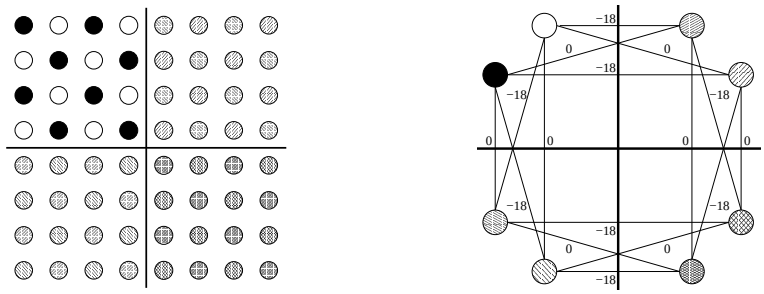


FIG. 4.2. *Application of the CGC algorithm to a 5-point stencil, distributed among 4 processors. The figure on the left shows the assignment of the points to the coarse grids, the figure on the right shows the weighted graph.*

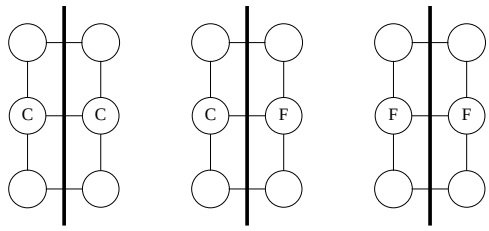


FIG. 4.3. *Three possible C/F-constellations at a processor's domain boundary.*

where $\mathcal{S}_p$ is defined as the set of processors with points strongly influencing points on processor p,

$$\mathcal{S}_p := \{q \neq p : \ \exists i \in \Omega_p, \ j \in \Omega_q : \ j \in S_i\}.$$

To determine the weight $\gamma(e)$ of the edge $e = (v, u)$, we consider the nodes $v \in V_p$, $u \in V_q$. Each of these particular nodes represents a C/F-splitting (C_p, F_p) or, respectively, (C_q, F_q). Together they form a C/F-splitting for the domain $\Omega_p \cup \Omega_q$. At the processor subdomain boundary, three grid configurations can occur, see Figure 4.3. We denote by $c_{C,C}$ the number of strong $C-C$-couplings (left), by $c_{C,F}$ the number of strong $C-F$-couplings (center) and by $c_{F,F}$ the number of strong $F-F$-couplings (right). Based on these classes of couplings we define the edge weight

$$\gamma(e) := c_{C,C}\gamma_{C,C} + (c_{C,F} + c_{FC})\gamma_{C,F} + c_{F,F}\gamma_{F,F}$$

with $\gamma_{C,C}, \ \gamma_{C,F}, \ \gamma_{F,F} \in \mathbb{R}$ defined as follows. The most important case are the $F-F$-couplings. Here, two fine grid points $i \in F_p$ and $j \in F_q$ are strongly coupled, which can lead to two problems: These two points may not have a common C-point to interpolate from, which violates condition (2.1). On the other hand, even if (2.1) is satisfied, we have to transfer the matrix rows A_i and A_j to construct a stable interpolation operator. Therefore, this situation must be avoided, which motivates us to penalize strong $F-F$-couplings with a large negative weight $\gamma_{F,F} := -8$.
The strong $C-C$-couplings should also be avoided because they can increase the operator and the grid complexity. We therefore set $\gamma_{C,C} := -1$. In the remaining case, which can be considered as the (optimal) sequential coarsening scenario, we do not add an additional weight, i.e. $\gamma_{C,F} := 0$. Figure 4.2 shows the graph $G = (V, E)$ obtained by our CGC algorithm for a 5-point discretization of the Laplacian. We can observe that constellations with strong $C-C$–couplings and strong $F-F$–couplings are heavily penalized, while constellations with strong $C-F$–couplings only are weighted by zero.
Now that we have constructed the graph, we can use it to choose an appropriate coarse grid for each processor. Observe that the number of vertices is related to the number of processors P; i.e.,

it is much smaller than the number of unknowns N. The cardinality of E is also small compared to N since edges are only constructed between neighboring processors. Thus, we can transfer the whole graph onto a single processor without high communication costs.[2] On this processor, we choose exactly one node v_p from each subset $V_p \subset V$ with the following algorithm, where we denote by $\mathcal{C}$ the set of the selected coarse grids. We now formulate three conditions on the coarse grid.

1. First, we define *heavy edges or couplings* H_v between the nodes v of the graph, where p denotes the processor v belongs to (i.e. $v \in V_p$),

$$H_v := \cup_{q \in \mathcal{S}_p} \{w \mid \gamma(v,w) = \max_{u \in V_q} \gamma(v,u)\}, \;\; H_v^T := \{w \mid v \in H_w\}.$$

   The heavy edges indicate which coarse grid on processor q can be fitted best to the coarse grid represented by $v \in V_p$. We assign a weight λ_v to each node v, where $\lambda_v = |H_v| + |H_v^T|$. This weight indicates how many coarse grids on other processors can be fitted to the coarse grid represented by v.
2. For some p, all nodes $v \in V_p$ might have weight $\lambda_v = 0$. As this means there are no strong couplings across the subdomain boundary, *any* grid constructed on this processor can be chosen. Here, we choose one $v \in V_p$ and remove V_p from the graph.
3. We choose the node v with maximal weight, put it into $\mathcal{C}$ and remove the subset V_p (v being a member of V_p) from the graph, as a coarse grid for domain Ω_p is now determined. We then increase the weight of each node $w \in H_v \cup H_v^T$ to the maximal weight of all remaining nodes in the graph plus one (so that one of these will be chosen in the next step), and repeat this step as long as the graph is not empty.

Program 3 *AmgCGCChoose*$(V, H, \mathcal{C})$

```
begin
  C ← ∅;
  U ← V;
  for v ∈ U do λ_v ← |H_v| + |H_v^T|;  od;
  for p ← {1, ..., P} do
      if λ_v = 0 for all v ∈ V_p then C ← {v};  U ← U \ V_p;  fi;          arbitrary v ∈ V_p
  od;
  while U ≠ ∅
        do
        v ← arg max_{w∈U} λ_w;  C ← C ∪ {v};  U ← U \ V_p such that v ∈ V_p;
        λ_max ← max_{w∈U} λ_w;  for w ∈ (H_v ∪ H_v^T) ∩ U do λ_w ← λ_max + 1;  od;
  od;
end
```

This procedure takes up to P steps, one for each processor domain, see Program 3 for details. After running the algorithm, we transfer the choice $v_p \in \mathcal{C} \cap V_p$ back to processor p. The union of all elements of $\mathcal{C}$ now defines the global consistent coarse grid for the complete domain, see Figure 3.1.

Recall that after applying the first phase of the Ruge–Stüben coarsening, there may exist a few fine grid points with strong connections to other fine grid points only. However, these strong couplings are very rare. In the sequential phase, this is corrected by the second pass of the classical coarsening scheme. To correct these very few couplings across processor boundaries, we employ a more straightforward method: We just check whether a fine grid point is strongly coupled to other fine grid points only and insert this point into the coarse grid if that is the case.

[2] For a very large number of processors, one can extend this algorithm by splitting the graph among a subset of the processors instead of transferring it onto a single processor. A local matching and a recursive application of our classification then gives the global consistent coarse grid.

5. Numerical Results. We now compare the coarsening algorithms experimentally. We implemented all coarsening techniques discussed in this paper using the library PETSc [1, 2, 3]. Two important measures for the quality of the hierarchy built during the setup phase are the *operator complexity* c_A and the *grid complexity* c_G,

$$c_A := \frac{\sum_{l=1}^{L_{\max}} nonzeros(A^l)}{nonzeros(A^1)}, \quad c_G := \frac{\sum_{l=1}^{L_{\max}} |\Omega^l|}{|\Omega^1|}. \tag{5.1}$$

These values give an indication of the increase of memory requirement during the coarsening algorithm compared to the size of the original linear system.
In our experiments, we use $\alpha = 0.25$ for the first pass and $\beta = 0.35$ for the second pass of the classical Ruge-Stüben algorithm. We interpolate using the truncated standard interpolation [14] using a truncation parameter $\epsilon_{tr} = 0.2$. If more than 70% of the points are boundary points, e.g. $i \in \partial\Omega_p$ for some p, then an agglomeration step takes place; i.e. two neighboring subdomains are merged. The setup phase is stopped if no strong couplings are present.
As an initial guess for the solution phase we use a random-valued vector u_0 with $\|u_0\|_{l_2} = 1$. We use a (subdomain) block-Jacobi smoother with one inner Gauss–Seidel relaxation step and employ one pre- and one post-smoothing step per V-cycle. The iteration is stopped if the l^2-norm of the residual $r_{it} = f - Au_{it}$ drops below 10^{-10}. The convergence factor is determined as $\rho = (\|r_{it}\|_{l^2}/\|r_1\|_{l^2})^{\frac{1}{it-1}}$. The examples were run on a massively parallel computer consisting of 41 IBM p690 with 32 CPUs and 128GB memory each, connected through an HPS network.

Example 5.1 (Laplace operator). In our first experiment we consider the Poisson problem on a rectangular domain,

$$-\Delta u = f, \tag{5.2}$$

with vanishing Dirichlet boundary conditions, discretized using a 5-point finite difference stencil on 511×511 points per processor. Table 5.1 shows the setup times and operator complexities for this

TABLE 5.1
Setup time in seconds and operator complexity (5.1) *for Example 5.1.*

P	NONE	MSB	RS3	CLJP	Falgout	CGC
1	21.3 / 2.59	21.3 / 2.59	21.3 / 2.59	26.8 / 3.85	21.3 / 2.59	21.3 / 2.59
2	28.2 / 2.61	29.2 / 2.64	28.5 / 2.62	35.9 / 3.87	30.1 / 2.61	27.7 / 2.59
4	32.3 / 2.60	33.7 / 2.68	31.8 / 2.62	40.5 / 3.89	34.2 / 2.62	32.3 / 2.60
8	40.6 / 2.60	42.6 / 2.70	45.2 / 2.62	52.6 / 3.89	44.8 / 2.64	41.1 / 2.61
16	59.6 / 2.61	61.7 / 2.72	59.2 / 2.64	78.0 / 3.90	62.7 / 2.64	60.2 / 2.61
32	99.6 / 2.61	104 / 2.73	99.7 / 2.65	140 / 3.90	104 / 2.64	99.8 / 2.61
64	343 / 2.62	354 / 2.74	349 / 2.66	451 / 3.90	350 / 2.64	343 / 2.61
128	836 / 2.62	853 / 2.75	847 / 2.67	1084 / 3.90	844 / 2.64	838 / 2.61
256	1731 / 2.63	1762 / 2.76	1757 / 2.67	2269 / 3.57	1746 / 2.64	1728 / 2.61

problem, obtained with up to 256 CPUs of the IBM cluster. We see the CGC method requires the same amount of time as the sequential algorithm (NONE), as no boundary treatment is involved and the additional coarsening runs require essentially no time compared to the construction of the coarse grid operator. The RS3 algorithm also allows comparable setup times, as the additional coarse grid points at the boundary simplify the construction of the interpolation operator. The most expensive algorithm is, as expected, the CLJP algorithm which does not allow fast coarsening. The large increase of the setup times for large processor numbers is due to the fact that only 32 processors on the same node can communicate very efficiently while slower network communication is involved for larger processor numbers. Regarding the operator complexities, we see the CLJP algorithm shows large values due to the amply selection of coarse grid points. The CGC algorithm

FIG. 5.1. *Free surfaces of the water drop and basin. Depicted is the constellation leading to the coarse grids in Figure 5.2.*

achieves even slightly better values than the sequential algorithm (NONE), as the grids are matched and $C - C$ couplings across the boundaries are avoided. The same holds for the grid complexity (5.1), which equals 1.97 for the CLJP algorithm, while the other methods reach values up to 1.69 only. For this standard test case, we have measured convergence factors of $\rho \approx 0.19$ for the CLJP algorithm whereas the convergence factors for the sequential algorithm (NONE) grow up to $\rho \approx 0.3$ as the number of processors increases. We also observed an increase of the convergence factors up to $\rho \approx 0.28$ for the RS3 algorithm. We found $\rho \approx 0.16$ for the MSB algorithm and $\rho \approx 0.14$ for the Falgout and the CGC algorithms. In summary, this experiment shows that the proposed CGC scheme gives a parallel AMG with similar scale-up behavior as the RS3 and Falgout scheme. Furthermore, the CGC scheme leads to operator and grid hierarchies with slightly better complexities. To fully assess the potential of the CGC method, however, we must be concerned also with its speed-up properties we consider in our next example.

Example 5.2 (Navier–Stokes Equations). In this example, we use our parallel AMG to solve the three-dimensional Pressure-Poisson equation in a two-phase flow simulation; i.e., there is a large jump in the diffusion coefficients across the phase boundary. Our finite difference based CFD solver [8] determines the pressure p^{n+1} of the next time step $n+1$ by solving the equation

$$\nabla \cdot \left(\frac{1}{\rho(\phi^{n+1})} \nabla p^{n+1} \right) = \nabla \cdot \frac{\vec{u}^*}{\delta t}, \tag{5.3}$$

where $\vec{u}^*$ is the velocity field, δt the time step width, ϕ^{n+1} a (time-depending) level set function which describes the location of the free surface and ρ denotes the density, which can be expressed depending on the level-set function. Near the surface, a large jump of the density can occur (in our case, a factor of ≈ 773). This leads to very large condition numbers for the resulting linear system.

In our simulation, we consider a box of size $2 \times 2 \times 1$ cm. Inside this box, a drop of water is falling into a water basin, see Figure 5.1. We first present the grids produced by the various parallelization techniques using $64 \times 64 \times 64$ points, see Figure 5.2. Note the coarse grid points are clustered near the phase boundaries. This is due to the anisotropic stencils at these points. We compare the RS3, the Falgout and the CGC algorithm on 4 processors each with the grid produced on a single processor. The RS3 algorithm inserts a band of additional coarse grid points near the processor subdomain boundary, the Falgout algorithm even adds coarse grid points away from the boundaries due to the CLJP algorithm used for boundary treatment. The grids produced by the CGC algorithm, however, resemble those produced in the sequential case very well. We find no artifacts due to the domain decomposition in the grids constructed by our CGC method.

Let us now turn to the speed-up properties of parallel AMG. To this end we consider a problem of size $128 \times 128 \times 128$. For large processor numbers, the CGC algorithm only needs about half the setup time (Table 5.2) the other methods require. The well-fitted grids produced by the CGC algorithm allow the computation of the coarse grid operator to be carried out in half the time the classical coarsening (NONE) needs, a feature that compensates the time needed for the additional coarsening iterations easily. The well-matched grids also yield a low operator complexity. The same

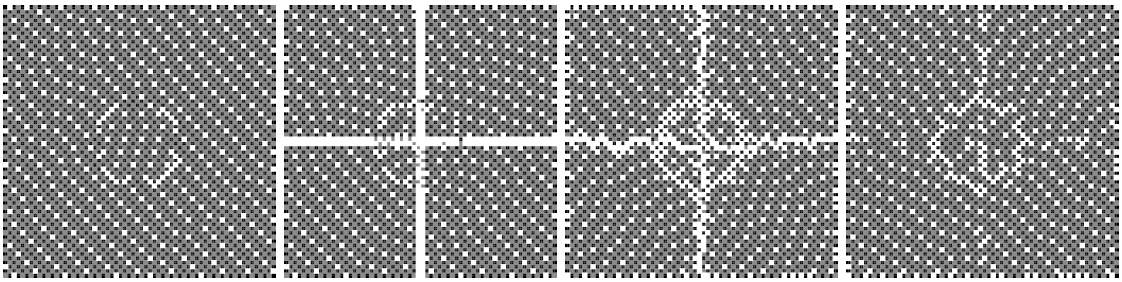


FIG. 5.2. *Results of various coarsening schemes. Depicted are the grid points of the $x-y$ slice at $z = 31h_z$ where h_z is the mesh width in z-direction. The black points belong to the fine grid Ω^1 only, the gray points also belong to grid Ω^2 and the white points belong to grid Ω^3. From left to right: sequential coarsening (one processor), RS3 algorithm, Falgout algorithm, CGC algorithm (on 4 processors each).*

TABLE 5.2
Setup time in seconds and operator complexity (5.1) *for Example 5.2.*

P	NONE	RS3	Falgout	CGC
1	473 / 3.89	473 / 3.89	473 / 3.89	473 / 3.89
2	280 / 4.98	287 / 4.54	319 / 6.43	290 / 4.09
4	143 / 6.10	151 / 5.14	160 / 6.99	142 / 4.19
8	68.7 / 5.73	80.2 / 5.26	77.8 / 7.77	74.6 / 3.80
16	50.0 / 6.14	56.4 / 5.50	52.5 / 8.37	42.5 / 4.01
32	32.9 / 5.91	36.1 / 5.57	33.6 / 8.77	24.4 / 4.21
64	22.2 / 5.90	22.3 / 5.66	28.7 / 9.33	13.0 / 4.53
128	20.1 / 5.86	15.7 / 5.90	19.3 / 9.79	8.47 / 4.60
256	14.5 / 6.26	10.5 / 6.48	13.1 / 10.5	5.51 / 5.07

holds for the grid complexity, which only increases from 1.67 to 1.74 as the number of processors increases from 8 to 256, while the Falgout algorithm produces a grid hierarchy of complexity 1.97 using 256 processors. We summarize the total compute time in Figure 5.3. The convergence factors

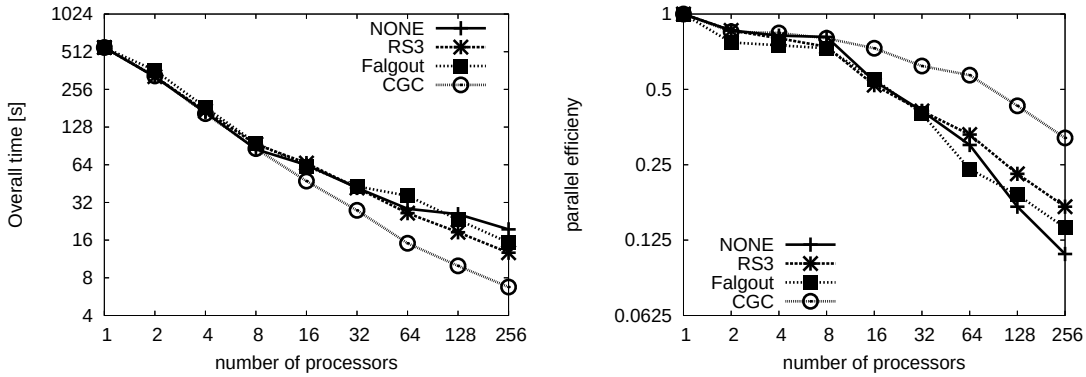


FIG. 5.3. *Overall time in seconds (left) and parallel efficiency (right) for Example 5.2.*

for this problem all stay below 0.1 independent of the number of processors and the parallelization technique used. Hence, the setup phase is the most expensive part of the AMG code and this phase also dominates the overall costs. From the graphs depicted in Figure 5.3 we can clearly observe that the CGC algorithm achieves a parallel efficiency over 50% up to 64 processors (i.e. $32 \times 32 \times 32$ points per processor), while the efficiency of all other methods drops below this level

for more than 16 processors. Hence, the CGC method allows us to utilize a larger part of a parallel computer with acceptable parallel efficiency than the other parallel AMG techniques. Note that the speed-up properties of a linear solver are especially important for time-dependent problems. Since the spatial resolution is coupled to the time discretization, engineers are very interested in solving a problem of given size faster simply by increasing the number of CPUs.

6. Concluding remarks. In this paper we presented a new approach to the construction of coarse grids for parallel AMG solvers. As the classical Ruge–Stüben coarsening scheme [12] performs excellently in the sequential case, our aim was to keep this scheme in the parallel case without coarsening the processor subdomain boundaries by another method. Our approach was to construct multiple coarse grids by the classical method, define a weighted graph describing the relationship between the constructed coarse grids and choosing one coarse grid per processor using graph partitioning techniques. This allowed us to match the coarse grid at the processor boundaries such that a further boundary treatment is not required in most cases.
Numerical experiments showed that the CGC coarsening algorithm led to a faster setup phase than other parallel AMG while maintaining the convergence behavior of sequential AMG. Finally, we have compared the RS3 method, the Falgout method and our CGC method for solving the Pressure-Poisson equation arising from the discretization of a two-phase flow simulation. In this case, the CGC method allowed a considerably faster setup phase and lower memory requirements compared to the other approaches and showed a much better parallel efficiency than the other parallel AMG techniques.

REFERENCES

[1] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *PETSc users manual*, Tech. Rep. ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.

[2] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *PETSc Web page*, 2001. http://www.mcs.anl.gov/petsc.

[3] S. Balay, V. Eijkhout, W. D. Gropp, L. C. McInnes, and B. F. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser Press, 1997, pp. 163–202.

[4] Brandt, A., *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–56.

[5] Brandt, A., McCormick, S.F., and Ruge, J., *Algebraic Multigrid (AMG) for automatic multigrid solution with application to geodetic computations*, 1982.

[6] ———, *Algebraic Multigrid (AMG) for sparse matrix equations*, in Sparsity and its Applications, D. Evans, ed., Cambridge University Press, Cambridge, 1984, pp. 257–284.

[7] Cleary, Andrew J., Falgout, Robert D., Henson, Van Emden, and Jones, Jim E., *Coarse-grid Selection for Parallel Algebraic Multigrid*, in Proc. of the Fifth International Symposium on Solving Irregularly Structured Problems in Parallel, vol. 1457 of Lecture Notes in Computer Science, Springer Verlag, 1998, pp. 104–115. Also available as Lawrence Livermore National Laboratory technical report UCRL-JC-130893.

[8] R. Croce, M. Griebel, and M. A. Schweitzer, *A Parallel Level-Set Approach for Two-Phase Flow Problems with Surface Tension in Three Space Dimensions*, Preprint 157, Sonderforschungsbereich 611, Universität Bonn, 2004.

[9] Edmond Chow, Robert D. Falgout, Jonathan J. Hu, Raymond S. Tuminaro, and Ulrike Meier Yang, *A Survey of Parallelization Techniques for Multigrid Solvers*, Tech. Rep. UCRL-BOOK-205864, Lawrence Livermore National Laboratory, Aug. 2004.

[10] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer series in Computational Mathematics, Springer-Verlag, Berlin, Heidelberg, 1985.

[11] V. E. Henson and U. M. Yang, *BoomerAMG: a Parallel Algebraic Multigrid Solver and Preconditioner*, Tech. Rep. UCRL-JC-141495, Lawrence Livermore National Laboratory, Mar. 2001.

[12] A. Krechel and K. Stüben, *Parallel Algebraic Multigrid Based on Subdomain Blocking*, Tech. Rep. REP-SCAI-1999-71, GMD, Dec. 1999.

[13] B. Metsch, *Ein paralleles graphenbasiertes algebraisches Mehrgitterverfahren*, Diplomarbeit, Institut für Numerische Simulation, Universität Bonn, Bonn, Germany, 2004.

[14] K. Stüben, *Algebraic Multigrid (AMG): An Introduction with Applications*, tech. rep., GMD - Forschungszentrum Informationstechnik GmbH, Mar. 1999.

[15] G. W. Zumbusch, *Adaptive Parallel Multilevel Methods for Partial Differential Equations*, Habilitation, Universität Bonn, 2001.

PARALLEL MULTI-LEVEL RESTRICTED SCHWARZ WITH POLLUTION REMOVING FOR PDE-CONSTRAINED OPTIMIZATION

ERNESTO E. PRUDENCIO AND XIAO-CHUAN CAI *

Abstract. Although Newton-Krylov-Schwarz (NKSz) methods have been succesfully applied to the numerical solution of many simulation PDEs, little is known about their suitability to the Karush-Kuhn-Tucker (KKT) systems arising from PDE-constrained optimization problems. In this paper we propose and study multi-level versions of the parallel fully coupled full space sequential quadratic programming class of methods known as Lagrange-Newton-Krylov-Schwarz (LNKSz). In LNKSz, a Lagrangian functional is formed and differentiated to obtain the KKT optimality system, which is then solved with NKSz algorithms. The multi-level preconditioner for the KKT Jacobian becomes a key component, combining restricted Schwarz smoothers with a pollution removing coarse-to-fine interpolation for the Lagrange multipliers. Our tests involve the parallel numerical solution of some boundary flow control problems with two-level LNKSz and show the method providing both the robustness with respect to Reynolds number and the scalability with respect to number of processors and mesh size.

Key words. Partial differential equations constrained optimization, inexact Newton, domain decomposition, multi-level methods, Schwarz preconditioners, parallel computing, flow control.

1. Introduction. Among optimization problems with constraints, those constrained by nonlinear PDEs are very challenging, both mathematically and computationally. Examples of such problems are inverse, optimal design and optimal control problems. In inverse problems some parameters are not known and must be estimated by analysis of experimental system output data. Optimal design usually refers to problems where one has to optimize an objective having the shape of a domain as the variable. In optimal control one usually searches for the best feasible control variables, such as boundary values or external forces, that optimize a system behavior.

Optimization problems can involve equality and inequality constraints. When only equalities are present, a sufficiently general description is given by

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbf{W}} & \mathcal{F}(\mathbf{x}) \\ \text{s.t.} & \mathbf{C}(\mathbf{x}) = \mathbf{0} \in \mathbf{Y}. \end{cases} \tag{1.1}$$

Here $\mathbf{W}$ and $\mathbf{Y}$ are normed spaces, $\mathbf{W}$ is the space of optimization variables, $\mathcal{F} : \mathbf{W} \to \mathbb{R}$ is the objective functional and $\mathbf{C} : \mathbf{W} \to \mathbf{Y}$. The associated Lagrangian functional $\mathcal{L} : \mathbf{W} \times \mathbf{Y}^* \to \mathbb{R}$ is defined as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \equiv \mathcal{F}(\mathbf{x}) + \langle \boldsymbol{\lambda}, \mathbf{C}(\mathbf{x}) \rangle_{\mathbf{Y}}, \quad \forall\ (\mathbf{x}, \boldsymbol{\lambda}) \in \mathbf{W} \times \mathbf{Y}^*,$$

where $\mathbf{Y}^*$ is the adjoint space of $\mathbf{Y}$, $\langle \cdot, \cdot \rangle_{\mathbf{Y}}$ denotes the duality pairing and variables $\boldsymbol{\lambda}$ are called Lagrange multipliers or adjoint variables. In many cases it is possible to prove that, if $\hat{\mathbf{x}}$ is a (local) solution of (1.1) then there exist Lagrange multipliers $\hat{\boldsymbol{\lambda}}$ such that $(\hat{\mathbf{x}}, \hat{\boldsymbol{\lambda}})$ is a critical point of $\mathcal{L}$ [19]. So, under sufficient smoothness assumptions, one obtains, as necessary condition for a solution, a system of equations, called Karush-Kuhn-Tucker (KKT) or optimality system. For the numerical solution of (1.1) with PDE constraints, the discretization with a mesh Ω_h of characteristic size $h > 0$ can happen after the deduction of the KKT system (*optimize-then-discretize*) or before the formulation of the Lagrangian (*discretize-then-optimize*). These two choices do *not* necessarily lead to the same solution because the differentiation and discretization operations do not commute [17]. In this paper we follow the latter approach, obtaining a finite dimensional optimization problem with $\mathbf{W} = \mathbb{R}^{n_h}$ and $\mathbf{Y} = \mathbb{R}^{m_h} = \mathbf{Y}^*$. The KKT system becomes $\nabla \mathcal{L}_h(\hat{\mathbf{x}}, \hat{\boldsymbol{\lambda}}) = \mathbf{0} \in \mathbb{R}^{n_h+m_h}$. There are then two major families of Newton techniques: reduced space sequential quadratic programming (SQP), characterized by the partition of the problem into smaller ones at each Newton step, and full space SQP. When memory capability is an issue, reduced methods are preferred, although many sub-iterations might be needed to converge the outer-iterations and the parallel scalability is less ideal. Full space methods also present difficulties: problem size, descent directions for line search not guaranteed and KKT Jacobians usually indefinite and ill-conditioned. Examples of full space methods are Lagrange-Newton-Krylov-Schur [3, 4] and one-level *Lagrange-Newton-Krylov-Schwarz* (LNKSz) [22].

*Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309 (prudenci@colorado.edu, cai@cs.colorado.edu). The research is supported in part by the National Science Foundation, CCR-0219190, ACI-0072089 and CCF-0305666, and in part by the Department of Energy, DE-FC02-01ER25479.

The rest of the paper is organized as follows. Section 2 discusses equality constrained optimization. In Section 3 we introduce multi-level LNKSz with pollution removing interpolation, while in Section 4 we test LNKSz on some boundary flow control problems. Final conclusions are given in Section 5.

2. Equality constrained optimization. Omitting symbols "h" and " $\hat{\cdot}$ ", and denoting $N \equiv n+m$ and $\mathbf{X} \equiv (\mathbf{x}, \boldsymbol{\lambda}) \in \mathbb{R}^N$, the KKT system becomes

$$\mathbf{F}(\mathbf{X}) \equiv \begin{array}{c} \nabla_{\mathbf{x}}\mathcal{L} \\ \nabla_{\boldsymbol{\lambda}}\mathcal{L} \end{array} = \begin{array}{c} \nabla\mathcal{F} + [\boldsymbol{\nabla}\mathbf{C}]^T\boldsymbol{\lambda} \\ \mathbf{C} \end{array} = \mathbf{0}, \tag{2.1}$$

where $\mathbf{F} : \mathbb{R}^N \to \mathbb{R}^N$, $\nabla_{\mathbf{x}}\mathcal{L}$ denotes the gradient of $\mathcal{L}$ w.r.t. $\mathbf{x}$ variables, with similar meaning for $\nabla_{\boldsymbol{\lambda}}\mathcal{L}$, and $\boldsymbol{\nabla}\mathbf{C}$ denotes the Jacobian of $\mathbf{C}$. System (2.1) can be solved with inexact Newton [15]: given an initial guess $\mathbf{X}^{(0)}$, at each iteration $k = 0, 1, 2, \ldots$ an approximate solution $\mathbf{p}^{(k)} \equiv (\mathbf{p}_{\mathbf{x}}^{(k)}, \mathbf{p}_{\boldsymbol{\lambda}}^{(k)})$ of the linearized KKT system

$$\begin{bmatrix} \boldsymbol{\nabla}_{\mathbf{xx}}\mathcal{L}^{(k)} & [\boldsymbol{\nabla}\mathbf{C}^{(k)}]^T \\ \boldsymbol{\nabla}\mathbf{C}^{(k)} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{p}_{\mathbf{x}}^{(k)} \\ \mathbf{p}_{\boldsymbol{\lambda}}^{(k)} \end{pmatrix} = - \begin{array}{c} \nabla_{\mathbf{x}}\mathcal{L}^{(k)} \\ \mathbf{C}^{(k)} \end{array} \tag{2.2}$$

is computed, where the terms should be understood as evaluated at $\mathbf{X}^{(k)}$. The KKT matrix is the transpose of the Hessian of the Lagrangian and is symmetric indefinite under appropriate assumptions [20]. If $\boldsymbol{\nabla}\mathbf{C}^{(k)}$ has full rank and $\boldsymbol{\nabla}_{\mathbf{xx}}\mathcal{L}^{(k)}$ is positive definite in the tangent space of the constraints (i.e., $\mathbf{d}^T[\boldsymbol{\nabla}_{\mathbf{xx}}\mathcal{L}^{(k)}]\mathbf{d} > 0$ for all $\mathbf{d} \neq \mathbf{0}$ such that $[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{d} = \mathbf{0}$), then we can interpret $(\mathbf{p}_{\mathbf{x}}^{(k)}, \mathbf{p}_{\boldsymbol{\lambda}}^{(k)})$ as being the *unique* solution and respective Lagrange multipliers of a quadractic programming problem on $\mathbf{p}_{\mathbf{x}} \in \mathbb{R}^n$, which justifies the use of SQP terminology for methods involving (2.2).

After computing $\mathbf{p}^{(k)}$ one may use a globalization method like *line search* or *trust region*. Here we focus on line search, i.e., $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \alpha^{(k)}\mathbf{p}^{(k)}$, with the step length $0 < \alpha^{(k)} \leqslant 1$ being selected by backtracking until the sufficient decrease

$$\phi(\mathbf{X}^{(k)} + \alpha^{(k)}\mathbf{p}^{(k)}) \leqslant \phi(\mathbf{X}^{(k)}) + \alpha^{(k)}c_1 \quad \nabla\phi(\mathbf{X}^{(k)})^{\ T}\mathbf{p}^{(k)} \tag{2.3}$$

is achieved, where ϕ is a merit function and $0 < c_1 < 1/2$ is a constant. For constrained optimization, the l_1 or augmented Lagrangian merit functions are commonly used. in order to balance the sometimes conflicting goals of reducing the objective function and satisfying the constraints [20]. In our experiments we use the augmented Lagrangian $\phi_{AL} : \mathbb{R}^N \to \mathbb{R}$. Given a penalty parameter $\rho > 0$, it is defined by

$$\phi_{AL}(\mathbf{X}; \rho) = \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) + \frac{\rho}{2}\ \|\mathbf{C}(\mathbf{x})\|_2^2 \quad \forall\ \mathbf{X} = (\mathbf{x}, \boldsymbol{\lambda}) \in \mathbf{W} \times \mathbf{Y}.$$

At iteration k, $\rho = \rho^{(k)}$ must be such that we obtain a descent direction $\mathbf{p}^{(k)}$, i.e.,

$$(\nabla\phi_{AL}(\mathbf{X}^{(k)}; \rho^{(k)}))^T\mathbf{p}^{(k)} = (\nabla\mathcal{L}^{(k)})^T\mathbf{p}^{(k)} + \rho^{(k)}\ \mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)} < 0.$$

Since $\mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)} = -\|\mathbf{C}^{(k)}\|_2^2$ for an exact step, it is reasonable to expect

$$\mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)} < 0 \tag{2.4}$$

for approximate steps, if $\mathbf{C}^{(k)} \neq \mathbf{0}$ and the tolerances for the Krylov subspace method are small enough and the preconditioner is good enough to guarantee that the Krylov subspace method does not stop by achieving the maximum allowed number of iterations. If (2.4) does not hold we can continue the Krylov iterations, with eventual smaller tolerances, until it does. Once (2.4) holds, we choose $0 < \beta < 1$ and use the fairly common strategy of demanding $\rho^{(k)}$ to satisfy

$$(\nabla\phi_{AL}(\mathbf{X}^{(k)}; \rho^{(k)}))^T\mathbf{p}^{(k)} \leqslant \beta\rho^{(k)}\ \mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)}.$$

Given an initial value $\rho^{(-1)} > 0$, we then set $\rho^{(k)} = \max\{\overline{\rho}^{(k)}, \rho^{(k-1)}\}$, where

$$\overline{\rho}^{(k)} = -\frac{1}{1-\beta} \cdot \frac{(\nabla\mathcal{L}^{(k)})^T\mathbf{p}^{(k)}}{\mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)}}.$$

However, if $\mathbf{C}^{(k)} = \mathbf{0}$ (although it is reasonable to expect $\mathbf{C}^{(k)} \neq \mathbf{0}$ for non-trivial PDEs) then there is no way to guarantee a descent direction. This is a fundamental issue with line search methods. Some algorithms handle this by modifying the Hessian to make it positive definite on the null space of $\boldsymbol{\nabla}\mathbf{C}^{(k)}$, but for problems of the size we are considering there is no efficient way to check positive definiteness.

3. Lagrange-Newton-Krylov-Schwarz. A key element of a successful full space approach is the preconditioning of the Jacobian of the KKT system. A useful preconditioner has to be able to simultaneously reduce the condition number and provide good parallel scalability. In *operator-splitting* type techniques a sequential block elimination step is needed to form the Schur complement w.r.t. some smaller group of variables. In contrast, Schwarz type preconditioners are *fully coupled*: all variables are treated equally, the partition is based completely on the physical domain Ω and one less sequential step is demanded since no variables need to be eliminated. Besides, overlapping methods are easier to be converted to nonlinear preconditioners [8] for highly nonlinear problems.

3.1. Multi-level restricted Schwarz preconditioners. Schwarz or overlapping domain decomposition methods can be used in one-level or multi-level approaches and, at each case, with a combination of additive and/or multiplicative algorithms [25]. They can be also used in linear [14] and nonlinear approaches [8, 13]. Let $\Omega \subset \mathbb{R}^2$ be a bounded open domain on which a differential problem is defined and a discretization is performed with a mesh Ω_h of characteristic size $h > 0$. To obtain the overlapping partition, we first divide Ω_h into non-overlapping subdomains Ω_j, $j = 1, \ldots, N_S$. We then expand each Ω_j to $\Omega_j^{'}$, i.e., $\Omega_j \subset \Omega_j^{'}$. The overlapping $\delta > 0$ is defined as the minimum distance between $\Omega_j^{'}$ and Ω_j. For boundary subdomains we simply cut off the part outside Ω. Let $H > 0$ denote the characteristic diameter of $\{\Omega_j\}$.

From now on we only consider simple box domains, uniform meshes and simple box decompositions, i.e., all subdomains Ω_j and $\Omega_j^{'}$ are rectangular and made up of integral number of mesh cells. Let N and N_j denote the number of degrees of freedom associated to Ω and $\Omega_j^{'}$, respectively. Let $\mathbf{K}$ be a $N \times N$ matrix of a linear system

$$\mathbf{K}\mathbf{p} = \mathbf{b} \tag{3.1}$$

that needs to be solved during the application of an algorithm for the numerical solution of the discretized differential problem. Let d indicate the degree of freedom per mesh point. For simplicity let us assume that d is the same throughout the entire mesh. We define the $N_j \times N$ matrix $\mathbf{R}_j^{\delta}$ as follows: its $d \times d$ block element $(\mathbf{R}_j^{\delta})_{\alpha,\beta}$ is either (a) an identity block if the integer indices $1 \leqslant \alpha \leqslant N_j/d$ and $1 \leqslant \beta \leqslant N/d$ are related to the same mesh point and this mesh point belongs to $\Omega_j^{'}$ or (b) a zero block otherwise. The multiplication of $\mathbf{R}_j^{\delta}$ with a $N \times 1$ vector generates a smaller $N_j \times 1$ vector by discarding all components corresponding to mesh points outside $\Omega_j^{'}$. The $N_j \times N$ matrix $\mathbf{R}_j^{0}$ is similarly defined, with the difference that its application to a $N \times 1$ vector also zeroes all those components corresponding to mesh points on $\Omega_j^{'} \setminus \Omega_j$. Let $\mathbf{B}_j^{-1}$ be either the inverse of or a preconditioner for $\mathbf{K}_j \equiv \mathbf{R}_j^{\delta}\ \mathbf{K}\ \mathbf{R}_j^{\delta^T}$. The one-level classical, right restricted (r-RAS) and left restricted (ℓ-RAS) additive Schwarz preconditioners for $\mathbf{K}$ are respectively defined as [6, 9, 14]

$$\mathbf{B}_{\delta\delta}^{-1} = \sum_{j=1}^{N_s} \mathbf{R}_j^{\delta^T} \mathbf{B}_j{}^{-1} \mathbf{R}_j^{\delta}, \quad \mathbf{B}_{\delta 0}^{-1} = \sum_{j=1}^{N_s} \mathbf{R}_j^{\delta^T} \mathbf{B}_j{}^{-1} \mathbf{R}_j^{0} \quad \text{and} \quad \mathbf{B}_{0\delta}^{-1} = \sum_{j=1}^{N_s} \mathbf{R}_j^{0^T} \mathbf{B}_j{}^{-1} \mathbf{R}_j^{\delta}.$$

For the description of multi-level Schwarz preconditioners, let us use index $i = 0, 1, \ldots, L-1$ to designate any of the $L \geqslant 2$ levels. Let $\mathbf{I}_i$ denote the identity operator and, for $i > 0$, let $\mathbf{R}_i^T$ denote the interpolation from level $i-1$ to level i. Multi-level Schwarz preconditioners are obtained through the combination of one-level Schwarz preconditioners $\mathbf{B}_i^{-1}$ assigned to each level. Here we focus on multi-level preconditioners that use exact coarsest solvers $\mathbf{B}_0^{-1}$ and that can be seen as multigrid V-cycle algorithms [5] having Schwarz preconditioned Richardson working as the pre and the post smoother at each level $i > 0$, with $\mathbf{B}_{i,\text{pre}}^{-1}$ preconditioning the $\mu_i \geqslant 0$ pre smoother iterations and $\mathbf{B}_{i,\text{post}}^{-1}$ preconditioning the $\nu_i \geqslant 0$ post smoother iterations. Then, as iterative methods for (3.1), with $\mathbf{r}^{(\ell)}$ denoting the residual at iteration $\ell = 0, 1, 2, \ldots$, they can be described in the case $L = 2$ as

$$\mathbf{r}^{(\ell+1)} = (\mathbf{I}_1 - \mathbf{K}_1 \mathbf{B}_{1,\text{post}}^{-1})^{\nu_1} (\mathbf{I}_1 - \mathbf{K}_1 \mathbf{R}_1^T \mathbf{B}_0^{-1} \mathbf{R}_1)(\mathbf{I}_1 - \mathbf{K}_1 \mathbf{B}_{1,\text{pre}}^{-1})^{\mu_1} \mathbf{r}^{(\ell)}. \tag{3.2}$$

3.2. Pollution removing interpolation. This constitutes a key procedure in our proposed multi-level LNKSz methods, due to the sharp jumps that often occur on the multiplier values over those regions of Ω where constraints are greatly affecting the behavior of the optimized system. Although the evidence of this discontinuity property of Lagrange multipliers is just empirical in our paper, it is consistent with

their interpretation [20]: the value of a Lagrange multiplier at a mesh point gives the rate of change of the optimal objective function value w.r.t. to the respective constraint at that point.

In the case of the problem corresponding to Figure 4.2-b, for instance, an external force causes the fluid to move on a clockwise way and the boundary consists of rigid slip walls. The vertical walls greatly affect the overall vorticity throughout the domain, i.e., the value of the objective function, because they completely oppose the horizontal velocity component v_1. The values of λ_1 at the walls then reflect this situation. In constrast, λ_2 develops sharp jumps at the other two walls opposing v_2. In all our experiments the discontinuities are located only over the boundary and not around it, even for very fine meshes. Common coarse-to-fine interpolation techniques will then smooth the sharp jumps present in coarse solutions, with a gradual jumping, from interior mesh points towards boundary mesh points, appearing over those fine cells (elements, volumes) located inside coarse boundary ones, as represented in Figure 3.1-*b*. That is, the good correction information previded by the coarse solution is lost with a common interpolation. We refer to the smoothed jump as "pollution", in contrast to the "clean" sharp jump that is expected at the fine level as well.

We therefore propose a *modified* coarse-to-fine interpolation procedure that is based on a general and very simple "removal of the pollution". Let $\mathbf{R}_i^T$ denote any unmodified interpolation procedure and $\boldsymbol{\mathcal{Z}}_i$ the operator that zeroes out, from a vector at level i, the Lagrange multipliers at all those mesh points with equations that have a greater influence on the objective function. For the case of PDEs describing physical systems, the number of such points are expected to be relatively small. Our modified interpolation is then expressed by

$$\mathbf{R}_{i,\text{modif}}^T = \mathbf{R}_i^T - \boldsymbol{\mathcal{Z}}_i \mathbf{R}_i^T (\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1}). \tag{3.3}$$

This procedure removes the smoothed contributions due to the coarse discontinuities, maintaining, at the fine level, the sharp jumps originally present at the coarse level. It should be noted that, once $\mathbf{R}_i^T$ is available, (3.3) can be applied to *any* mesh in *any* dimension, with *any* number of components.

In the case of the problems in this paper, $\boldsymbol{\mathcal{Z}}_i$ zeroes the Lagrange multiplier components located at the boundary. A schematic representation of (3.3) in then given in Figure 3.1. In our tests we apply the modified interpolation only on the Lagrange multiplier components of coarse solutions, while the optimization variables continue to be interpolated with the unmodified process $\mathbf{R}_i^T$. Also, the restriction process remains $\mathbf{R}_i$ for all variables, that is, (3.2) becomes

$$\mathbf{r}^{(\ell+1)} = (\mathbf{I}_1 - \mathbf{K}_1 \mathbf{B}_{1,\text{post}}^{-1})^{\nu_1} (\mathbf{I}_1 - \mathbf{K}_1 \mathbf{R}_{1,\text{modif}}^T \mathbf{B}_0^{-1} \mathbf{R}_1)(\mathbf{I}_1 - \mathbf{K}_1 \mathbf{B}_{1,\text{pre}}^{-1})^{\mu_1} \mathbf{r}^{(\ell)}. \tag{3.4}$$

The Lagrange multipliers reflect the eventual "discontinuity" on the type of equations (or their physical dimensions) between equations in different regions of $\overline{\Omega}$: in the case of the problems in Section 4, between those equations on Ω and those at $\partial\Omega$. From this point of view, it seems "natural" to employ different interpolations to the Lagrange multiplier components depending on their location on $\overline{\Omega}$.

3.3. Expected scalability properties. When classic Schwarz preconditioners are applied to symmetric positive definite systems arising from the discretization of elliptical problems defined in $H_0^1(\Omega)$, the condition number κ of the preconditioned system satisfies $\kappa \leqslant C\,(1 + H/\delta)\,/H^2$ for one-level methods and $\kappa \leqslant C\,(1 + H/\delta)$ for two-level methods, where C is independent of h, H and δ. The factor $1/H^2$, associated to the number of subdomains on the fine level, relates itself to the increase on the number of iterations (needed for the exchange of information among distant subdomains) with the increase in the total number of subdomains. The use of a coarse level helps the exchange of information. The necessity of information exchange among distant domain regions can be understood through the expression of the solutions of elliptic problems in terms of Green's functions: although the solution value at a point strongly depends on surrounding values, there is weaker dependence w.r.t. the entire domain [25]. Regarding the application of two-level methods to indefinite model problems, the study in [10] suggests that the coarse mesh needs to be sufficiently fine for the two-level Schwarz preconditioner to perform well.

Theoretically, however, these results may not be directly applied to the case of symmetric indefinite KKT Jacobians. Let $\overline{\ell}$ be the average number of linear iterations per Newton step. We then look for the following more general properties when applying two-level LNKSz:

$$\text{For fixed } h \text{ and } H/\delta,\ \overline{\ell} \text{ is not very sensitive to } H \text{ decreasing,} \tag{3.5}$$

$$\text{For fixed } H \text{ and } \delta,\ \overline{\ell} \text{ is not very sensitive to the mesh refinement.} \tag{3.6}$$

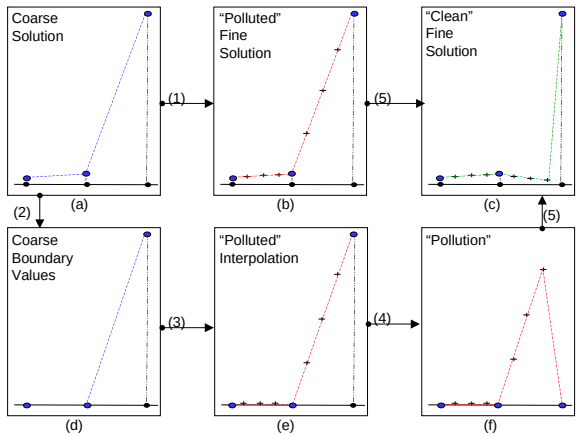


FIG. 3.1. *Representation of the modified interpolation* (3.3)*, with (a) input* $\boldsymbol{\varphi}_{i-1}$ *and (c) output* $\boldsymbol{\varphi}_i = \mathbf{R}_{1,modif}^T \boldsymbol{\varphi}_{i-1}$*. The steps are: (1) unmodified interpolation* $\mathbf{R}_i^T \boldsymbol{\varphi}_{i-1}$*, (2) coarse boundary values* $\tilde{\boldsymbol{\varphi}}_{i-1} = (\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1})\boldsymbol{\varphi}_{i-1}$*, (3)* $\tilde{\boldsymbol{\varphi}}_i = \mathbf{R}_i^T \tilde{\boldsymbol{\varphi}}_{i-1}$*, (4) "pollution"* $\boldsymbol{\mathcal{Z}}_i \tilde{\boldsymbol{\varphi}}_i$*, (5) "clean"* $\boldsymbol{\varphi}_i = \mathbf{R}_i^T \boldsymbol{\varphi}_{i-1} - \boldsymbol{\mathcal{Z}}_i \tilde{\boldsymbol{\varphi}}_i$*.*

Also, it might happen that a presumed "enhancement" of the preconditioner, aiming to cluster the eigenvalues around 1, eventually results in negative eigenvalues getting too close to 0. Although there is no necessary relationship between the eigenvalue distribution and ill-conditioning for general matrices [21], this process might qualitatively explain an eventual degradation of the linear convergence before it finally gets better. Indeed, we have observed such temporary degradation in some of our numerical experiments when increasing H or δ.

4. Numerical experiments. Our numerical experiments in this paper focus on optimal control problems, where the optimization space in (1.1) is generally given by $\mathbf{W}$=$\mathbf{S} \times \mathbf{U}$, with $\mathbf{S}$ being the state space and $\mathbf{U}$ the control space. Upon discretization, one has n=n_s+n_u, where n_s (n_u) is the number of discrete state (control) variables. In the case of optimal flow control problems, for instance, typical state variables are velocity, pressure and temperature. Control variables might be defined on the boundary or inside the flow domain and can be the temperature, velocity, external force etc. The objective can be, for instance, minimization of turbulence, or maximization of mixing [26]. The constraints are given by the physical equations that must be obeyed by the system being controlled, e.g. Navier-Stokes equations, and by eventual inequalities, e.g. bounds on the state and control variables.

All currently important numerical methods for optimal control differ themselves by the way (2.2) is solved. In contrast to LNKSz, however, many of them usually assume that the linearized forward operator $\boldsymbol{\nabla}_\mathbf{s}\mathbf{C}$ is a square matrix of order $m = n_s$ and can then be understood through the block-LU factorization

$$\boldsymbol{\nabla}^2 \mathcal{L}^{(k)} = \begin{bmatrix} \mathbf{W_{ss}}[\boldsymbol{\nabla}_\mathbf{s}\mathbf{C}]^{-1} & \mathbf{0} & \mathbf{I} \\ \mathbf{W_{su}}[\boldsymbol{\nabla}_\mathbf{s}\mathbf{C}]^{-1} & \mathbf{I} & \mathbf{L}^T \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\nabla}_\mathbf{s}\mathbf{C} & \boldsymbol{\nabla}_\mathbf{u}\mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_2 & [\boldsymbol{\nabla}_\mathbf{s}\mathbf{C}]^T \end{bmatrix} \tag{4.1}$$

of the KKT Jacobian, where $\mathbf{H}_1 = \mathbf{W_{uu}} - \mathbf{L}^T\mathbf{H}_2 - \mathbf{W_{su}}\mathbf{L}$, $\mathbf{H}_2 = \mathbf{W_{us}} - \mathbf{W_{ss}}\mathbf{L}$, and $\mathbf{L} = [\boldsymbol{\nabla}_\mathbf{s}\mathbf{C}]^{-1}[\boldsymbol{\nabla}_\mathbf{u}\mathbf{C}]$. The matrix $\mathbf{H}_1$ is the Schur complement for the compoenent $\mathbf{p}_\mathbf{u}^{(k)}$ of step $\mathbf{p}^{(k)} = (\mathbf{p}_\mathbf{s}^{(k)}, \mathbf{p}_\mathbf{u}^{(k)}, \mathbf{p}_{\boldsymbol{\lambda}}^{(k)})$, and is called the reduced Hessian of the Lagrangian [3, 4].

More specifically, we treat the boundary control of two-dimensional steady-state incompressible Navier-Stokes equations in the velocity-vorticity formulation [23]: $\mathbf{v} = (v_1, v_2)$ is the velocity and ω is the vorticity. Let $\Omega \subset \mathbb{R}^2$ be an open and bounded smooth domain, $\Gamma = \partial\Omega$ its boundary, $\boldsymbol{\nu}$ the unit outward normal vector along Γ and $\mathbf{f}$ a given external force defined in Ω. Let $\mathbf{S} = L^2(\Omega)$ and $\mathbf{U} = L^2(\Gamma_u)$ be the spaces of square Lebesgue integrable functions in Ω and $\Gamma_u \subset \Gamma$ respectively. The control problems consist on finding $(\mathbf{s}, \mathbf{u}) = (v_1, v_2, \omega, u_1, u_2)$ such that the minimization

$$\min_{(\mathbf{s},\mathbf{u})\in\mathbf{S}\times\mathbf{U}} \mathcal{F}(\mathbf{s}, \mathbf{u}) = \frac{1}{2}\int_\Omega \omega^2 \, d\Omega + \frac{c}{2}\int_{\Gamma_u} \|\mathbf{u}\|_2^2 \, d\Gamma_u \tag{4.2}$$

is achieved subject to the constraints

$$\begin{cases} -\Delta v_1 - \frac{\partial \omega}{\partial x_2} & = & 0 & \text{in } \Omega, \\ -\Delta v_2 + \frac{\partial \omega}{\partial x_1} & = & 0 & \text{in } \Omega, \\ -\Delta \omega + Re\ v_1 \frac{\partial \omega}{\partial x_1} + Re\ v_2 \frac{\partial \omega}{\partial x_2} - Re \text{ curl } \mathbf{f} & = & 0 & \text{in } \Omega, \\ v_i - v_{D,i} & = & 0 & \text{on } \Gamma_{D,i}, \quad i = 1,2, \\ \frac{\partial v_i}{\partial \boldsymbol{\nu}} - v_{N,i} & = & 0 & \text{on } \Gamma_{N,i}, \quad i = 1,2, \\ \mathbf{v} - \mathbf{u} & = & \mathbf{0} & \text{on } \Gamma_u, \\ \omega + \frac{\partial v_1}{\partial x_2} - \frac{\partial v_2}{\partial x_1} & = & 0 & \text{on } \Gamma, \\ \int_\Gamma \mathbf{v} \cdot \boldsymbol{\nu}\ d\Gamma & = & 0, & \end{cases} \tag{4.3}$$

where curl $\mathbf{f} = -\partial f_1/\partial x_2 + \partial f_2/\partial x_1$ and, for $i = 1, 2$, $\Gamma = \Gamma_{D,i} \cup \Gamma_{N,i} \cup \Gamma_u$ and $\Gamma_{D,i}$ ($\Gamma_{N,i}$) is the part of the boundary where the velocity component v_i is specified through a Dirichlet (Neumann) condition with a prescribed velocity $v_{D,i}$ ($v_{N,i}$). The parameter $c > 0$ is used to adjust the relative importance of the control norms on achieving the minimization, so indirectly constraining their sizes. The physical objective in (4.2)-(4.3) is the minimization of turbulence [1, 18]. The last constraint is necessary for the consistency with the physical law of mass conservation, making $m \neq n_s$ and causing the complexity of parallel finite difference approximation of Jacobians to increase, since non-neighboring mesh points become coupled by the integral. An alternative formulation, that compromises between the physical law of mass conservation and the complex computation of Jacobians, eliminates the integral constraint but adds to the objective function the term $\tilde{c} \left[\int_\Gamma \mathbf{v} \cdot \boldsymbol{\nu}\ d\Gamma\right]^2 /2$, with $\tilde{c} \gg 1$ [4]. We restrict our numerical experiments to *tangential* boundary control problems (BCPs), i.e., $\mathbf{u} \cdot \boldsymbol{\nu} = 0$ on Γ_u and the velocity $\mathbf{v}$ is assumed to guarantee mass conservation along $\Gamma \setminus \Gamma_u$, so that $m = n_s$.

We consider rectangular domains $\Omega = (0, L_1) \times (0, L_2)$, $L_2 \leqslant L_1$. Figure 4.1 depicts all the related notation. We define $E_{1,a} = \{(x_1, x_2) \in E_1 : 0 < x_1 \leqslant L_2\}$, $E_{1,b} = E_1 \setminus E_{1,a}$, $E_{4,a} = \{(x_1, x_2) \in E_4 : \frac{L_2}{2} \leqslant x_2 < L_2\}$ and $E_{4,b} = E_4 \setminus E_{4,a}$. Two flow problems are considered: cavity and backward-facing step. In each case we consider both simulation problems and tangential BCPs. In the descriptions below we only define Γ_u and the velocity boundary conditions on $\Gamma \setminus \Gamma_u$, since all control problems seek the minimization of the objective function (4.2) with $c = 10^{-2}$ and have in common the three PDEs in Ω, the ω boundary condition on Γ and the tangential boundary control on Γ_u. In the case of *cavity* flows, $L_1 = L_2 = 1$, $\Gamma_u = \Gamma$ and $\mathbf{f} = (f_1, f_2) = \ -\sin^2(\pi x_1)\cos(\pi x_2)\sin^2(\pi x_2), \sin^2(\pi x_2)\cos(\pi x_1)\sin^2(\pi x_1)\ $. In the case of *backward-facing step* flows, $L_1 = 6$, $L_2 = 1$, $\Gamma_u = E_{4,b} \cup \{C_1\} \cup E_{1,a}$ and $\mathbf{f} = \mathbf{0}$. The velocity boundary conditions on $\Gamma \setminus \Gamma_u$ are $v_1 = 0$ on $E_{1,b} \cup \overline{E_3} \cup E_{4,b}$, $v_1 = 8(1 - x_2)(x_2 - 1/2)$ on $E_{4,a}$, $v_1 = x_2(1 - x_2)$ on $\overline{E_2}$ and $v_2 = 0$. All corresponding simulation problems, used for comparison with control problems, impose $\mathbf{v} \cdot \boldsymbol{\nu} = 0$ and $\partial \mathbf{v}/\partial \boldsymbol{\nu} = 0$ on Γ_u, i.e., Γ_u becomes a (set of) slip rigid wall(s).

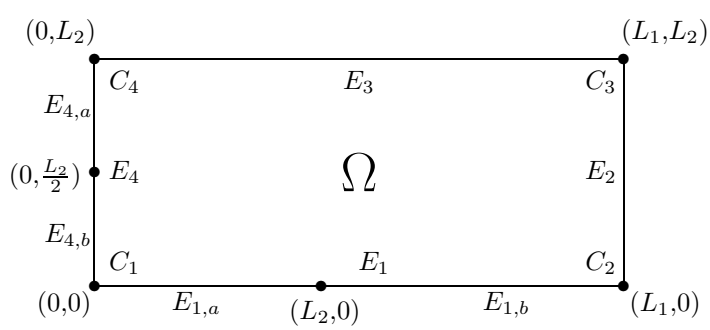


FIG. 4.1. *Domain $\Omega = (0, L_1) \times (0, L_2)$, $L_2 \leqslant L_1$, involved in our numerical experiments.*

4.1. Details of numerical approaches. For discretization we use a five-point second order finite difference method on a uniform nonstaggerd mesh. Since our focus in this paper is the LNKSz method, we do not use staggered grids [12, 16]. All derivative terms are discretized with a second order scheme, including the ω boundary condition. For parallel performance reasons, only mesh points adjacent to the boundary are used by applying the definition $\omega = -\partial v_1/\partial x_2 + \partial v_2/\partial x_1$ at these points as well [22].

In order to form the algebraic system of nonlinear discretized equations, we need to order the unknowns and the corresponding functions. The unknowns are ordered mesh point by mesh point, in contrast to physical variable by physical variable as usually required by methods related to (4.1). At each mesh point the unknowns are ordered as $v_1, v_2, \omega, u_1, u_2, \lambda_1, \lambda_2$ and λ_3. The mesh points are ordered subdomain by subdomain, for the purpose of parallel processing. The ordering of the subdomains is not important since we use additive methods whose performance has nothing to do with the subdomain ordering. In order to avoid pivoting during the sparse LU method (used in our experiments), the corresponding functions are ordered as $\nabla_{\lambda_1}\mathcal{L}$,$\nabla_{\lambda_2}\mathcal{L}$,$\nabla_{\lambda_3}\mathcal{L}$, $\nabla_{u_1}\mathcal{L}$,$\nabla_{u_2}\mathcal{L}$, $\nabla_{v_1}\mathcal{L}$,$\nabla_{v_2}\mathcal{L}$ and $\nabla_{\omega}\mathcal{L}$. Because the orderings for the unknowns and for the function components are different, the Jacobian matrix becomes nonsymmetric and so we use GMRES [24]. The Jacobian matrix is constructed approximately using a multi-colored central finite difference method [11] with step size 10^{-5}, which provides KKT matrices closer to be symmetric than the ones computed by forward finite differences. To solve the Jacobian systems we use restarted GMRES with an absolute (relative) tolerance equal to 10^{-8} (10^{-4}), a restart parameter equal to 90 and a maximum number of iterations equal to 5000. When using left preconditioning, the GMRES tolerances are checked over preconditioned residuals. Regarding the one-level additive Schwarz preconditioner, the number of subdomains is equal to the number of processors, and the extended subdomain problems have zero Dirichlet interior boundary conditions and are solved with sparse LU. The computation of coarse matrices at each Newton iteration proceeds recursively from the finest coarse level until the coarsest level by simply first injecting the finer iterate and then computing the Jacobian. We use a direct solver at the coarsest level is the interpolation is linear. Line search is performed with augmented Lagrangian merit function, cubic backtracking, $c_1 = 10^{-4}$ in (2.3), a minimum allowed step length of 10^{-6} and with the penalty parameter strategy explained in Section 2 with $\rho^{(-1)}$=10 and $\beta = 1/2$. For Newton iterations, the maximum allowed number is 100 and the absolute (relative) stopping tolerance is 10^{-6} (10^{-10}). Simulation problems are solved with NKSz [7]. We do not use Reynolds continuation in any of the algorithms.

4.2. Results of numerical experiments. We performed tests on a cluster of Linux PCs and developed our software using the Portable, Extensible Toolkit for Scientific Computing (PETSc) from Argonne National Laboratory [2]. Since the cluster network is relatively slow and is shared with other processes, our analysis focuses on the number of Newton and Krylov iterations, not on CPU times.

Since MPI is not deterministic, i.e., consecutive runs perform arithmetic operations with eventual different orders of processors, an indication of a good preconditioner is a stable number of Krylov iterations accross repeated runs of harder problems. We observed such stable behavior in the experiments with multi-level LNKSz. Regarding overlapping, a proper overlap for the control problems in our tests was two to four times bigger than a proper overlap for the corresponding simulation problems. With respect to line search, we have not experienced difficulty on guaranteeing inequality (2.4) and the value of $\rho^{(k)}$ always became fixed after some iterations, i.e., the penalty parameter heuristic described in Section 2 worked well for the examples of this paper.

Table 4.1 shows the efficacy of the modified interpolation process, which performs much better than the unmodified one, causing the two-level preconditioner to outperform the one-level preconditioner. Table 4.2 shows the flexibility of the two-level LNKSz, which provides a similar average number of Krylov iterations throughout all seven situations in the table. Figure 4.2-*a* shows the controlled velocity field for the cavity flow with $Re = 200$: although the fluid in the interior follows the clockwise direction imposed by the external force, the movement near the boundary is much less intense. In fact, a zoom on the bounday velocity shows the control acting in the counter-clockwise way, making the integral of ω^2 on Ω to decrease from the value of 55.4 for the simulation with slip walls to the controlled value of 32.5. Figure 4.2-*b* shows the sharp jump of the Lagrangian multiplier function λ_1 at the boundary. The other two Lagrangian multipliers behave similarly, no matter how fine the mesh is. Finally, Figures 4.2-*c* and 4.2-*d* clearly show the stabilization on the average number of Krylov iterations provided by the two-level preconditioner with modified interpolation. The one-level preconditioner fails with 100 processors for $Re = 250$ and $Re = 300$.

Regarding the experiments on the backward-facing step flows, Figure 4.3-*a* shows the computed velocity field for the simulation problem with $Re = 200$. The computed control acts on the left bottom corner of the domain, again in a counter-clockwise way, diminishing the intesity of the clockwise vortex observed in that region and causing the objective functional value to decrease. Although the one-level method now converges with 100 processors for all three Re, Figures 4.3-*b* and 4.3-*c* also show the stabilization on the average number of Krylov iterations provided by the two-level preconditioner with modified interpolation. Although in the case of backward-facing step control problems the unmodified interpolation provides a

TABLE 4.1

Results of LNKSz, with right preconditioned GMRES, for the cavity flow control problem with Re=250, a 280×280 mesh ($631,688$ variables), 49 processors, relative overlapping $\delta/H = 1/4$ and a 70×70 coarse mesh, for different combinations of number L of levels, linear interpolation type, number σ of pre and post smoother iterations, and RAS preconditioner. k is the total number of Newton iterations and $\overline{\ell}$ is the average number of Krylov iterations per Newton iteration.

L	Linear Interpolation Type	σ	RAS preconditioner	
			ℓ-RAS	r-RAS
1	–	–	$k=8$; $\overline{\ell}=336$	$k=9$; $\overline{\ell}=973$
2	Unmodified	1	$k=8$; $\overline{\ell}=1,110$	$k=8$; $\overline{\ell}=1,150$
2	Unmodified	2	$k=8$; $\overline{\ell}=356$	$k=8$; $\overline{\ell}=222$
2	Modified	1	$k=8$; $\boldsymbol{\overline{\ell}=21}$	$k=8$; $\boldsymbol{\overline{\ell}=28}$

TABLE 4.2

Results of two-level LNKSz, with right preconditioned GMRES, for the cavity flow control problem with Re=300 and a 70×70 coarse mesh, for different situations of number N_p of processors and mesh size. k is the total number of Newton iterations and $\overline{\ell}$ is the average number of Krylov iterations per Newton iteration. To each situation corresponds a combination of the number σ of Richardson iterations, the RAS preconditioner and the relative overlapping δ/H used in the pre and post smoothers. The number of variables is $2,517,768$ in the case of finest mesh.

N_p	$\frac{\delta}{H}$	140×140	280×280	560×560
25	$\frac{1}{4}$	$\sigma=1$; r-RAS $k=13$; $\boldsymbol{\overline{\ell}=20}$	$\sigma=1$; r-RAS $k=10$; $\boldsymbol{\overline{\ell}=23}$	– –
49	$\frac{1}{2}$	$\sigma=1$; r-RAS $k=13$; $\boldsymbol{\overline{\ell}=18}$	$\sigma=1$; r-RAS $k=10$; $\boldsymbol{\overline{\ell}=21}$	– –
100	$\frac{1}{2}$	$\sigma=1$; ℓ-RAS $k=13$; $\boldsymbol{\overline{\ell}=18}$	$\sigma=1$; ℓ-RAS $k=10$; $\boldsymbol{\overline{\ell}=25}$	$\sigma=2$; r-RAS $k=8$; $\boldsymbol{\overline{\ell}=27}$

better convergence than the one-level preconditioner as well, the use of the modified interpolation decreases the average number of Krylov iterations by a further factor of two.

Theoretically, the solution of the KKT system through proper steps is guaranteed only to be a local minimum, i.e., one might obtain a larger objective function value than the simulation problem used for comparison. In our tests, however, this was never the case. Finally, Figures 4.2-d and 4.3-c are consistent with prediction (3.5), while the information in Table 4.2 for 100 processors is consistent with (3.6).

5. Conclusions. We have developed a parallel multi-level LNKSz algorithm that showed a robust performance when tested on some tangential boundary flow control problems.

In general, the success of Lagrange-Newton methods with line search rests on four tasks: the computation of the Jacobian, the computation of a Newton step with descent direction, the choice of a penalty parameter in the merit function, and the computation of the step length. Although we address all these issues, our main contribution in this paper consists on the combination of a general multigrid V-cycle preconditioner with (1) RAS preconditioned Richardson smoothers and (2) a modified interpolation procedure that removes the pollution often generated by the application of common interpolation techniques to the Lagrange multipliers. Such combination was key for the successfull application of the two-level LNKSz in our experiments and the consequent improvement over the one-level method, handling flow control problems with higher Reynolds number, finer meshes and more processors. Surprisingly, RAS preconditioners performed much better than the classical ones.

Multi-level LNKSz is a flexible algorithm, and since it is also fully coupled (in contrast to operator-splitting, Schur complement, reduced space techniques), the original sparsity of a discretized PDE constrained optimization problem is maintained throughout its entire application and fewer sequential preconditioning steps are needed. We expect multi-level LNKSz to have a wide application in many PDE-constrained optimization problems.

6. Acknowledgements. We would like to thank the PETSc team for helping us using the PETSc library, the Hemisphere cluster team at the University of Colorado at Boulder for their support and Professor Richard Byrd for his useful comments.

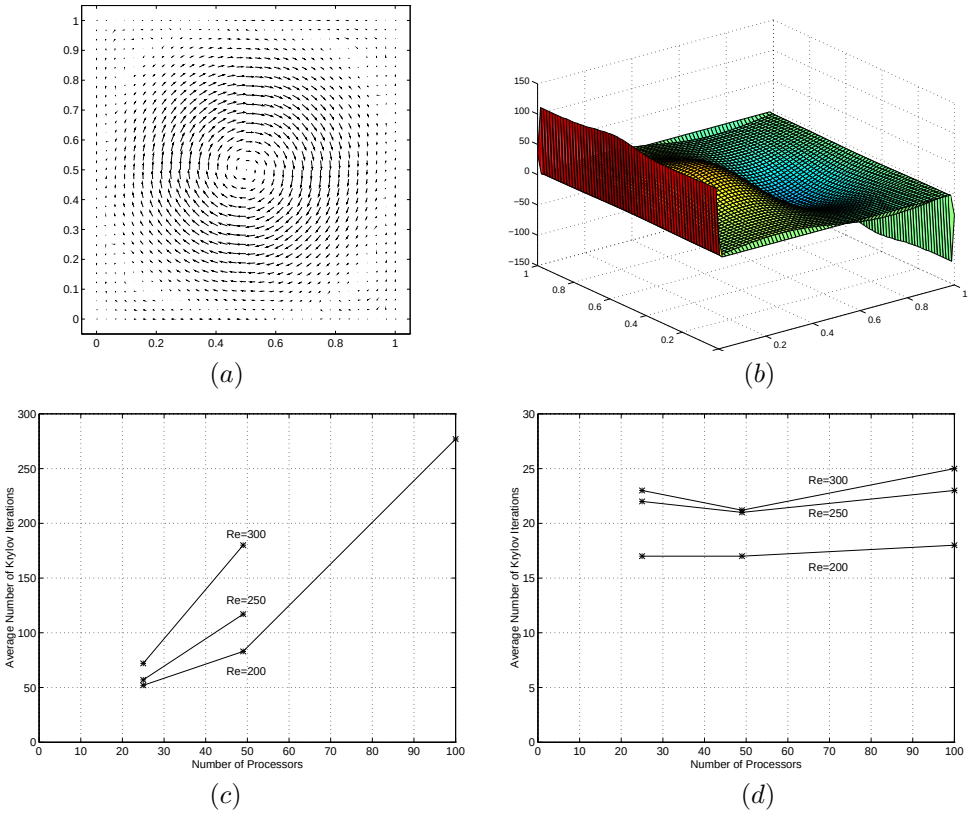


FIG. 4.2. *Information on cavity flow problems: (a) controlled velocity field with* $Re = 200$ *and (b) corresponding Lagrange multiplier* λ_1*; results for (c) one-level and (d) two-level LNKSz with right-preconditioned GMRES, 49 processors, a* 280×280 *mesh (*$631,688$ *variables), and a* 70×70 *coarse mesh.*

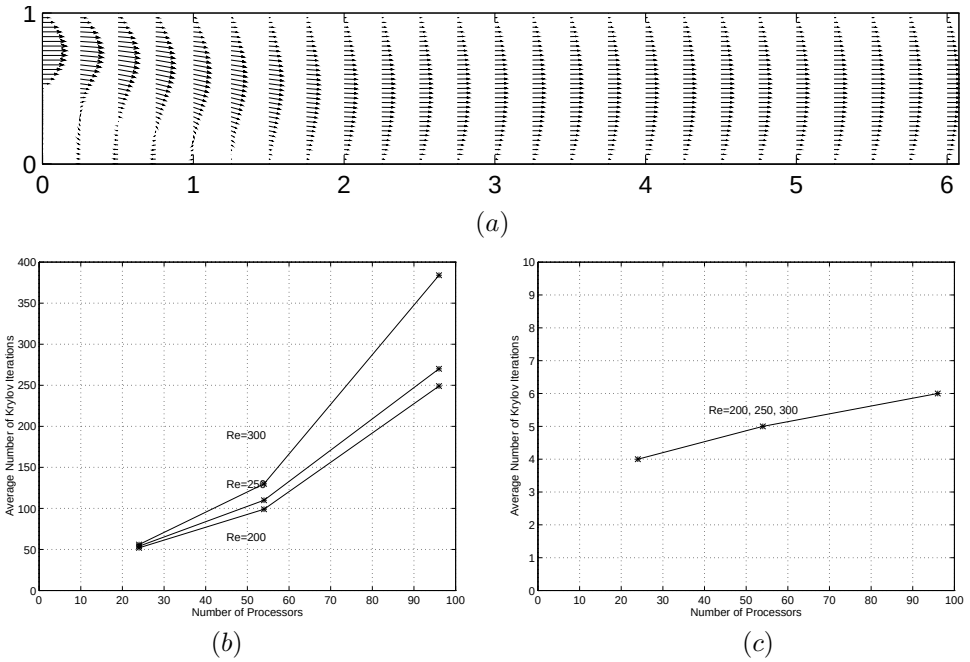


FIG. 4.3. *Information for backward facing-step flow problems: (a) simulated velocity field for* $Re = 200$*; results for (b) one-level and (c) two-level LNKSz with left-preconditioned GMRES, 54 processors, a* 576×96 *mesh (*$447,752$ *variables), and a* 144×24 *coarse mesh.*

REFERENCES

[1] F. Abergel and R. Temam, *On some control problems in fluid mechanics*, Theoret. Comput. Fluid Dynamics, 1 (1990), pp. 303–325.

[2] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *Portable, Extensible Toolkit for Scientific Computation (PETSc) home page*. http://www.mcs.anl.gov/petsc, 2004.

[3] G. Biros and O. Ghattas, *Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization, part I: The Krylov-Schur solver*, SIAM J. Sci. Comput., (to appear).

[4] ———, *Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization, part II: The Lagrange-Newton solver and its application to optimal control of steady viscous flows*, SIAM J. Sci. Comput., (to appear).

[5] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, SIAM, 2nd ed., 2000.

[6] X.-C. Cai, M. Dryja, and M. Sarkis, *Restricted additive Schwarz preconditioners with harmonic overlap for symmetric positive definite linear systems*, SIAM J. Numer. Anal., 41 (2003), pp. 1209–1231.

[7] X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, and D. P. Young, *Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput., 19 (1998), pp. 246–265.

[8] X.-C. Cai and D. E. Keyes, *Nonlinearly preconditioned inexact Newton algorithms*, SIAM J. Sci. Comput., 24 (2002), pp. 183–200.

[9] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797.

[10] X.-C. Cai and O. B. Widlund, *Domain decomposition algorithms for indefinite elliptic problems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 243–258.

[11] T. F. Coleman and J. J. Moré, *Estimation of sparse Jacobian matrices and graph coloring problems*, SIAM J. Numer. Anal., 20 (1983), pp. 187–209.

[12] O. Daube, *Resolution of the 2D Navier-Stokes equations in velocity-vorticity form by means of an influence matrix technique*, J. of Comput. Phys., 103 (1992), pp. 402–414.

[13] M. Dryja and W. Hackbush, *On the nonlinear domain decomposition method*, BIT, (1997), pp. 296–311.

[14] M. Dryja and O. Widlund, *Domain decomposition algorithms with small overlap*, SIAM J. Sci. Comp., 15 (1994), pp. 604–620.

[15] S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton method*, SIAM J. Optim., 4 (1994), pp. 393–422.

[16] G. Guj and F. Stella, *Numerical solutions of high-Re recirculating flows in vorticity-velocity form*, International Journal for Numerical Methods in Fluids, 8 (1988), pp. 405–416.

[17] M. D. Gunzburger, *Perspectives in Flow Control and Optimization*, SIAM, 1st ed., 2003.

[18] L. S. Hou and S. S. Ravindran, *Numerical approximation of optimal flow control problems by a penalty method: Error estimates and numerical results*, SIAM J. Sci. Comput., 20 (1999), pp. 1753–1777.

[19] A. Ioffe and V. Tihomirov, *Theory of Extremal Problems*, North-Holland Publishing Company, 1979. Translation from russian edition ©NAUKA, Moscow, 1974.

[20] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1st ed., 2000.

[21] S. C. R. Noël M. Nachtigal and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.

[22] E. Prudencio, R. Byrd, and X. C. Cai, *Parallel full space SQP Lagrange-Newton-Krylov-Schwarz algorithms for PDE-constrained optimization problems*, SIAM J. Sci. Comp., (submitted).

[23] L. Quartapelle, *Numerical Solution of the Incompressible Navier-Stokes Equations*, International Series of Numerical Mathematics, Vol. 113, Birkhäuser Verlag, 1996.

[24] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2nd ed., 2003.

[25] B. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1st ed., 1996.

[26] S. S. Sritharan, *Optimal Control of Viscous Flow*, SIAM, 1998.

Discrete Multistep Projection Methods for Incompressible Fluid Flow Problems

Jörg-M. Sautter*

Abstract. Efficient time integration methods for incompressible fluid flow simulations in arbitrary (two- or three-dimensional) domains are of great importance in many industrial applications. New numerical simulation techniques based on discrete multistep projection methods are introduced and their increased efficiency compared to standard methods is demonstrated. Furthermore, coupling with other highly nonlinear physical effects, such as fluid-structure interaction, is described and a the results of a fluid-structure interaction simulation are presented.

1 Introduction

It is shown in [21] that transient flow of isothermal incompressible Newtonian or non-Newtonian fluids is governed by the following Navier-Stokes initial boundary value problem in velocity-pressure form:

$$\varrho\left(\frac{\partial u}{\partial t}+(u\cdot\nabla)u\right)=\nabla\cdot\left(-pI_{d\times d}+\eta(\nabla u+(\nabla u)^T)\right)+b \qquad \text{in } \Omega \text{ for } t\in[0,t_{\text{final}}] \tag{1a}$$

$$\nabla\cdot u=0 \qquad \text{in } \Omega \text{ for } t\in[0,t_{\text{final}}] \tag{1b}$$

$$u=u_{\text{in}} \qquad \text{on } \Gamma_{\text{in}}\subset\Gamma^D \tag{1c}$$

$$u=u_D \qquad \text{on } \Gamma_{\text{solid}}\subset\Gamma^D \tag{1d}$$

$$f_{\text{st}}=(\eta(\nabla u+(\nabla u)^T)-pI_{d\times d})n \qquad \text{on } \Gamma_{\text{fb}}\subset\Gamma^N \tag{1e}$$

$$f_{\text{out}}=(\eta(\nabla u+(\nabla u)^T)-pI_{d\times d})n \qquad \text{on } \Gamma_{\text{out}}\subset\Gamma^N \tag{1f}$$

$$u=u_0 \qquad \text{in } \Omega \text{ at } t=0 \tag{1g}$$

$$\nabla\cdot u_0=0 \qquad \text{in } \Omega \tag{1h}$$

$$n\cdot u_0=n\cdot u_D \qquad \text{on } \Gamma^D \tag{1i}$$

where u is the fluid velocity, p the pressure, η the viscosity, ϱ the density, and b a given body force. Ω denotes the area occupied by the moving fluid and $\Gamma=\Gamma^D\cup\Gamma^N=\Gamma_{\text{in}}\cup\Gamma_{\text{out}}\cup\Gamma_{\text{solid}}\cup\Gamma_{\text{fb}}$ its boundary consisting of parts where fluid flows into and out of the domain and parts where the fluid suffices a no-slip condition or has a free boundary. Γ^D and Γ^N denote the parts where u is constrained by Dirichlet and Neumann boundary conditions, respectively. f_{st} represents the surface tension in case of a free boundary and f_{out} is the force acting upon the fluid at the outlet.

For Newtonian fluids, i.e. fluids of which the viscosity is constant in space and time, the momentum equation (1a) can be simplified to the more common form

$$\frac{\partial u}{\partial t}+(u\cdot\nabla)u=-\nabla p'+\nu\Delta u+b' \tag{2}$$

with the normalized pressure $p'=\frac{p}{\varrho}$, the kinematic viscosity $\nu=\frac{\eta}{\varrho}$, and the normalized body force density $b'=\frac{b}{\varrho}$. In contrast to (1a), the form (2) is not suitable for non-Newtonian fluid

*Applied Mathematics Group, Universität Düsseldorf, Universitätsstr. 1, 40225 Düsseldorf, Germany. E-Mail: `sautter@am.uni-duesseldorf.de`

flow and boundary conditions at an outflow or free boundary are no longer natural boundary conditions if the finite element Galerkin method is used for discretization in space. In fact, it leads to non-physical outflow boundary conditions in this case (cf. [21]).

It is important to note that the Navier-Stokes equations require no a priori boundary conditions on the pressure. Velocity boundary conditions applied to the momentum equations are sufficient to allow the determination of both velocity and pressure. For solving the Navier-Stokes equations numerically, the situation may be different (see section 3.3).

Solutions of (1) do not only depend on initial data, boundary values, and the viscosity, but also on the Reynolds number. For viscous fluids governed by the Navier-Stokes equations, the solution may show singularities when the domain contains vertices, the boundary values are not continuous, or the type of boundary condition changes between Dirichlet and Neumann. Adaptivity in space and time to control these phenomenons is not covered in this work. The goal of this work is to derive time integration schemes which are more efficient than standard methods.

Discretizing (1) in space with the Galerkin finite element method yields the discrete Navier-Stokes equations,

$$M\frac{du_h}{dt} = -A(u_h, u_h) + D(u_h) - Cp_h + f \tag{3a}$$

$$C^T u_h = g. \tag{3b}$$

For Newtonian fluids, the discrete diffusion term $D(u_h)$ is linear, i.e. $D(u_h) := Du_h$. The spatial discretization process and the efficient incorporation of boundary conditions is described in detail in [21].

2 Classical projection methods

In computational fluid dynamics, projection methods have been introduced as a useful and efficient way to significantly reduce the computational cost of time-dependent incompressible viscous flow simulations in the velocity-pressure formulation (see also [13] and references therein). This means that the system (1a)–(1b) is split into a series of simpler equations, such as advection-diffusion equations and Poisson equations, for example, and explicit or implicit updates.

The projection approach is strongly coupled with the type of time discretization and goes back to Chorin [2, 3, 4] and Temam [24, 25]. It is certainly one of the most controversial issues in computation fluid dynamics of incompressible viscous fluid flow, the controversy stemming from the boundary conditions satisfied by the pressure (see also [8, 14, 15, 16, 17] and references therein). The rigorous underlying analysis and understanding is still incomplete (cf. [18, 19]), particularly concerning spurious pressure boundary layers. Additionally, smaller time steps have to be chosen due to the more explicit character of the schemes.

There are two approaches for applying projection methods: continuous in space projection methods (typically referred to as projection methods) like the methods of Chorin [2, 3, 4], Goda [7], Van Kan [13] which belong to the class of Rothe methods and discrete projection methods such as Turek's (cf. [30]) discrete counterparts of Chorin and Van Kan which belong to the methods of lines class.

The continuous in space approach is most common in computational fluid dynamics, especially when finite differences or pseudospectral methods are used for the spatial discretization.

The so-called discrete projection method belongs to the methods of lines class. We shall first carry out a semi-discretization, i.e. proceed with spatial discretization while retaining the temporal derivative. Time marching is then treated separately in a second step.

A detailed comparison of the continuous in space approach and the method of lines approach (typically used in the finite element context), which leads to discrete projection methods, can be found in [21].

So far, little attention has been paid to improving the efficiency of the discrete methods by incorporating other schemes than implicit Euler or Crank-Nicolson or variants thereof. Discrete projection methods based on implicit Euler and Crank-Nicolson scheme have been extensively studied by Turek (cf. [27, 28, 29, 30]).

In the following, it is shown how the advantages of discrete projection methods can be carried over to BDF methods. The result are more efficient integrators than discrete versions of Chorin and Van Kan. In addition, the techniques suggested in [28] may also be applied.

3 Discrete multistep projection methods

3.1 Multistep methods

BDF methods are multistep methods for the numerical solution of systems of ordinary differential equations. Using the ε-embedding method (cf. [9]), a BDF method can also be applied to the (infinitely stiff) differential algebraic equation (3). This yields

$$L^n_{\Delta t,k}(Mu_h^{n+1}) = -A(u_h^{n+1}, u_h^{n+1}) + D(u_h^{n+1}) - Cp_h^{n+1} + f^{n+1} \tag{4a}$$

$$C^T u_h^{n+1} = g^{n+1} \tag{4b}$$

provided that u_h^n is discretely divergence-free. The operator $L^n_{\Delta t,k}$ represents the BDF scheme approximation of $\frac{d}{dt}$,

$$L^n_{\Delta t,k}(My^{n+1}) = \frac{1}{\Delta t}\sum_{m=0}^{k} \beta_m My^{n+1-m} \tag{5}$$

with appropriate coefficients β_i, $i = 1, \ldots, k$. For $k = 1$ and $\beta_0 = 1$, $\beta_1 = -1$ we get the (fully coupled) implicit Euler method. For $k = 2$ we set $\beta_0 = \frac{3}{2}$, $\beta_1 = -2$, $\beta_2 = \frac{1}{2}$ and for $k = 3$ we use $\beta_0 = \frac{11}{6}$, $\beta_1 = -3$, $\beta_2 = \frac{3}{2}$, $\beta_3 = \frac{1}{3}$ (see [5]). (4) represents a fully coupled nonlinear system of equations. Although it is more efficient than the discrete θ-method (if k is sufficiently large), the cost per time step is approximately the same. Therefore, velocity and pressure are decoupled by introducing a projection step.

3.2 Velocity-pressure decoupling

Velocity and pressure can be decoupled by introducing a projection step in order to reduce the numerical cost per time step. This is a generalization and extension of the continuous in space approach suggested by [10, 11] for the continuous in space linearized Navier-Stokes equations. These approaches are based on pseudospectral discretizations and are only demonstrated to work on very simple geometries and with homogeneous Dirichlet boundary conditions. Furthermore, boundary conditions for the projection step are chosen to be too simple for this approach, namely homogeneous Neumann boundary conditions. In contrast to these

approaches, the discrete approach presented here takes care of pressure modes, works on arbitrary geometries in combination with a large variety of physically relevant boundary conditions (see [21]) and the nonlinearity is treated fully implicitly which allows the simulation of non-Newtonian fluid flows. If desired, the nonlinear advection term may also be treated explicitly in the case of Newtonian fluids. The full set of options is described in [21].

In order to reduce the computational cost, consider the following splitting scheme:

$$\frac{1}{\Delta t} \quad \beta_0 M \widetilde{u}_h^{n+1} + M \sum_{m=1}^{k} \beta_m u_h^{n+1-m} \Bigg) = -A(\widetilde{u}_h^{n+1}, \widetilde{u}_h^{n+1}) + D(\widetilde{u}_h^{n+1}) - C\widetilde{p}_l^{n+1} + f^{n+1} \tag{6}$$

with pressure correction

$$\beta_0 M \frac{u_h^{n+1} - \widetilde{u}_h^{n+1}}{\Delta t} + C(p_h^{n+1} - \widetilde{p}_l^{n+1}) = 0$$
$$C^T u_h^{n+1} = g^{n+1}$$

and

$$\widetilde{p}_l^{n+1} = \sum_{m=0}^{l-1} \gamma_m p_h^{n-m} \tag{7}$$

with appropriate coefficients γ_m, $m = 0, \ldots, l-1$. For $l = 1$ we have $\gamma_0 = 1$ and for $l = 2$ $\gamma_0 = 2$, $\gamma_1 = -1$, for $l = 3$: $\gamma_0 = 3$, $\gamma_1 = -3$, $\gamma_2 = 1$. For $k = 1$ and $l = 0$ this is the discrete projection method of Chorin type. The projection step is realized, in principle, by solving

$$C^T M^{-1} C \lambda_h = C^T \widetilde{u}_h - g^{n+1} \tag{8}$$

(see [21] for details) and then performing two updates

$$u_h^{n+1} = \widetilde{u}_h - M^{-1} C \lambda_h \tag{9a}$$
$$p_h^{n+1} = \widetilde{p}_h + \frac{\beta_0}{\Delta t} \lambda_h. \tag{9b}$$

Using this approach, the nonlinear iteration in order to solve (6) is only in the velocity unknowns due to the pressure decoupling, i.e. only "small" linear systems need to be solved. Therefore, the nonlinear iteration is significantly cheaper compared to a fully coupled approach. The linear systems which need to be solved within the nonlinear iteration always have full rank, no matter whether there are pressure modes or not. In other words, pressure modes don't affect the velocity step. In addition, only one "large" linear system or a pressure Poisson equation must be solved per time step in the projection step. If there are pressure modes, they have to be taken care of only once per time step. Theoretically, the approximations u_h^{n+1} are discretely divergence-free. In practice, however, this depends on how the projection step is actually performed (see [21] for details).

3.3 Concomitant pressure field

In contrast to the continuous in space equations (1) which do not require initial data for the pressure, all discrete projection methods (except those of Chorin type) do require an initial

pressure. Given a discrete velocity field u_h, the concomitant pressure field can be obtained by differentiating (3b) with respect to time and plugging into (3a). This yields

$$C^T M^{-1} C p_h(t) = C^T M^{-1}(-A(u_h, u_h) + D(u_h) + f(t)) - \dot{g}(t).$$

The efficient solution of the pressure Poisson equation is described in [21]. Note that the issue with boundary conditions for the pressure Poisson equation does, in contrast to continuous in space projection methods, not arise here.

4 Algebraic multigrid

4.1 Treatment of the nonlinear subproblems

For non-Newtonian fluids or Newtonian fluids where the advection term is treated implicitly, the methods in section 3 require the solution of a nonlinear system of equations in the eliminated velocity unknowns in every time step or in every stage of the method. These nonlinear equations are solved by a simplified Newton method which yields a series of linear subproblems.

4.2 Treatment of the linear subproblems by AMG

The projection methods presented in section 3.2 require the solution of linear systems

$$Mu = b \tag{10}$$

and

$$(M + c_1 \Delta t J_A - c_2 \Delta t J_D))u = b, \qquad c_1, c_2 \in \mathbb{R}. \tag{11}$$

in steps (6), (8), and (9a). These linear subproblems are solved with an algebraic multigrid (AMG) solver (cf. [1, 6, 12, 22, 23]) based on the Ruge-Stüben algorithm (cf [20]). In contrast to geometric multigrid methods, AMG requires no a priori given hierarchy of coarse grids. In fact, the construction of a problem-dependent hierarchy – including the coarsening process itself, the transfer operators as well as the coarse-grid operators – is part of the AMG algorithm, based solely on algebraic information contained in the given system of equations (cf. [26]).

Geometrically oriented approaches, however, can hardly cope with the complex geometries occurring in simulations of real-life problems. There is generally no natural grid hierarchy which could easily be exploited. But even if there was such a hierarchy, the coarsest level would still be required to be fine enough to resolve the geometry to some extend. For industrially relevant configurations, such coarse grids would still be much too fine for efficient multilevel solutions (cf. [26]). For example, there is typically no hierarchy of coarse grids a priori available for grids coming form an unknown source. It is shown in the following that AMG also works very well on unstructured grids for a modified driven cavity problem (see figure 3). More numerical examples can be found in [21].

Although AMG has originally been developed for symmetric, positive definite problems, it can also be applied to non-symmetric systems. However, the convergence behavior is not clear in this case. The performance of AMG applied to non-symmetric convection-dominated systems is studied in the following. However, it should be said that at this point, AMG

for non-linear problems is still a field of major ongoing research and there is no well-settled approach known yet.

If not specified otherwise, Gauß-Seidel smoothing is used within AMG V-cycles with ν_1 pre-smoothing relaxations and ν_2 post-smoothing relaxations (abbreviated by AMG(ν_1,ν_2) in the following), typically $\nu_1 = \nu_2 = 2$. The grid coarsening is performed until the number of unknowns on the coarsest grid drops below 50. A direct method is then used to solve on the coarsest level.

Let d^m, $m \in \mathbb{N}\backslash\{0\}$, denote the defect of AMG(ν_1,ν_2) after m V-cycles. Then the defect reduction factors are defined as

$$q^{(m)} := \frac{||d^m||}{||d^{m-1}||}.$$

If not specified otherwise, the Euclidean norm is used. The convergence factor ρ of the AMG(ν_1,ν_2) method can be estimated by the average defect reduction factor $\widehat{q}^{(m)}$, i.e.

$$\rho \approx \widehat{q}^{(m)} := \sqrt[m]{q^{(m)} q^{(m-1)} \cdots q^{(1)}} = \sqrt[m]{\frac{||d^m||}{||d^0||}}$$

(see also [26]).

The convergence histories and rates of AMG(2,2) applied to (10) on different grids of the modified driven cavity problem presented in section 5.1 are shown in table 1 and figure 1(a). It can be seen that the convergence is essentially grid independent. Note that the different grids are not regular refinements of each other and therefore true grid independent convergence can not be expected.

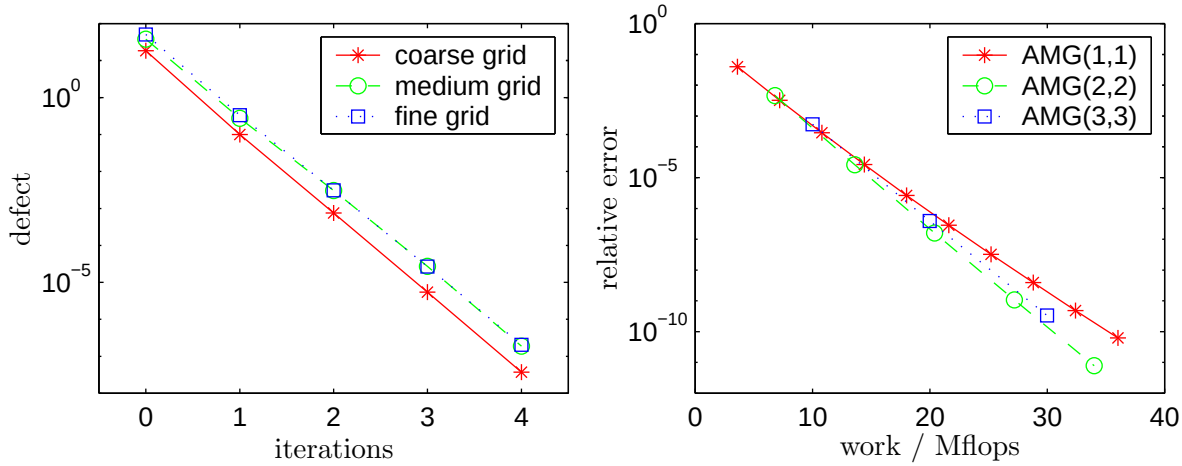
grid	$q^{(1)}$	$q^{(2)}$	$q^{(3)}$	$q^{(4)}$	$\widehat{q}^{(4)} \approx \rho$
coarse	0.0051	0.0074	0.0075	0.0066	0.0066
medium	0.0072	0.0111	0.0090	0.0070	0.0084
fine	0.0062	0.0092	0.0085	0.0074	0.0077

Table 1: Convergence rates on different grids

The efficiency of AMG(ν_1,ν_2) applied to (10) on the fine grid is shown in figure 1(b). It turns out that $\nu_1 = \nu_2 = 2$ is the optimal choice for this test problem. The performance of AMG(1,2), AMG(2,1), AMG(3,2), AMG(2,3), etc. has been found to be inferior to that of AMG(2,2).

4.2.1 AMG in the nonlinear iteration

The convergence histories of AMG(2,2) applied to the linear systems (11) occurring within the (simplified) Newton method have been studied on the finest grid of the test problem. The convergence history of AMG depends on the temporal step size of the underlying time integration scheme and whether the advection term is treated fully implicitly or explicitly, as in the case of semi-implicit methods. The results are shown in figure 2(a) for different step sizes. The convergence rates are shown in table 2. Figure 2(b) shows the dependence of the convergence factor on the temporal step size for both, fully implicit and semi-implicit methods. These numerical tests show that the convergence factors are excellent for both fully implicit and semi implicit treatment of the velocity step. In fact, as the time step size decreases, the efficiency of AMG increases and therefore contributes to a lower increase of numerical work of the time stepping schemes for small step sizes.



(a) Grid independent convergence of AMG

(b) Efficiency of AMG

Figure 1: AMG convergence

	$\Delta t = 0.1$	0.05	0.02	0.01	0.005	0.002	0.001
fully implicit	0.3832	0.3260	0.2219	0.1272	0.0834	0.0232	0.0047
semi implicit	0.4638	0.3764	0.2305	0.1446	0.0847	0.0241	0.0059

Table 2: Convergence rates $\rho(\Delta t)$ for different step sizes

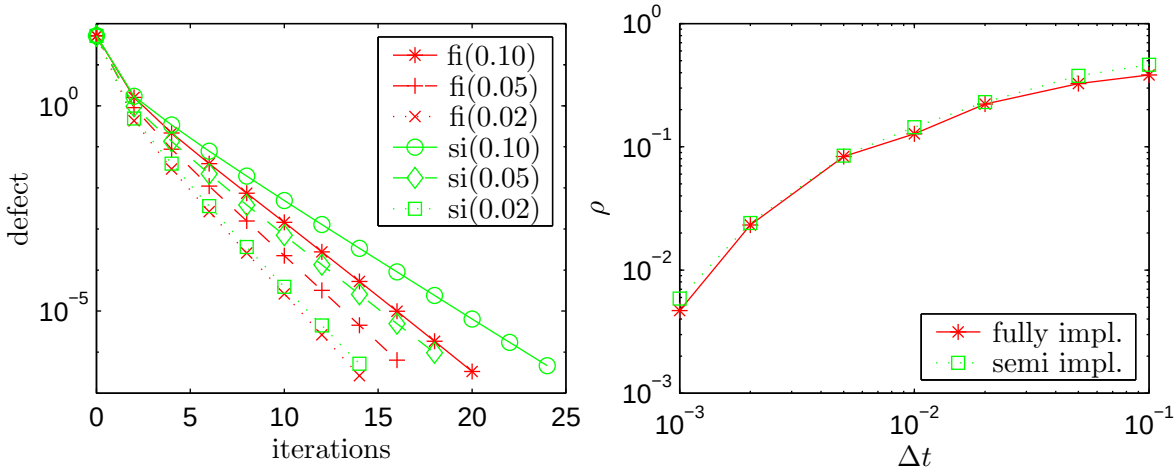
5 Numerical examples

5.1 Modified driven cavity

The discrete multistep projection methods have been implemented in the codes SpexBDF(k,l) with k and l as defined in section 3.2. The performance of the discrete multistep projection methods compared to the discrete version of Van Kan's projection method, which is one of the most popular and most efficient methods used in practice (cf. [27, 29, 30]), applied to a modified driven cavity problem (see figure 3) is shown in figure 4. The error has been measured in the infinity norm. It can be seen that SpexBDF(3,2) is the most efficient code. This is due to the fact that SpexBDF(4,l) is unstable for $l = 3$ and therefore the order of the extrapolation had to be decreased, i.e. SpexBDF(4,2) was used instead. Nevertheless, this numerical example shows the advantages, the good performance of SpexBDF(3,2), and the limits of the velocity-pressure decoupling approach, i.e. SpexBDF(4,2) is less efficient than SpexBDF(3,2).

5.2 Fluid-structure interaction

As an object moves through a fluid, the viscosity of the fluid acts on the moving object with a force that resists the motion of the object. At the same time, the fluid flow depends on the moving obstacle. The fluid velocity on each structure boundary must be constrained to match the rigid-body motion of the structure. The approach presented here is based on body fitted unstructured moving grids for optimal spatial resolution and implicit or semi-implicit



(a) Convergence depending on the step size (fi(Δt)=fully implicit, si(Δt)=semi implicit)

(b) Convergence factor depending on the step size

Figure 2: AMG convergence in the velocity step

discrete projection methods.

In order to simulate the motion of $N_{\text{fsi}} \in \mathbb{N}$ moving obstacles within a fluid, the Navier-Stokes equations are coupled with the equations for rigid body motions. The initial boundary value problem for a fluid-structure interaction problem is therefore given by the complete initial boundary value problem (1) augmented by the following equations for $i = 1, \ldots, N_{\text{fsi}}$ moving structures fsi(i),

$$m_{\text{fsi}(i)} \frac{du_{\text{fsi}(i)}}{dt} = m_{\text{fsi}(i)} g + F_{\text{fsi}(i)}(u) \tag{12a}$$

$$\frac{dx_{\text{fsi}(i)}}{dt} = u_{\text{fsi}(i)} \tag{12b}$$

$$I_{\text{fsi}(i)} \frac{d\omega_{\text{fsi}(i)}}{dt} = T_{\text{fsi}(i)}(u) \tag{12c}$$

$$\frac{d\theta_{\text{fsi}(i)}}{dt} = \omega_{\text{fsi}(i)} \tag{12d}$$

$$F_{\text{fsi}(i)}(u) = \int_{\Gamma_{\text{struct}}} (-pI_{d\times d} + \eta(\nabla u + (\nabla u)^T))n \tag{12e}$$

$$T_{\text{fsi}(i)}(u) = \int_{\Gamma_{\text{struct}}} r_{\text{fsi}(i)} \tau^T (-pI_{d\times d} + \eta(\nabla u + (\nabla u)^T))n \tag{12f}$$

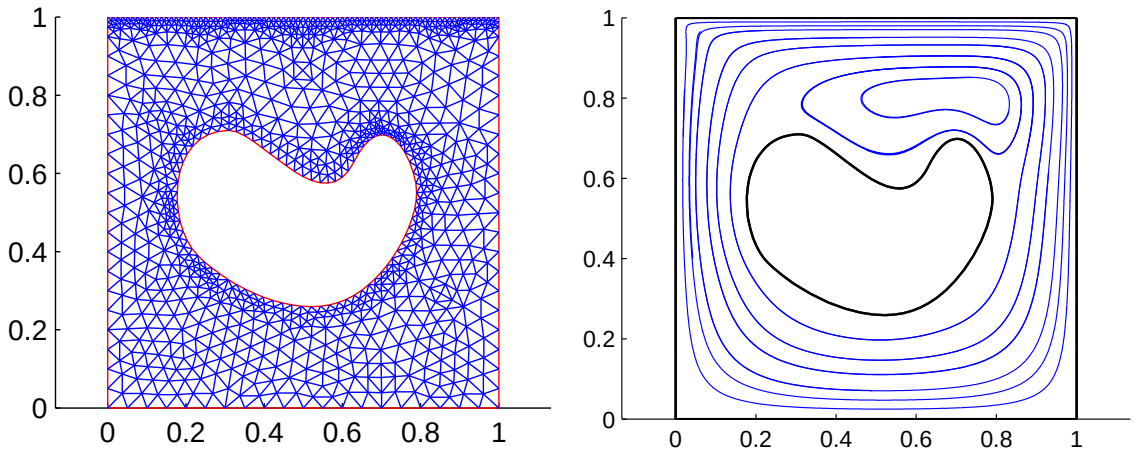
$$u = u_{\text{fsi}(i)} + \omega_{\text{fsi}(i)} \times r_{\text{fsi}(i)} \qquad \text{on } \Gamma_{\text{fsi}(i)}(t) \tag{12g}$$

$$u_{\text{fsi}(i)}(0) = u_{\text{fsi}(i),0} \tag{12h}$$

$$x_{\text{fsi}(i)}(0) = x_{\text{fsi}(i),0} \tag{12i}$$

$$\theta_{\text{fsi}(i)}(0) = \theta_{\text{fsi}(i),0} \tag{12j}$$

$$\omega_{\text{fsi}(i)}(0) = \omega_{\text{fsi}(i),0}. \tag{12k}$$



(a) $N_\Delta = 2006$, $N_{\mathrm{DOFe}} = 8816$

(b) Streamlines

Figure 3: Flow of the test problem

Mass, velocity of the center of mass, position of the center of mass, moment of inertia, angular velocity, and angle of rotation of the ith structure are denoted by $m_{\mathrm{fsi}(i)}$, $u_{\mathrm{fsi}(i)}$, $x_{\mathrm{fsi}(i)}$, $I_{\mathrm{fsi}(i)}$, $\omega_{\mathrm{fsi}(i)}$, and $\theta_{\mathrm{fsi}(i)}$, respectively, and gravity (or any other type of body force) is denoted by g. Equations (12e) and (12f) represent the force and torque exerted upon the ith structure by the flow. Most important for the simulation is the fluid-structure coupling expressed by the transient no-slip boundary conditions (12g). The initial boundary value problem (12) describes fluid-structure interaction problems as long as there are no collisions between the moving structures.

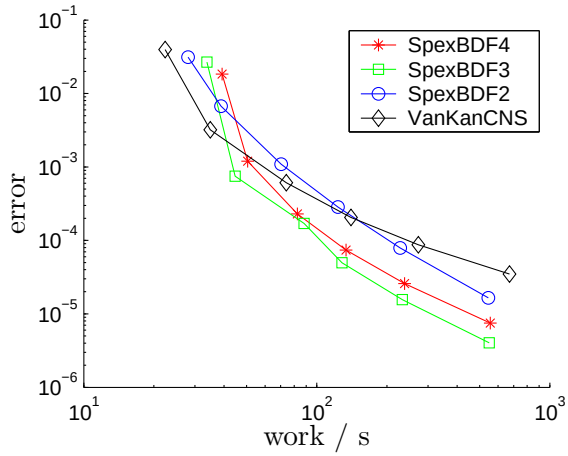


Figure 4: Work vs. accuracy

5.3 (Basic) Algorithm

The idea of body fitted grids is that the nodes on the structure are assumed to move with the obstacle. At each time step, the grid is updated according to the motion of the obstacle. Then a new grid is generated using the new edge elements in order to prevent distortion of the grid and the flow field is projected onto the new grid. The simulation of fluid flow in a time-dependent domain is significantly more expensive since all matrices and vectors are time-dependent. Furthermore, the grid generation within the simulation is very expensive.

The initial boundary value problem (1) and (12) is solved by the following splitting procedure: first, the fluid equations are solved subject to the correct boundary conditions on the moving structure which couple fluid motion and rigid body motion. Then, the equations of rigid body motion are integrated. This process is repeated while at the same time the grid is constantly updated.

As an example of fluid-structure interaction, consider the situation shown in figure 5. A reservoir is filled with fluid and more fluid is pumped into the reservoir through the inlet on the left side while fluid leaves the reservoir through the outlet on the right side. Within the reservoir, there is a solid structure (a circle) which moves under body forces (gravity) and the forces of the fluid acting upon the structure. The domain of the reservoir is represented by 18 boundary segments. Boundary segment Γ_1 represents the inlet where the velocity profile of the flow is given as

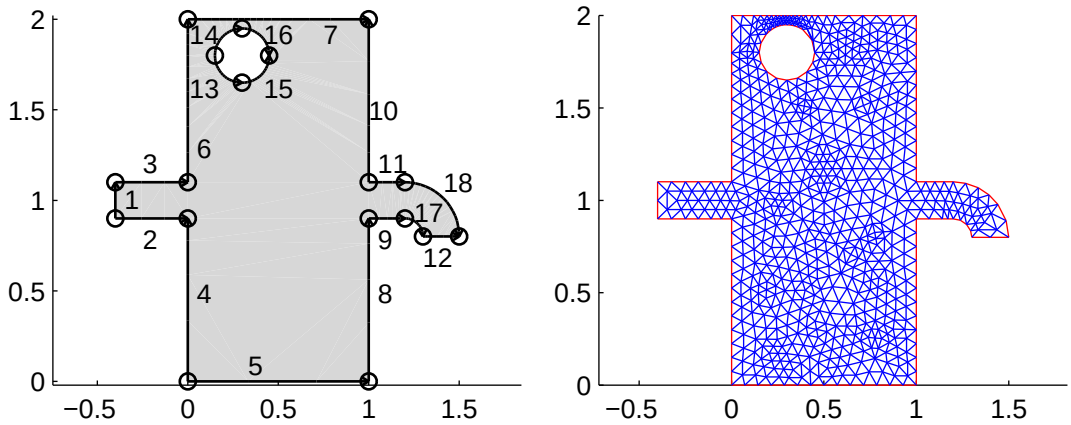
$$u_{\text{in}} = u_{\text{in}}(s) = (s(1-s), 0)^T$$

where $s \in [0, 1]$ linearly parameterizes Γ_1. Boundary segment Γ_{12} is the outlet where homogeneous traction boundary conditions are applied, i.e. the force acting upon the fluid at the outlet is $f_{\text{out}} = 0 \in \mathbb{R}^2$ on Γ_{12}. Let the density of the fluid be $\varrho = 1$, the viscosity $\eta = 0.1$, and the body force $b = (0, -2)^T$. The parameters of the structure are as follows: The circle's radius is $r = 0.15$ and it has a homogeneous mass density. Its mass is $m_{\text{fsi}} = 1$ and its moment of ineratia $I_{\text{fsi}} = 1$.

At $t = 0$, the fluid is at rest and the initial position of the center of mass and velocity of the structure are $x_{\text{fsi}}(0) = (0.3, 1.8)^T$ and $u_{\text{fsi}}(0) = (0, 0)^T$, respectively. In addition, the initial angular velocity of the structure is $\omega(0) = 2$ and its initial angular orientation is $\theta(0) = 0$.

As the sphere moves through the fluid, the viscosity of the fluid acts upon the moving sphere with a force that resists the motion of the object. Since the circle rotates at the beginning, it can be observed that the angular velocity of the structure decreases due to frictional forces of the viscous fluid. The evolution of the angular velocity $\omega(t)$ is shown in figure 6(a).

As the structure starts moving downward, it also moves slightly to the left since fluid mainly from the upper right follows the circles' path. Then, as the circle comes closer to the inlet, it gets swept away by the stream going mainly from inlet to outlet. Figure 6(b) shows the evolution of the horizontal position of the center of mass of the sphere. The gain of horizontal momentum, when the sphere gets caught by the stream, can be clearly seen. The results of this simulation have been computed using the techniques presented in section 3. Figure 7 shows a sequence of plots of the magnitude of the velocity and figure 8 shows the corresponding velocity vector field.



(a) Initial geometry

(b) Initial mesh

Figure 5: Initial configuration

6 Conclusions

It has been shown that discrete multistep projection methods can be much more efficient than the discrete version of Van Kan's projection method, which is one of the most popular and most efficient methods used in industrial applications nowadays. The discrete multistep projection methods excel in non-trivial two- and three-dimensional domains. For maximal geometric flexibility, algebraic multigrid is used for the linear subproblems occuring in the method. Applicability to non-trivial real-life applications such as fluid-structure interaction has been demonstrated. Adaptivity in space and time is currently under construction.

References

[1] W. L. Briggs, V. E. Henson, and S. McCormick. *A Multigrid Tutorial*. SIAM, Philadelphia, 2000.

[2] A. J. Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 2:12–26, 1967.

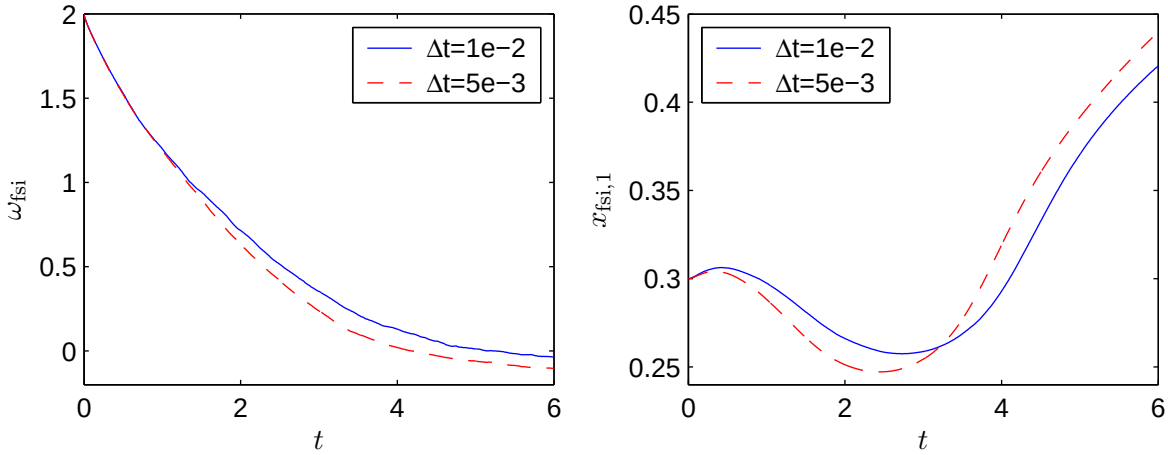
[3] A. J. Chorin. The numerical solution of the Navier-Stokes equations for an incompressible fluid. *Bull. Am. Math. Soc.*, 73:928–931, 1967.

[4] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computations*, 22:745–762, 1968.

[5] P. Deuflhard and F. Bornemann. *Numerische Mathematik II*. de Gruyter, Berlin, 1994.

[6] R. D. Falgout. Adaptive algebraic multigrid. Lawrence Livermore National Laboratory, 2002.

[7] K. Goda. A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows. *Journal on Computational Physics*, 30:76–95, 1979.



(a) Angular velocity

(b) Horizontal position

Figure 6: Evolution of angular velocity and horizontal position of center of mass

[8] P. M. Gresho and R. L. Sani. *Incompressible Flow and the Finite Element Method*, volume 2. John Wiley & Sons, Chichester, 2000.

[9] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II*. Springer-Verlag, Berlin, 2002.

[10] H. Haschke and W. Heinrichs. Splitting techniques for the Navier-Stokes equations. In P. Neittaanmk̈i, T. Tiihonen, and P. Tarvainen, editors, *Numerical Mathematics and Advanced Applications*, 2000.

[11] W. Heinrichs and H. Haschke. Splitting techniques with staggered grids for the Navier-Stokes equations. *Journal on Computational Physics*, 168:131–154, 2001.

[12] V. E. Henson. An algebraic multigrid tutorial. Lawrence Livermore National Laboratory, 1999.

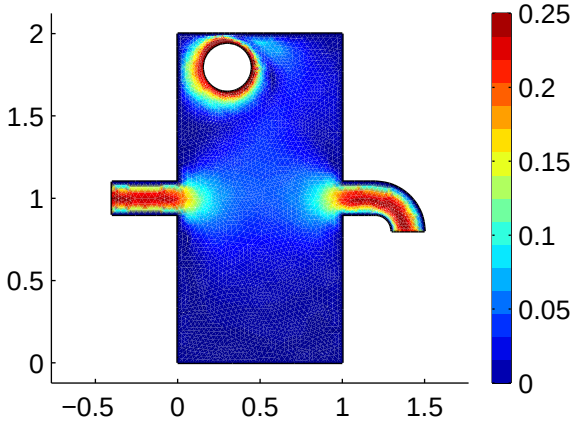
[13] J. V. Kan. A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM J. Sci. Stat. Comput.*, 7:870–891, 1986.

[14] M. Marion and R. Temam. Navier-Stokes equations. In *Handbook of Numerical Analysis*, volume 1, pages 197–462. North-Holland, Amsterdam, 1998.

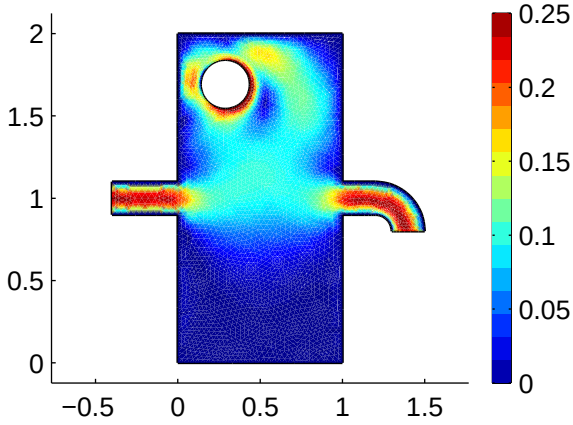
[15] A. Prohl. *Projection and quasi-compressibility methods for solving the incompressible Navier-Stokes equations*. Teubner, Stuttgart, 1997.

[16] L. Quartapelle. *Numerical Solution of the Incompressible Navier-Stokes Equations*. Birkhäuser, Basel, 1993.

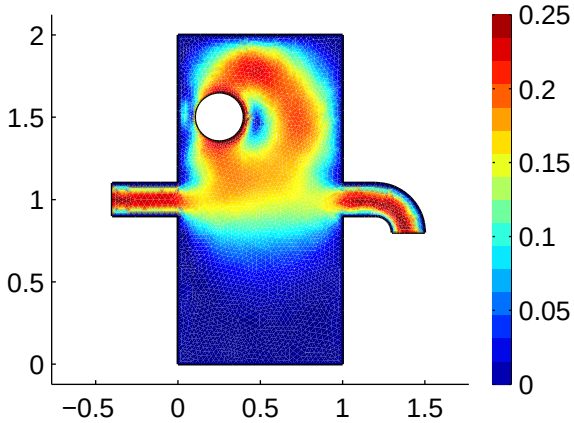
[17] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*. Springer-Verlag, Berlin, 1994.



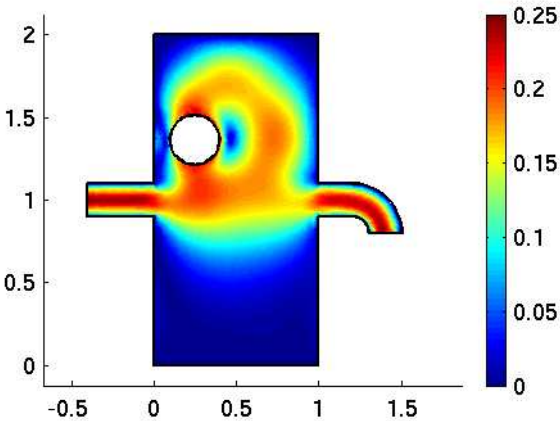
(a) $t = 0.1$



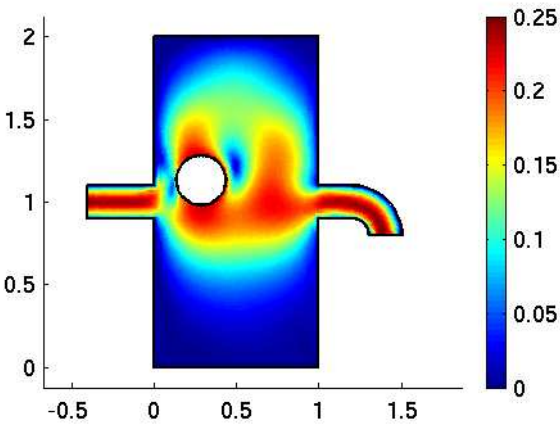
(b) $t = 1$



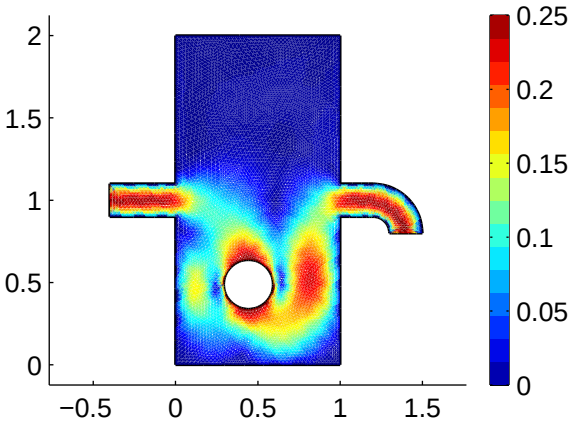
(c) $t = 2$



(d) $t = 3$

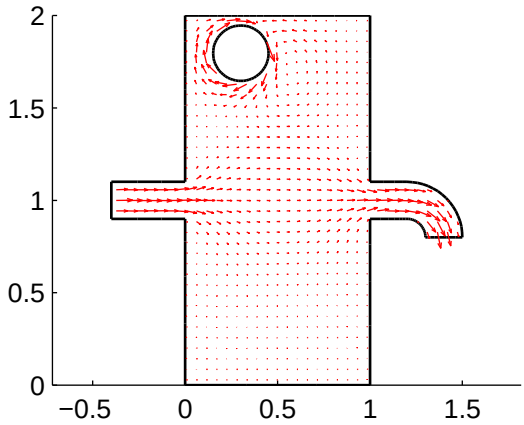


(e) $t = 4$

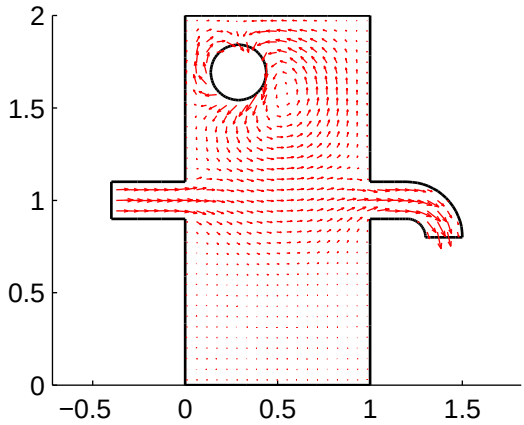


(f) $t = 6$

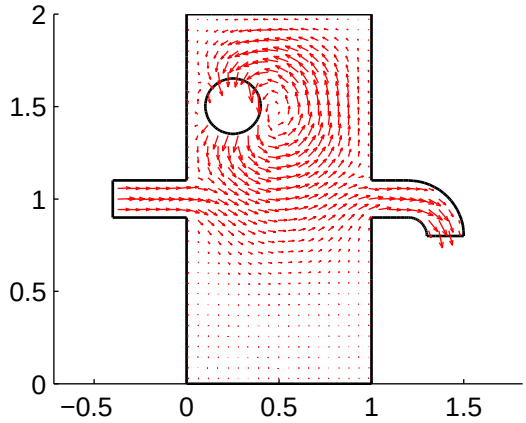
Figure 7: Magnitude of velocity



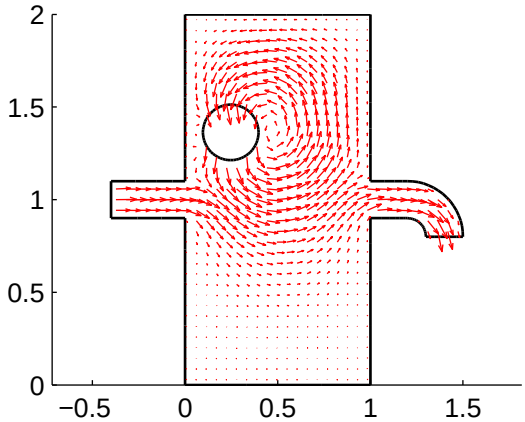
(a) $t = 0.1$



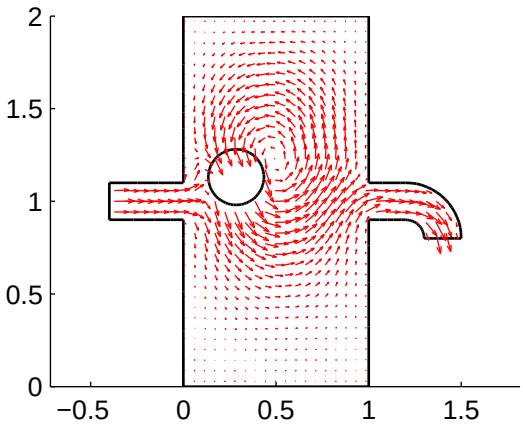
(b) $t = 1$



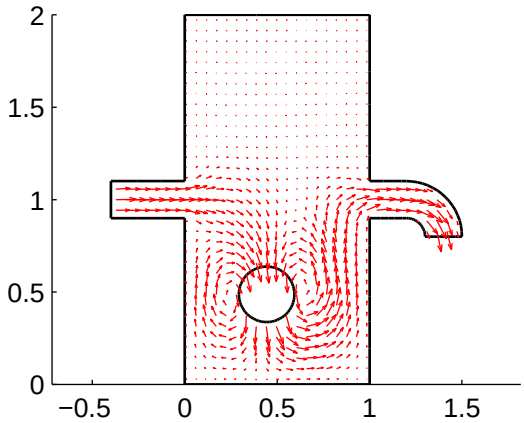
(c) $t = 2$



(d) $t = 3$



(e) $t = 4$



(f) $t = 6$

Figure 8: Velocity field

[18] R. Rannacher. On Chorin's projection method for the incompressible Navier-Stokes equations. In J. G. H. et al., editor, *The Navier-Stokes equations II - Theory and Numerical Methods*, volume 1530 of *Lecture Notes in Mathematics*, Berlin, 1992. Springer-Verlag.

[19] R. Rannacher. Finite element methods for the incompressible Navier-Stokes equations. Unpublished, University of Heidelberg, 1999.

[20] J. W. Ruge and K. Stüben. Algebraic multigrid. In S. F. McCormick, editor, *Multigrid Methods*, number 3 in Frontiers in Applied Mathematics. SIAM, Philadelphia, 1987.

[21] J.-M. Sautter. *Discrete Projection Methods for Time-Dependent Viscous Incompressible Fluid Flow Simulations in Arbitrary Domains.* PhD thesis, University of Düsseldorf, Germany, June 2004.

[22] K. Stüben. *Algebraic Multigrid (AMG): An Introduction with Applications.* GMD Report No. 70. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, Bonn, 1999.

[23] K. Stüben. *A Review of Algebraic Multigrid.* GMD Report No. 69. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, Bonn, 1999.

[24] R. Temam. Sur l'approximation de la solution des equations de Navier-Stokes par la méthode des fractionnaires I. *Arch. Rational Mech. Anal.*, 32(2):135–153, 1969.

[25] R. Temam. Sur l'approximation de la solution des equations de Navier-Stokes par la méthode des fractionnaires II. *Arch. Rational Mech. Anal.*, 33(5):377–385, 1969.

[26] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid.* Academic Press, London, 2001.

[27] S. Turek. A comparative study of some time-stepping techniques for the incompressible Navier-Stokes equations: From fully implicit nonlinear schemes to semi-implicit projection methods. *International Journal for Numerical Methods in Fluids*, 22:987–1011, 1996.

[28] S. Turek. *Multilevel Pressure Schur Complement Techniques for the Numerical Solution of the Incompressible Navier-Stokes Equations.* Habilitationsschrift, Universität Heidelberg, 1997.

[29] S. Turek. On discrete projection methods for the incompressible Navier-Stokes equations: An algorithmical approach. *Comp. Methods Appl. Mech. Engrg.*, 143:271–288, 1997.

[30] S. Turek. *Efficient solvers for incompressible flow problems: An algorithmic and computational approach*, volume 6 of *Lecture Notes in Computational Science and Engineering.* Springer-Verlag, Berlin, 1999.

A Multiscale Filter to Accelerate Multigrid Methods

Haim Waisman [a] Jacob Fish [a]

[a] *Department of Civil Engineering, Rensselaer Polytechnic Institute*

Raymond S. Tuminaro [b] John N. Shadid [b]

[b] *Sandia National Laboratories*

Abstract

We present the Generalized Global-Basis (GGB) method aimed at enhancing performance of multigrid solvers for difficult systems such as those arising from indefinite and nonsymmetric matrices. The GGB method is a multiscale filter that constructs an auxiliary coarse model from the largest eigenvalues of the iteration matrix. It projects these modes which would cause slow convergence to a coarse problem which is then used to eliminate these modes. For nonlinear problems the filter is modified (MGGB) to selectively reuse the same prolongation and thus reduce the amount of eigensolver calculations. This selection criteria is based on principal angles between subspaces spanned between the previous and current prolongation operators. Numerical examples show that best performance is obtained when GGB is accelerated by GMRES and used for problems with multiple right hand sides. In addition it is demonstrated that GGB can enhance restarted GMRES strategies by retention of important subspace information. For nonlinear problems, MGGB scheme indicate significant time savings.

1 Introduction

Multigrid methods are very efficient solvers for symmetric positive definite systems arising from partial differential equations. For such systems, multigrid preconditioners require computational work proportional to the number of unknowns. However, when the system is nonsymmetric or highly indefinite, multigrid may not perform as well [1]. Such systems arise in a variety of applications including linearized Navier-Stokes equations, saddle-point problems

⋆ This work was supported by the ASC DOE NNSA program at Sandia National Laboratories under contract number DE-ACD4-94AL85000.

* Corresponding author: waismh@rpi.edu (Haim Waisman)

Submitted to Copper Mountain conference on multigrid methods (2005)

and other. Some multigrid methods have been applied for certain weakly indefinite systems. However, the proposed strategies impose restrictions on the coarse grid, requiring that these grids are sufficiently fine for the algorithm to be convergent. For highly indefinite systems, convergence is not guaranteed and often multigrid methods are used to precondition Krylov methods, such as GMRES or QMR. Yet, for some problems GMRES might stall in the first iterations until it resolves the indefinite modes of the system. This can often lead to poor convergence rates [2].

In this paper, we present the Generalized Global-Basis (GGB) multiscale method [3] that generalizes the GB method [4]. The idea is to filter out modes that are slow to converge and resolve them on an additional coarse grid. This stabilizes and accelerates the iterative process, and yields rates of convergence similar to the application of the unaccelerated multigrid method applied to a positive definite system. Consequently, any multigrid method may be applied to difficult systems, assuming only a small number of those eigenmodes need to be filtered. In this sense, the GGB filter fits into the "self-correcting" multigrid methodology presented in [5]. In terms of computational work, the method is the most efficient for problems that can reuse the information of the eigenspace, such as multiple right hand side problems. For nonlinear problems solved by Newton's method we modify the filter (MGGB) to selectively reuse the same prolongation and thus reduce the amount of eigensolver calculations.

2 Overview of the GGB multiscale filter

Consider an $N \times N$ linear algebraic system of equations $Ku = f$ with the solution $u \in \mathbb{R}^N$, and the right hand side vector $f \in \mathbb{R}^N$. The system matrix K is considered to be generally nonsymmetric indefinite. A generic two-level multigrid V-cycle is illustrated in Figure 1. We introduce the following notation: the prolongation operator from the coarse grid to the fine grid $Q : \mathbb{R}^m \to \mathbb{R}^N$, where the coarse model is $m \times m$ and $m < N$. The restriction operator is taken to be transpose of the prolongation operator i.e. $Q^T : \mathbb{R}^N \to \mathbb{R}^m$. The coarse grid operator is defined by $K_0 = Q^T K Q$, and the smoothing or relaxation procedure is given by $M \in \mathbb{R}^{N \times N}$. The subscript zero is used to denote variables on the coarse space. If the error after iteration i is $e^i = u - u^i$, then reduction of the error after one V-cycle is controlled by the multigrid iteration matrix, given as

$$e^{i+1} = S^{\nu_2} T S^{\nu_1} e^i = R_{\mathcal{MG}} e^i, \tag{1}$$

where ν_1 and ν_2 are the number of pre and post smoothing. $S \in \mathbb{R}^{N \times N}$ is the smoothing iteration matrix written as

$$S = I - M^{-1} K, \tag{2}$$

and $T \in \mathbb{R}^{N \times N}$ is the coarse grid correction given by

$$T = I - Q(Q^T K Q)^{-1} Q^T K \tag{3}$$

Step 1: Pre-smoothing

$$u^{i+1} \leftarrow u^i + M^{-1}(f - Ku^i) \qquad \text{do } \nu_1$$

Step 2: Coarse grid correction

a. Residual restriction:

$$r_0^i = Q^T(f - Ku^i)$$

b. Coarse grid solution:

$$u_0^i = K_0^{-1} r_0^i \quad \text{where} \quad K_0 \leftarrow Q^T K Q$$

b. Correction:

$$u^{i+1} \leftarrow u^i + Q u_0^i$$

Step 3: Post-smoothing

$$u^{i+1} \leftarrow u^i + M^{-1}(f - Ku^i) \qquad \text{do } \nu_2$$

Fig. 1. Generic two-level multigrid V-cycle

where I is the $N \times N$ identity matrix. T is a projector satisfying $T = T^2$ with a spectral radius of $\rho(T) = 1$. The two-level cycle is guaranteed to converge iff the spectral radius of the two-level iteration matrix is inside the unit circle, $\rho(R) < 1$.

Multigrid methods consist of two major elements: smoothing and coarse grid correction. When symmetric positive definite (SPD) systems are considered, classical iterative methods, used as smoothers, eliminate the oscillatory components of the error leaving the smooth components almost untouched. This motivates the use of a coarse grid correction, where smooth components of the error are effectively approximated on a coarser grid. However, for difficult systems such as indefinite systems, smoothing may leave some oscillatory modes untouched, and thus standard multigrid methods might magnify these modes rather than reducing them [1].

The GGB multiscale method accelerates (stabilizes) the entire multigrid procedure in the following way. It first identifies all the troublesome modes of the applied multigrid method by solving for the largest magnitude eigenvalues λ_i of the multigrid iteration matrix in equation (1)

$$R_{\mathcal{MG}}\phi_i = \lambda_i \phi_i \qquad i = 1, ..., N. \tag{4}$$

The troublesome modes are the highest eigenvalues that are either not converging (indefinite) or "slow-to-converge" modes. The non converging eigenvalues are those that lie outside the unit circle $|\lambda_i| > 1$, and the "slow-to-converge" are the ones that lie inside the unit circle, however very close to one, i.e. $1 - \delta < |\lambda_i| < 1$, for some small positive constant δ. Next, based on the computed eigenvalues, say k eigenvalues, the GGB method constructs an additional coarse grid correction, with the prolongation operator spanned by the corresponding eigenvectors $Q_f = span\,\{\phi_i\}_{i=1}^k$. The additional coarse grid is used as a multiscale filter, eliminating those troublesome modes. As such, this method belongs to the class of "self correcting" multigrid methods, which find the algebraically smooth error components unresolved by multigrid [5]. How-

ever, as opposed [5] the algebraically smooth error components are obtained directly from the eigenvalue problem (4). Figure 2 schematically illustrates the architecture of the method used in the paper. Black circles denote local smoothing at each level, and GMRES/QMR is an outer accelerator. The GGB cycle is used to precondition Krylov methods. The overall error reduction of a

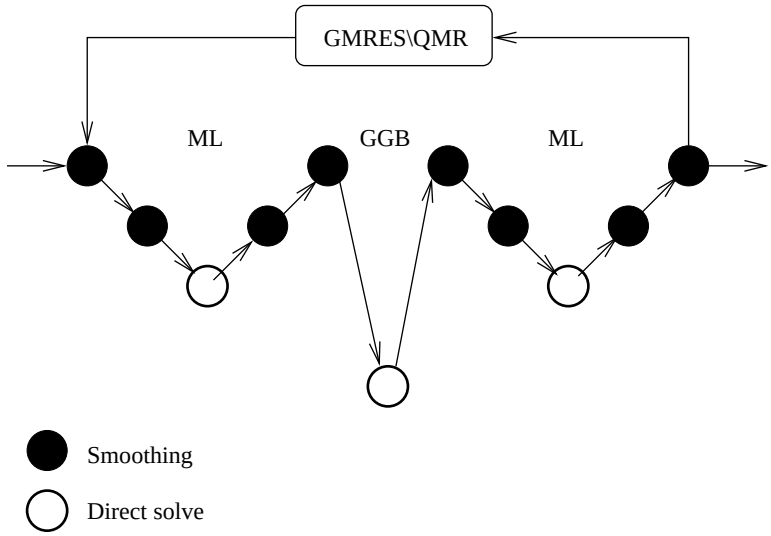


Fig. 2. Generalized Global Basis (GGB) cycle

single GGB cycle illustrated in Figure 2, without an external accelerator and one smoothing iteration at each level can be written as

$$e^{i+1} = (STS)^{\nu_2} F_{GGB} (STS)^{\nu_1} e^i = R_{\mathcal{MG}}^{\nu_2} F_{GGB} R_{\mathcal{MG}}^{\nu_1} e^i, \tag{5}$$

where STS is the multigrid iteration matrix and ν_1, ν_2 correspond to the number of V-cycles. F_{GGB} is the additional projector (filter), given as

$$F_{GGB} = I - Q_f (Q_f^* K Q_f)^{-1} Q_f^* K. \tag{6}$$

In the next section we analyze the behavior of the GGB filter. We show that the filter eliminates all unstable modes leading to convergence rates that is similar to the application of the unaccelerated multigrid method applied to the definite system.

3 Analysis of the GGB filter

Consider a one-dimensional, Helmholtz model problem $-u'' - k^2 u = f$ on the interval $\Omega = [0, 1]$ with homogeneous Dirichlet boundary conditions $u(0) = u(1) = 0$. The domain is divided into $N + 1$ segments with equal length $h = \frac{1}{N+1}$. A central difference discretization is employed leading to a $N \times N$ linear system. The problem is indefinite for $k^2 > \pi^2$. The eigenvalues and

eigenvectors of the system matrix K are respectively

$$\lambda_j = \frac{2(1-\cos j\pi h)}{h^2} - k^2 = \frac{4}{h^2}s_j^2 - k^2 \qquad j = 1, \ldots, N$$

and $v_j = \sqrt{2h}\,[\sin ij\pi h]_{i=1}^N$, were $s_j = \sin\frac{j\pi h}{2}$. We define the matrix V as the span of all eigenvectors $V = span\,\{v_i\}_{i=1}^N$. Applying a damped Jacobi relaxation to the linear system is equivalent to the following iteration on the error $e^{i+1} = (I - \omega D^{-1}K)e^i = Se^i$, where $D = \left(\frac{2}{h^2} - k^2\right) I$ is the diagonal of K and ω is a damping factor. The smoothing iteration matrix S shares the same eigenvectors as K. Specifically,

$$Sv_j = \quad 1 - \frac{\omega}{\frac{2}{h^2} - k^2}\lambda_j\Bigg) v_j = \mu_j v_j,$$

where $\mu_j = 1 - \omega\left(\frac{4s_j^2-(kh)^2}{2-(kh)^2}\right)$. We denote $\Lambda_{a,b}$ as a diagonal matrix of size $(b-a+1)\times(b-a+1)$, with all sorted eigenvalues μ_j such that $a \le j \le b$. It is apparent that for indefinite cases some eigenvalues of S are outside the unit circle, i.e. $|\mu_j| > 1$ and thus applying a standard multigrid method to this system will magnify the error of the corresponding eigenmodes rather than reducing it [1].

To overcome this difficulty, we apply on top of a standard multigrid method the GGB filter. We show that the multigrid method enhanced by GGB is convergent and positive definite with rate of convergence governed by spectral characteristics of the convergent modes (of the unfiltered operator). Superscript zero denotes the coarse grid variables. Linear interpolation Q_{mg} is used as prolongation and Q_{mg}^T is used as the restriction operator. It is easily verified that the prolongation operator applied to a coarse grid eigenvector yields a linear combination of one smooth and one oscillatory mode of the fine grid: $Q_{mg}v_j^0 = c_j^2 v_j - s_j^2 v_{N+1-j}$ where $c_j = \cos\frac{j\pi h}{2}$. It is shown in [3] that the multigrid iteration matrix $R_{\mathcal{MG}} = STS$ satisfies,

$$STS\,[v_j, v_{N+1-j}] = [v_j, v_{N+1-j}]\begin{bmatrix} \mu_j^2\left(1 - c_j^4\frac{\lambda_j}{\lambda_j^0}\right) & s_j^2c_j^2\frac{\lambda_{N+1-j}}{\lambda_j^0}\mu_j^2\mu_{N+1-j} \\ s_j^2c_j^2\frac{\lambda_j}{\lambda_j^0}\mu_j^2\mu_{N+1-j} & \mu_{N+1-j}^2\left(1 - s_j^4\frac{\lambda_{N+1-j}}{\lambda_j^0}\right) \end{bmatrix}.$$

The above implies that the Fourier transform can be used to reduce the multigrid iteration operator to a block diagonal matrix, with 2×2 blocks. The eigenvectors of STS, ϕ_j, are a linear combination of $[v_j, v_{N+1-j}]$.

We now examine how GGB enhances the multigrid scheme eliminating the indefinte modes. We define the prolongation operator for the GGB scheme $Q_f = span\,\{\phi_i\}_{i=N-k+1}^N$. The operator is spanned by all modes corresponding to eigenvalues $|\theta_j| > 1$, say k eigenvalues. The eigenvalue problem for STS can be written as $(STS)\,\Phi = \Phi\Theta$, where Φ is the complete set of eigenvectors and Θ is the eigenvalue matrix. We partition Φ into three subpaces: the null

space Φ_0, the convergent space Φ_c and the indefinite space Q_f we want to filter out. Thus the eigen problem for STS can be written as

$$(STS)[\Phi_0, \Phi_c, Q_f] = [\Phi_0, \Phi_c, Q_f] \begin{bmatrix} 0_{n\times n} & & \\ & \Theta_{n+1,N-k} & \\ & & \Theta_{N-k+1,N} \end{bmatrix}. \tag{7}$$

The eigenvalues of the GGB iteration matrix in equation (4) can now be analytically resolved. We write the error reduction of the GGB cycle given in (5) as the following eigenvalue problem: $(STS)F_{GGB}(STS)\Phi = \Phi\Sigma$ with the goal to obtain Σ. Using a similarity transform, that is possible due to the orthogonality properties of the subspaces, and substituting the terms for STS we get the following result

$$\Sigma = \Theta\Phi^T F_{GGB}\Phi\Theta = \Theta\,[I, I, 0]\,\Theta = \begin{bmatrix} 0_{n\times n} & & \\ & \Theta^2_{n+1,N-k} & \\ & & 0_{k\times k} \end{bmatrix}. \tag{8}$$

This completes the proof (for more details see [3]). It implies that the indefinite subspace was eliminated and that the rate of convergence is governed by $c = \rho(\Sigma)$ or the highest eigenvalue within the convergent space $\mathcal{C}$, which is given by

$$\theta_k^2 = \left[\mu_{k+1}^2 \quad 1 - c_{k+1}^4 \frac{\lambda_{k+1}}{\lambda_{k+1}^0}\right) + \mu_{N-k}^2 \quad 1 - s_{N-k}^4 \frac{\lambda_{N-k}}{\lambda_{k+1}^0}\Bigg)\Bigg]^2.$$

4 Acceleration of GGB applied to nonlinear problems

The GGB filter is useful for solving very difficult problems that require one linear solve or a sequence of linear solves. For the latter, the method is most attractive for problems with multiple right hand sides such as linear transient problems or shift-and-invert eigenvalue problems, since the indefinite and "slow-to-converge" eigenspace given in (4) has to be computed only once at the setup phase and can later be reused throughout the entire sequence of linear solves. However, if the method is applied to a sequence of linear solves, for instance, those that arise from nonlinear problems solved by Newton's method, then the left hand side (the Jacobian matrix) as well as the right hand side changes from one Newton iteration to the other. This results in an eigen computation for each linear solve in the sequence which may dominate the entire computational cost. However, if the sequence of Jacobians are slowly changing, then the eigenspace information can be reused without significantly affecting the convergence rate of the GGB method. Thus we modify the method (now called MGGB) to selectively reuse the same prolongation.

That is, the same GGB filter is applied to several nonlinear iterations. In practice, the total linear solve iterations will sometimes increase but overall CPU time can be significantly reduced. On the other hand, automatic reuse can be unsafe since the indefinite subspace might not be eliminated.
To evaluate whether a new filter has to be computed or not, we measure how far are the eigenvectors $\{q_i^j\}$ computed at nonlinear iteration j from being eigenvectors of $R_{\mathcal{MG}}^{j+1}$ (see equation (4)). We start by writing a Rayleigh quotient type measure $\frac{\|R_{\mathcal{MG}}^{j+1}q_i^j-\rho_i q_i^j\|_2}{|\rho_i|} \leq \delta$ and $i = 1, ..., k$ where k is the number of non-converging and "slow-to-converge" eigenvalues and ρ_i is the Rayleigh quotient defined as $\rho_i = \frac{(q_i^j)^* R_{\mathcal{MG}}^{j+1} q_i^j}{(q_i^j)^* q_i^j} = (q_i^j)^* R_{\mathcal{MG}}^{j+1} q_i^j$, and δ is some small constant. Moreover, we relax this measure to an angle β between the vector $u = R_{\mathcal{MG}}^{j+1} q_i^j$ and the subspace $Q^j = \{q_1^j, q_2^j, ..., q_k^j\}$ (as opposed to a single vector). The analysis show that

$$\cos^2 \beta_i \geq \frac{1}{\frac{\|R_{\mathcal{MG}}^{j+1}q_i-\rho_i q_i\|_2}{|\rho_i|}+1}.$$

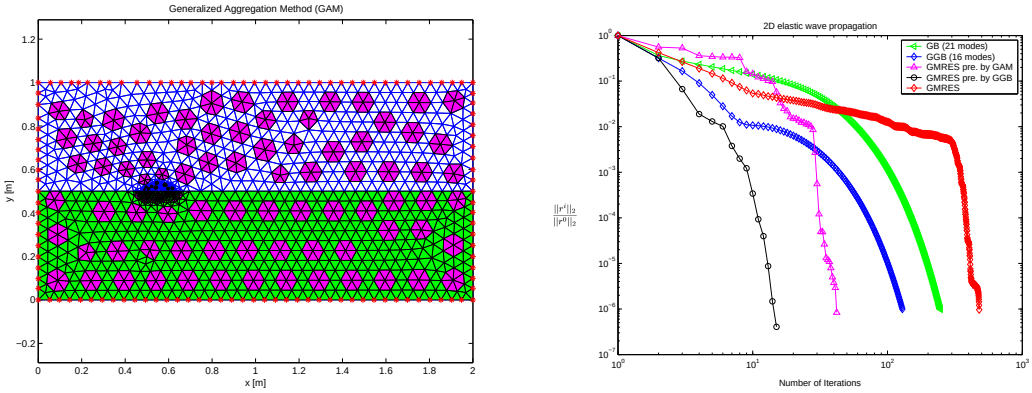
In practice, a new filter is then recomputed if $\max\{\beta_i\} \geq \beta_{cr}$ (more details are given in [6]).

5 Numerical results

We apply the GGB scheme to a 2D elastic wave propagation problem, and a 3D nonlinear thermal-convection flow application. We employ the Generalized Aggregation Multilevel (GAM) [7] method to the wave propagation problem and the smoothed aggregation multilevel method from the ML package [8] to the flow problem. All eigen computations used by GGB are computed by the Implicit Restarted Arnoldi method implemented in ARPACK.

5.1 2D wave propagation problem preconditioned by GAM

We study performance of various solvers for a 2D elastic wave propagation problem in a cracked membrane with softening zones at the tips of the crack. A standard Galerkin finite element semi-discretization in space produces a system of ordinary differential equations of the form $M\ddot{u} + Ku = f$ for the transverse displacement u. The membrane is constrained all around and a point force is applied at a single node. The crack tips are damaged and thus the problem become indefinite. An Implicit Newmark predictor-corrector method is used for the time integration (average acceleration). The multigrid method used as preconditioner is GAM [7], which is similar in spirit to AMGe. GAM constructs the prolongation operator based on lowest modes of aggregate stiffness matrices. Figure 3(a) shows the formation of such aggregates. The convergence of the various solvers for the first time step are illustrated in Figure 3(b). We use GMRES without restarting. The iterations count increase linearly over the time integration as only the right hand side varies. The fastest method



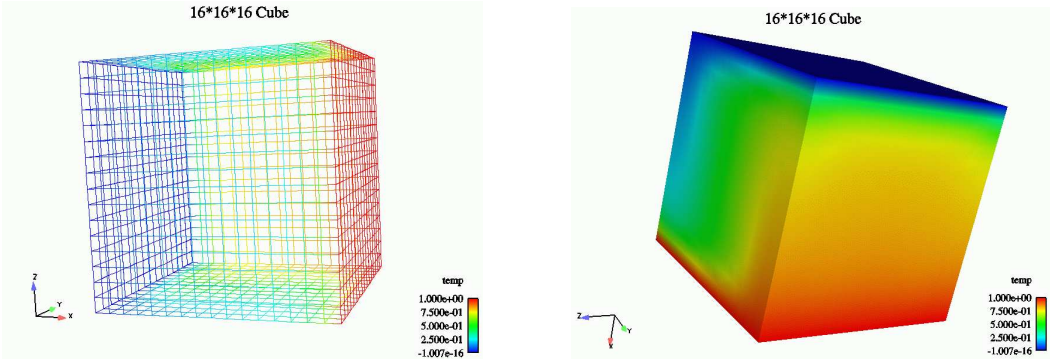
(a) Formation of large aggregates (b) convergence of the solvers

Fig. 3. 2D elastic wave propagation problem

to converge is GMRES preconditioned by GGB. The eigenvector calculation time is negligible (for this problem) as it is only performed in the first step and then reused throughout the time integration. This is in contrast to GMRES which builds information during the computation of a linear solve but then normally throws this information away for subsequent linear solve. Restarting the GMRES iteration makes it even harder and the computational cost increases tremendously, where sometimes convergence is not guarantied. This example illustrates the main advantage of the GGB filter when the matrix is constant but the right hand side varies. GGB computes eigenspace information that it is able to reuse over many linear solves.

5.2 Thermal-convection flow preconditioned by Smooth Aggregation

In this section we demonstrate the performance of GGB and MGGB applied to steady, 3D thermal-convection flow in a cube. The governing PDEs are the Navier-Stokes with thermal energy equations. The equations are approximated by a Galerkin Least Squares formulation. The resulting nonlinear system of equations gives rise to a system of coupled, nonlinear and nonsymmetric equations. We employ MPSalsa [9] to generate the system of equations, and a smooth aggregation multilevel method implemented in ML package [8] as preconditioner to GMRES. Due to nonsymmetry of the problem our experience shows that best performance is obtained when piecewise constants are used as grid interpolants (unsmoothed aggregation). A no-slip condition is enforced on all surfaces. A hot temperature is set to one face of the cube and a cold temperature is set to the parallel. $ilu(0)$ is applied as a smoother to the aggregation method at each level excluding the coarse one. We consider a cube consisting of $32 \times 32 \times 32$ mesh elements that yield systems with $179,685$ number of unknowns. A 4-level aggregation method is applied. We use GMRES(30) as an outer accelerator. We also apply a modified GGB scheme (MGGB) where the eigenspace is not always recomputed for each linear solve of Newton's method (see section 4). The mesh and temperature distribution



(a) mesh (b) Temperature solution

Fig. 4. 3D thermal convection flow in a cube

(for $16 \times 16 \times 16$ cube) are illustrated in Figure 4. Table 1 presents the convergence results. It is shown that GGB outperformed all the other methods in terms of iteration count. However, the best CPU time is obtained by MGGB due to the inexpensive cost of the eigensolutions compared to GGB as the projection is computed selectively. To this end it is important to mention that GGB plays an important complementary role to GMRES. Restarting GMRES is crucial for convergence. However, once GMRES is restarted, all the prior information is lost, causing temporal stall until it reconstructs a sufficient space for further reduction of the error. When GGB is used, the information about the indefinite modes which span the most difficult space is stored (the space that is slow to converge), which enables GMRES to recover rapidly and to converge faster.

Table 1
CPU time and iteration summary for 3D flow in a $32 \times 32 \times 32$ cube.

	GMRES(30) pre. by ML	GMRES(30) pre. by GB	GMRES(30) pre. by GGB	GMRES(30) pre. by MGGB
Total CPU time [sec]	6998.40	9191.40	7183.63	5817.99
Total Eigen time [sec]	-	2306.10	2033.50	206.9
Eigen work [%]	-	25%	28.31%	3.56%
Total linear solves iter.	1521	1849	501	616

6 Conclusions

We introduce and study the Generalized Global Basis (GGB) multiscale filter. We show that GGB is robust for variety of problems considered and can be used to accelerate any multigrid method. If sufficient number of modes is taken, convergence is ensured for indefinite and nonsymmetric systems. The efficiency of the method hinges on the highest eigenmodes computations. Thus,

it is most attractive for problems with multiple right hand sides as the indefinite eigenspace is computed only once and reused. The method also enhances restarted GMRES strategies by providing significant information. For nonlinear problems we propose a modified (MGGB) method that selectively reuse the same prolongation. This selection criteria is based on the angle between a subspace and a vector. Numerical examples clearly show that MGGB preconditioned by GMRES performs the best of all the methods considered and provides significant time savings.

References

[1] A. Brandt and S. Ta'asan. Multigrid methods for nearly singular and slightly indefinite problems. In W. Hackbusch and U. Trottenberg, editors, *Lecture Notes in Mathematics 1228:Multigrid Methods II*, pages 100–122. Springer-Verlag, 1985.

[2] C. W. Oosterlee and T. Washio. An evaluation of parallel multigrid as a solver and as a preconditioner for singularly perturbated problems. *SIAM Journal of Scientific Computing*, 19(1):87–110, 1998.

[3] H. Waisman, J. Fish, R. S. Tuminaro, and J. Shadid. The Generalized Global Basis (GGB) method. *International Journal for Numerical Methods in Engineering*, 61(8):1243–1269, 2004.

[4] J. Fish and Y. Qu. Global-basis two-level method for indefinite systems. part 1:convergence studies. *International Journal for Numerical Methods in Engineering*, 49(3):439–460, 2000.

[5] M. Brezina, R. Falgout, S. Maclachlan, T. Manteuffel, S. Mccormick, and J. Ruge. Adaptive smooth aggregation (αSA). *SIAM Journal on Scientific Computing*, 25(6):1896–1920, 2004.

[6] H. Waisman, J. Fish, R. S. Tuminaro, and J. Shadid. Acceleration of the Generalized Global Basis (GGB) method for nonlinear problems. *Submitted to Journal of Computational Physics*.

[7] J. Fish and V. Belsky. Generalized Aggregation Multilevel solver. *International Journal for Numerical Methods in Engineering*, 40(23):4341–4361, 1997.

[8] C. Tong and R. S. Tuminaro. ML 2.0 smoothed aggregation user's guide. Technical Report SAND2001-8028, Sandia National Laboratories, 2000.

[9] J. Shadid, H. K. Moffat, S. A. Hutchinson, G. L. Hennigan, K. D. Devine, and A. G. Salinger. MPSalsa-A finite element computer program for reacting flow problems. Part 1-Theoretical development. Technical Report SAND95-2752, Sandia National Laboratories, May 1996.

A Cache-aware Multigrid Navier-Stokes Solver

Tobias Weinzierl, Miriam Mehl, Christoph Zenger
Institut für Informatik, TU München,
Boltzmannstraße 3, 85748 Garching, Germany
weinzier@in.tum.de

January 12, 2005

Abstract

Sophisticated numerical algorithms like multigrid techniques on adaptive grids require hierarchical multilevel data, which are commonly organized in tree structures. The resulting non–locality of interconnections between data causes jumps in the memory space during the run of the algorithm. This makes an efficient usage of cache–hierarchies and prefetching algorithms impossible. In this paper, we present a Navier–Stokes solver using an additive multigrid algorithm for the pressure equation resulting from Chorin's projection method. We achieve a very high cache–efficiency by the construction of particular linear data structures based on the concepts of space–filling curves and space–trees. In fact, we can avoid memory jumps completely and, in addition, we get along with only two bits per degree of freedom for administrational information — one for the geometry description and one for the description of the refinement structure. Last but not least new velocity elements are introduced that fulfill the side condition (incompressiblity) pointwise. Thus, the algorithm becomes energy and momentum preserving in terms of spatial discretization.

1 Introduction

The aim of this paper is to develop a stable and — both in terms of numerical methods and hardware usage — efficient algorithm to solve the Navier-Stokes equations

$$\partial_t u + (u\nabla)\, u - \frac{1}{Re}\Delta u + \nabla p = 0, \tag{1}$$

$$\mathbf{div}\ u = 0 \tag{2}$$

for incompressible flows in two dimensions. To achieve this, we use a special Finite–Element discretization of the momentum equation enabling us to satisfy the continuity equation pointwise and to preserve energy and momentum. We combine it with a sophisticated algorithm and data structure allowing a very efficient implementation of multigrid methods on adaptive grids.

In more detail, we use Chorins projection method [4] combined with an explicit Euler step for simplicity. The explicit Euler step can easily be replaced by more accurate explicit time steps [5]. This approach leads to a sequence of decoupled PDEs where one is able to calculate the pressure explicitly at any time step.

Our new continuity preserving shape functions for the Finite–Element discretization of the spatial derivatives of the momentum equation enable us to fulfill the continuity equation pointwise. Together with the duality of the discrete gradient and divergence operators this leads to a stable, both energy and momentum preserving system in terms of spatial discretization. Because our discrete solution fulfills the continuous continuity equation exactly, convergence theory is much simpler and does not use the theory of mixed finite element. One can easily observe that a discrete solution in the vicinity of the exact solution exists. Applying Ceas lemma one gets error bounds for the discrete Galerkin solution.

To solve the Poisson equation for the pressure in each time step, we use a multigrid scheme implemented in a very efficient manner in terms of storage usage and access. Hereby, efficiency is achieved by using linear data structures with locally deterministic access. Thus, in contrast to many other approaches, we end up with an algorithm which is highly efficient both in terms of numerics and in terms of hardware usage.

Section 2 shortly describes the essentials of the grid used, the location of data within the grid and the traversion order of the grid cells. In section 3, we define our continuity preserving Finite–Element basis functions. The discretization of the pressure equation and the ingredients of the multigrid algorithm are described in 4. Finally, we give some results for the driven–cavity scenario.

2 Space–Trees, Cell–oriented Operator Evaluation and a Cache–aware Hierarchical Domain Traversion

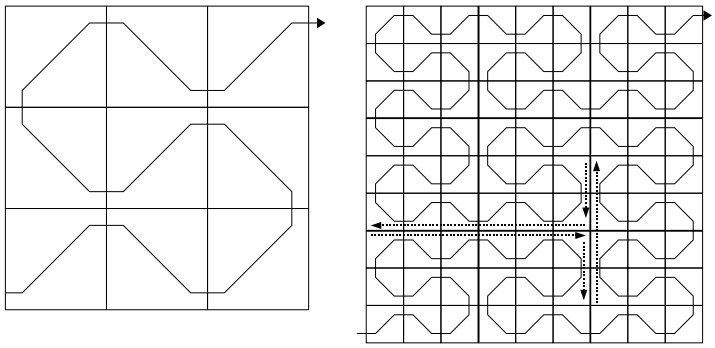


Figure 1: The Peano space filling curve used for the traversion of the spacetree grid. The right picture indicates the processing order of some selected vertices.

The algorithmic concept of this paper is based on the interplay of space–filling curves and space–tree discretizations of a quadratic domain [6, 17]. Thus, the computational domain is embedded into the unit square and the grid is created by recursive decomposition of each cell (square) into nine subcells. The refinement rules depend on the geometry and numerical requirements. This allows the construction of very flexible adaptive grids without loosing the structuredness of the grid.

During our computation, we process the grid cell–wise and perform a cell–wise operator evaluation [3]. Hereby, the velocity degrees of freedom are located on the cells vertices and the cells edges (see figure 2). Thus, we have to establish an algorithm that accesses the vertex and edge data of the cells from suitable data structures in a very efficient way. The first ingredient of such an algorithm is the definition of a traversion order of the cells:

Using a structured grid as defined above, one is able to traverse the grid along the corresponding approximative polygon of the Peano curve [17] — a space–filling curve. The recursive definition of the space–filling curve induces an unique ordering of grid cells for adaptive space–tree grids in a top–down–depth–first manner. Figure 1 shows an example for a regular grid with the fitting Peano polygon.

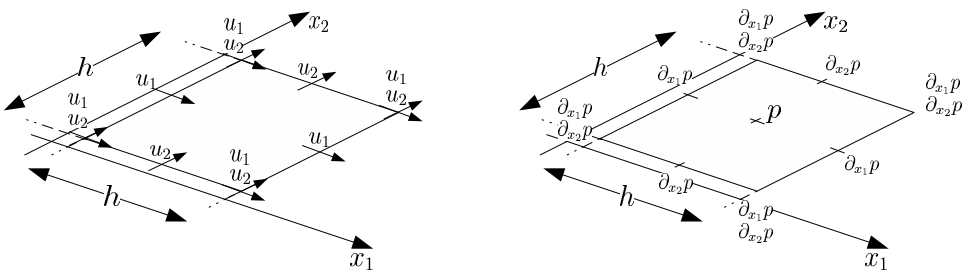


Figure 2: Arrangement of the degrees of freedom per grid cell for both velocity, pressure and pressure derivatives.

If we now examine the order vertex data are used during our traversion, we observe a property which will be fundamental for the construction of our data structures: Let $\sqsubseteq_1$ be the order the vertices are passed by the curve for the first time. $\sqsubseteq_2$ is the order the vertices are passed for the second time. Then $x_1 \sqsubseteq_1 x_2 \Leftrightarrow x_1 \sqsupseteq_2 x_2$ holds. Thus, we can store all data on stacks which mirror the resulting requirements: put data on top of a stack after using it for the first time, pop data of the stack when it is needed for the second time. On a regular grid, one is able to split up all the vertices of the grid into two classes — half of the vertices are on the left hand side, the others are on the right hand side of the Peano curve. If we handle nodal data only, one can easily see that in this case it is sufficient to use two stacks representing this two classes of points. Stacks are cache–efficient due to the inherent local memory access. Furthermore, we can show that if we use stacks, the storage of further administrational information like pointers to neighbors and/or subsequent points is obsolete.

In [10, 16, 13], it was shown that this principle can be generalized to adaptive grids with hierarchical data (generating systems) [9] suited for sophisticated multigrid algorithms. The number of stacks is small and independent of the mesh refinement depth. [12] has developed a dimension–generic algorithm. Furthermore [14] shows some applications of this concept adding parallel programming, adaptivity and extrapolation techniques. [10, 14] have proven that such an algorithm ends up with only 10 percent more cache–misses than absolutely inevitable in a code which does not make explicit use of a prefetching strategy. [16] and [12] have shown that this holds for arbitrary dimensions, too.

3 Shape Functions Preserving Continuity

Solving both the momentum (1) and the continuity equation (2) in a weak formulation, the FEM minimizing problem requires an extended theory for mixed elements [3, 8]. In contrast to this, we eliminate all complications induced by the side condition by solving it exactly (pointwise). For this, we approximate the velocity space in a very special way:

Let $\mathcal{T}$ be a domain discretization, where every geometric element $V \in \mathcal{T}$ is a square. Then, there is a collocated ansatzspace $\mathbb{U}$ such that

$$Mu_h := \int_V \mathbf{div}\ u_h\ dx = 0 \quad \forall V \in \mathcal{T} \quad \Leftrightarrow \quad \mathbf{div}\ u_h = 0 \quad \textit{almost everywhere} \tag{3}$$

holds for all $u_h \in \mathbb{U}$.

As shown in [8], this is the precondition for both energy and momentum conservation with respect to space discretizations[1]. Such an ansatzspace with square support for the velocity was presented in [2] and extended in [20]. It was also used in a more complicated setting in [5]. Figure 3 shows the two components of one basis function. The idea is to subdivide the support into triangles and couple the two velocity components. On each triangle, the shape function is linear. The "hat" ϕ connecting each velocity component with its own coefficients has the value 1 at the associated grid point, $\frac{1}{4}$ at the midpoints of the adjacent cells and 0 at all other grid points. The coupling function φ connecting each velocity component with the coefficients of the other component has the values $\pm\frac{1}{4}$ at the midpoints of the adjacent cells and 0 at all other grid points. Thus we get the basis representation

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \sum_i u_{1,i} \begin{pmatrix} \phi_i \\ \varphi_i \end{pmatrix} + u_{2,i} \begin{pmatrix} \varphi_i \\ \phi_i \end{pmatrix}.$$

To avoid a decomposition of cells in a checkerboard pattern, we had to introduce additional degrees of freedom at the cells edges (see figure 2). The corresponding additional basis functions are generated by scaling and translation of the ones described above. To describe the complete ansatzspace is a very technical task exceeding the extent of this paper. For more details see [20].

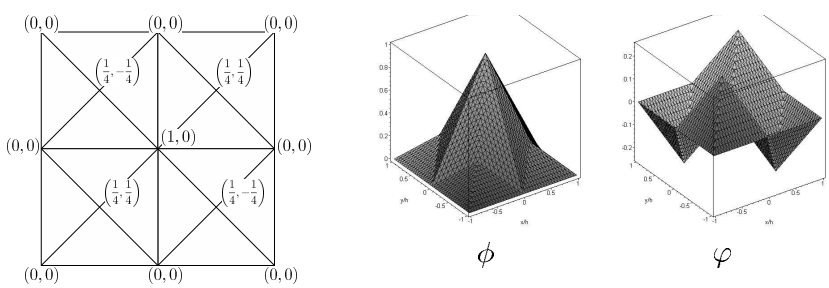


Figure 3: The original velocity shape function presented in [2].

We note here in addition that the nonstaggered arrangement of the velocity components is sometimes critizized because it may produce the so called checkerboard

[1] The development of an optimal time discretization is not content of this work. Thus, conservation of energy and momentum holds only for $\tau \mapsto 0$, but for any grid width!

instability. But this happens only if the Dirichlet boundary conditions are not introduced in a consistent way. If the boundary conditions are chosen appropriately, this instability is avoided easily.

4 The Pressure Equation

According to Chorins projection method with an explicit Euler step, we use the pressure p to achieve a solenoidal solution at time $t+\tau$. Using a FEM approach, equation (1) becomes

$$\Omega u_h^{(t+\tau)} = \underbrace{\Omega u_h^{(t)} + \tau \left(\frac{1}{Re} D u_h^{(t)} - C(u_h^{(t)}) u_h^{(t)} \right)}_{=:b} - \tau \Omega \nabla_h p. \tag{4}$$

Here, Ω denotes the mass matrix, D and $C(u)$ are the discrete diffusion and the convection operator and ∇_h is the discrete gradient. Considering the equivalence given by (3) and the duality of divergence and gradient ($\Omega\nabla = M^T$), the continuity equation (2) becomes

$$\mathbf{div}\ u^{(t+\tau)} = 0 \quad \Leftrightarrow \quad M\nabla_h p = \frac{1}{\tau} M\Omega^{-1}\ b \tag{5}$$

$$\Leftrightarrow \quad M\Omega^{-1}M^T p = \frac{1}{\tau} M\Omega^{-1} b, \tag{6}$$

resulting in a staggered pressure grid[2]. Such a pressure layout is energy preserving and stable with respect to checkerboard patterns [18, 19]. In (6) we used the duality of divergence and gradient operator, which is, in fact, again a precondition for energy and momentum conservation [19]. The entries of the discrete divergence operator M can easily be calculated using the Gauss integral theorem and are given in figure 4.

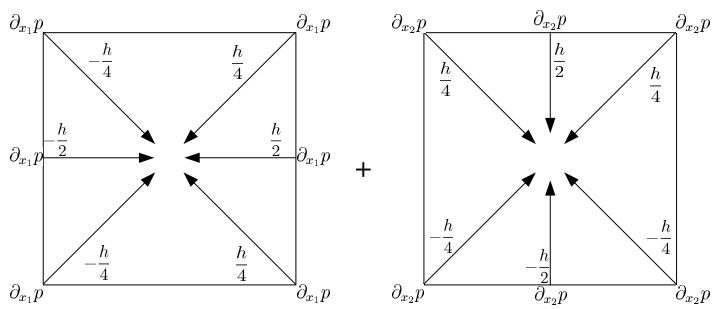


Figure 4: Stencil $M|_V$ on a micro cell V divided up into x_1 and x_2 contributions. Matrix M is composed by all $M|_V$.

.

[2]We interprete the pressure being a staggered spline function space.

4.1 The Fine Grid Relaxation Scheme

To implement a relaxation method for the pressure, we have to integrate both the storage scheme of the pressure (cell–wise) and the evaluation of the discrete operator $M\Omega^{-1}M^T$ into our algorithmic concept based on stacks and cell–wise operator evaluation. Since the pressure values are visited in a strict linear order, it is sufficient to add one additional stack for the pressure. In contrast, the evaluation of the operator turns out to be a litte more complex for the operator offends the locality property, as for the direct evaluation of $M\Omega^{-1}M^T$ we would have the pressure values of the neighbouring cells.

To prevent this, it is a good idea to store the gradients of the pressure, too, which are located at the cell vertices and edges analog to the location of the velocity values. A relaxation smoother for this equation system can be formalized as

$$\begin{aligned} p^{(k+1)} &= p^{(k)} + \omega\ diag^{-1}\ \left(\frac{1}{\tau}M\Omega^{-1}b - M\Omega^{-1}M^T p^{(k)}\right) \\ &=: p^{(k)} + \omega\ diag^{-1}\ res, \end{aligned} \tag{7}$$

where $diag$ is the diagonal entry of $M\Omega^{-1}M^T$. Assume that we start with actual values of p and ∇p, and $\frac{1}{\tau}\Omega^{-1}b$ is given. Then, one iteration of our relaxation method can be decomposed into three steps:

- Evaluate $res_V = \frac{1}{\tau}M_V\Omega^{-1}b - M_V\nabla p^{(k)}$ (completed when leaving the cell),
- change pressure within cell $p^{(k+1)} = p^{(k)} + \omega\ diag^{-1}\ res_V$ (completed when leaving the cell) and finally
- update pressure gradients $\nabla p^{(k+1)} = \nabla p^{(k)} + \Omega_V^{-1}\ M_V^T\ \omega\ diag^{-1}\ res_V$. (Accumulation process over adjacent cells. $\nabla p^{(k+1)}$ completed when leaving the last adjacent cell.)

Here Ω_V is the lumped mass matrix. Since the pressure gradient is updated after each correction step immediately, this algorithm equals a Gauss-Seidel smoother and is therefore faster than [10, 12, 16]. The drawback compared to collocated Jacobi-smoothers is the number of variables to be stored per pressure degree of freedom.

Figure 6 shows the performance of a SOR Gauss–Seidel smoother for different relaxation factors (see also 5). For further information on boundary conditions and actual values for $\omega, diag^{-1}$, etc see [8, 15, 20].

4.2 The Coarse Grid Correction

The aim of the following chapter is to develop a Galerkin multigrid algorithm for the pressure Poisson equation. Since we are solving a time–dependent problem, the main focus is on reducing low frequency error components fast as they are changing more substantially in each time step than the high frequency components. This can be done using a multigrid approach. Due to our stack concept, the implementation of a multigrid algorithm does not cause any loss of hardware efficiency.

In this paper, all hierarchical values (coefficients of the generating system) and operators are denoted with a hat. The concept of hierarchical generating systems [9] fits the concept of space–trees and top–down–depth–first tree traversion in a very

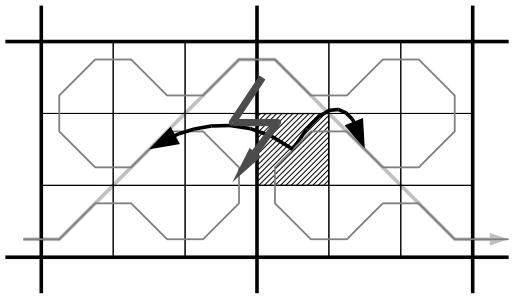


Figure 5: A residual on the micro cell marked would affect the macro cell residual on the left side. This cell is not "available" ascending the space–tree.

natural way. We introduced a generating system for the pressure, whereas the velocity is given on the fine grid only[3].

Since for the fine grid smoother the nodal pressure gradient has to be computed, let's assume the hierarchical gradient is given on every grid level. Descending the space–tree with the aim of computing the nodal, interpolated values of $\nabla\hat{p}$ on the fine grid, one is able to calculate

$$\Omega^{-1} M^T P_{cell} \hat{p} = P_{\nabla} \nabla \hat{p} \tag{8}$$

directly using a suitable gradient interpolation operator P_{∇}. Afterwards, we perform one fine grid relaxation step. So, instead of working on the generating system for the pressure the problem is transformed to a generating system for the pressure gradient.

Unfortunately, one is not able to restrict the pressure residual in one step ascending the spacetree, as every fine grid residual on a micro cell — except the one in the center — would affect the residuals of all neighbouring macro cells within the space–tree as the restriction which is the transposed of the bilinear interpolation offends the locality property of the space filling curve algorithm as illustrated in figure 5.

The description of the solution of this problem is a very technical task and exceeds the length constraints of this paper. The complete algorithm is described in [20]. The critical point is to update the hierarchical pressure gradient after each coarse grid smoothing step immediately. This is again a reason for working on the hierarchical pressure gradients instead of the pressure itself.

Finally, we get full multigrid performance. Because we are using the pressure of the preceding time step as starting value, the number of iterations becomes small very soon using exploiting the property of multigrid algorithms that smooth error components which are predominant in our situation are reduced rapidly.

5 Results and Conclusion

In this paper, we presented a Navier–Stokes solver based on Chorins projection method [4] combined with a new Finite–Element ansatzspace enabling us to solve the incompressiblity exactly [2]. A highly efficient storage usage and access was achieved by the construction of linear data structures with the help of space–filling curves [10, 16].

[3]Future work will concentrate on multilevel representation of the velocities

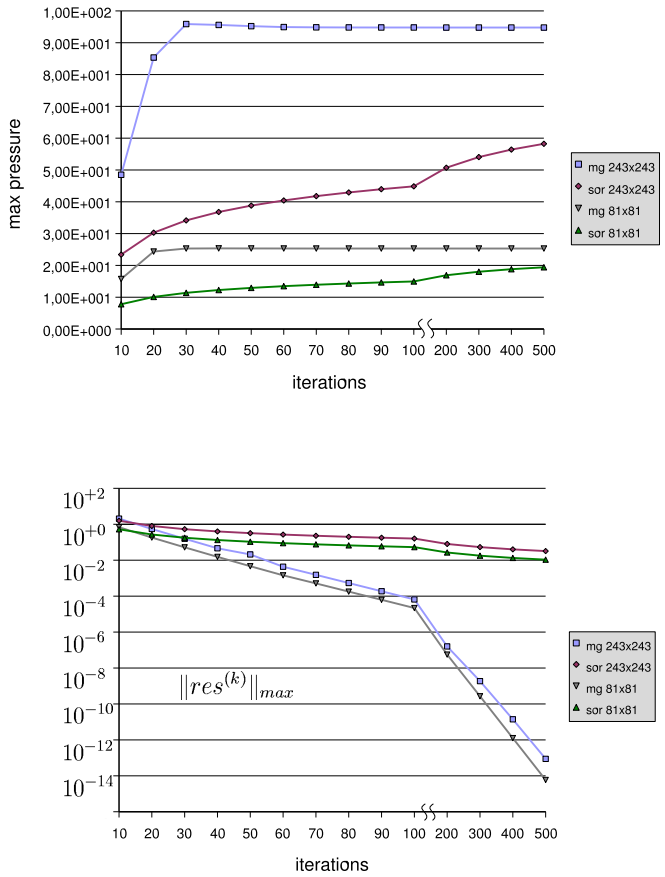


Figure 6: Some results for the driven–cavity scenario with $Re = 10$ comparing the multigrid algorithm with a Gauss–Seidel relaxation scheme on two different grids.

Furthermore, by modifying the storage scheme of the pressure (gradients instead of function values) we were able to implement a pressure solver with multigrid efficiency [3, 9, 18]. Figure 6 shows this fact. For all grid sizes, around 20 relaxation steps are needed to reduce the error by a factor of 10. This is much worse compared to multiplicative multigrid schemes but already good enough for our purpose. If we use the method e.g. in the scenario of the time dependant driven–cavity only very few relaxation steps in every time step are enough to reduce the error sufficiently. But it should not be difficult to improve the order of convergence substantially by a more sophisticated relaxation strategy. This will be subject of further investigations. The L2–cache hit–rate is almost minimal in accord with earlier results for simpler PDEs [10, 12, 16].

In our opinion there are five important facts to point out:

1. Since the side condition is fulfilled exactly, the standard FEM theory can be ap-

plied. Particularly with regard to the discretization of the momentum equation no extended theory for mixed/nonconforming elements is required.

2. Current computers suffer from a lack of L2–cache making memory access pretty expensive compared to register access. The sophisticated traversion algorithm used here is by construction cache–optimal with respect to hit–rates on all memory hierarchies, since it uses stacks for data and geometry management.

3. Although the algorithm is designed for adaptive hierarchical meshes, the geometry is stored sequentially using space filling curves. Therefore, one bit per grid point is adequate to store the complete geometry data. In addition a second bit describes whether the cell is refined or not. So, geometry description and refinement structure are stored in a very efficient way.

4. The algorithm provides full multigrid performance.

5. The energy and momentum conservation laws are satisfied as the side condition is fulfilled exactly and all properties of the continuous operators are carried over to the discrete operators.

References

[1] M. Bader, A.C. Frank, Ch. Zenger: *An Octree-Based Approach for Fast Elliptic Solvers*, published in: M. Breuer, F. Durst, Ch. Zenger (Eds.): High Performance Scientific and Engineering Computing, Springer-Verlag, Berlin Heidelberg, 2001

[2] C. Blanke: *Kontinuitätserhaltende Finite–Elemente–Diskretisierung der Navier–Stokes–Gleichungen*, Diploma thesis, Chair of Computer Science in Engineering, Science, and Numerical Programming, Department of Informatics, Technical University Munich, 2004

[3] D. Braess: *Finite Elemente*, Springer-Verlag, Berlin Heidelberg, 2002

[4] A. J. Chorin: *Numerical solution of the Navier-Stokes equations*, Math. Comp. , 22 (1968), pp. 745-762, 1968

[5] M. Emans: *Numerische Simulation des unterkhlten Blasensiedens in turbulenter Strmung: Ein Euler-Lagrange-Verfahren auf orthogonalen Gittern*, Dissertation, Chair of Computer Science in Engineering, Science, and Numerical Programming, Department of Informatics, Technical University Munich, 2003

[6] A. Frank: *Organisationsprinzipien zur Integration von geometrischer Modellierung, numerischer Simulation und Visualisierung*, Dissertation, Chair of Computer Science in Engineering, Science, and Numerical Programming, Department of Informatics, Technical University Munich, 2000

[7] P.M. Gresho, R.L. Sani: *On pressure boundary conditions for the incompressible Navier-Stokes equations*, International Journal for Numerical Methods in Fluids, Vol. 7, p. 1111-1145, 1987

[8] P.M. Gresho, R.L. Sani, M.S. Engelman: *Incompressible Flow and the Finite Element Method - Advection-Diffusion and Isothermal Laminar Flow*, John Wiley & Sons Ltd, Chichester, 1998

[9] M. Griebel: *Multilevel algorithms considered as iterative methods on indefinite systems*, In: Proceedings of the 2nd Copper Mountain Conference on Iterative Methods, University of Colorado at Denver, 1992

[10] F. Günther: *Eine cache–optimale Implementierung der Finite–Elemente–Methode*, Dissertation, Chair of Computer Science in Engineering, Science, and Numerical Programming, Department of Informatics, Technical University Munich, 2004

[11] F. Günther, M. Mehl, M. Pögl, Ch. Zenger: *A cache-aware algorithm for PDEs on hierarchical data structures based on space-filling curves*, SIAM Journal on Scientific Computing, in review

[12] J. Hartman: *Entwicklung eines cache–optimalen Finite–Element–Verfahrens zur Lösung d-dimensionaler Probleme* , Diploma thesis, Chair of Computer Science in Engineering, Science, and Numerical Programming, Department of Informatics, Technical University Munich, 2004

[13] A. Krahnke: *Adaptive Verfahren höherer Ordnung auf cache–optimalen Datenstrukturen für dreidimensionale Probleme*, Dissertation, Chair of Computer Science in Engineering, Science, and Numerical Programming, Department of Informatics, Technical University Munich, 2004

[14] M. Langlotz, M. Mehl, T. Weinzierl, Ch. Zenger: *Cache-Optimal Parallel Solution of PDEs on High Performance Computers Using Space-Trees and Space-Filling Curves*, in: Proceedings of the final KONWIHR conference — Competence Network for Technical, Scientific High Performance Computing in Bavaria (to appear), Springer-Verlag, Berlin Heidelberg, 2005

[15] T. Neckel: *Einfache 2D–Fluid–Struktur–Wechselwirkungen mit einer cache–optimalen Finite–Element–Methode*, Diploma thesis, Chair of Computer Science in Engineering, Science, and Numerical Programming, Department of Informatics, Technical University Munich, 2005

[16] M. Pögl: *Entwicklung eines cache-optimalen 3D Finite-Element-Verfahrens für große Probleme*, Dissertation, Chair of Computer Science in Engineering, Science, and Numerical Programming, Department of Informatics, Technical University Munich, 2004

[17] H. Sagan: *Space-Filling Curves*, Springer-Verlag, Berlin Heidelberg, 1994

[18] U. Trottenberg, C. Oosterlee, A. Schüller: *Multigrid*, Academic Press, London 2000

[19] R.W.C.P. Verstappen, A.E.P. Veldman: *Symmetry-Preserving Discretization of Turbulent Channel Flow*, published in: M. Breuer, F. Durst, Ch. Zenger (Eds.): High Performance Scientific and Engineering Computing, Springer-Verlag, Berlin Heidelberg, 2001

[20] T. Weinzierl: *Eine Cache-optimale Implementierung eines Navier–Stokes–Lösers unter besonderer Berücksichtigung physikalischer Erhaltungssätze*, Diploma thesis, Chair of Computer Science in Engineering, Science, and Numerical Programming, Department of Informatics, Technical University Munich, 2005