MULTILEVEL ACCELERATED OPTIMIZATION FOR PROBLEMS IN GRID GENERATION[3]

Markus Berndt[1] Mikhail Shashkov[2]

[1]*Los Alamos National Laboratory, Los Alamos, NM, U.S.A. berndt@lanl.gov*
[2]*Los Alamos National Laboratory, Los Alamos, NM, U.S.A. shashkov@lanl.gov*
[3]*LA-UR-03-3474*

ABSTRACT

The quality of numerical simulations of processes that are modeled by partial differential equations strongly depends on the quality of the mesh that is used for their discretization. This quality is affected, for example, by mesh smoothness, or discretization error. To improve the mesh, a functional that is in general nonlinear must be minimized (for example, the L^2 approximation error on the mesh). This minimization is constrained by the validity of the mesh, since no mesh folding is allowed. Classical techniques, such as nonlinear CG, or Gauss-Seidel steepest descent, perform very poorly on this class of minimization problems. We introduce a new minimization technique that utilizes the underlying geometry of the problem. By coarsening the mesh successively, in a multilevel-like fashion, minimizing appropriate coarse grid quality measures, and interpolating finer meshes from coarser ones, a more rapid movement of fine mesh points results, and the overall convergence of the minimization procedure is accelerated.

Keywords: mesh generation, optimization, multilevel methods

1. INTRODUCTION

Grid generation often requires the minimization of a functional that describes mesh smoothness or approximation quality of a specific function. Optimization methods that are commonly used in this context are of the steepest descent type [1, 2]. These methods perform very poorly, if vertices, or clusters of vertices have to move large distances from an initial grid to reach a final optimal configuration.

Our multilevel approach effectively achieves an accelerated vertex movement by coarsening the grid and then solving an appropriate optimization problem on the coarsened grid. The coarse grid is then interpolated to the fine grid. On a coarser grid, vertices can move larger distances than on fine grids, without the mesh becoming invalid. By interpolating an improved coarse grid to a finer grid we effectively move clusters of fine grid vertices by moving only a single coarse grid vertex. We apply the idea of coarsening a fine grid, approximately solving an appropriate coarse grid optimization problem, and interpolation the resulting grid back to the fine level recursively. This describes our multilevel accelerated optimization procedure.

In this paper, we first give an overview of classical multigrid methods to motivate the applicability of multilevel-type ideas in the context of optimization problems that arise in the context of grid generation. We then introduce our target optimization problem and construct the multilevel components coarsening and interpolation. After defining the coarse grid optimization problems we define the multilevel optimization procedure, give a complexity analysis and a numerical example to underline the performance gain that can be achieved with this type of approach to optimization.

2. OVERVIEW OF MULTILEVEL METHODS

Multilevel methods are the most efficient methods for solving linear systems that typically arise in the discretization of elliptic partial differential equations (PDEs). A good introduction to these methods is given in the Multigrid Tutorial [3]. For a more comprehensive overview, see [4]. Here, we give a short introduction of multilevel methods, to motivate their applicability to of some grid generation problems.

One class of methods for solving linear systems is called relaxation methods. As an example, we introduce Jacobi relaxation. A given linear system

$$Ax = b \tag{1}$$

can be re-written, using the diagonal D, the lower triangular part L, and the upper triangular part U of A in the following way,

$$Dx = b - Lx - Ux. \tag{2}$$

Equation (2) gives rise to the iterative procedure commonly referred to as Jacobi relaxation,

$$x_{n+1} = D^{-1}(b - Lx_n - Ux_n). \tag{3}$$

For the class of linear systems that arise in the discretization of elliptic PDEs, it is easily seen that iteration (3) converges to the solution $x^\star$ of equation (1).

The obvious advantage of such an iteration is its simplicity and therefore its ease of implementation. However, the convergence of such relaxation methods is typically prohibitively slow. To illustrate the convergence behavior, we discretize

$$u'' = 0, \quad \text{on } (0,1), \tag{4}$$

with homogeneous Dirichlet boundary conditions $u(0) = u(1) = 0$, using finite differences on a regular grid. Note that in this example, an iterate equals the error.

Figure 1 shows an oscillatory initial guess (top), the solution after one Jacobi relaxation (middle), and the error after five Jacobi relaxations (bottom).

The figure illustrates that already after one Jacobi relaxation, the high frequency error components are gone (middle), and after five relaxations only a very low frequency error remains (bottom).

An important observation is that the notion of frequency is linked to the mesh size. The function in the bottom graph in Figure 1 can be adequately represented on a mesh consisting of five equally spaced grid points. On such a grid, this function can be interpreted as being of high frequency. Jacobi relaxation

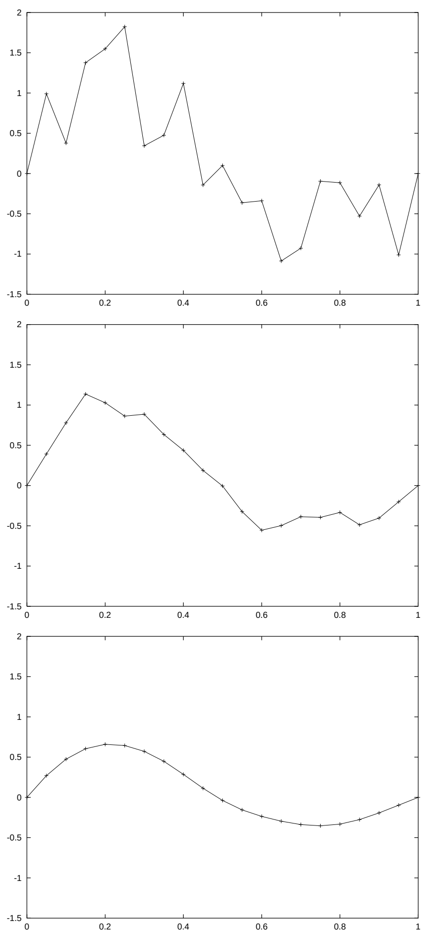

Figure 1: Error reduction in Jacobi relaxation. The top graph shows an oscillatory initial guess, the middle graph shows an intermediate solution after one Jacobi smoothing step, and the bottom graph shows the solution after five Jacobi smoothing steps.

would dampen it in a couple of iterations. This observation motivates multilevel methods.

We first note that, if x is any vector and $r = b - Ax$ the residual, then solving for the correction $\tilde{x}$

$$A\tilde{x} = r \tag{5}$$

enables us to write down the solution $x^\star = x + \tilde{x}$. Using a relaxation method on the finest grid, transferring the residual to a coarser grid, solving the correction equation there, and correcting the fine grid solution with an interpolation of the correction from the coarse grid, summarizes a two grid method. Of course, this two grid method can be used to solve the coarse grid linear system (if the coarse grid can be further coarsened). This constitutes the recursive definition of the multigrid method.

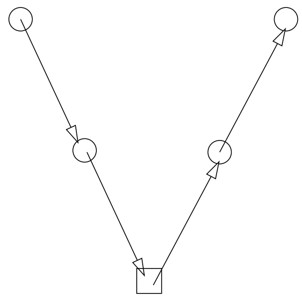

Figure 2: V-cycle for three levels (circles – relaxation, arrows down – restriction, arrows up – interpolation, square – coarse grid solve)

Figure 2 illustrates one iteration of the multilevel method. Relaxation is represented by circles, restriction and interpolation are represented by arrows that are pointing down or up, respectively, and the coarse grid solve is represented by a square. Because of the characteristic shape of this diagram, the iteration is called V-cycle.

The most important property of the multilevel method is that it is typically of complexity $O(N)$ for the V-cycle, where N is the number of unknowns. In other words, it is scalable, whereas most other known linear solvers are not.

3. OPTIMIZATION IN GRID GENERATION

Many problems in grid generation can be stated as a discrete nonlinear optimization problem, where the objective function has the form

$$F(G) = \sum_{vertices} F_{loc}(G). \tag{6}$$

Here, G is the grid. An example for this is the reference Jacobi objective functional that is used in mesh smoothing (see, for example, [1]).

Methods that are commonly used to solve these types of optimization problems include nonlinear conjugate gradient methods, Gauss-Seidel Newton, and Newton Gauss-Seidel. These methods have two common flaws. Their convergence is typically slow, and they do not scale well with the size of the mesh.

To exemplify the problems that all these methods exhibit, we briefly describe the Gauss-Seidel Newton method. Denote by $\{x_i^{(0)}\}_{i=1,\ldots,n}$ the vertices, and by E the edges in the initial grid $G^{(0)} = (\{x_i^{(0)}\}_{i=1,\ldots,N}, E)$. The k-th iteration is as follows.

1. Loop over the vertices $i = 1, \ldots, N$ and solve for each i the local minimization problem

   $$x_i^{(k)} = \arg\min_x F(x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x, x_{i+1}^{(k-1)}, \ldots, x_N^{(k-1)})$$

   by Newton's method.
2. Set $G^k = (\{x_i^{(k)}\}_{i=1,\ldots,N}, E)$.

Of course, in each Newton step, the new optimal position for vertex i can only be chosen inside a set that preserves mesh validity. We refer to this set at the feasible set. The problem with this approach is that if the optimal grid is far away from the initial grid, the convergence will be very slow.

We propose an optimization algorithm that is based on ideas from multilevel methods. These methods build on the flaws that the aforementioned methods have, in much the same way, that multilevel methods build on the flaws of simple relaxation methods.

4. COARSENING AN UNSTRUCTURED TRIANGULAR GRID – THE RESTRICTION OPERATOR

The coarsening procedure that we present is very similar to Delaunay-Coarsening (DC), introduced in [5]. In an initial step in DC, the list of vertices is reordered in such a way that all boundary vertices come first. In a loop over the vertices in this list the current vertex is added to the list of coarse vertices and its neighbors are deleted from the list of vertices. Hence, initially the boundary is coarsened, and then the interior. As a slight modification of this algorithm, we consider such boundary vertices first that are necessary to properly resolve the shape of the domain. An example for such vertices are the four corner vertices of a square.

An alternative coarsening algorithm that could be employed in the context of our algorithm is presented in [6]. This algorithm is based on edge contraction and it works in two as well as in three dimensions. For the purposes of this paper, it was simpler to use a strategy that builds on readily available software for two dimensional Delaunay triangulation. We now formalize our coarsening procedure.

We will need the notion of distance for vertices in a grid. To that end, we first define a path $p_{i,j}$ connecting vertices x_i and x_j through a grid as a sequence of edges $\{e_{i_j}\}_{j=1,\ldots,K}$, where e_{i_j} and $e_{i_{j+1}}$ are adjacent, and e_{i_1} is adjacent to x_i, and e_{i_K} is adjacent to x_j. We use the notation $|p_{i,j}| = K$ to denote the length of $p_{i,j}$. Now, denote by $d_G(x_i, x_j)$ the distance function for a particular grid. It is defined as

$$d_G(x_i, x_j) = \min\{|p| : p \in P_{i,j}\}, \tag{7}$$

where $P_{i,j}$ is the set of all paths that connect vertices x_i and x_j.

As a first step in our coarsening procedure, we add all fine vertices to the set of available vertices $L = \{x_i^f\}_{i=1,\dots,N}$, and initialize the set of coarse vertices as the empty set: $C = \emptyset$. In the next step, we mark all boundary vertices that are necessary to properly resolve the geometry of the domain as coarse vertices and remove them from L: If $x_i^f \in L$ is such a vertex, let $L \leftarrow L \backslash \{x_i^f\}$, and let $C \leftarrow C \cup \{x_i^f\}$. Then, all of their neighbors, that is, all fine vertices $x_j^f \in L$ for which $d_G(x_j^f, x_i) = 1$, for any $x_i \in C$, are removed from L.

In a second step, we loop over the remaining boundary vertices in L. The current vertex x_i^f in this loop is removed from L: $L \leftarrow L \backslash \{x_i^f\}$, and added to C: $C \leftarrow C \cup \{x_i^f\}$. Then all neighbors of x_i^f in L are also removed from L.

All other coarse vertices are determined in a greedy algorithm to find a maximal independent set. Pick any vertex $x^f \in L$, mark it as a coarse vertex $C \leftarrow C \cup \{x^f\}$, and remove it from L: $L \leftarrow L \backslash \{x^f\}$. Then, remove all its neighbors from L: For all $\tilde{x}^f \in L$, such that $d_G(x^f, \tilde{x}^f) = 1$, $L \leftarrow L \backslash \{\tilde{x}^f\}$. The algorithm terminates, when L is empty.

We now have a set C of vertices that are marked as coarse. To create a coarse grid from these, we use a the classic sweep line algorithm by S. Fortune (see [7]) to generate a Delaunay triangulation. For the simple domains we have considered so far, there have been no problems with boundary preservation. As a fix we plan to use constrained Delaunay

Figure 3 illustrates the coarsening process. Given a fine grid (top), a maximal independent set of coarse points is determined (middle), from which a Delaunay triangulation is generated (bottom).

4.1 Restriction of the fine to the coarse grid

The described procedure yields a coarse grid, as well as a relationship between this coarse and the initial fine grid. We use this relationship to define the restriction operator. We keep the connectivity of both the fine and the coarse grid fixed. When the fine grid moves, the coarse grid moves along with it by virtue of each coarse vertex having the same location as its corresponding fine vertex. We call the process of updating the coordinates of the coarse grid vertices to the values of their corresponding fine grid vertices the restriction of the fine to the coarse grid.

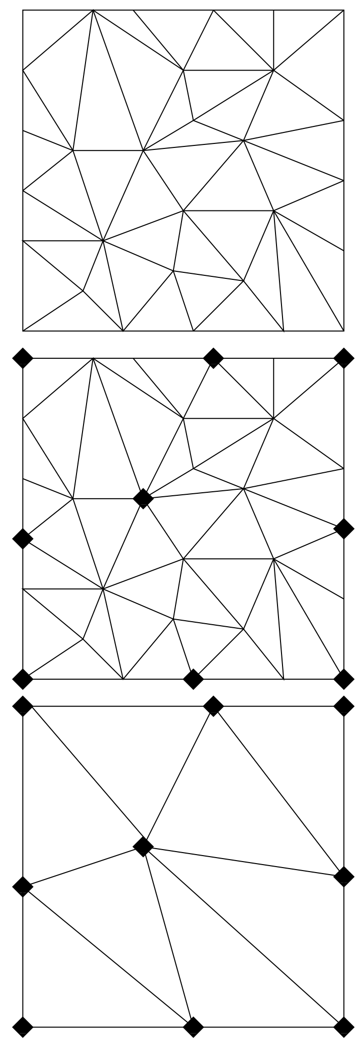

Figure 3: Coarsening an unstructured triangular grid.

5. INTERPOLATING A FINE GRID FROM A COARSE GRID

In the previous section, we have introduced the restriction operator. After a coarse grid is created, we record for each fine vertex what its relation to the coarse grid is. There are two possible cases.

- First, a fine vertex x^f was marked as coarse in the coarsening procedure ($x^f \in C$), and
- second, a fine vertex was discarded in the coarsening procedure ($x^f \notin C$).

In the first case, we interpolate by injection. More specifically, we replace the coordinates of the fine vertex with the coordinates of the coarse vertex. In the second case, we initially find the coarse triangle τ^c, such that the coordinates of the fine vertex x^f are inside of τ^c. Then, we store the barycentric coordinates $(\xi^{(1)}_{x^f,\tau^c}, \xi^{(2)}_{x^f,\tau^c}, \xi^{(3)}_{x^f,\tau^c})$ of vertex x^f inside of τ^c. To interpolate the coordinates of x^f from the coarse grid after the coarse grid has changed, we compute new physical for vertex x^f using these barycentric coordinates.

5.1 Mesh folding induced by interpolation

It is possible that after interpolating a fine grid from a coarser grid, the fine grid is folded (i.e. some cell volumes are negative) in some places. Here we give an example to illustrate this possibility.

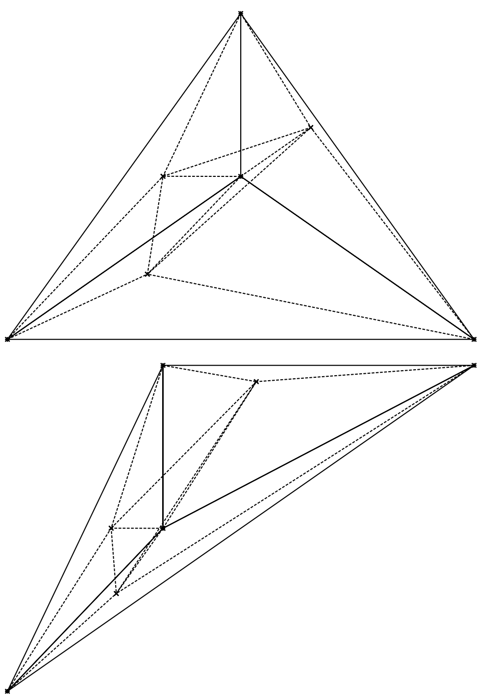

Figure 4: An example for mesh folding induced by interpolation. The dashed lines are the fine grid and the solid lines the coarse grid. The initial grid is shown on top. In the bottom grid, the bottom right corner has moved up, and the fine grid has moved with it, consequently being folded at center coarse grid vertex.

Figure 4 illustrates this case. The top grid shows the initial configuration, with dashed lines representing the fine grid and solid lines representing the coarse grid. In the bottom grid the bottom right corner of the initial grid has moved up. With it the fine grid has moved, and consequently one fine grid triangle is folded in the area of the center coarse grid vertex.

This grid folding can be fixed using the mesh untangling procedure introduced in [8]. In this procedure, a three stage approach is taken. In the first stage, the grid is untangled using the feasible set method. The feasible set for a vertex is defined as the set of all positions of the vertex for which each connected element is valid at the corners affected by the position of the vertex (see [8, chapter 2]). The feasible set can be interpreted as a set of inequality constraints, given by the the feasible half planes. See Figure 5 for an example of the feasible area of a vertex. Here the feasible area

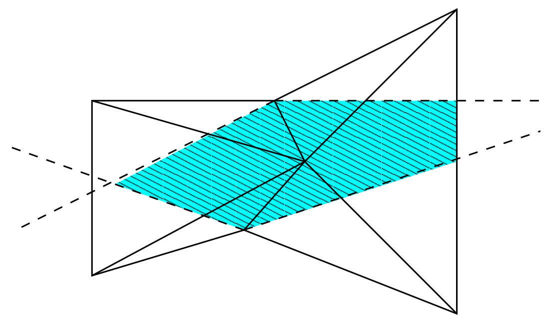

Figure 5: The feasible set for the placement of the central vertex is the shaded area.

is the shaded area. In practice, the feasible set can be computed by using the simplex method to minimize simple linear functions (e.g. $f(x, y) = 1, -1, y, -y$) to find three distinct corners of the feasible set polygon. The vertex is then placed at the center of this triangle that is formed by these three corners.

The feasible set for a vertex can be empty. In this case the feasible set method fails. In a second stage a functional is minimized. Denote by $\alpha(\tau)$ the area of element τ, and let $\beta > 0$ be a small parameter. The functional

$$f(x) = \sum_{\tau \in G} (|\alpha(\tau) - \beta| - (\alpha(\tau) - \beta))^2 \qquad (8)$$

is minimized using standard minimization methods such as the conjugate gradient method [2, 9]. Note that functional f has only contributions from elements that have negative area. All other elements contribute zero. It is also important to observe that f is smooth and convex. In a third step, another sweep of the feasible set approach is applied.

The untangling method outlined above is not guaranteed to work. However, the cases of tangled meshes that arise in the multilevel procedure are well suited to this untangling procedure. Typically, only few cells become tangled in the interpolation procedure. In most cases one sweep of the feasible set approach is enough to untangle the grid.

6. DEFINING A COARSE GRID OPTIMIZATION PROBLEM

In the previous sections we have defined a restriction and, derived from it, an interpolation operator. Both operate on triangular grids. Our goal is to accelerate the original optimization problem, by solving a similar problem on a coarser mesh.

We give an example for which it is easy to derive an appropriate coarse optimization problem. We first define the L^2 projection Π_{τ,V_τ}

$$\Pi_{\tau,V_\tau}(f) = \arg\min_{\phi\in V_\tau} \|f-\phi\|_{L^2(\tau)} \tag{9}$$

for a triangle τ and a function space V_τ on τ. As examples, we can think of V_τ as the space of constant functions on τ

$$V_\tau^{const} = \{\phi(x,y) = c : c \in \Re\} \tag{10}$$

or the space of linear functions on τ

$$V_\tau^{lin} = \{\phi(x,y) = ax + by + c : a, b, c \in \Re\}. \tag{11}$$

Note that $\Pi_{\tau,V_\tau^{const}}(f) = \overline{f}$, the average of f on τ. Then, the objective functional is

$$F(G) = \sum_{\tau\in G} \|f - \Pi_{\tau,V_\tau}(f)\|_{L^2(\tau)}. \tag{12}$$

The grid G_{min} that is the minimizer of optimization problem

$$G_{min} = \arg\min_G F(G) \tag{13}$$

yields the best possible L^2 approximation of the function f with piecewise constant or piecewise linear functions, on grids that have the same connectivity as the starting grid.

Based on (12) it is straightforward to define an appropriate coarse grid minimization problem. In fact, we can use the same objective functional as in (12), and apply it to a coarser grid.

$$F(G_{coarse}) = \sum_{\tau\in G_{coarse}} \|f - \Pi_{\tau,V_\tau}(f)\|_{L^2(\tau)}. \tag{14}$$

7. THE MINIMIZATION PROCEDURE

First a hierarchy of grids is created, using the procedure described in section 4. A natural stopping criterion for the successive coarsening is whether or not the boundary, or, for example, interior features, such as material interfaces, can be resolved on a coarser grid.

On each level, we employ a few iterations of a simple optimization algorithm, such as the Gauss-Seidel Newton method described in section 3. After that, the mesh is restricted to the next coarser level (see section 4.1). This is repeated until we arrive at the coarsest level. Here more effort is spent on the optimization step, to find the optimal coarsest grid. Then, the mesh is interpolated to the next finer level, and a few sweeps of the Gauss-Seidel Newton method are applied on the finer level. We repeat this interpolation and optimization step until we arrive at the finest level.

We refer to this procedure as a V-cycle. This V-cycle can be formulated recursively. We call the following procedure $V(k, \nu_1, \nu_2)$ the iteration on level k.

1. ν_1 iterations of Gauss-Seidel Newton on level k.
2. Coarsen the grid on level k to level $k-1$.
3. If $k > 2$, call $V(k-1, \nu_1, \nu_2)$, else solve the minimization problem on level 1 to a high level of accuracy.
4. Interpolate the grid from level $k-1$ to level k.
5. ν_2 iterations of Gauss-Seidel Newton on level k.

8. COMPLEXITY ANALYSIS OF THE V-CYCLE MINIMIZATION

We first consider the coarsening procedure. The selection of coarse vertices on level k with N_k fine vertices takes $O(N_k)$ steps. The number of selected coarse vertices depends on the connectivity of the fine grid. On a regular grid, the number of coarse points will be $N_{k-1} \approx N_k/4$, if the fine vertices are numbered in a lexicographic fashion. See figure 6 for an example. For

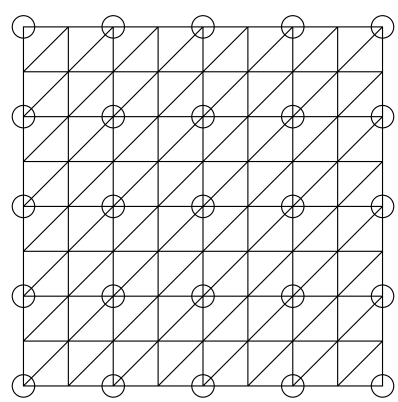

Figure 6: Coarsening of a regular grid, circles denote coarse vertices. The number of fine vertices is 81, and number of coarse vertices is 25.

an unstructured grid, it is difficult to give a good estimate of the ratio of the number of fine and coarse vertices after coarsening. However, the number of coarse vertices depends on the average degree of a fine vertex. In our coarsening algorithm, coarse grids are generated using a Delaunay triangulation. The expected maximum degree of a vertex in a Delaunay triangulation is $\Theta(\log n/\log\log n)$, where n is the number of vertices [10]. In practice, the coarsening ratio is close to 1/4.

The work per V-cycle is relative to N_{total}, the total number of vertices on all levels combined. Assuming a coarsening factor of $0 < \delta < 1$, we get

$$
\begin{aligned}
N_{total} &\approx N_K + N_{K-1} + \ldots + N_1 && (15)\\
&= N_K \sum_{i=1}^{K} \delta^{K-i} && (16)\\
&\leq \frac{N_K}{1-\delta} && (17)
\end{aligned}
$$

Here, N_K is the number of vertices in the finest level grid. For a coarsening factor of $\delta = 1/4$, this means $N_{total} \approx 4N_K/3$.

It is difficult to estimate the complexity of the Newton step that is performed inside the Gauss-Seidel loop. It is however reasonable to assume that this complexity is similar on all levels.

The complexity of the interpolation is $O(N_k)$ on level k, if no mesh folding occurs. So, in this case, over all levels, it is $O(N_{total})$. In practice, mesh folding only occurs in few locations on the fine grid, and can be very efficiently remedied with the untangling procedure outlined above. The computational effort spent in the untangling procedure is negligible.

The coarsening procedure, after the coarse grids have been created in an initial step, is of complexity $O(N_{total})$. The creation of coarse grids consists of the selection of coarse vertices, which is of complexity $O(N_k)$, on level k, and a Delaunay triangulation, which is of complexity $O(N_k \log N_k)$ on level k.

In conclusion the initial coarse grid creation is of complexity $O(N_K \log N_K)$. After that, each V-cycle is of complexity $O(N_K)$. Hence, the complexity of a V-cycle is the same as the complexity of a Gauss-Seidel Newton iteration.

9. A NUMERICAL EXAMPLE

We present a numerical example that is representative of the performance gain that can be achieved with our multilevel optimization procedure. The code is written in the C programming language.

We start with a regular triangular grid G of with 20×20 vertices on the domain $(0,1) \times (0,1)$. The objective is to find the grid that minimizes functional

$$F(G) = \sum_{\tau \in G} \|f - \Pi_{\tau, V_\tau^{const}}(f)\|_{L^2(\tau)}, \qquad (18)$$

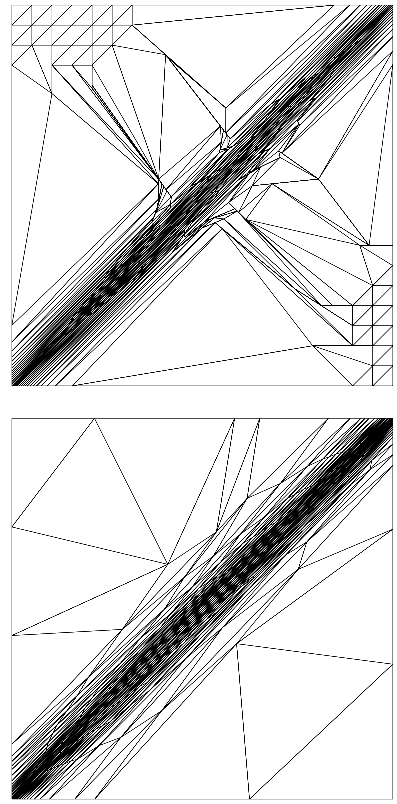

Figure 7: Fine grid after 500 iterations of Gauss-Seidel steepest descent (top), and 3 V-cycle iterations (bottom).

with

$$f(x,y) = 20\tanh(x-y). \qquad (19)$$

This function has a very steep gradient along the line $x = y$, and is essentially flat elsewhere. An optimal grid will have most grid points clustered along this line.

First, we used a Gauss-Seidel steepest descent procedure to minimize the objective functional. The grid that resulted after 500 iterations is depicted in figure 7 (top image).

Second, we used a V-cycle iteration with $\nu_1 = \nu_2 = 3$, i.e. three sweeps of the Gauss-Seidel steepest descent method were employed before restriction and after interpolation. The coarse grid optimization problem was solved using $\nu_c oarse = 10$ iterations of a Gauss-Seidel steepest descent procedure. The grid that resulted after three V-cycle iterations is depicted in figure 7 (bottom image). Table 1 shows the value of the objective functional and the time required to reach this value. Note that the same computer code was used for the Gauss-Seidel steepest descent and the relaxation step inside the V-cycle iteration.

	Gauss-Seidel	V-cycle
$F(G) =$	6.44(-5)	4.74(-5)
# iterations	500	3
time (sec)	1027	25.9

Table 1: Comparison of iteration counts and wall clock times for the Gauss-Seidel steepest descent iteration (left column), and the V-cycle iteration (right column).

The V-cycle iteration converges much faster than the fine grid Gauss-Seidel steepest descent iteration. The number of iterations, as well as the time are much smaller for the multilevel method. Additionally, the V-cycle iteration reaches a minimum after only three V-cycle iterations. The Gauss-Seidel steepest descent iteration was terminated at a maximum iteration count of 500, for the lack of a good stopping criterion.

For a fair comparison between the two approaches, it is not useful to compare numbers of Gauss-Seidel iterations. Note that Gauss-Seidel iterations on coarse levels are less expensive than on fine levels. Assuming a coarsening factor of 1/4, we can estimate the cost of one $V(k, \nu_1, \nu_2)$ iteration relative to the cost of one Gauss-Seidel steepest descent iteration at

$$\sum_{i=1}^{k-1} \frac{\nu_1 + \nu_2}{4^{i-1}} + \frac{\nu_{coarse}}{4^{k-1}}. \tag{20}$$

Thus, in our example, we estimate the cost of one V-cycle iteration equivalent to $6 + 6/4 + 10/16 = 8.75$ Gauss-Seidel steepest descent iterations. So, for the total cost of $3 \times 8.75 = 26.25$ Gauss-Seidel steepest descent iterations the problem is converged.

In the V-cycle iteration, six Gauss-Seidel steepest descent iterations are performed on each level. To estimate the overhead that is induced by coarse levels, we divide the time per V-cycle iteration $t_{V-cycle} = 8.63sec$ by the number of Gauss-Seidel steepest descent iteration per V-cycle:

$$\frac{t_{V-cycle}}{4} = 2.15sec. \tag{21}$$

The average time for one iteration of the Gauss-Seidel steepest descent procedure is

$$t_{Gauss-Seidel} = 2.05sec. \tag{22}$$

Comparing the two, we observe that the additional work associated with coarser levels in a V-cycle iteration is not very significant. We also note that the Gauss-Seidel steepest descent iterations that are performed as part of a V-cycle tend to require less costly line searches than for the fine level case. This also helps to keep the average time per V-cycle per fine level Gauss-Seidel steepest descent iteration $t_{V-cycle}/4$ small.

10. EXTENSION TO 3D

The algorithm presented in this paper is a 2D algorithm. The coarsening procedure can be extended to 3D using Delaunay tessellations in 3D. However, as mentioned above, a coarsening procedure such as the one described in [6] might be better suited, since Delaunay tessellations in three dimensions can produce slivers. The untangling procedure is also extensible to 3D (see [8]). This indicates that an extension of our algorithm to 3D is possible. We are planning to pursue this direction in our future research.

11. CONCLUSIONS

We have introduced a new multilevel-type optimization procedure that is well suited to very efficiently solve optimization problems that commonly occur in grid generation applications. Our complexity analysis indicates that our algorithm scales linearly with the number of unknowns. Our research has been focused on the two dimensional case, however, all components of the algorithm are available in three dimensions, as well. We will investigate the three dimensional case in a future paper. It is essential for this approach that a coarse grid representation of the objective function that is to be minimized can be derived. In other words, the change in the initial grid that is required to obtain the optimal grid must be expressible as small changes of vertex positions relative to the positions of neighbor vertices, plus larger changes of positions of groups of vertices. This is possible for applications where the objective is to find a grid that is optimal for the approximation of a function. We plan to extend this work to include the case where an error estimate, and not the actual error, is to me minimized.

References

[1] Knupp P., Margolin L., Shashkov M. "Reference Jacobian Optimization-Based Rezone Strategies for Arbitrary Lagrangian Eulerian Methods." *J. Comput. Physics*, vol. 176, 93–128, 2002

[2] Knupp P.M., Steinberg S. *The Fundamentals of Grid Generation.* CRC Press, 1993

[3] Briggs W.L., Henson V.E., McCormick S.F. *A Multigrid Tutorial, Second Edition.* SIAM, 2001

[4] Trottenberg U., Osterlee C.W., Schüller A. *Multigrid.* Academic Press, 2001

[5] Guillard H. "Node Nested Multigrid with Delaunay Coarsening." Tech. rep., INRIA Report No. 1898, 1993

[6] Ollivier-Gooch C. "Coarsening unstructured meshes by edge contraction." *Int. J. Numer. Meth. Engng.*, vol. 57, 391–414, 2003

[7] Fortune S.J. "A Sweepline Algorithm for Voronoi Diagrams." *Algorithmica*, pp. 153–174, 1987

[8] Vachal P., Garimella R.V., Shashkov M.J. "Untangling of 2D Meshes in ALE Simulations." *under review in J. Comput. Phys.*

[9] Nocedal J., Wright S.J. *Numerical Optimization.* Springer, 1999

[10] Bern M., Eppstein D., Yao F. "The expected extremes in a Delaunay triangulation." *Int. J. Comput. Geom. & Appl.*, vol. 1, no. 1, 79–92, 1991