Efficient Nonlinear Solvers for Laplace-Beltrami Smoothing of Three-Dimensional Unstructured Grids

Markus Berndt and J. David Moulton
Mathematical Modeling and Analysis Group
Theoretical Division, MS B284
Los Alamos National Laboratory
Los Alamos, NM 87545

Glen Hansen
Multiphysics Methods Group
Advanced Nuclear Energy Systems Dept.
Idaho National Laboratory
Idaho Falls, ID 83415-3840

November 20, 2007

Abstract– The Laplace-Beltrami system of nonlinear, elliptic, partial differential equations has utility in the generation of computational grids on complex and highly curved geometry. Discretization of this system using the finite element method accommodates unstructured grids, but generates a large, sparse, ill-conditioned system of nonlinear discrete equations. The use of the Laplace-Beltrami approach, particularly in large-scale applications, has been limited by the scalability and efficiency of solvers. This paper addresses this limitation by developing two nonlinear solvers based on the Jacobian-Free Newton-Krylov (JFNK) methodology. A key feature of these methods is that the Jacobian is not formed explicitly for use by the underlying linear solver. Iterative linear solvers such as the Generalized Minimal RESidual (GMRES) method do not technically require the stand-alone Jacobian;

instead its action on a vector is approximated through two nonlinear function evaluations. The preconditioning required by GMRES is also discussed. Two different preconditioners are developed, both of which employ existing Algebraic Multigrid (AMG) methods. Further, the most efficient preconditioner, overall, for the problems considered is based on a Picard linearization. Numerical examples demonstrate that these solvers are significantly faster than a standard Newton-Krylov approach; a speedup factor of approximately 26 was obtained for the Picard preconditioner on the largest grids studied here. In addition, these JFNK solvers exhibit good algorithmic scaling with increasing grid size.

Keywords– Tangent stiffness matrix, Unstructured mesh, Grid smoothing, Elliptic smoothing, Laplace-Beltrami, Jacobian free, Newton-Krylov, Algebraic multigrid.

1 Introduction

The desire to simulate more complex and detailed physical processes continues to drive research in grid generation algorithms. Advances in computational physics applications demand continued improvement in grid quality, and ever-increasing grid refinement. To support these changing application requirements, grid generation algorithms are becoming more flexible, complex, and computationally intensive. Additionally, grid topologies are becoming more general, with hybrid or mixed element grids and complex boundary structure becoming important in a variety of applications.

The Laplace-Beltrami system is classified as an *elliptic grid generation method*, and has been studied extensively for structured grids (see [1, 2, 3, 4, 5]). In d-dimensions, this system may be written as a coupled set of d nonlinear elliptic (diffusion) equations. Each coordinate is defined by one nonlinear diffusion equation in which the metric tensor acts as the diffusion coefficient. These nonlinear equations are coupled through the dependence of the metric tensor on all d coordinates. Recently, Hansen et al.[6, 7] developed a new methodology, based on the Laplace-Beltrami system, for smoothing unstructured grids in two-dimensional and three-dimensional problems. The metric tensor that appears in the Laplace-Beltrami equations is typically viewed as being *descriptive* of the particular element in the grid. However, a metric tensor might also be constructed by the user with the goal to modify the

current state of the grid; indeed, this notion of the metric tensor may be considered as being *prescriptive* in nature. This prescriptive tensor (*i.e., target* metric tensor) is quite general in concept. It is required to be symmetric positive definite, however. The resulting freedom in the specific choice of the metric tensor has previously been employed for the purpose of grid adaptation to various solution properties on two-dimensional structured grids [8, 9, 10, 11]. In the *target metric* approach presented in [7], this freedom is exploited to improve grid quality within the constraints of the geometry of the domain and the connectivity of the grid.

There are many other algorithms that aim to improve the quality of a grid. Many such algorithms can be implemented very efficiently and are easily parallelized (e.g. averaging algorithms, such as Laplacian grid smoothing [12]). Such algorithms are well suited to incrementally improve a grid, for example, by applying a few iterations of such a grid smoothing algorithm in every time step of a time-dependent simulation. However, for meshes with a concave boundary naive averaging approaches fold the mesh in this region, and more sophisticated averaging generates severely compressed and distorted cells near the boundary. An alternative to averaging algorithms are minimization algorithms (see, for example, [13, 14]) that improve grid quality by minimizing an objective functional that describes element quality. Such algorithms are also well suited for incremental grid improvement, while they are typically less efficient than averaging algorithms. Both these classes of algorithms become very inefficient and, in particular, do not scale well algorithmically when they are run to convergence. The nonlinear solver presented here exhibits good algorithmic scaling, and hence we demonstrate that the Laplace-Beltrami grid smoothing algorithm that was introduced in [6, 7] is feasible for large computational physics applications in steady-state or with implicit time integration.

Solving the nonlinear system of diffusion equations that arises in both conventional Laplace-Beltrami grid generation algorithms and the *target metric* approach to Laplace-Beltrami grid smoothing is a challenging problem. This paper focuses only on the latter, but notes that the problems are quite similar. Discretization of this nonlinear system of diffusion equations using a Finite Element Method (FEM) results in a large, sparse, and *ill-conditioned* system of nonlinear equations [7]. For a grid with N vertices, this system has

$3N$ unknowns. It is common for complex applications to use grids consisting of hundreds of thousands to several million elements; it is thus paramount that the solution algorithm scales well with increasing grid size. In addition, the target metric tensor will typically be discontinuous across element boundaries. Thus, the solver must be robust with respect to these jumps, as well as capable of handling unstructured grids on complex geometries. Finally, parallel scalability must be considered in the design as many applications require the additional processing power and memory that is only available through the use of parallel architectures.

In [6] and [7], a Newton-Krylov (NK) solver was used that formed the full Jacobian for each Newton Step. Since the Jacobian is a large sparse nonsymmetric matrix, the restarted Generalized Minimal Residual method (GMRES) was chosen as the Krylov solver. A block incomplete factorization, BILU(1), was used in conjunction with symmetric successive over-relaxation (SSOR) as a preconditioner for GMRES. Although this NK solution algorithm performed well enough to demonstrate the potential of the target metric smoothing methodology, it is not practical for large-scale applications. Specifically, the computation of the Jacobian is very expensive because the target metric introduces dependencies in a neighborhood of each element. Hence, even though this component of the solver scales in a linear fashion with grid size, the cost is prohibitive. In addition, assuming that the movement of the vertices in each nonlinear iteration is relatively small, the Jacobian is diffusion dominated, and hence, the cost of the preconditioned GMRES solution scales superlinearly with the grid size.

This work develops efficient nonlinear solvers that address these two weaknesses of the standard NK solver. First, motivated by the need to eliminate the evaluation of the full Jacobian, this study uses the Jacobian-Free Newton-Krylov (JFNK) class of nonlinear solvers. These methods have been used successfully in a broad range of applications [15, 16] and are based on the observation that Krylov methods do not explicitly need a matrix, but rather build a solution based on the action of the matrix on a vector. This matrix-vector product may be obtained by a finite-difference approximation using the nonlinear functional itself, allowing the computation of the Jacobian to be avoided.

To address the second weakness, an effective preconditioner for the restarted GMRES

solver that leverages a multilevel solution algorithm to deliver the necessary efficiency, is sought. There are two aspects to the preconditioning problem. First, a preconditioner $\mathcal{P}$ must be developed that is sufficiently close to the full Jacobian $\mathcal{J}$, such that the number of GMRES iterations is independent of the grid size. In this context, the perfect preconditioner is the Jacobian itself, $\mathcal{P} = \mathcal{J}$. While this is too costly to form in each Newton step, it is not uncommon to perform this computation once during the first Newton step and reuse this *frozen* Jacobian throughout the remainder of the solution process [15]. An alternative preconditioning approach is suggested by JFNK studies of a similar system of nonlinear partial differential equations (PDEs) that describe nonequilibrium radiation diffusion [17, 18]. In these studies, the Picard linearization was found to be an effective preconditioner. For both preconditioners, their effectiveness reflects the fact that they capture the underlying elliptic nature of the Jacobian. For elliptic problems, multilevel iterative methods are the only solvers whose computational cost scales optimally with problem size. An introduction to multigrid methods is given in [19], and a comprehensive review of the field is presented in [20]. Multigrid methods have been studied previously in the context of eliptic grid generation (see, for example, [21]). In particular, Laplace-Beltrami grid-generation was studied in [22], where the linear solver for the Picard linearization of a three-dimensional Laplace-Beltrami grid-generator was accelerated by a two-dimensional structured multigrid algorithm, and in [11] a "matrix-light" structured multigrid solver was used as a preconditioner in a two-dimensional grid-generation application. The latter is based on re-discretizing the linearized differential operator on coarser grids, while the multigrid code employed in the former relies on matrix-dependent prolongation and restriction operators. While these are good solver strategies for structured grid applications; this paper seeks to address the needs of applications employing 3D unstructured grids. This generalization will be addressed with the use of algebraic multigrid (AMG) [23, 24, 25], to approximately-invert the preconditioner using a small number of V-cycles. AMG methods were developed to handle unstructured grids, and are robust with respect to jumps in the diffusion coefficients. Although the underlying theory and heuristics in AMG methods are motivated by scalar diffusion problems, they have been used successfully for systems of PDEs and nonsymmetric problems. It is important to note that efficient implementations of both Krylov solvers (e.g., the PCG [26] and

HYPRE [27] packages) and AMG solvers (e.g., the LAMG package [28] and BoomerAMG in HYPRE [27]) with good parallel scalability, are available. Thus, the nonlinear solvers developed here are extensible to parallel architectures.

The objective of this research is to demonstrate the efficient Laplace-Beltrami target-metric smoothing of unstructured grids and the feasibility of this technique for large-scale applications. Section 2 reviews the *target metric* approach to Laplace-Beltrami grid smoothing first introduced in [6, 7], and discusses the associated boundary conditions. Section 3 highlights key elements of the JFNK approach and introduces the *frozen* Jacobian and the *Picard* linearization preconditioners. The issues associated with using AMG to approximately invert these preconditioners are considered in Section 3.2. Section 4 presents a performance comparison of JFNK solvers that use the Frozen and Picard preconditioners with the reference NK solver that uses a BILU(1)/SSOR preconditioner. The comparison includes results for three grids from [7] (Section 4.1) and a scaling study on two sequences of logically-structured hexahedral grids (Section 4.2). Finally, conclusions are developed in Section 5.

2 The target metric approach to grid smoothing

The use of a target metric tensor in conjunction with the Laplace-Beltrami equation system was recently proposed by Hansen et al.[6, 7]. Elliptic equation systems have been used extensively in grid generation applications, due to their robustness, smoothing behavior, and other desirable characteristics that they impart on the final grid. Each elliptic approach differs in implementation, depending on the goals and requirements of the application that they were developed to support. The Laplace-Beltrami system is an elliptic system derived from the Laplacian operator in general coordinates [2]. In the Laplace-Beltrami approach, the metric tensor is typically viewed as descriptive in nature; it describes the mapping of each element into the final computational grid. The Laplace-Beltrami target metric approach [6, 7] employs the metric tensor in a prescriptive manner; a tensor is created to describe an "improved" version of each of the elements contained in the original grid. This new tensor, the *target* element metric tensor, is used in the solution of the Laplace-Beltrami system to

enhance the previous grid state. In the interior of the grid, the target metric method is based on prescribing deviations from a unity ratio and/or the angular relationship of the fundamental coordinate directions. This is typically best implemented using the method of *coarse-graining* that was introduced in [7], especially on an unstructured or hybrid-element grid. At the grid boundary, the nodal coordinates of the grid are typically fixed, forming a Dirichlet boundary condition. This relationship is shown schematically in the two-dimensional diagram, Figure 1. In both coordinate systems shown in the illustration, $\mathbf{u} = (u, v)$ and $\mathbf{x} = (x, y)$, there is a single physical domain. Given this domain with an interior grid that characterizes $\mathbf{u}$, the objective to grid improvement is to modify the interior grid while preserving the shape of the boundary of the domain.

Given a domain $\Omega \in \Re^3$, the coordinate systems of interest are written as $\mathbf{x} = (x^1, x^2, x^3)$ and $\mathbf{u} = (u^1, u^2, u^3)$, such that the covariant components of the metric tensor may be expressed as

$$g_{\alpha\beta} = \sum_{i=1}^{3} \frac{\partial x^i}{\partial u^\alpha} \frac{\partial x^i}{\partial u^\beta} \,. \tag{1}$$

The contravariant components of the metric tensor are written as $g^{\alpha\beta}$; these components are simply those of the inverse of $g_{\alpha\beta}$. Omitting the details of the derivation (c.f., [7, Section 2]), the Laplace equations for the coordinates x^i may be written as

$$\Delta x^i = \frac{1}{\sqrt{g}} \frac{\partial}{\partial u^\alpha} \left(\sqrt{g} g^{\alpha\beta} \frac{\partial x^i}{\partial u^\beta} \right) = 0 \,, \tag{2}$$

where g is the determinant of the covariant metric tensor.

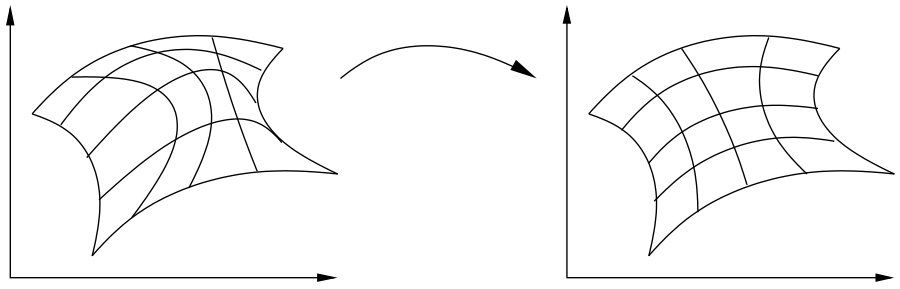

Figure 1: The objective of combining a *target metric* with the Laplace-Beltrami system is to improve the quality of the interior grid while fixing the physical boundary of the object.

Equation (2) is the basis for the finite element discretization. To derive this finite element approximation, multiply (2) by a sufficiently smooth test function w and integrate over the domain. After integration by parts, one obtains

$$\int_\Omega \frac{\partial w}{\partial u^\alpha}\sqrt{g}g^{\alpha\beta}\frac{\partial x^i}{\partial u^\beta}d\mathbf{u} - \int_{\partial\Omega} w\sqrt{g}g^{\alpha\beta}\frac{\partial x^i}{\partial u^\beta}ds_\alpha = 0. \tag{3}$$

In case of a Dirichlet problem, the grid coordinates x^i are specified on the boundary $\partial\Omega$. A finite element discretization of the coordinates x^i using linear basis functions $\psi_n(\mathbf{u})$ yields

$$x^i(\mathbf{u}) = \sum_{n=1}^{N} x_n^i \psi_n(\mathbf{u}). \tag{4}$$

Substituting (4) into (3) and omitting the boundary term results in the nonlinear algebraic system

$$\sum_{n=1}^{N} K_{mn}(\mathbf{x})x_n^i = 0, \qquad \text{for } i = 1, 2, 3, \text{ and } m = 1, \ldots, N, \tag{5}$$

where the entries in the stiffness matrix are given by

$$K_{mn}(\mathbf{x}) = \int_\Omega \frac{\partial \psi_m}{\partial u^\alpha}\sqrt{g}g^{\alpha\beta}\frac{\partial \psi_n}{\partial u^\beta}d\mathbf{u}\,. \tag{6}$$

The stiffness matrix $K(\mathbf{x}) = [K_{mn}(\mathbf{x})]_{m,n=1,\ldots,N}$ depends on the vector of expansion coefficients $\mathbf{x}$, since the components of the element metric tensor $g^e_{\alpha\beta}$ depend on it. As is common in finite element methods, a computationally convenient representation of these components is attained by transforming to a reference element with coordinates (ξ, η, ζ). Thus, for element Ω_e with M nodes,

$$g^e_{\alpha\beta} = \sum_{m=1}^{M}\sum_{n=1}^{M}(x^e_m x^e_n + y^e_m y^e_n + z^e_m z^e_n)\frac{\psi_m}{\partial\xi^\alpha}\frac{\psi_n}{\partial\xi^\beta}\,, \tag{7}$$

where

$$x_e = \sum_{m=1}^{M} x^e_m \psi(\xi, \eta, \zeta) \tag{8}$$

is the expansion of the x-coordinate of element e in terms of the finite element basis function ψ on the reference element. The expansions of the y- and z-coordinates of element Ω_e are analogous to (8).

Thus far, the grid and its properties have been defined in terms of the current or initial grid. Specifically, the metric tensor (7) is consistent with the current grid; hence, the current

grid is actually the solution of the nonlinear algebraic system given in Equation (5). The target metric method is designed to drive this system to move the current grid to a more desirable state by modifying (7). The process of coarse-graining was introduced in [7] as a means to compute a target metric based on the given unstructured grid. Figure 2 illustrates this coarse-graining process using a two-dimensional example grid that is comprised of quadrilaterals and triangles.

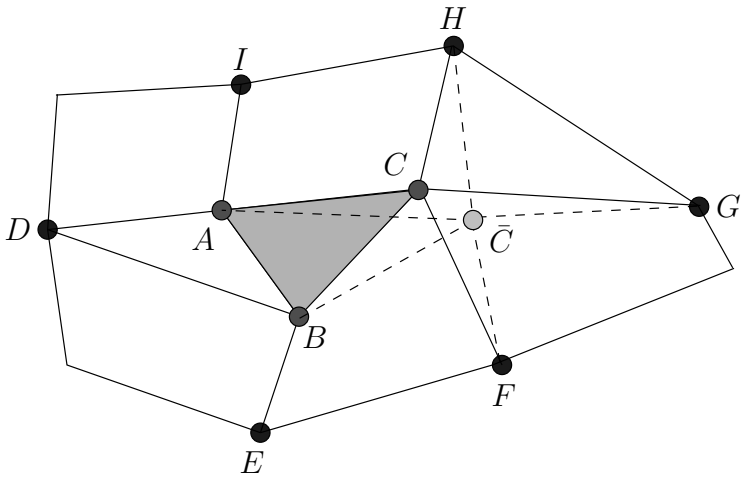

Figure 2: Coarse Graining: The target metric is based on averaged vertex positions, such as $\bar{C}$, which is the average of vertices A, B, F, G, and H.

To compute the target metric tensor for an element, averaged coordinates for its vertices are computed using their edge-connected neighbor vertices. For example, to compute the target metric tensor for the shaded triangle in Figure 2, averaged coordinates for vertices A, B, and C are computed. In this case, the averaged coordinates of C, denoted $\bar{C}$ are given by

$$\bar{C} = \frac{1}{5}(A + B + F + G + H)\,. \tag{9}$$

This coarse-grained strategy is employed to compute the target metric tensor

$$\tilde{g}^e_{\alpha\beta} = \sum_{m=1}^{M}\sum_{n=1}^{M}(\bar{x}^e_m\bar{x}^e_n + \bar{y}^e_m\bar{y}^e_n + \bar{z}^e_m\bar{z}^e_n)\frac{\psi_m}{\partial\xi^\alpha}\frac{\psi_n}{\partial\xi^\beta}\,, \tag{10}$$

where $(\bar{x}^e_m, \bar{y}^e_m, \bar{z}^e_m)$ are the coarse-grained coordinates of the vertices of element Ω_e. The arithmetic average of coordinates was inspired by Laplacian grid smoothing [29]. It is

important to note that, unlike in Laplacian smoothing where these average positions are used as the *new* location of the grid nodes, coarse-graining is used to merely formulate a "more-optimal" state for the final element for input to the Laplace-Beltrami solution method. The nodes are subsequently placed to satisfy the Laplace-Beltrami system using the target metric within the solution of Equation (5).

It is also useful to note that, although simple averaging is employed in this study, the target metric approach is a flexible methodology that offers a natural way to build a number of features of interest into the final grid. These include reference or weighting of the original grid (grid-memory), preservation of specific features of the grid, physics-based adaptivity, and/or impedance-matched smoothing along interfaces [30].

2.1 Boundary conditions

The Laplace-Beltrami method outlined above requires the specification of boundary conditions on both the nodal coordinates $\mathbf{x}$ and the target metric tensor. In [6, 7], the nodal coordinates of internal interfaces and boundaries were held fixed at their initial location. This Dirichlet condition is used in this study as well.

The boundary condition on the target metric tensor is more subtle, as it arises indirectly through its dependence on averaged coordinates (see Equation (10)), which on boundary elements include fixed boundary nodes. In [6], a *dual grid* target metric was presented, where *element ghosting* at the boundary provides a straightforward mechanism to compute the target metric over the entire domain. This approach is quite robust, and applies equally well to the *coarse-grained* target metric introduced in [7] and employed here. Element ghosting was used in [7] for the sphere, torus, and extruded horseshoe grids. Alternative boundary conditions on the target metric may be defined if ghosting is inconvenient. For example, with convex or planar boundaries it is possible to replace the averaged coordinates $(\bar{x}^e_m, \bar{y}^e_m, \bar{z}^e_m)$ in (10) at boundary nodes, by their fixed original position. However, a more reflective condition may be required near severely concave boundaries (i.e., re-entrant edges or corners).

3 The Nonlinear Solver: Jacobian-Free Newton-Krylov

This paper presents two efficient iterative solvers for the nonlinear system given in Equation (5), both are based on the Jacobian-Free Newton-Krylov (JFNK) methodology (cf. [15] and references contained therein). These new solvers offer a significant improvement in performance over the standard Newton-Krylov method used in [7].

For this presentation, it is convenient to write the full nonlinear system in the form,

$$\mathcal{F}(\mathbf{x}) = [\mathbf{F}^1(\mathbf{x}), \mathbf{F}^2(\mathbf{x}), \mathbf{F}^3(\mathbf{x})]^T = \mathbf{0}\,, \tag{11}$$

where the components of $\mathcal{F}(\mathbf{x})$ are taken directly from Equation (5),

$$\mathbf{F}^i(\mathbf{x}) = \sum_{n=1}^{N} K_{mn}(\mathbf{x}^i) x_n^i = 0 \qquad \text{for } i = 1, 2, 3\,. \tag{12}$$

Hence $\mathbf{F}^i(\mathbf{x}) : \Re^{3N} \rightarrow \Re^N$, where N is again the number of nodes in the grid. The Jacobian of this system is a $3N \times 3N$ sparse matrix,

$$\mathcal{J}(\mathbf{x}) = \frac{\partial \mathcal{F}(\mathbf{x})}{\partial \mathbf{x}}\,. \tag{13}$$

If one orders the unknowns first by coordinate and then by node index,

$$\mathbf{x} = [x_1^1, \ldots, x_N^1, x_1^2, \ldots, x_N^2, x_1^3, \ldots, x_N^3]^T\,,$$

then the Jacobian may be naturally written as a 3×3 block matrix,

$$\mathcal{J}^{(i,j)} = \frac{\partial F^i}{\partial x^j} \qquad i = 1, 2, 3 \quad \text{and} \quad j = 1, 2, 3. \tag{14}$$

Each block in the above is an $N \times N$ matrix with entries given by,

$$\mathcal{J}^{(i,j)}_{(m,n)} = \frac{\partial F^i_m}{\partial x^j_n} \qquad m = 1, \ldots, N \text{ and } n = 1, \ldots, N\,. \tag{15}$$

Given the Jacobian in this form, it is straightforward to express the Newton iteration,

$$\mathbf{x}^{(k+1)} \longleftarrow \mathbf{x}^{(k)} + \delta\mathbf{x}^{(k)}, \text{ and} \tag{16}$$

$$\mathcal{J}(\mathbf{x}^{(k)})\, \delta\mathbf{x}^{(k)} = -\,\mathcal{F}(\mathbf{x}^{(k)}), \tag{17}$$

where the superscript k denotes the iteration count of the Newton iteration. Using Newton's method as shown here amounts to implementing a sequence of steps:

1. Form the Jacobian matrix.

2. Solve the sparse linear system (17) to obtain $\delta\mathbf{x}^{(k)}$.

3. Apply this update (16) to obtain the next iteration of the solution state vector, $\mathbf{x}^{(k+1)}$.

These steps are usually challenging to perform in an actual grid generation application. Formation of the Jacobian matrix (*i.e.,* the tangent stiffness matrix), is very costly in terms of both memory and computer time. Secondly, solving the linear system that arises internal to Newton's method is typically a daunting problem in its own right. This system is usually very large, sparse, and ill-conditioned in a typical application; making its efficient solution quite challenging.

3.1 Jacobian-Free Newton-Krylov Steps

The nonlinear solver described in [7] computes the Jacobian matrix each Newton iteration. Even for moderately-large grids, the cost of forming the Jacobian is high and typically dominates the computation, making this grid smoothing algorithm impractical for most situations. Fortunately, Krylov iterative solvers such as the generalized minimum residual (GMRES) algorithm [31], which is used here to solve the Jacobian system, do not require the Jacobian matrix itself but simply the action of the Jacobian matrix on a vector. Approximating this matrix-vector product by differencing, which requires two nonlinear function evaluations, is the basis of the Jacobian-Free Newton-Krylov (JFNK) method. Specifically, to evaluate the matrix-vector product $\mathcal{J}(\mathbf{x}^{(k)})\mathbf{v}$, a finite-difference approach,

$$\mathcal{J}(\mathbf{x}^{(k)})\mathbf{v} \approx \frac{\mathcal{F}(\mathbf{x}^{(k)} + \varepsilon\mathbf{v}) - \mathcal{F}(\mathbf{x}^{(k)})}{\varepsilon}, \tag{18}$$

is commonly used [15, 32]. Here, ε is chosen to avoid problems with machine precision,

$$\varepsilon = \frac{\sqrt{(1 + \|\mathbf{u}\|_2)\hat{\varepsilon}}}{\|\mathbf{v}\|_2}, \tag{19}$$

with $\hat{\varepsilon} = 10^{-12}$.

Using this Jacobian-free approach, the dominant cost of the algorithm shifts from evaluating the Jacobian to the solution of the linear system. Indeed, the solution cost of GMRES for elliptic problems scales quadratically with the number of unknowns in the grid, unless effective preconditioning is used [33].

3.2 Multigrid Preconditioning of GMRES

To develop efficient preconditioners for the Krylov solution of the Jacobian system requires an examination of the Jacobian structure and properties. Linearizing the strong form of the continuous model (2) about a smoothed grid $\mathbf{x} = \mathbf{x}^*$, one obtains

$$\mathcal{J} = \mathcal{D} + \mathcal{C} \tag{20}$$

where $\mathcal{D} = diag\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ is the contribution to the diagonal arising from the elliptic component, and $\mathcal{C}$ represents the lower order terms. Specifically,

$$\mathcal{D}_i = \nabla_u \cdot \mathcal{G}(\mathbf{x}^*)\nabla_u \tag{21}$$

where $\mathcal{G}(\mathbf{x}^*) = g(\mathbf{x}^*)g^{\alpha\beta}(\mathbf{x}^*)$, and the lower order terms are given by

$$\mathcal{C}_{i,j} = \nabla_u \cdot \frac{\partial \mathcal{G}}{\partial x_j} \nabla_u x_i \Big|_{\mathbf{x}=\mathbf{x}^*} . \tag{22}$$

The diagonal terms $\mathcal{D}_i$ are general diffusion operators with a discontinuous full tensor diffusion coefficient corresponding to the metric tensor. In contrast, while the entries of $\mathcal{C}$ are not derivative operators, they contain derivatives of both the solution and the metric tensor. Clearly, in a typical application, the signs and magnitude of these terms may vary dramatically. Lastly, $\mathcal{C}$ is not symmetric.

There are a number of significant challenges for the efficient and scalable solution of the full Jacobian linear system. First, the solution involves a coupled system of three partial differential equations that are likely to be strongly elliptic over part or all of the domain. This implies that, in order for the computational cost of the solver to scale linearly with the number of unknowns (i.e., provide optimal algorithmic scaling), the preconditioner must be a multilevel algorithm. Also, to accommodate unstructured grids, the preconditioner must be based solely on the fine-scale discrete system. The Ruge–Stüben algebraic multigrid (AMG) method [23, 24] and its various descendants meet both of these requirements. Unfortunately, most of the heuristics that form the basis of these AMG algorithms are motivated by scalar elliptic equations (such as the blocks $\mathcal{D}_i$ given in Equation (21)) as opposed to a coupled system of PDEs. Secondly, the full Jacobian is nonsymmetric. The theoretical

foundation of AMG methods for nonsymmetric matrices is much weaker than it is for symmetric matrices. Finally, but perhaps most importantly, the local character of the equations may shift from well-posed elliptic PDEs to a more "Helmholtz-like" system, especially in the neighborhood of highly-curved boundaries. Nevertheless, despite these weaknesses in the theoretical foundation, this study seeks to demonstrate that AMG is capable of solving the full Jacobian linear system with only moderate degradation in performance and scaling. As an aside, it is important to note that robust and efficient MPI-based parallel versions of AMG are available (e.g., [34, 27, 28]) and scalability to large numbers of processors has been demonstrated [35]. Although parallel scalability is not directly addressed in this study, meeting this requirement is critical for the extension of the Laplace-Beltrami method to larger applications.

Given the high cost and the challenges of computing the Jacobian, this study explores two JFNK solution algorithms. Both of these will employ the finite difference technique given in Equation (18) to approximate the matrix-vector product required by the restarted GMRES iteration. Both methods use a restart length of 50 (i.e., GMRES(50)) and are converged to a relative tolerance of 3×10^{-2} in the two-norm of the residual. In both cases, right preconditioning is employed, and AMG V-cycles are used to approximately invert the preconditioner, $\mathcal{P}$, to a relative tolerance of 1×10^{-2} in the two-norm of the residual. These two approaches differ only in the preconditioning approach used for this linear system.

The first preconditioning method is based on forming a full Jacobian for use as a preconditioner. Instead of recomputing this Jacobian in each nonlinear iteration, however, this approach will use the initial Jacobian for all nonlinear steps. Clearly, this idea does not eliminate the cost of forming the Jacobian; it seeks to amortize this cost by forming $\mathcal{J}$ only once and using this as the preconditioner, $\mathcal{P}_{frozen} = \mathcal{J}(\mathbf{x}^{(1)})$, for all linear iterations in all Newton steps. This technique is termed the *frozen* preconditioner method. One critical aspect of this frozen Jacobian approach is that the Newton iteration is not technically altered by the frozen approximation; only the efficiency of the linear solution algorithm is affected (cf. [15]). Specifically, in the first Newton step, $\mathcal{P}_{frozen}$ is the ideal preconditioner, while its efficacy will decrease somewhat as the Newton iteration proceeds. In many applications, the grid movement is relatively small, as is the number of Newton steps required to converge

the system. For these problems, the frozen preconditioner remains effective for the entire solution process. Further, the setup phase of AMG is rather costly, as it creates a hierarchy of grids and discrete operators. Thus, a desirable feature of the frozen preconditioner is that the setup phase is only performed once. Unfortunately, as noted above, a strong theoretical foundation for the application of AMG to this class of problems is lacking. Indeed, in the examples that follow, convergence rates ranging from approximately 0.15 on small problems to 0.90 on larger, more challenging grids (4) are observed. While these results are still a significant distance away from the bound at 1, they further reinforce that AMG performs significantly better for scalar diffusion applications.

The second (and *fastest*) JFNK method improves on both weaknesses present in the frozen preconditioner. This method is based on preconditioning using a Picard linearization $\mathcal{P}_{Picard} = \mathcal{D} = diag\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$. In this case, the connection to the discrete linear system is given by

$$\mathcal{D}_i(\mathbf{x}^{(k)}) = \left.\frac{\partial \mathbf{F}^i(x^i, \mathcal{G}(\mathbf{x}^{(k)}))}{\partial x^i}\right|_{\mathbf{x}=\mathbf{x}^{(k)}} = K(\mathbf{x}^{(k)}). \tag{23}$$

Here, the dependence of $\mathbf{F}^i(\mathbf{x})$ is explicitly decomposed into a dependence on the coordinate x^i and the target metric $\mathcal{G}(\mathbf{x}^k)$. This formulation highlights that the Picard approximation freezes the dependence on the target metric at the current iterate before taking the derivative, and in so doing, removes the coupling across coordinates. Thus, the cost of forming the Picard matrix is solely that of evaluating the finite element stiffness matrix at the value $\mathbf{x}^{(k)}$. Furthermore, this approximation to the Jacobian results in a symmetric, positive definite matrix (i.e., each $\mathcal{D}_i$ is a scalar diffusion operator) that is ideally suited for AMG. Indeed, one would expect excellent AMG performance with this result; in testing convergence rates ranging from approximately 0.10 (for small problems), to roughly 0.30 for larger grids were observed.

One expects to see fewer AMG iterations per GMRES iteration with the Picard preconditioner, $\mathcal{P}_{Picard}$, than with the frozen preconditioner, $\mathcal{P}_{frozen}$. Unfortunately, the Picard preconditioner is not generally as effective as $\mathcal{P}_{frozen}$; a small increase in the number of GMRES iterations per Newton step is anticipated. The results presented in the next section illustrate this behavior, but also confirm that the Picard preconditioner is significantly

faster albeit not quite as effective. Of final note, the AMG setup phase is performed only once for the first GMRES iteration in each Newton step, and then the setup is reused without modification for subsequent GMRES iterations.

4 Numerical Examples

In this section, the two proposed JFNK solvers are compared with the Newton-Krylov solver used in the original Laplace-Beltrami grid generator developed by Hansen et al.[7]. This comparison will involve some of the examples considered there, followed by testing on larger problems to explore the scalability of the solvers. Section 4.1 examines solver performance on a representative sample of three grids from [7]. Then, in Section 4.2, a sequence of structured and semi-structured grids are used to examine the algorithmic scaling of the three solvers with increasing grid size.

In all of the results presented, nonlinear convergence is defined by performing Newton iterations until a relative convergence criteria in the two norm of the nonlinear residual,

$$TOL = \frac{||\mathcal{F}(\mathbf{x}^{(k)})||_2}{||\mathcal{F}(\mathbf{x}^{(0)})||_2} \leq 10^{-6}\,, \tag{24}$$

is satisfied. The initial state of the grid is used as the initial guess.

The implementation of these algorithms is in the form of an ANSI-C++ code that was derived from the original code used in [7], which uses existing C++ and Fortran (77/95) based solver packages. The nonlinear solver employed in [7] uses the Block Preconditioning Toolkit (BPKIT) [36] as a linear solver and preconditioner. For the JFNK solvers developed here, the restarted GMRES solver from the PCG software package [26] is used, along with Ruge's AMG1r6 implementation of the Ruge-Stüben AMG [24].

To simplify the discussion of these results, each solver will be denoted by the preconditioner that it employs. Specifically, the NK solver from the original grid-smoothing research [7] is termed *BILU(1)*, the JFNK solver that uses the frozen Jacobian preconditioner is called *frozen*, and the JFNK solver that uses the Picard linearization as the preconditioner is denoted *Picard*. In the list below, a summary of the arrangement of components and relevant parameter values that characterize each of these three solvers, is outlined.

1. *BILU(1)*: This is the Newton-Krylov solver used in the original development by Hansen et al.[7]. It forms the full Jacobian each Newton step and uses the restarted flexible-GMRES(50) Krylov method from BPKIT. This Krylov iteration is converged to a relative tolerance of 1×10^{-2} in the two norm of the residual. The preconditioner used is Block ILU(1), with 16 blocks, followed by two passes of Symmetric Successive Over-Relaxation (SSOR) (also native to BPKIT).

2. *frozen:* This is the first JFNK based solver considered, where restarted GMRES(50) is used as the Krylov method, and Equation (18) approximates the required matrix-vector multiplication. The preconditioner is the Jacobian formed for the first iteration, $\mathcal{P}_{frozen} = \mathcal{J}(\mathbf{x}^{(1)})$, which is then reused for all subsequent linear system solutions. The preconditioner is approximately inverted with AMG V-cycles to a relative tolerance of 3×10^{-2} in the two norm of the residual. Similarly, the restarted GMRES(50) iteration is converged to a relative tolerance of 1×10^{-2}.

3. *Picard:* This is the second JFNK based solver, where restarted GMRES(50) is used as the Krylov method, and Equation (18) again approximates the required matrix-vector multiplication. The preconditioner is a Picard linearization, $\mathcal{P}_{Picard} = \mathcal{D} = diag\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$, which is simply the stiffness matrix evaluated at the current iterate. The preconditioner is approximately inverted with AMG V-cycles to a relative tolerance of 3×10^{-2} in the two norm of the residual. Similarly, the restarted GMRES(50) iteration is converged to a relative tolerance of 1×10^{-2}.

The majority of computations in the solver study which follows were run on a Pentium 4 Xeon at 2.8GHz, using the Intel 8.0 compiler suite. The only exception is the scaling study on the sequence of corner grids depicted in Figure 7, which was run on an AMD Opteron system at 2GHz, using the Portland Group 6.0 compiler suite.

4.1 A Suite of Grids

In the first part of the solver evaluation, three grids, namely the cylinder, extruded horseshoe, and turbine grids, from the original three-dimensional grid-smoothing paper [7] are considered. The extruded horseshoe, shown in Figure 3, is a two-dimensional horseshoe grid

extruded into the third dimension along a quarter circle (topologically a quarter of a torus), and is composed of hexahedral elements. The turbine, shown in Figure 4, is an unstructured tetrahedral grid representing a turbine nozzle; this grid is courtesy of the *amira*$^{\text{TM}}$software package by TGS, Inc. (www.tgs.com). In both Figures, cross-sections through the original and smoothed grids are shown.

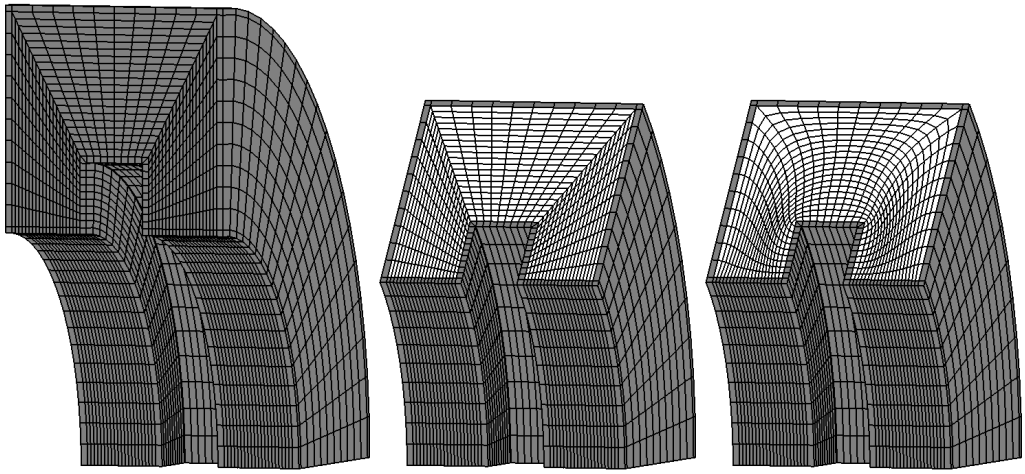

Figure 3: The extruded horseshoe grid (see [7]). From left to right: the full initial grid, a cross-section of the initial grid, and a cross-section of the final grid.

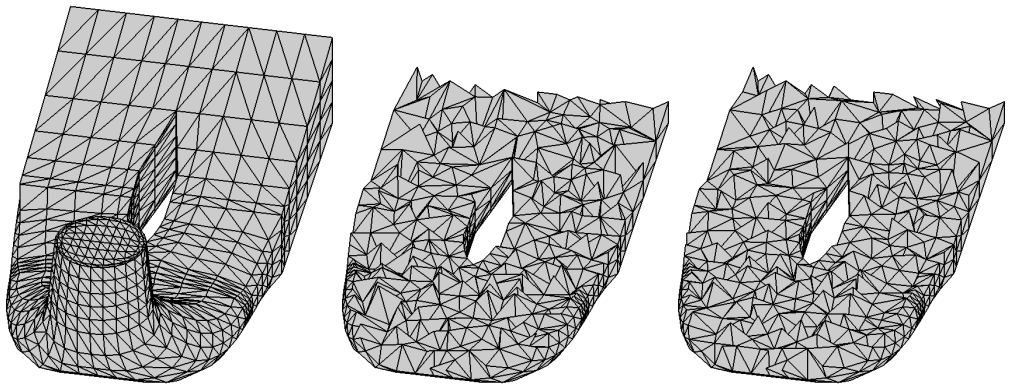

Figure 4: The turbine grid (see [7]). From left to right: the full initial grid, a cross-section of the initial grid, and a cross-section of the final.

All three of the solvers outlined previously in Section 4 (i.e., BILU(1), frozen, and

Picard) were studied on all three of these grids. The results of this study are presented in Table 1 in the form of the number of iterations required to achieve convergence, where the execution timings and speedup factors are shown in Table 2. In the case of the iteration counts, it is apparent that the cylinder and turbine grids are too small to differentiate the iterative behavior of the three solvers. In these cases, all three solvers have a very low GMRES/Newton count (around two), and both the frozen and Picard solvers have a very low AMG V-cycles/GMRES count (also around two). However, even for these small grids, the solution times are significantly different. Both of the JFNK methods show improvement beyond the BILU(1) method; the frozen solver is approximately 4.5 times faster and the Picard solver ranges from 37 to 51 times faster.

The extruded horseshoe grid is more than an order of magnitude larger than the other grids; as a result the grid is large enough that differences between the solution approaches become more apparent. Specifically, it becomes clear that using AMG to approximately invert $\mathcal{P}_{frozen}$ is the best preconditioner of the three studied, showing an average of 7.50 GMRES iterations per Newton step compared with 11.83 and 20.14 for the BILU(1) and Picard solver, respectively. However, $\mathcal{P}_{frozen}$ is a larger and more complicated matrix than the Picard preconditioner $\mathcal{P}_{Picard}$, which is a set of three decoupled scalar diffusion problems. Hence, the average number of AMG V-cycles per GMRES iteration is significantly higher for the frozen preconditioner, 19.53, compared with 6.11 for the Picard preconditioner. In addition, the speedup for the extruded horseshoe grid is excellent; resulting in factors of 4.0 and 30.9 for the frozen and Picard solvers respectively.

Table 1: Average iteration counts, GMRES/Newton and AMG V-cycles/GMRES, are shown for the three solvers introduced above, for each of three grids from [7].

Grid Name	Size ($3N$)	GMRES/Newton			AMG V-cycles/GMRES	
		BILU(1)	Frozen	Picard	Frozen	Picard
cylinder	2202	2.80	1.75	2.00	1.86	2.25
turbine	1926	1.80	1.75	2.00	1.71	2.00
horseshoe	24795	11.83	7.50	20.14	19.53	6.11

Table 2: The overall solution time (seconds) for the three solvers, and the speedup factors for the frozen and Picard solvers relative to the original BILU(1) solver, are shown for each of three grids from [7].

Grid Name	Size ($3N$)	Solution Time (s)			Speedup	
		BILU(1)	Frozen	Picard	Frozen	Picard
cylinder	2202	86.6	18.9	1.69	4.6	51.2
turbine	1926	58.1	13.0	1.56	4.5	37.3
horseshoe	24795	2572.2	648.1	83.2	4.0	30.9

4.2 Scaling Studies

The second phase of this study examines the algorithmic scaling of the three solvers and their various components over two sequences of grids. The first example set involves a sequence of progressively-finer grids with randomly perturbed vertices on a cubic domain. The second set involves progressively-finer grids in a more-complex "corner" grid configuration.

4.2.1 Randomly Perturbed Structured Grids on a Cube

The first grid sequence studied is a logically structured hexahedral grid on a cubic domain. Each grid in the sequence is created by first generating a uniform cubic grid with spacing h, and then randomly perturbing each vertex within a cubic neighborhood of the vertex, $[-0.2h, 0.2h]^3$. This random perturbation is constrained for vertices that lie on the surface of the cube (e.g., on the planar boundaries of the cube, vertices are perturbed within each plane). Figure 5 shows a grid of $8 \times 8 \times 8$ elements, as well as a cross-section view of this grid and the corresponding smoothed grid. The main feature of this Figure is that at vertices away from the boundary of the cube, the memory of the distorted grid is lost and the vertices relax back to the unperturbed uniform orthogonal grid. This is a consequence of how the *target metric* is defined in conjunction with the grid topology. If a greater attraction to, or memory of, specific features in the initial grid is desired, modifications to the target metric may be introduced to provide the desired behavior. As was discussed

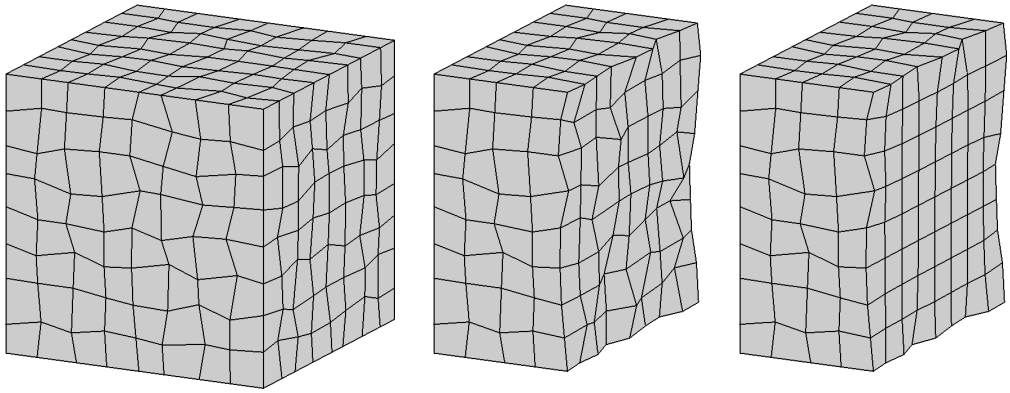

Figure 5: An $8 \times 8 \times 8$ logically square grid inside a unit cube. From left to right: the full initial grid, a cross-section of the initial grid, and a cross section of the final grid. The initial grid was generated by randomly perturbing the vertices of a uniform cubic grid. The sequence of grids in the scaling study have 4, 8, 16, and 32 elements in each coordinate direction.

in Section 2.1, Dirichlet boundary conditions are imposed on the vertex coordinates that rest on the boundary. In this example, the suggested modified form of the coarse-graining equation (10) was used to obtain the target metric for the boundary elements, which acts to indirectly set the boundary condition for the boundary metric.

All three of the solvers discussed in Section 4 (i.e., BILU(1), frozen, and Picard), were studied on this sequence of perturbed grids. The iteration counts are shown in Table 3, and the execution times are plotted in Figure 6. Once again, the iteration count confirms that the frozen Jacobian is the most effective preconditioner, showing only a modest growth in the average number of GMRES iterations per Newton step for the first three grids. The marked increase in iteration count for the largest grid, from 2.8 to 5, is noteworthy, but it is not a conclusive indicator of scaling behavior. In contrast, for both the BILU(1) and Picard preconditioners, the average number of GMRES iterations per Newton step increases steadily with grid size, reaching nearly identical values of approximately 28 for the largest grid. Most importantly, the rate of growth is significantly higher for BILU(1) (≈ 2.5), compared with Picard (≈ 1.7). Finally, note that the different characteristics of the full Jacobian matrix (coupled system of three PDEs) and the Picard preconditioner

Table 3: Average iteration counts, GMRES/Newton and AMG V-cycles/GMRES, are shown for the three solvers introduced above, for the sequence of square grids with randomly perturbed vertices (see Figure 5).

Dimensions (Elements)	Size ($3N$)	GMRES/Newton			AMG V-cycles/GMRES	
		BILU(1)	frozen	Picard	frozen	Picard
$4 \times 4 \times 4$	81	2.00	1.80	5.60	2.00	1.93
$8 \times 8 \times 8$	1029	4.20	2.50	9.60	2.90	2.40
$16 \times 16 \times 16$	10125	11.17	2.80	17.33	7.64	3.00
$32 \times 32 \times 32$	89373	28.29	5.00	28.14	45.67	3.91

(three decoupled scalar diffusion equations) are reflected in the average number of AMG V-cycles per GMRES iteration, which reach a maximum of 45.67 and 3.91, respectively, for the largest grid. Moreover, the growth in the number of AMG V-cycles per GMRES iteration is significant and superlinear for solving the Jacobian system, and nearly flat for the Picard linearization.

The differences in the performance of these components translates to a significant speedup relative to the BILU(1) solver for the Picard solver, a factor of 26.38 on the largest grid, and a modest speedup for the frozen Jacobian, a factor of 3.63 on the largest grid. Note that the speedup for the Picard solver reached a high of 52.1 on the second smallest grid, and then declined to 45.40 and finally to 26.38. This decline in speedup is contrary to what is expected from the iteration count data. This effect is due to the computational cost of the AMG solver. The AMG1r6 algorithm exhibits suboptimal scaling in its setup phase and increasing computational complexity of the constructed hierarchy of components for large three-dimensional problems. Indeed, the compromise between optimal scaling of the setup phase and the potential loss of optimality in the solve phase of algebraic multigrid methods remains an active area of research. It is expected that using a more advanced AMG solver with options for aggressive coarsening (e.g., [35]) would result in speedup factors that are closer to the expected performance suggested by the iteration counts.

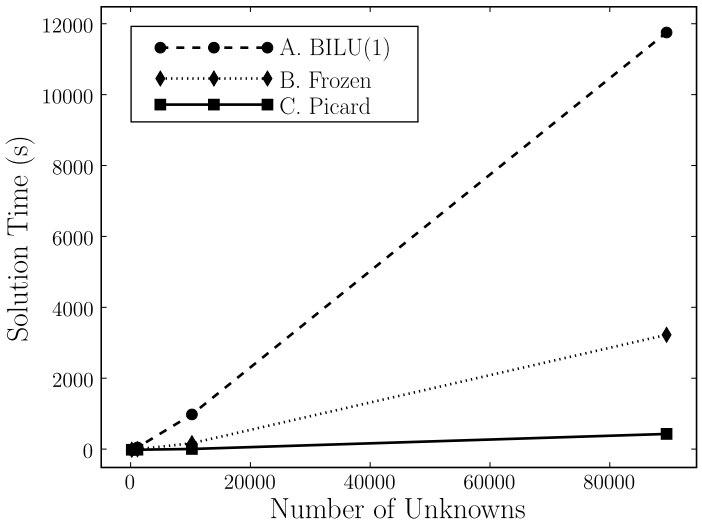

Figure 6: Solution times of all three solvers for the sequence of square grids with randomly perturbed vertices (see Figure 5) plotted against the number of unknowns.

4.2.2 Uniform Refinement of a Corner Grid

The second sequence of grids was created using the CUBIT Mesh Generation Toolkit [37] and is based on uniform refinement of a corner gridded with hexahedral elements. Figure 7 shows a cross-section view of the third grid in this sequence, which is contains 12,288 hexahedral elements. These corner grids pose challenges similar to the extruded horseshoe discussed above, including a triple-point junction along the reentrant corner. High curvature features of this sort are known to cause traditional Laplacian smoothing to fail if convergence is attempted, and also challenge the nonlinear solver in the Laplace-Beltrami target-metric methodology. The right view of Figure 7 shows that the smoothed grid in the neighborhood of the triple point should be well balanced when the grid generator has converged.

Dirichlet boundary conditions are prescribed on the vertices, but in this example element ghosting is employed to calculate the boundary metric. All three of the solvers listed in Section 4 (i.e., BILU(1), frozen, and Picard), were studied on this sequence of corner grids. Iteration counts were collected in Table 4 and the timings are plotted in Figure 8. The iteration counts follow the same trends observed in the previous examples. However,

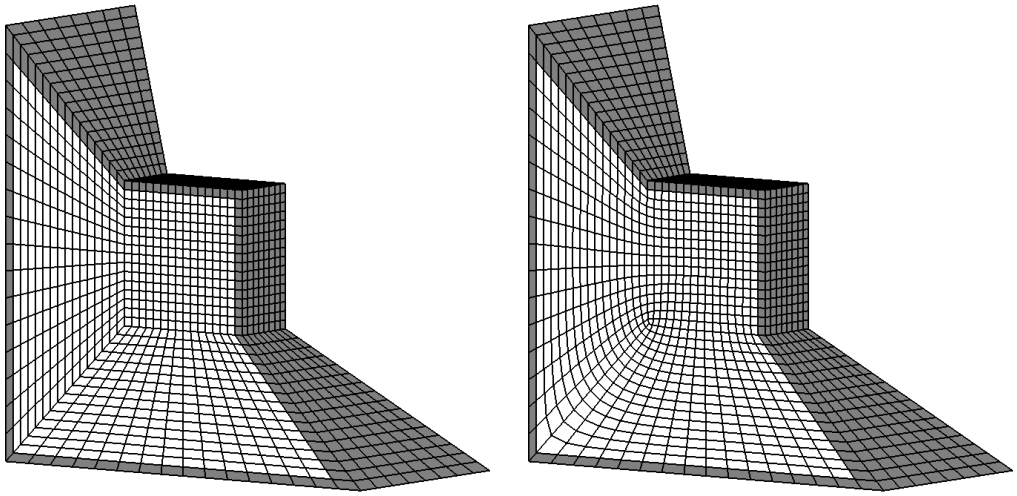

Figure 7: A grid consisting of 12288 hexahedrons. Shown are a cut through the initial grid (on the left) and a cut through the final grid to expose their interior. This grid is part of a sequence of successively refined grids with 192, 1536, 12288, and 98304 hexahedrons.

the scaling in these trends now shows that the Picard solver is not only the fastest, but scales significantly better than the either of the frozen or the BILU(1) solvers. Specifically, the average number of GMRES iterations per Newton step increases by over a factor of 5.5 from the third to fourth grid in the study for the BILU(1) solver, while it only increases by a factor of 1.8 for the Picard solver. Moreover, where the apparent scaling of the frozen Jacobian preconditioner was inconclusive before, here the number of GMRES iterations per Newton step is increasing by approximately 1.6 with each increase in grid size. Thus, the scaling in terms of GMRES/Newton is very similar for the frozen and the Picard preconditioners. However, the scaling of the AMG V-cycles per GMRES iteration remains significantly better, indeed nearly flat, for the Picard solver. This is in comparison with the frozen solver, which increases at a rate of approximately 2.5 with each grid.

The overall solution time is plotted in Figure 8, where the differences in the solver components result in the Picard-based JFNK solver being the fastest and most scalable solver. The speedup factors for both the Picard and frozen solvers are similar to those seen in the previous scaling study, with the Picard solver attaining a factor of 26.30 on the largest grid, and a corresponding speedup for the frozen solver of 3.49. Also, the speedup

Table 4: Average iteration counts, GMRES/Newton and AMG V-cycles/GMRES, are shown for the three solvers introduced above, for the sequence of triple point corner grids (see Figure 7).

Grid Size ($3N$)	GMRES/Newton			AMG V-cycles/GMRES	
	BILU(1)	frozen	Picard	frozen	Picard
57	2.00	1.50	3.75	1.83	1.80
1905	6.00	3.40	9.33	4.35	2.59
24609	17.17	5.33	17.00	11.78	2.98
242817	95.00	8.71	24.38	29.53	3.91

factor of the Picard solver follows the same behavior seen in the previous study, peaking at the second grid, and then declining, despite the obvious superior scaling of the iteration counts. As noted before this is an issue with the algorithmic scaling of the AMG1r6 setup phase and the complexity of the resulting hierarchy of components. These issues are better controlled with the aggressive coarsening algorithms present in more advanced AMG solvers [35].

5 Conclusions

This paper presents two efficient Jacobian-Free Newton-Krylov (JFNK) solvers for the Laplace-Beltrami grid generation system of equations. Although the Laplace-Beltrami approach is effective as a grid generation method for complex structured and unstructured applications with highly-curved boundaries [6, 7], its application to large three-dimensional unstructured grids has been limited by the lack of effective solvers. This paper presents JFNK solvers that use a matrix-free matrix-vector product for the underlying GMRES iterations, and preconditioners that are readily treated with algebraic multigrid solvers (AMG). The two JFNK solvers differ only in the preconditioner, with the first using a frozen Jacobian throughout the solve, and the second defined by the Picard linearization. Numerical results contrast the performance of these solvers with the standard Newton-Krylov method

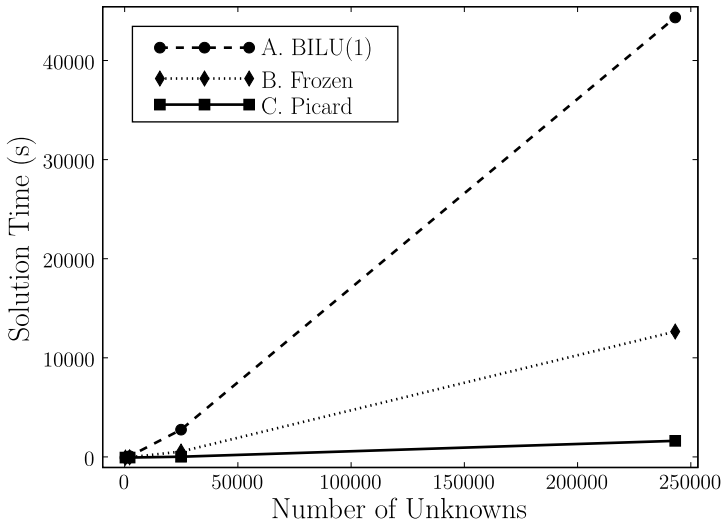

Figure 8: Solution times of all three solvers for the sequence of triple point corner grids (see Figure 7), plotted against the number of unknowns.

from [7], which used a block ILU(1) preconditioner. A significant performance gain is achieved on all grids considered, and the use of AMG leads to improved scaling with grid size. The overall speedup factors on the largest grids considered here are approximately 4 for the frozen Jacobian preconditioner, and 26 for the Picard linearization. These results, in conjunction with the availability of parallel implementations of all solver components, confirms that the new JFNK solvers provide the computational efficiency and algorithmic scaling necessary to use Laplace-Beltrami grid smoothing in large-scale applications.

Future work includes exploring a more advanced AMG solver with options for aggressive coarsening (e.g., [35]) to resolve the discrepancy between the speedup factors that were obtained and the algorithmic scaling observed in the iteration counts. In addition, an investigation of more advanced preconditioners that further improve the results obtained by Picard linearization is warranted. This study might proceed by capturing some of the lower-order terms from the true Jacobian. The challenge here is to incorporate more information from the Jacobian without approaching the complexity of forming the full Jacobian, and without negatively impacting the performance of the AMG solution algorithm. Parallel implementations of the GMRES and AMG algorithms are readily available, hence, enabling the study

of the parallel performance of the new solvers on large-scale parallel architectures. Such a study is required to increase the application relevance of the Laplace-Beltrami method. Finally, the potential benefit of enhancing the target-metric tensor specification to include new capabilities; such as the the preservation of grid features, smoothing along interfaces and boundaries, is clearly indicated.

Acknowledgments

This work was performed under the auspices of the US Department of Energy by Los Alamos National Laboratory under Contract W-7405-ENG-36 (LA-UR-06-3998). We wish to thank Andrew Zardecki of the Los Alamos National Laboratory (retired) for providing the code and grid examples that were used in [2] and that we extended in this study.

References

[1] J. F. Thompson, Z. U. A. Warsi, C. W. Mastin, Numerical Grid Generation: Foundations and Applications, Elsevier, New York, 1985.

[2] G. A. Hansen, R. W. Douglass, A. Zardecki, Mesh enhancement: selected elliptic methods, foundations and applications, Imperial College Press, London, 2005.

[3] A. Khamayseh, C. W. Mastin, Computational conformal mapping for surface grid generation, J. Comput. Phys. 123 (1996) 394–401.

[4] V. D. Liseikin, Grid Generation Methods, Springer, Berlin, Heidelberg, New York, 1999.

[5] V. D. Liseikin, A Computational Differential Geometry Approach to Grid Generation, Springer, Berlin, Heidelberg, New York, 2004.

[6] G. Hansen, A. Zardecki, D. Greening, R. Bos, A finite element method for unstructured grid smoothing, J. Comput. Phys. 194 (2) (2004) 611–631.

[7] G. Hansen, A. Zardecki, D. Greening, R. Bos, A finite element method for three-dimensional unstructured grid smoothing, J. Comput. Phys. 202 (1) (2005) 281–297.

[8] J. U. Brackbill, J. S. Saltzman, Adaptive zoning for singular problems in two dimensions, J. Comput. Phys. 46 (1982) 342–368.

[9] J. U. Brackbill, An adaptive grid with directional control, J. Comput. Phys. 108 (1993) 38–50.

[10] W. Cao, W. Huang, R. D. Russell, An r-adaptive finite element method based upon moving mesh PDEs, J. Comput. Phys. 149 (1999) 221–244.

[11] L. Chacon, G. Lapenta, A fully implicit, nonlinear adaptive grid strategy, J. Comput. Phys. 212 (2006) 703–717.

[12] W. H. Frey, D. A. Field, Mesh relaxation. a new technique for improving triangulations, International Journal for Numerical Methods in Engineering 31 (6) (1991) 1121–1133.

[13] L. Freitag, P. Knupp, Tetrahedral element shape optimization via the Jacobian determinant and condition number, in: Proceedings of the 8th International Meshing Roundtable, Lake Tahoe, California, 1999, pp. 247–258.

[14] L. Freitag, P. Plassmann, Local optimization-based simplicial mesh untangling and improvement, International Journal of Numerical Methods in Engineering 49 (1–2) (2000) 109–125.

[15] D. A. Knoll, D. E. Keyes, Jacobian-free Newton-Krylov methods: A survey of approaches and applications, J. Comput. Phys. 193 (2) (2004) 357–397.

[16] C. T. Kelly, Iterative methods for linear and nonlinear equations, Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1995.

[17] D. A. Knoll, W. J. Rider, G. L. Olson, An efficient nonlinear solution method for non-equilibrium radiation diffusion, J. Quant. Spect. Rad. Tran. 63 (1) (1999) 15–29.

[18] W. J. Rider, D. A. Knoll, G. L. Olson, A multigrid Newton-Krylov method for multimaterial equilibrium radiation diffusion, J. Comput. Phys. 152 (1) (1999) 164–191.

[19] W. L. Briggs, V. E. Henson, S. F. McCormick, A Multigrid Tutorial, SIAM Books, Philadelphia, 2000, second edition.

[20] U. Trottenberg, C. Oosterlee, A. Schüller, Multigrid, Academic Press, 2001.

[21] A. L. Codd, T. A. Manteuffel, S. F. McCormick, J. W. Ruge, Multilevel first-order system least squares for elliptic grid generation, SIAM J. Numer. Anal. 41 (6) (2003) 2210–2232.

[22] S. P. Spekreijse, Elliptic grid generation based on Laplace equations and algebraic transformations, J. Comput. Phys. 118 (1995) 38–61.

[23] A. Brandt, S. F. McCormick, J. W. Ruge, Algebraic multigrid (AMG) for sparse matrix equations, in: D. J. Evans (Ed.), Sparsity and its Applications, Cambridge University Press, Cambridge, 1984, pp. 257–284.

[24] J. W. Ruge, K. Stüben, Algebraic multigrid (AMG), in: S. F. McCormick (Ed.), Multigrid Methods, Vol. 3 of Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1987, pp. 73–130.

[25] K. Stüben, A review of algebraic multigrid., J. Comp. Appl. Math. 128 (1/2) (2001) 281–309.

[26] W. D. Joubert, G. F. Carey, N. A. Berner, A. Kalhan, H. Kohli, A. Lorber, R. T. Mclay, Y. Schen, PCG Reference Manual: A Package for the Iterative Solution of Large Sparse Linear Systems on Parallel Computers Version 1.0, Los Alamos National Laboratory, Los Alamos, NM (1994).

[27] Center for Applied Scientific Computing (CASC), Lawrence Livermore National Laboratory, Hypre high performance preconditioners User's Manual, software version 1.9.0b Edition (February 2005).

[28] W. D. Joubert, LAMG: Los Alamos Algebraic Multigrid Code User's Manual, A Parallel Software Library for the Solution of Systems of Linear Equations Using Algebraic Multigrid Methods, LA-UR 05-4041 Edition (June 2005).

[29] P. Knupp, S. Steinberg, The Fundamentals of Grid Generation, CRC Press, Boca Raton, FL, 1993.

[30] A. Khamayseh, G. Hansen, Quasi-orthogonal grids with impedance matching, SIAM J. Sci. Comput. 22 (4) (2000) 1220–1237.

[31] Y. Saad, Iterative Methods for Sparse Linear Systems, The PWS Series in Computer Science, PWS Publishing Company, Boston, MA, 1995.

[32] M. Pernice, H. F. Walker, NITSOL: a Newton iterative solver for nonlinear systems, SIAM J. Sci. Comp. 19 (1) (1998) 302–318.

[33] D. A. Knoll, W. J. Rider, Multigrid preconditioned Newton-Krylov method, SIAM J. Sci. Comp. 21 (2) (1999) 691–710.

[34] M. Sala, M. W. Gee, J. J. Hu, R. S. Tuminaro, Ml 4.0 smoothed aggregation user's guide, Tech. Rep. SAND2004-4819, Computational Math & Algorithms Department, Sandia National Laboratories, Albuquerque, NM (August 2005).

[35] W. Joubert, J. Cullum, Scalable algebraic multigrid on 3500 processors, Tech. Rep. LA-UR 05-4129, Continuum Dynamics Group, Los Alamos National Laboratory, Los Alamos, NM (2005).

[36] E. Chow, M. A. Heroux, Object-oriented framework for block preconditioning, ACM Transactions on Mathematical Software 24 (2) (1998) 159–183.

[37] S. J. Owen (Project Lead), Cubit: Geometry and mesh generation toolkit.
URL `http://cubit.sandia.gov`