The Hierarchical Subspace Iteration Method for Laplace–Beltrami Eigenproblems

AHMAD NASIKUN, Delft University of Technology, The Netherlands and Universitas Gadjah Mada, Indonesia
KLAUS HILDEBRANDT, Delft University of Technology, The Netherlands

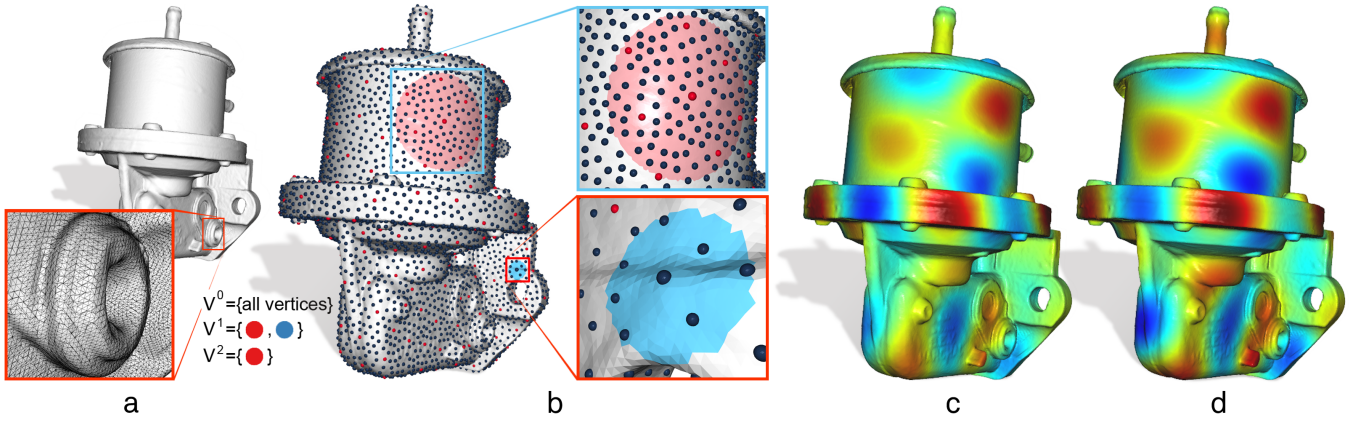


Fig. 1. The novel Hierarchical Subspace Iteration Method (HSIM) can efficiently solve eigenvalue problems, such as the computation of the p lowest modes of the discrete Laplace–Beltrami operator on a surface mesh (a). HSIM constructs a set of nested subspaces of the spaces of functions on the mesh by building a vertex hierarchy (b) and prolongation operators between the levels of the hierarchy. HSIM is initialized by solving a dense eigenproblem on the coarsest level, an example of a resulting eigenfunction is shown in (c). Then the eigenproblem is solved on each level, from coarse to fine, using subspace iterations initialized with the result from the previous level to finally produce the sought eigenpairs. An example of an eigenfunction is shown in (d).

Sparse eigenproblems are important for various applications in computer graphics. The spectrum and eigenfunctions of the Laplace–Beltrami operator, for example, are fundamental for methods in shape analysis and mesh processing. The Subspace Iteration Method is a robust solver for these problems. In practice, however, Lanczos schemes are often faster. In this paper, we introduce the Hierarchical Subspace Iteration Method (HSIM), a novel solver for sparse eigenproblems that operates on a hierarchy of nested vector spaces. The hierarchy is constructed such that on the coarsest space all eigenpairs can be computed with a dense eigensolver. HSIM uses these eigenpairs as initialization and iterates from coarse to fine over the hierarchy. On each level, subspace iterations, initialized with the solution from the previous level, are used to approximate the eigenpairs. This approach substantially reduces the number of iterations needed on the finest grid compared to the non-hierarchical Subspace Iteration Method. Our experiments show that HSIM can solve Laplace–Beltrami eigenproblems on meshes faster than state-of-the-art methods based on Lanczos iterations, preconditioned conjugate gradients and subspace iterations.

CCS Concepts: • **Computing methodologies** → **Shape analysis**.

Additional Key Words and Phrases: Laplace–Beltrami operator, Laplace matrix, spectral methods, multigrid, eigensolver, subspace iteration method

Authors' addresses: Ahmad Nasikun, Delft University of Technology, Department of Intelligent Systems, Delft, The Netherlands and Universitas Gadjah Mada, Department of Electrical and Information Engineering, Yogyakarta, Indonesia, ahmad.nasikun@ugm.ac.id; Klaus Hildebrandt, Delft University of Technology, Department of Intelligent Systems, Delft, The Netherlands, K.A.Hildebrandt@tudelft.nl.

ACM Reference Format:
Ahmad Nasikun and Klaus Hildebrandt. 2021. The Hierarchical Subspace Iteration Method for Laplace–Beltrami Eigenproblems. *ACM Trans. Graph.* 1, 1 (November 2021), 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Large-scale sparse eigenvalue problems arise in many applications of computer graphics. An important example is the computation of the low and medium frequency spectrum and the corresponding eigenfunctions of the Laplace–Beltrami operator of a surface. These are used in a range of applications in shape analysis and mesh processing. Commonly used methods for solving Laplace–Beltrami eigenproblems are based on Lanczos iterations. These are highly efficient solvers for sparse eigenvalue problems. However, in order to be efficient, they combine various extensions of the basic Lanczos iterations, which makes the algorithms complex and introduces parameters that need to be set. One problem is that Lanczos iterations are inherently unstable, which can be counteracted by re-starting strategies. Another issue is that Lanczos iterations lead to orthogonal eigenvectors only if the arithmetic is exact. Due to rounding errors, re-orthogonalization strategies are required. An alternative to Lanczos schemes is the Subspace Iteration Method (SIM). This method does not suffer from instabilities and is therefore easier to analyze and implement. On the other hand, the SIM is often slower than Lanczos schemes.

In this paper, we introduce the Hierarchical Subspace Iteration Method (HSIM). This method is suitable for computing the eigenpairs in the low and mid frequency part of the spectrum of an

operator defined on a mesh, such as the discrete Laplace–Beltrami operator. Our goal is to maintain the benefits of the SIM while reducing the computational cost significantly. One reason why the SIM is expensive is that many iterations are needed before the method converges. Our idea is to take advantage of the fact that low and mid frequency eigenfunctions can be approximated on coarser grids. Instead of working only on the finest grid, we shift iterations to coarser grids. This enables us to perform effective subspace iterations with little computational effort on coarse grids and substantially reduce the number of iterations needed on the finest grid.

We design the hierarchical solver so that it starts on the coarsest grid. The complexity of this grid is chosen such that the relevant matrices can be represented as dense matrices and all eigenfunctions on the coarsest grid can be efficiently computed with a standard dense eigensolver. Then, the hierarchy is traversed from coarse to fine, whereby the eigenproblem on each grid is solved to the desired accuracy by subspace iterations and the solution on the previous grid is used as an initialization for the subspace iterations. To make the subspace iterations more efficient, we use the eigenvalues computed on one grid to determine a value by which we shift the matrix on the next grid. To construct the hierarchy, we use vertex sampling to create a vertex hierarchy and build prolongation operators based on the geodesic vicinity of the samples. The prolongation operators are used to define a hierarchy of nested function spaces on the mesh whose degrees of freedom are associated with the vertex hierarchy. The advantage of the resulting hierarchy over alternatives, such as mesh coarsening-based hierarchies, is that we obtain a hierarchy of nested spaces. This is of benefit for our purposes because the prolongation to the finer grids then preserves properties of a subspace basis, like its orthonormality.

We evaluate our HSIM scheme on the computation of the lowest p eigenpairs of the Laplace–Beltrami operator, where p ranges from 50 to 5000. Our experiments show that HSIM significantly reduces the number of iterations needed on the finest grid and thus accelerates the SIM method. HSIM has also outperformed three state-of-the-art Lanzcos solvers and the Locally Optimal Block Preconditioned Conjugate Gradient Method in our experiments. HSIM was consistently faster than the fastest of the three Lanczos solvers over a range of computations on a variety of meshes and different numbers of eigenpairs to be computed. In particular, for challenging settings, in which more than a thousand eigenpairs needed to be computed, HSIM was up to six times faster than the fastest Lanczos solver.

We expect that applications that need to compute low and medium frequency eigenfunctions of the Laplace-Beltrami operator will benefit from the properties of HSIM, in particular, methods that need to continuously solve new eigenproblems, for example in the context of isospectralization [Cosmo et al. 2019; Rampini et al. 2019] and geometric deep learning [Bronstein et al. 2017], and methods that need to compute a larger number of eigenfunctions, for example, for shape compression [Karni and Gotsman 2000; Váša et al. 2014], filtering [Vallet and Lévy 2008], and shape signatures [Sun et al. 2009][1].

[1]In the supplementary material, we demonstrate that projections into subspaces spanned by Laplace–Beltrami eigenfunctions, and, at the example of the heat kernel signature, that shape signatures can benefit from using a larger number of eigenfunctions.

2 RELATED WORK

Spectral shape analysis and processing. The eigenfunctions of the Laplace–Beltrami operator on a surface have many properties that make them useful for applications. First, the eigenfunctions form an orthonormal basis of the space of functions on the surface, which generalizes the Fourier basis of planar domains to curved surfaces. With the help of the spectrum and the eigenfunctions, a frequency representation can be associated to functions on a surface and spectral methods from signal and image processing can be generalized to methods for the processing of surfaces. Examples of mesh processing applications that use the Laplace–Beltrami spectrum and eigenfunctions are surface filtering [Vallet and Lévy 2008], mesh and animation compression [Karni and Gotsman 2000; Váša et al. 2014], quad meshing [Dong et al. 2006; Huang et al. 2008; Ling et al. 2014], surface segmentation [Huang et al. 2009; Sharma et al. 2009], vector field processing [Azencot et al. 2013; Brandt et al. 2017], mesh saliency [Song et al. 2014] and shape optimization [Musialski et al. 2015]. Further properties of the Laplace–Beltrami eigenfunctions are that they are invariant under isometric surface deformation and that they reflect the symmetries of a surface. These properties make them a powerful tool for non-rigid shape analysis. For example, they are used to efficiently compute shape descriptors, such as the the Diffusion Distance [Nadler et al. 2005], the Shape-DNA [Reuter et al. 2005, 2006], the Global Point Signature [Rustamov 2007], the Heat Kernel Signature [Sun et al. 2009], the Auto Diffusion Function [Gebal et al. 2009] and the Wave Kernel Signature [Aubry et al. 2011]. Moreover the eigenfunctions are the basis for Functional Maps [Kovnatsky et al. 2013; Litany et al. 2017; Ovsjanikov et al. 2012, 2016; Rodolà et al. 2017; Rustamov et al. 2013], isospectralization [Cosmo et al. 2019; Rampini et al. 2019] and spectral methods in Geometric Deep Learning [Boscaini et al. 2015; Bronstein et al. 2017; Bruna et al. 2014; Sharp et al. 2020].

Krylov schemes. Krylov methods, such as Lanczos schemes for symmetric and Arnoldi schemes for general matrices, are effective solvers for large scale eigenproblems. For a comprehensive introduction to Krylov schemes, we refer to the textbook by Saad [2011]. One way to apply Lanczos schemes to generalized eigenproblems, such as the Laplace–Beltrami problem we consider, is to convert them to ordinary eigenproblems by a change of coordinates. In particular, if the scalar product is given by a diagonal mass matrix, the change of coordinates is not costly [Vallet and Lévy 2008]. For non-diagonal matrices, the coordinate transformation can be done using a Cholesky decomposition of the mass matrix [Saad 2011]. ARPACK [Lehoucq et al. 1998] provides implementations of the Implicitly Restarted Lanczos Method for symmetric eigenproblems and the Implicitly Restarted Arnoldi Method for non-symmetric eigenproblems. ARPACK is so widely used that it can be consider to provide reference implementations of the Implicitly Restarted Lanczos and Arnoldi Methods. For example, MATLAB's sparse eigensolver `eigs` interfaces ARPACK. SPECTRA [Qiu 2015] is a library offering a C++ implementation of an Implicitly Restarted Lanczos Method build on top of the EIGEN matrix library [Guennebaud et al. 2010]. An alternative to the implicitly restarted Lanczos method is the band-by-band, shift-and-invert Lanczos solver for Laplace–Beltrami eigenproblems on surfaces that was introduced in [Vallet and Lévy 2008].

Subspace iterations. An alternative to Krylov schemes is the Subspace Iteration Method (SIM). It is a robust method for solving generalized sparse eigenproblems and is well-suited for parallelization [Bathe 2013]. A comprehensive introduction to the SIM can be found in the textbook by Bathe [2014]. Matrix shifting is important to make the subspace iterations effective. Different heuristics have been proposed ranging from conservative choices [Bathe and Ramaswamy 1980; Gong et al. 2005] to more aggressive shifting strategies [Zhao et al. 2007]. A recent development is the concept of turning vectors [Kim and Bathe 2017] and its extension that includes the turning of turning vectors [Wilkins 2019].

Preconditioned Eigensolvers. The lowest eigenpairs of a matrix can be computed by minimizing the Rayleigh coefficient. The Locally Optimal Block Preconditioned Conjugate Gradient Method (LOBPCG) [Knyazev 2001] uses a preconditioned conjugate gradient solver for this minimization. A property of the method is that it does not need to explicitly access the matrix but only needs to evaluate matrix-vector products, which can be of benefit when dealing with large matrices. In recent work [Duersch et al. 2018], an improved basis selection strategy is proposed that improves the robustness of the method when larger numbers of eigenpairs are computed. LOBPCG was used for solving Steklov eigenproblems in [Wang et al. 2019]. A method that uses hierarchical preconditioning to approximate a few of the lowest eigenpairs was presented in [Krishnan et al. 2013]. While LOBPCG is reported to be effective for different eigenproblems, our experiments, see Section 6, indicate that for the Laplace–Beltrami eigenproblems we consider, HSIM is more efficient.

Approximation schemes. Schemes for the approximate solution of eigenproblems are static condensation [Bathe 2014] in engineering and the Nyström method [Williams and Seeger 2001] and random projections [Halko et al. 2011] in machine learning. Approximation schemes for the Laplace–Beltrami eigenproblem on surfaces have been introduced in [Chuang et al. 2009; Lescoat et al. 2020; Liu et al. 2019; Nasikun et al. 2018]. In contrast to the eigensolvers we consider in this work, these schemes do not provide any guarantee on the approximation quality of the eigenpairs.

Multigrids on surfaces. The multigrid hierarchy we need is challenging since we are working with an irregular mesh on a curved surface. One way to build a multigrid hierarchy for a surface mesh is to use mesh coarsening algorithms [Aksoylu et al. 2005; Hoppe 1996]. This is, however, not ideal for our setting because the resulting spaces are not nested, as each space is defined on a different surface. Another possibility is to build hierarchical grids on ambient space and then restrict the functions to the surface [Chuang et al. 2009]. The function spaces generated by this approach, however, do not resemble the linear Lagrange finite elements on the mesh that we want to work with. Algebraic multigrids [Stüben 2001] are an alternative that would fit our setting. However, unlike the proposed hierarchy, algebraic multigrids only use the operator to build the hierarchy, while we also use the geometry of the surface. A multi-level approach for the computation of the heat kernels on surfaces was introduced in [Vaxman et al. 2010]. In recent work, an intrinsic prolongation operator based on mesh coarsening has been proposed [Liu et al. 2021].

Multilevel eigensolvers. A traditional multigrid approach to eigenproblems is to treat them as a nonlinear equation and to apply nonlinear multigrid solver to the equation [Brandt et al. 1983; Hackbusch 1979]. These methods have the advantage that they can be extended or even applied directly to nonlinear eigenproblems. For linear eigenvalue problems, however, this technique is not always efficient because the specific properties of eigenvalue problems are not used when a general nonlinear solver is used.

Another approach is to integrate a multigrid scheme for solving linear systems into an eigensolver [Arbenz et al. 2005; Bank 1982; Martikainen et al. 2001; McCormick 1981]. A solver for linear eigenproblems that needs to solve linear systems in every iteration, such as Krylov and subspace iteration methods, is used as an outer iteration. In every outer iteration, the linear systems are solved in an inner multigrid loop. For our HSIM solver, we use sparse direct solvers for the linear systems, as these are more efficient in our setting than multigrid solvers, see [Botsch et al. 2005]. In a different application context, however, it could be useful to use a multigrid linear solver.

An approach in which also the outer iterations operate on two different grids was proposed in [Xu and Zhou 2001]. In this method, the lowest eigenpair of an elliptic operator is approximated by first computing the eigenpair on the coarse grid and then correcting it by a boundary value problem on the fine grid. This two-grid correction scheme was accelerated in [Hu and Cheng 2011] and extended to include matrix shifting in [Yang and Bi 2011]. A multigrid extension of the scheme was introduced in [Chen et al. 2016; Lin and Xie 2015] and later integrated with wavelet bases [Xie et al. 2019] and algebraic multigrid procedures [Zhang et al. 2015]. The multigrid correction scheme has been used for the computation of Laplace spectra on planar domains [Hu and Cheng 2011] and parametrized surfaces [Brannick and Cao 2015]. A key difference to HSIM is that HSIM provides users with explicit control of the residual of the resulting eigenpairs. In contrast, the multigrid correction schemes do not provide control over the residual. Instead, the resulting residual depends on the approximation quality of the grids in the hierarchy. We include a discussion and comparison in Section 6.

3 BACKGROUND

In this section, we first briefly review the Laplace–Beltrami eigenproblem, which we use for evaluating the proposed eigensolver. Then, we describe the Subspace Iteration Method, which will be the basis of the novel Hierarchical Subspace Iteration Method.

3.1 Laplace–Beltrami eigenproblem

In the continuous case, we consider a compact and smooth surface Σ in $\mathbb{R}^3$. A function ϕ is an eigenfunction of the Laplace–Beltrami operator Δ on Σ with eigenvalue $\lambda \in \mathbb{R}$ if

$$-\Delta\phi = \lambda\phi \tag{1}$$

holds. For discretization, the weak form of (1) is helpful. This can be obtained by multiplying both sides of the equation with a continuously differentiable function f and integrating

$$\int_\Sigma \operatorname{grad}\phi \cdot \operatorname{grad} f \, \mathrm{d}A = \lambda \int_\Sigma \phi f \, \mathrm{d}A. \tag{2}$$

On the left-hand side of the equation, we applied integration by parts. A function ϕ is a solution of (1) with eigenvalue λ if and only if (2) holds for all continuously differentiable functions f. A benefit of the weak form is that evaluating both integrals in (2) only requires functions to be weakly differentiable (with square-integrable weak derivative) and does not involve differentials of the surface's metric tensor.

In the discrete case, Σ is a triangle mesh and we consider a finite-dimensional space of functions defined on the mesh, usually the space F of continuous functions that are linear polynomials over every triangle. Then, for functions $\phi, f \in F$, the integrals in (2) can be evaluated and ϕ is an eigenfunction of the discrete Laplace–Beltrami operator if there is a $\lambda \in \mathbb{R}$ such that (2) holds for any $f \in F$.

Any function in F is uniquely determined by its function values at the vertices of the mesh. The nodal representation of a function in F is a vector $\Phi \in \mathbb{R}^n$ that lists the function values at all vertices. If a nodal vector Φ is given, the corresponding function in F can be constructed by linear interpolation of the function values at the three vertices in every triangle. Let $\varphi_i \in F$ be the function that takes the value one at vertex i and vanishes at all other vertices. Then, the stiffness, or cotangent, matrix S and the mass matrix M are given by

$$S_{ij} = \int_\Sigma \operatorname{grad}\varphi_i \cdot \operatorname{grad}\varphi_j \mathrm{d}A \qquad \text{and} \qquad M_{ij} = \int_\Sigma \varphi_i \varphi_j \, \mathrm{d}A. \tag{3}$$

Explicit formulas for S_{ij} and M_{ij} can be found in [Vallet and Lévy 2008; Wardetzky et al. 2007]. The eigenfunctions Φ and eigenvalue λ can be computed as the solution to the eigenvalue problem

$$S\,\Phi = \lambda M\,\Phi. \tag{4}$$

This is a sparse, generalized eigenvalue problem where M is symmetric and positive definite and S is symmetric. We refer to [Crane et al. 2013a; Hildebrandt et al. 2006] for more background on the discretization of the Laplace–Beltrami operator on surfaces.

3.2 Subspace iteration method

The subspace iteration method (SIM) is an approach for computing eigenpairs of generalized eigenvalue problems such as (4). We outline SIM in Algorithm 1. The input to the method are the stiffness and mass matrices $S, M \in \mathbb{R}^{n\times n}$, a matrix $\Phi \in \mathbb{R}^{n\times q}$ that specifies an initial subspace basis, the number of desired eigenpairs p, a tolerance ε and a shifting value μ. The dimension q of the subspace needs to be larger or equal to p. We will first discuss the subspace iterations without shifting, *i.e.* assuming $\mu = 0$, and then discuss choices of convergence test, subspace dimension, initial subspace basis, shifting value and linear solver.

SIM iteratively modifies the initial basis, which consists of q vectors Φ_i, such that it more and more becomes the desired eigenbasis. In each iteration, first an inverse iteration is applied to all q vectors (Algorithm 1, line 4), thereby increasing the low-frequency components in the vectors. For this, q linear systems of the form

$$(S - \mu M)\Psi_i = M\Phi_i \tag{5}$$

need to be solved. The second step in each iteration is to solve the eigenproblem restricted to the subspace spanned by the vectors Ψ_i (Algorithm 1, lines 5–7). For this, the reduced stiffness and mass matrices are computed and the q-dimensional dense eigenproblem is solved using a dense eigensolver, *e.g.* based on a QR factorization. The third step is to replace the current subspace basis with the eigenbasis (Algorithm 1, line 8). The inverse iterations amplify the low frequencies in the subspace basis. The second and third steps are needed in order to prevent the vectors from becoming linearly dependent. Without these steps, the vectors would all converge to the lowest eigenvector.

ALGORITHM 1: Subspace Iteration Method

Input: Stiffness matrix $S \in \mathbb{R}^{n\times n}$, mass matrix $M \in \mathbb{R}^{n\times n}$, initial vectors $\Phi \in \mathbb{R}^{n\times q}$, number of eigenpairs p, tolerance ε, shifting value μ

Output: Matrix $\bar{\Lambda}$ with lowest eigenvalues of (4) on diagonal and Φ listing eigenvectors as columns. First p pairs converged.

1 **Function** SIM($S, M, \Phi, p, \varepsilon, \mu$)**:**
2 Compute sparse factorization: $LDL^T = S - \mu M$
3 **repeat**
4 Solve using factorization: $(S - \mu M)\Psi = M\Phi$
5 Compute reduced stiffness matrix: $\bar{S} \leftarrow \Psi^T S \Psi$
6 Compute reduced mass matrix: $\bar{M} \leftarrow \Psi^T M \Psi$
7 Solve dense eigenproblem: $\bar{S}\bar{\Phi} = \bar{M}\bar{\Phi}\bar{\Lambda}$
8 Update vectors: $\Phi \leftarrow \Psi\bar{\Phi}$
9 **until** *pairs* $(\bar{\Lambda}_{ii}, \Phi_i)$ *pass convergence test (6) for all* $i \le p$
10 **return** $\bar{\Lambda}$ and Φ
11 **End Function**

Convergence test. The final step of each iteration is the convergence check, which tests whether or not the first p eigenvectors have converged. For each eigenpair Φ_i and λ_i, the relative M^{-1}-norm[2] of the residual of equation (4) is computed

$$\frac{\|S\Phi_i - \lambda_i M\Phi_i\|_{M^{-1}}}{\|S\Phi_i\|_{M^{-1}}} < \varepsilon \tag{6}$$

and the test is passed if it is below the threshold ε. The choice of the value for the convergence tolerance depends on the application context. In most of our experiments, we used $\varepsilon = 10^{-2}$, which based on our experiments, see Section 5, we consider appropriate for applications in shape analysis and spectral mesh processing.

[2]The M^{-1}-norm is given by $\|S\Phi\|_{M^{-1}} = \sqrt{(S\Phi)^T M^{-1} S\Phi}$. The reason, we use the M^{-1}-norm is that $S\Phi$ is an integrated quantity and $M^{-1}S\Phi$ is the corresponding function (pointwise quantity). The M-norm, which is the discrete L^2-norm, of the pointwise quantity is the same as the M^{-1}-norm of the integrated quatity, $\|S\Phi\|_{M^{-1}} = \sqrt{(S\Phi)^T M^{-1} S\Phi} = \sqrt{(M^{-1}S\Phi)^T M M^{-1} S\Phi} = \left\|M^{-1}S\Phi\right\|_M$. For more background, we refer to [Wardetzky et al. 2007].

Subspace dimension. The choice of the dimension q affects the computational cost per iteration and the number of iterations needed for convergence. A larger subspace size increases the computational cost per iteration as more linear systems have to be solved (line 4 of Algorithm 1) and the dimension of the dense eigenproblem (line 7) increases. On the other hand, the algorithm terminates when the subspace contains (good enough approximations of) the lowest p eigenvectors. This is easier to achieve if the subspace is larger. Therefore, with a larger subspace, fewer iterations may be needed. It is suggested to set $q = \max\{2p, p + 8\}$ in [Bathe 2013]. In our experiments, we found $q = \max\{1.5p, p + 8\}$ to be more efficient for the eigenproblems we consider.

Initialization. The subspace basis Φ can be initialized with a random matrix. An alternative is to use information extracted from the matrices for initialization, which can help to reduce the required number of subspace iterations. One heuristic from [Bathe 2014] is to use the diagonal of the mass matrix M as the first column of the matrix representing initial vectors, random entries for the last column, and unit vectors e_i with entry +1 at the degree of freedom with the smallest ratio of k_{ii}/m_{ii} for the remaining $q - 2$ columns.

Shifting. One way to make the subspace iterations more effective is to shift the matrix S, which means to replace it with the shifted matrix $S-\mu M$. The shifted matrix keeps the same eigenvectors while the eigenvalues are shifted by $-\mu$. As a consequence, the inverse iteration, line 4 of Algorithm 1, focuses on enhancing the frequencies around μ instead of around zero. This can help to reduce the number of iterations required for convergence. Different heuristics for setting the shifting value have been proposed. A conservative choice is to set μ to the average of the last two converged eigenvalues [Bathe and Ramaswamy 1980]. Alternative shifting strategies are to set μ to the average of the last converged and the first non-converged eigenvalue [Wilson and Itoh 1983] or to the average of the first two non-converged eigenvalues [Gong et al. 2005]. An aggressive shifting technique that places μ further into the range of the non-converged eigenvalues is shown to accelerate the SIM in [Zhao et al. 2007].

Direct solver. For the inverse iterations of the subspace basis, line 4 of Algorithm 1, q linear systems with the same matrix $S - \mu M$ need to be solved. It can be effective to use a direct solver for this task since a factorization once computed can be used to solve all the systems. Since the shifted matrix is not positive definite, we use a sparse symmetric indefinite decomposition $LDL^T = S - \mu M$.

4 HIERARCHICAL SUBSPACE ITERATION METHOD

In this section, we introduce the Hierarchical Subspace Iteration Method (HSIM). We first describe the construction of the hierarchy of function spaces on a mesh. Then, we detail the multilevel eigensolver that operates on the hierarchy.

4.1 Hierarchy construction

Important goals for the construction of the hierarchy are that the construction is fast since the hierarchy must be built as part of the HSIM algorithm, that the basis functions are locally supported and the prolongation and restriction operators are sparse, and that the functions spaces are nested. Moreover, the function spaces need to be able to approximate low and mid frequency functions well.

We describe the construction of the subspaces in three steps. First, we describe the construction of a hierarchy on the set of vertices of the mesh. Then, we define prolongation and restriction operators that act between the levels of the vertex hierarchy. Finally, we explain how the vertex hierarchy and the operators can be used to obtain the hierarchy of nested function spaces.

ALGORITHM 2: Construction of the vertex hierarchy

Input: Surface mesh Σ, number of levels T, number of vertices per level $n^1, n^2, \ldots, n^{T-1}$
Output: Sets of vertex indices $V^1, V^2, \ldots, V^{T-1}$

```
1  V^T ← {Random number from {0, 1, ..., |V_Σ| − 1}}
2  τ ← T − 1
3  repeat
4  |   V^τ ← V^{τ+1}
5  |   repeat
6  |   |   V^τ ← V^τ ∪ {Index of vertex farthest away from V^τ}
7  |   until |V^τ| = n^τ
8  |   τ ← τ − 1
9  until τ = 0
10 return V^1, V^2, ..., V^{T−1}
```

Vertex hierarchy. We consider a hierarchy with T levels ranging from 0 to $T - 1$, where 0 is the finest level. We denote by V^τ the set of vertices in level τ and by n^τ the number of vertices in V^τ. The sets V^τ are nested, $V^\tau \subset V^{\tau-1}$, and V^0 is the set of all vertices of the mesh. Since we will solve a dense eigenproblem to get all eigenpairs at the coarsest level, we want to control the number n^{T-1} of vertices in V^{T-1}, which we set to

$$n^{T-1} = \max\{\lceil 1.5p \rceil, 1000\}. \tag{7}$$

The numbers of vertices in the other levels are determined by the growth rate μ

$$n^\tau = \mu\, n^{\tau+1}, \tag{8}$$

where μ is given by

$$\mu = \sqrt[T]{\frac{n^0}{n^{T-1}}}. \tag{9}$$

The trade-off for the choice of the number of levels is that a larger number of levels helps to reduce the required number of iterations on the finest level. On the other hand, each level adds computational cost, *e.g.* for computing the reduced matrices S^τ and M^τ. In our experiments, we found HSIM to be most effective with a low number of levels. We used three levels in most cases and opted for two levels when only a small number of eigenpairs, *i.e.* $p \leq 200$, needs to be computed.

To form the sets V^τ, we use a scheme based on farthest point sampling [Eldar et al. 1997]. The set V^{T-1} is initialized to contain one random vertex. Then, iteratively the vertex farthest away from all the vertices that are already in V^{T-1} is added to V^{T-1} until the desired number of vertices is reached. The sets V^{T-2} to V^1 are created in a similar manner. The scheme is summarized in Algorithm 2. Most expensive in this algorithm is the repeated computation of the

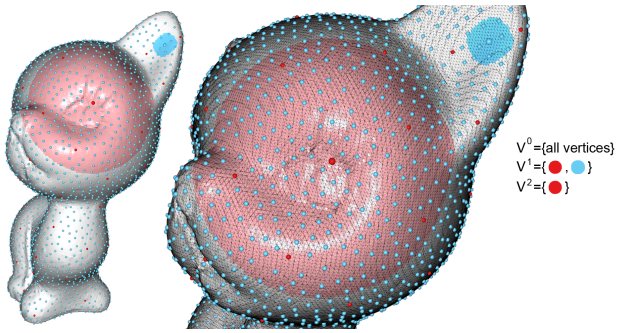


Fig. 2. Illustration of a vertex hierarchy with three sets $V^2 \subset V^1 \subset V^0$. The coarsest set V^2 consists of the red vertices, V^1 of the blue and the red vertices and V^0 of all vertices of the mesh. The light red and light blue regions are geodesic disks of radii ρ^2 and ρ^1 around the highlighted red and blue vertices in the centers of the regions and illustrate the support regions of the highlighted vertices.

farthest points (line 6). These computations can be accelerated by maintaining a distance field that stores for each vertex of the mesh the distance to the closest vertex in the current set V^τ. Since the vertices are inserted one after another, in each iteration the distance field only needs to be updated locally around the newly inserted vertex, and the maximum of the field has to be computed. We compute the distances between vertices using Dijkstra's algorithm on the edge graph with weights corresponding to the length of the edges. We found Dijkstra's distance a sufficient approximation of the geodesic distance for our purposes in our experiments. Alternatively, the Short-Term Vector Dijkstra (STVD) algorithm [Campen et al. 2013] could be used, which computes a more accurate approximation of the geodesic distance while still keeping computations localized. The supplementary material includes examples that illustrate that farthest point sampling generates hierarchies that are suitable for our purposes.

Prolongation and restriction. A function on level τ is represented by a vector $f^\tau \in \mathbb{R}^{n^\tau}$. We will first describe the prolongation and restriction operators and show in the next paragraph how the prolongation operator can be used to construct the piecewise linear polynomial corresponding to a vector f^τ. The τ^{th} prolongation operator is given by a matrix $U^\tau \in \mathbb{R}^{n^\tau \times n^{\tau+1}}$ that maps vectors $f^{\tau+1} \in \mathbb{R}^{n^{\tau+1}}$ representing functions on level $\tau+1$ to vectors $f^\tau \in \mathbb{R}^{n^\tau}$ representing functions on the finer level τ. The restriction operator maps from level τ to the coarser level $\tau+1$ and is given by the transpose $U^{\tau T}$ of the prolongation matrix. This relationship of the prolongation and restriction operators ensures that the restricted matrices S^τ and M^τ, see lines 4 and 5 of Algorithm 3 for a definition of the matrices, on all levels are symmetric.

The i^{th} row of U^τ describes how the value associated with the i^{th} vertex of level $\tau+1$ is distributed among the vertices on level τ. This means that the entry U^τ_{ij} is a weight describing how strongly the vertex j on level τ is influenced by the vertex i on level $\tau+1$ during the prolongation. This weight decreases with increasing geodesic distance of the vertices. To obtain sparse operators, the weight vanishes when the distance of the vertices reaches a threshold ρ^τ, which differs per level. We set ρ^τ to be

$$\rho^\tau = \sqrt{\frac{\sigma A}{n^\tau \pi}}. \tag{10}$$

where A is the area of the surface and σ is a control parameter. This choice of ρ^τ yields matrices U^τ that have about σ non-zero entries per row. For our experiments, we choose $\sigma = 7$. The reasoning behind (10) is that we want the sum of the areas of the geodesics disks of radius ρ^τ around all the vertices of level τ to be σ times the area of the surface. To make this idea easily computable, we replace the combined areas of all the geodesic disks by n^τ times the area of the Euclidean disk of radius ρ^τ.

To construct the matrices U^τ, we first construct preliminary matrices $\tilde{U}^\tau \in \mathbb{R}^{n^\tau \times n^{\tau+1}}$ that have the entries

$$\tilde{U}^\tau_{ij} = \begin{cases} 1 - \frac{d(v_i^{\tau+1}, v_j^\tau)}{\rho^\tau} & \text{for } d(v_i^{\tau+1}, v_j^\tau) \leq \rho^\tau \\ 0 & \text{for } d(v_i^{\tau+1}, v_j^\tau) > \rho^\tau \end{cases}, \tag{11}$$

where $d(v_i^{\tau+1}, v_j^\tau)$ is the geodesic distance of the i^{th} vertex of $V^{\tau+1}$ to the j^{th} vertex of V^τ. The matrix U^τ is then obtained by normalizing the rows of $\tilde{U}^\tau$

$$U^\tau_{ij} = \frac{1}{\sum_{j=1}^{n^\tau} \tilde{U}^\tau_{ij}} \tilde{U}^\tau_{ij}. \tag{12}$$

The normalization ensures that all function spaces will include the constant functions. This is of benefit for our purposes as the constant functions make up the kernel of the Laplace–Beltrami operator. Another property is that the set of functions on each level forms a partition of unity. As for the sampling scheme, we use Dijkstra's distance on the weighted edge graph of the mesh in our experiments to approximate the geodesic distance. A discussion of two alternatives, the Short-Term Vector Dijkstra algorithm [Campen et al. 2013] and the Heat Method [Crane et al. 2013b], is included to the supplementary material.

Function spaces. So far we have considered abstract vectors $f^\tau \in \mathbb{R}^{n^\tau}$. Now, we describe how the continuous piecewise linear polynomial corresponding to f^τ can be constructed. On the finest level, any $f^0 \in \mathbb{R}^{n^0}$ is the nodal vector, which lists the function values of the continuous, piecewise linear polynomial at the vertices. To get the continuous, piecewise linear polynomial that corresponds to a $f^\tau \in \mathbb{R}^{n^\tau}$ for any τ, we use the prolongation operators to lift f^τ to the finest level. The resulting vector

$$U^0 U^1 ... U^{\tau-1} f^\tau \tag{13}$$

is the nodal vector of the continuous, piecewise linear polynomial corresponding to f^τ. By construction, the resulting function spaces are nested and the functions are locally supported. The HSIM algorithm does not need to lift the functions using (13). Instead, the reduced stiffness and mass matrices S^τ and M^τ are directly computed for each level.

4.2 Hierarchical Solver

The HSIM is outlined in Algorithm 3. The algorithm starts with preparing the multilevel subspace iterations. First, the number of

levels, the vertex hierarchy and the prolongation matrices U^τ are computed. Then the reduced stiffness and mass matrices, S^τ and M^τ, for all levels are constructed from fine to coarse starting with level 1. In this computation, we benefit from the fact that the prolongation matrices U^τ are highly sparse. The next step is to determine the dimension q of the subspace that is used. Our experiments indicate that values between $q = 1.5p$ and $q = 2p$ are suitable. Following [Bathe 2013], we set $q = p + 8$ for small values of p

$$q = \max\{\lceil 1.5p \rceil, p + 8\}. \tag{14}$$

The last step before the multilevel iterations start is the computation of an initial subspace. This is done by solving the eigenproblem on the coarsest level of the hierarchy completely using a dense eigensolver. The dimension of the coarsest space is chosen, see (7), such that the dense eigenproblem can be solved efficiently.

ALGORITHM 3: Hierarchical Subspace Iteration Method

Input: Stiffness and mass matrices of finest level $S^0, M^0 \in \mathbb{R}^{n\times n}$, number of eigenpairs p, number of levels T, tolerance ε
Output: p lowest eigenpairs of the generalized eigenproblem (4)

Function HSIM($S^0, M^0, p, T, \varepsilon$)**:**
  Compute vertex hierarchy (Section 4.1)
  Build matrices U^τ for $\tau = 0, 1, \ldots, T-2$ (Section 4.1)
  for $\tau \leftarrow 1$ *to* $T-1$ **do**
    Build level τ stiffness matrix: $S^\tau \leftarrow (U^{\tau-1})^T S^{\tau-1} U^{\tau-1}$
    Build level τ mass matrix: $M^\tau \leftarrow (U^{\tau-1})^T M^{\tau-1} U^{\tau-1}$
  end
  Set size of subspace: $q \leftarrow \max(\lceil 1.5p \rceil, p + 8)$
  Compute first q eigenpairs of $S^{T-1}\Phi^{T-1} = \Lambda^{T-1} M^{T-1}\Phi^{T-1}$
  for $\tau \leftarrow (T-2)$ *to* 0 **do**
    Prolongation of subspace basis: $\Phi^\tau \leftarrow U^\tau \Phi^{\tau+1}$
    Set shifting parameter: $\mu \leftarrow \Lambda^{\tau+1}_{jj}$ with $j = \lfloor \frac{p}{10} \rfloor$
    $(\Lambda^\tau, \Phi^\tau) \leftarrow$ **SIM**$(S^\tau, M^\tau, \Phi^\tau, p, \varepsilon, \mu)$
  end
  return First p diagonal entries of Λ^0 and first p columns of Φ^0
End Function

The multilevel iterations traverse the hierarchy from coarse to fine starting with the second coarsest level. At each level, the eigenproblem is solved up to the tolerance by subspace iterations. The subspace iterations are initialized with the eigenvectors computed at the coarser level. To make the subspace iteration more effective, we use the approximate eigenvalues computed on the previous level to specify a shifting parameter for the iterations on the current level. We employ an aggressive shifting strategy, which sets the shifting value to be the estimated eigenvalue with index $\lfloor p/10 \rfloor$. The shifting value is set only once for each level and used for all subspace iterations on this level. Then, only one sparse factorization of the shifted stiffness matrix $S^\tau - \mu^\tau M^\tau$ has to be computed per level. This way we achieve that, on the one hand, the shifting value is regularly updated, while, on the other hand, no additional factorizations have to be computed.

To further accelerate the subspace iterations, we do not perform additional inverse iterations, step 4 of the Algorithm 1, on the lowest r vectors that are already converged. However, to avoid error accumulation, we stop the iteration of vectors only after their residual, eq. (6), has reached one tenth of the specified tolerance ε. A further acceleration is achieved by performing two inverse iterations before orthonormalizing the vectors. Thus we execute step 4 of Algorithm 1 twice before we continue with step 5.

The subspace iteration method converges quickly when the desired eigenspace is close to the initial subspace. Our hierarchical method makes use of this property by providing the subspace iterations on each level with the solution from the coarser level. As a result, only few iterations are needed on each level. In particular, the multilevel strategy substantially reduces the necessary number of iterations on the finest level compared to SIM. The price to pay is that the hierarchy has to be built and iterations on the coarse levels are needed. Nevertheless, HSIM is about 4-8 times faster than SIM in our experiments. The highest acceleration is achieved in the difficult case that a large number of eigenvectors must be computed.

5 EXPERIMENTS

Implementation. Our implementation of HSIM uses Eigen [Guennebaud et al. 2010] for linear algebra functionalities and LibIGL [Jacobson et al. 2016] for geometry processing tasks. OpenMP is used to solve the linear systems in each subspace iteration, step 4 of Algorithm 1, in parallel and to compute the prolongation and projection matrices in parallel during hierarchy construction. Moreover, we solve the low-dimensional eigenproblems at the coarsest level of the hierarchy, step 9 of Algorithm 3, and in each subspace iteration, step 7 of Algorithm 1, on the GPU using a direct solver for dense generalized eigenproblems from CUDA's cuSolver library. For our experiments, we used an Alienware Area-51 R3 home desktop with a AMD Ryzen Threadripper 1950x (16 core) processor and 24GB of RAM, equipped with NVIDIA GeForce GTX 1080 Ti graphics card with 11GB memory.

Timings. Table 1 lists timings of our HSIM implementation for the computation of the p lowest eigenpairs of the discrete Laplace–Beltrami operator, eq. (4), for meshes with different sizes and values of p. Individual timings for hierarchy construction and solving the eigenproblems using the hierarchy are listed. Moreover, iteration counts for subspace iterations on the individual levels are provided. For the coarsest level, a dense solver is used instead of the subspace iteration, therefore, the table lists F's instead of a number for the coarsest level. The solver's convergence tolerance is set to 10^{-2} for all examples. In all cases, the required number of iterations on the finest level is reduced to one by the hierarchical approach. Figure 4 provides more details for one example, the computation of the lowest 200 eigenpairs on a dragon model with $150k$ vertices

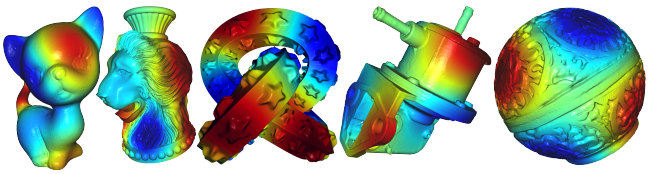


Fig. 3. The 10th eigenfunction of the Laplace-Beltrami operator of the models listed in Table 1.

Model (#Verts)	#Eigs	#Iters	Timings of HSIM		
			Hier.	Solver	Total
Kitten (137k)	50	F|1	1.9	6.2	8.1
	250	F|1|1	3.7	28.4	32.1
	1000	F|2|1	4.3	118.9	123.2
Vase-Lion (200k)	50	F|1	3.2	5.3	8.5
	250	F|2|1	7.3	41.2	48.5
	1000	F|3|1	9.2	188.0	197.2
Knot-Stars (450k)	50	F|1	9.9	28.4	38.3
	250	F|2|1	29.1	131.0	160.1
	1000	F|3|1	36.3	505.7	542.0
Oilpump (570k)	50	F|1	9.2	31.9	41.1
	250	F|2|1	31.6	122.6	154.2
	1000	F|3|1	40.3	650.6	690.9
Red-Circular (700k)	50	F|1	10.2	65.5	75.7
	250	F|2|1	40.6	199.3	239.9
	1000	F|4|1	55.0	1061.2	1116.2

Table 1. Timings of HSIM for the computation of the lowest eigenpairs of the Laplace–Beltrami operator on surface meshes with different numbers of vertices. The error tolerance ε is set to 10^{-2}. Individual timings for constructing the hierarchy and for solving the eigenproblem using the hierarchy are listed (in seconds). Meshes are shown in Figure 3.

using a hierarchy with three levels. The figure shows (a) how the runtimes split over the different levels of the hierarchy, (b) for the finest level the division between prolongation of the solution for the second finest level and subspace iterations, and (c) the breakdown of the timings of the individual steps of the subspace iterations (Algorithm 1) on the finest level. The figure illustrates that, when three levels are used, most of the runtime is spent on the finest level, almost 80% for the shown example, and that the restrictions of the stiffness and mass matrices and solving the linear systems are the most costly steps of HSIM.

Figure 5 lists runtimes for different numbers of eigenpairs to be computed. In our experiments, we found that the runtime grows linearly even when computing several thousand eigenpairs. This

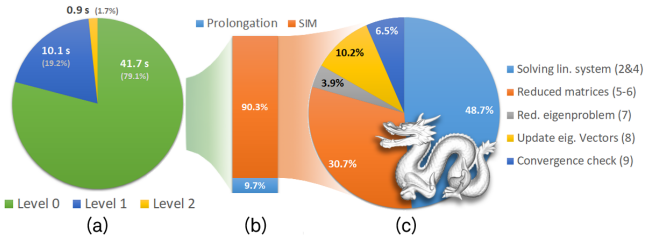


Fig. 4. Analysis of timings of the HSIM for the Laplace–Beltrami eigenproblem. Distribution of the runtimes to the three levels (a), split of the time spent at the finest level between the prolongation of the solution from level 1 and the subspace iterations at the finest level (b) and distribution of the time of the subspace iteration to the individual steps in Algorithm 1. The 200 lowest eigenpairs are computed on the Dragon mesh with 150k vertices.

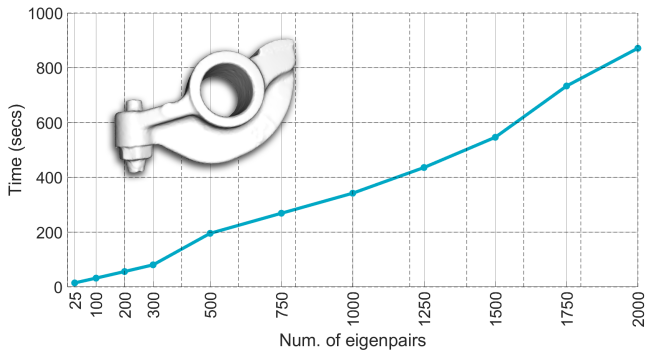


Fig. 5. Plot listing the runtime of HSIM over the number of Laplace–Beltrami eigenpairs to be computed on the Rocker Arm model with 270k vertices.

is illustrated by the timings listed in the figure. We expect this linear trend to continue as long as the runtime is dominated by the time needed for the solving of the linear systems (step 4 of Algorithm 1). At some point, solving the dense eigenproblems (step 7 of Algorithm 1), which does not scale linearly with the number of eigenpairs, will be the most expensive step and the trend will no longer be linear.

For most experiments, we set the convergence tolerance, ϵ in Algorithm 3, to 10^{-2}. Figure 6 lists runtimes over the convergence tolerance for the computation of 100 eigenpairs on two different meshes, the Blade model with 200k vertices and the Chinese Dragon with 127k vertices. The figure illustrates that low tolerances such as 10^{-9} can be achieved and that the time grows proportional with the relative residual. Roughly speaking, we observe in our experiments that the number of iterations that are needed on the finest grid grows by two for a decrease of one order of magnitude in the relative residual.

Termination criterion. To test for convergence, see line 9 of Algorithm 1, we use the criterion stated in (6). This test ensures the convergence of the eigenvalues as well as the convergence of the eigenvectors. To determine a suitable value for the convergence tolerance ε, we performed several experiments. We discuss two experiments in this paragraph, the supplementary material includes

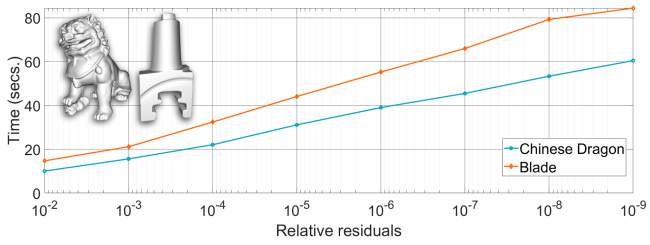


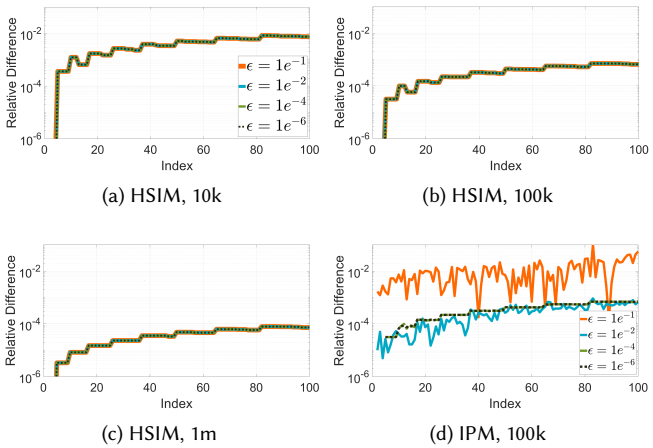
Fig. 6. The plot of the required computation time for different error tolerances when computing the first 100 eigenpairs of Laplace–Beltrami operator on the Chinese Dragon (127k vertices) and on the Blade model (200k vertices).

Model (#Vert)	#Eigs	SIM		par. SIM	HSIM		Lanczos methods			Prec. Solver
		#Iters	Time	Time	#Iters	Time	MATLAB	MH	SPECTRA	LOBPCG
Sphere (160k)	50	7	49.4	23.8	F|1	11.7	16.2	27.2	24.4	48.0
	250	7	274.5	155.7	F|2|1	51.3	94.3	285.3	124.8	268.3
	1000	7	1088.0	642.2	F|2|1	165.6	921.8	1132.2	1235.5	2601.1
	2500	7	3228.2	1930.7	F|2|1	529.8	7784.5	2987.1	7552.7	Mem. bound
	4000	8	10687.8	8913.0	F|2|1	1431.2	11745.1	5836.1	13100.1	Mem. bound
Rocker Arm (270k)	50	8	74.9	42.2	F|1	14.6	18.6	26.4	25.8	126.5
	250	8	541.7	300.1	F|2|1	79.6	130.3	178.7	185.3	711.5
	1000	8	2118.9	1228.0	F|2|1	342.1	1549.0	696.4	1359.4	4014.5
	2500	8	10278.5	8658.4	F|2|1	1108.1	13018.3	1798.9	9543.0	Mem. bound
Rolling stage (660k)	50	7	212.5	102.5	F|1	57.9	62.7	100.1	77.7	384.2
	250	7	1308.8	664.4	F|2|1	206.6	362.5	773.3	675.4	1885.2
	1000	7	8058.2	5358.9	F|3|1	937.8	4072.6	3034.5	8396.0	Mem. bound

Table 2. Comparison of HSIM to the (non-hierarchical) SIM, different Lanczos solvers, and LOBPCG. Runtimes are listed in seconds.

additional experiments. Based on the results of these experiments, we used a tolerance of $\varepsilon = 10^{-2}$ for the evaluation of HSIM. In the first experiment, we consider three different discretizations of the unit sphere with regular meshes (having 10k, 100k and 1m vertices) and measure the difference between the computed eigenvalues for different tolerances ($\varepsilon = 10^{-1}, 10^{-2}, 10^{-4}, 10^{-6}$) and the analytical solution. The results are shown in Figure 7. For all three discretizations, the difference between the numerical solutions for different tolerances is small compared to the approximation error, that is, the difference to the analytical solution. We would like to note that the convergence test establishes an upper bound on the convergence of the eigenpairs. In particular, for the lowest tolerance, $\varepsilon = 10^{-1}$, the solutions computed by HSIM are often already more accurate when the process terminates. One reason for this is that the method terminates only after all eigenpairs pass the convergence test. We therefore conducted an additional experiment using the inverse power method to compute the eigenpairs one by one and stop the iteration for each eigenpair when the convergence tolerance is reached. The results are shown in Figure 7 (d). In this experiment, differences in accuracy occur between the numerical solution for $\varepsilon = 10^{-1}$ and the other solutions ($\varepsilon = 10^{-2}, 10^{-4}, 10^{-6}$), which indicates that a tolerance of $\varepsilon = 10^{-1}$ is not sufficient.

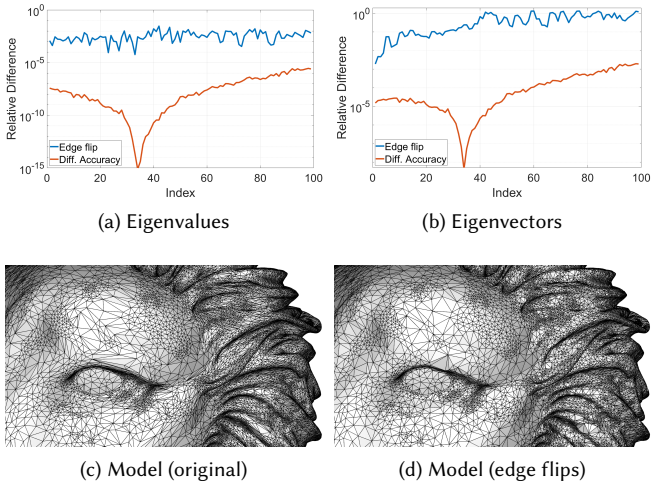
In a second experiment, we compute eigenpairs for two different meshes approximating the same surface. The second mesh was created by flipping edges of the first mesh. For both meshes, we compute the lowest eigenpairs for the tolerance $\varepsilon = 10^{-2}$ and as reference for $\varepsilon = 10^{-8}$. Since the two meshes have the same vertices, we can compare both the eigenvalues and the eigenvectors. Figure 8 shows the difference between the reference solutions ($\varepsilon = 10^{-8}$) on both meshes (blue graph) and for one mesh, the difference between the solutions for $\varepsilon = 10^{-2}$ and $\varepsilon = 10^{-8}$ (red graph). To measure



(a) HSIM, 10k (b) HSIM, 100k

(c) HSIM, 1m (d) IPM, 100k

Fig. 7. Relative difference of numerical approximations of the eigenvalues of the unit sphere to the analytic solutions are shown.



(a) Eigenvalues (b) Eigenvectors

(c) Model (original) (d) Model (edge flips)

Fig. 8. Comparison of the relative difference of the eigenvalues and the eigenvectors between two meshes that approximate the same surface (blue graph) and solutions for different convergence tolerance on one of the meshes (red graph).

Model (#Verts)	#Eigs	Tol	Level=2		Level=3		Level=4		Level=5	
			#Iters	Time	#Iters	Time	#Iters	Time	#Iters	Time
Rocker Arm (270k)	50	1e-2	F|1	17.2	F|1|1	31.1	F|1|1|1	66.0	F|1|1|1|1	123.8
		1e-4	F|3	26.0	F|3|2	38.2	F|2|2|3	76.8	F|2|2|2|2	135.3
	2000	1e-2	F|3	1078.6	F|2|1	650.7	F|1|1|1	885.3	F|1|1|1|1	1412.0
		1e-4	F|8	2595.4	F|7|4	2137.1	F|6|4|4	2967.8	F|5|3|3|4	3586.4
Ramses (820k)	50	1e-2	F|1	45.3	F|1|1	101.0	F|1|1|1	244.0	F|1|1|1|1	495.0
	300		F|2	246.4	F|2|1	234.5	F|2|1|1	417.6	F|1|1|1|1	700.6
	750		F|2	969.6	F|2|1	657.5	F|2|1|2	1521.1	F|2|1|1|1	1607.0

Table 3. Performance of HSIM with different numbers of levels.

#Eigs	Residue	Shift ratio						
		No shift	0.1	0.2	0.25	1/3	0.4	0.45
50	1e-2	F|1	F|1	F|1	F|1	**F|1**	F|2	F|2
250		F|2|1	F|2|1	F|2|1	F|2|1	**F|2|1**	F|2|1	F|2|1
1000		F|3|1	F|2|1	F|2|1	F|2|1	**F|2|1**	F|3|2	F|4|2
50	1e-4	F|5	F|5	F|4	F|4	**F|4**	F|4	F|5
250		F|6|4	F|6|4	F|5|4	F|5|3	F|5|3	**F|4|3**	F|5|4
1000		F|8|4	F|8|4	F|7|4	F|7|4	**F|6|3**	F|6|4	F|7|5

Table 5. Iteration counts for different choices of shifting values are shown. Computations are done using the Gargoyle model with 85k vertices.

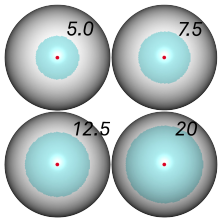
the difference of eigenvectors the relative L^2-norm is used. It can be seen that the difference between the reference solutions on the two meshes is more than three orders of magnitude larger than the difference between the solutions for different tolerances.

We want to note that the convergence test (6) does not directly measure the deviation from the exact solution. In our experiments (for example in Figure 8), we see that the relative difference between the solution for a tolerance of $\varepsilon = 10^{-2}$ and the reference solution, which is computed with $\varepsilon = 10^{-8}$, is usually much smaller than 10^{-2}. In Figure 8, and also in Figure 12, one can observe that the errors generated by HSIM are smaller for the eigenvalue pairs whose index is about one-third of the total number of computed eigenvalues than for the others. This is due to our shifting strategy, which makes these eigenpairs converge faster.

When evaluating the convergence criterion, equation (6), the standard norm of $\mathbb{R}^n$ is commonly used to replace the M^{-1}-norm. The reason is that the evaluation of the M^{-1} norm can be costly. For our experiments, we used the M^{-1}-norm at the finest level as the M matrices are diagonal, and, therefore, can be easily inverted. At the coarser levels, however, the restricted matrices M^τ (see line 6 of Algorithm 3) are no longer easy to invert. To save the effort of computing a factorization of the M^τ matrices, we replace the M^{-1}-norm by the standard norm for the convergence check on all but the finest level. We want to emphasize that since on the finest level we use the M^{-1}-norm, the error tolerances are respected. The simplification could only lead to more or fewer iterations on the coarser levels. Since the convergence check uses the relative norm, a global scaling factor to better match the standard norm and the M^{-1}-norm is not needed. We did not observe differences in the numbers of iterations, when using the standard norm instead of the M^{-1}-norm for the convergence test on the coarser levels in our experiments.

Number of levels. A parameter HSIM needs as user input is the number of levels of the hierarchy, see Algorithm 3. By increasing the number of levels, one can reduce the number of iterations required on the finest grid. On the other hand, increasing the number of levels leads to additional computational costs on the levels below the finest level. Table 3 lists computation times and iteration counts for the individual levels for computations with different meshes sizes, number of eigenpairs and convergence tolerances. For most of these examples, three levels yield the shortest runtime.

Support region. For the construction of the prolongation matrices U^τ, the radius of its domain of influence, ρ^τ, must be defined individually for each level. We use eq. (10), which allows us to set the radii on all levels by means of a control parameter σ. This value is the average expected number of non-zero entries per row of the matrices U^τ. The inset figure shows the areas of influence around one point for different values σ. A smaller value for σ results in matrices U^τ with less non-zero entries and thus less computational effort per iteration. On the other hand, a too small value for σ can increase the number of iterations needed on each level.



In our experiments, we have identified a value of $\sigma = 7$ as a good trade-off. This means that in each level, each vertex of V^τ in average is coupled to six neighbor vertices, which agrees with the average valence in a triangle mesh. Table 4 shows iteration counts and runtimes for different values of σ for eigenproblems on two meshes with 200k and 475k vertices and different numbers of eigenpairs to be computed. The value $\sigma = 7$ reaches in all cases either the lowest runtime or a time close to the lowest runtime.

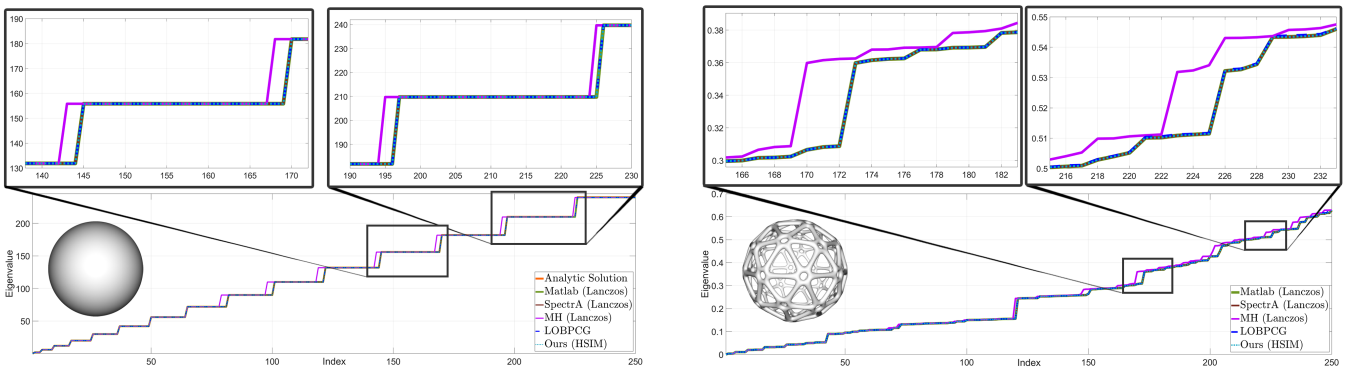
Shifting strategy. Matrix shifting can reduce the number of required subspace iterations on all levels. In our experiments, we use a heuristic, which is described in Step 12 of Algorithm 3, to automatically determine the shifting parameter μ. This heuristic is based on the aggressive shifting technique from [Zhao et al. 2007]. We

Model (#Verts)	#Eigs	2.5		5		7		10		20	
		#Iters	Time	#Iters	Time	#Iters	Time	#Iters	Time	#Iters	Time
Vase-Lion (200k)	100	F|2	15.9	F|2	17.0	F|1	12.2	F|1	13.4	F|2	23.6
	400	F|3|2	78.3	F|3|1	60.4	F|3|1	69.6	F|2|1	74.3	F|2|1	110.4
	750	F|4|2	175.4	F|3|2	174.8	F|3|1	138.9	F|3|2	219.5	F|3|2	301.5
Eros (475k)	100	F|2	55.7	F|1	42.2	F|1	43.0	F|2	65.3	F|2	71.7
	400	F|2|3	273.6	F|2|2	231.9	F|2|1	169.8	F|4|1	207.2	F|4|1	292.7
	750	F|3|4	710.2	F|2|3	580.5	F|4|1	465.5	F|4|2	567.8	F|5|2	758.7

Table 4. Runtimes and iteration counts for different values of the parameter σ that determines the supports of the functions.

Boundary	#Eigs	#Iters	Timing		
			Hier.	Solve	Total
Dirichlet	50	F|2	5.8	30.0	35.8
	250	F|2|1	19.3	93.7	113.0
Neumann	50	F|2	5.7	29.8	35.4
	250	F|2|1	18.9	94.9	113.3

Table 6. Timings and iteration counts for solving eigenproblems with boundary conditions on the Julius Caesar model with 370k vertices.

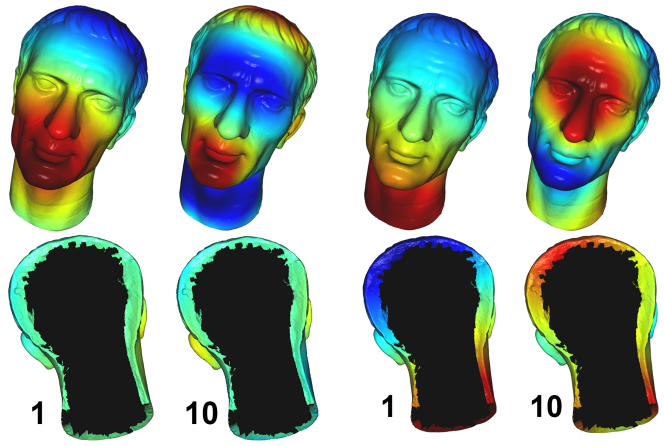


(a) Sphere, 160k (b) Ball, 90k

Fig. 9. The lowest 250 Laplace–Beltrami eigenvalues computed with HSIM and three different Lanczos solvers on a discrete sphere with 160k vertices (left) and a surface with many symmetries and 90k vertices (right). For the sphere, the analytic solution is shown as a reference.

set μ equal to the current approximate eigenvalue Λ_{jj} with index $j = \lfloor \alpha p \rfloor$. Here, α should take a value between 0 and 0.5. In Algorithm 3, we set $\alpha = 0.1$. Table 5 lists iteration counts for different values of α. Results for different numbers of eigenpairs and error margins are shown. We used values between 0.1 and 1/3 for α in our experiments.

Surface with boundary. We applied HSIM to the computation of Laplace–Beltrami eigenproblems on surfaces with boundary. We experimented with Dirichlet and Neumann boundary conditions and used the same hierarchy and basis construction as for surfaces without boundary. Examples of eigenfunctions on surfaces with boundary are shown in Figure 10. Table 6 shows for an example mesh the runtimes and iteration counts for Dirichlet and Neumann boundary conditions. The runtimes are comparable to the runtimes we observe for meshes without boundary and a comparable number of vertices.



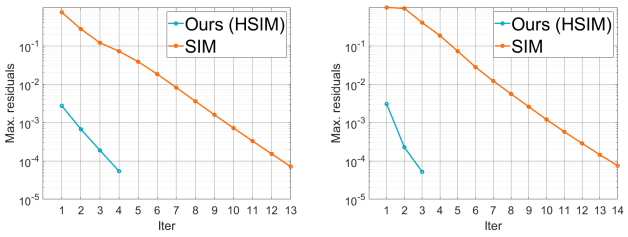
(a) Dirichlet (b) Neumann

Fig. 10. The first (left) and tenth (right) eigenfunction of the Laplace–Beltrami operator on a surface with boundary using Dirichlet and Neumann boundary conditions are shown.

6 COMPARISONS

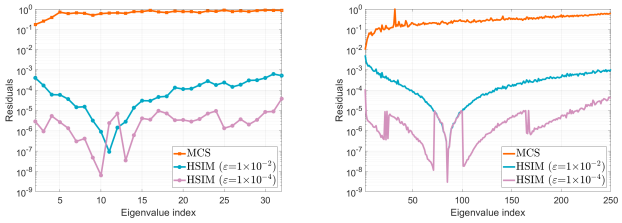
In this section, we discuss comparisons of HSIM to alternative methods. Laplace–Beltrami eigenproblems are commonly solved in graphics applications using Lanczos methods [Vallet and Lévy 2008]. Therefore, we begin this section with the comparison to Lanczos solvers. An alternative to Lanczos schemes is the SIM [Bathe 2013]. Since HSIM is based on SIM, this comparison provides a basis to quantify the gains resulting from our hierarchy. The third solver to which we compare HSIM is the Locally Optimal Block Preconditioned Conjugate Gradient Method [Knyazev 2001]. Lastly, we compare HSIM to the multilevel correction scheme that was introduced in [Chen et al. 2016; Lin and Xie 2015]. In our comparisons, we use the same convergence test (6) for all methods and set the tolerance to $\varepsilon = 10^{-2}$, which is the value we determined in our experiments, see Section 5. To implement this for the methods we compare to, we check for convergence after every iteration and stop when the convergence test is passed. Once the required number of iterations is known, we re-run the computation without convergence test and record the timings.

Lanczos schemes. Schemes based on Lanczos iterations are commonly used for solving large-scale, sparse, symmetric eigenproblems. These methods have been studied and improved over decades. ARPACK's implementation of the Implicitly Restarted Lanczos Method is well-established [Lehoucq et al. 1998]. We compare HSIM with MATLAB's `eigs` (MATLAB 9.8, R2020a) that interfaces ARPACK and with SPECTRA [Qiu 2015] that offers an alternative implementation of the Implicitly Restarted Lanczos Method. In addition to that, we compare to the authors' implementation of the band-by-band, shift-and-invert Lanczos solver that was introduced in [Vallet and Lévy 2008]. We denote this solver by Manifold Harmonics (MH). If a diagonal, or lumped, mass matrix is used in (4), the generalized



(a) 250 eigs (Vase-Lion, 200k V) (b) 100 eigs (Eros, 475k V)

Fig. 11. Plot of the maximum residual and the numbers of iterations for SIM and HSIM are shown. For HSIM the number of iterations on the finest level is used.



(a) 32 eigenpairs (b) 250 eigenpairs

Fig. 12. Plot of the residuals of MCS and our novel HSIM eigensolver. Not only HSIM is substantially more accurate, it also has explicit control on the accuracy of the eigenvalues and corresponding eigenvectors. (Dragon model, 150k vertices.)

eigenproblem can easily be transformed to an 'ordinary' eigenproblem as described in [Vallet and Lévy 2008]. We have tested all three Lanczos solvers on the 'ordinary' eigenproblem.

The runtimes for meshes of different complexity and with different numbers of eigenpairs are given in Table 2. The listed runtimes for HSIM also include the construction of the hierarchy and prolongation operators. In our experiments, HSIM was consistently faster than all three Lanczos schemes. This is also reflected in the table where HSIM is the fastest method for all combinations of mesh complexity and numbers of eigenpairs. In particular, for the difficult cases where a larger number of eigenpairs is computed, HSIM is significantly faster.

Figure 9 shows plots of the lowest eigenvalues for two surfaces computed with different solvers. On the left side of the figure, numerical approximations of the eigenvalues of the unit sphere computed with the different solvers on a mesh with 320k triangles approximating the sphere are shown. For reference, the analytical solution is included to the plot. On the right side of the figure, results for a surface that exhibits different symmetries are shown. SpectrA and Matlab applied to the ordinary eigenproblem provided accurate results in our experiments that for the sphere example well-approximate the analytic solution. The results obtained with HSIM match the accuracy of SpectrA and Matlab. The band-by-band, shift-and-invert solver [Vallet and Lévy 2008] meets the convergence tolerance for the individual eigenpairs, but some eigenpairs are skipped. This seems to happen at the transitions between the bands and we have observed it in our experiments consistently for different bandwidths.

SIM. In addition to the runtimes for Lanczos schemes, Table 2 also lists times and iteration counts for the (non-hierarchical) SIM. If one compares the number of iterations required by HSIM on the finest level with the number of iterations required by SIM, one sees that HSIM effectively reduces the number of iterations from 7-8 to 1. Accordingly, we observe that HSIM is 4-8 times faster than SIM. The table lists additional runtimes for an optimized SIM implementation in which the linear systems in step 4 of Algorithm 1 are solved in parallel using OpenMP and the dense eigenproblems, step 7 of Algorithm 1, are solved on the GPU using CUDA's cuSolver library. Figure 11 shows for two examples how the number of iterations changes if a lower convergence tolerance is requested. It can be seen that the increase of iterations is lower for HSIM than for SIM.

LOBPCG. The last column of Table 2 lists timings for the Locally Optimal Block Preconditioned Conjugate Gradient Method (LOBPCG). To generate the timings, we used the author's implementation [Knyazev et al. 2007]. We experimented with Jacobi preconditioners, incomplete Cholesky factorizations and the preconditioner $S - \nu Id$, which is suggested in [Knyazev 2001]. The latter produced the best results, which we report. Here S is the stiffness matrix (of the transformed ordinary eigenvalue problem that we also used for the Lanczos solvers), $\nu \in \mathbb{R}$ is approximately in the middle of the first ten eigenvalues [Knyazev 2001], and Id is the identity matrix. The results demonstrate that HSIM can solve the eigenproblems faster than LOBPCG with the preconditioners we tested.

Multilevel correction scheme. We compare with the multilevel correction scheme (MCS) from [Chen et al. 2016; Lin and Xie 2015], which is an extension of the two-grid scheme from [Hu and Cheng 2011]. This method has in common with our HSIM method that for initialization, an eigenvalue problem on the coarsest grid is solved. However, the multilevel iterations differ substantially from HSIM. In their method, the coarse space is used in all levels and it is enriched by vectors that are computed in the multilevel iterations. An essential difference to HSIM is that HSIM reduces the error on each level to the desired tolerance margin, while MCS does not offer direct control over the accuracy of the solution. The accuracy depends on the approximation quality of the coarse grid and the growth rate between the grids. Therefore, an aggressive growth rate, which is essential to the performance of our scheme, would lead to an increase in approximation error. Another substantial difference is that MCS is focused on computing only one or a few eigenpairs. This contrasts this work from our setting in which we compute more than a thousand eigenpairs. In Figure 12, we show a plot of the accuracy of the eigenpairs computed with MCS and HSIM with convergence tolerance 10^{-2} and 10^{-4}. The error produced by MCS is orders of magnitude higher than that produced by HSIM. Moreover, the plot on the right shows that the error increases with the index of the eigenvalue. This illustrates the point that MCS is focused on the computation of a few of the lowest eigenpairs. Since the MCS

scheme is formulated for regular grids, we use our hierarchy with three levels in the comparisons for both schemes, MCS and HSIM.

7 CONCLUSION

We introduce HSIM, a hierarchical solver for sparse eigenvalue problems and evaluate HSIM on the computation of the lowest p eigenpairs of the discrete Laplace–Beltrami operator on triangle surface meshes. HSIM first constructs a hierarchy of nested subspaces of the space functions on the mesh. Then, it iterates from coarse to fine over the hierarchy solving the eigenproblem on all levels to the desired accuracy. HSIM is initialized with the solution of the eigenproblem on the coarsest level, which is computed by solving a low-dimensional dense eigenproblem. Our comparisons show that HSIM outperforms state-of-the-art Lanczos solvers and demonstrate the advantages of the hierarchical approach over the plain SIM.

We think that the benefits of HSIM over Lanczos and SIM solvers make HSIM attractive for methods in shape analysis and mesh processing. Therefore, we plan to release our implementation of HSIM.

Future work. One direction of future work is to explore alternative hierarchies, e.g. operator-dependent bases or wavelets on surfaces. This could improve the performance of HSIM for certain types of operators, such as strongly anisotropic operators. Another direction could be to extend the method such that not only the lowest but arbitrary eigenpairs can be efficiently computed. Moreover, the method could be improved by further exploring the possibilities of parallelization of the method and by integrating out-of-core techniques for the computation of large eigenbases. Another aspect is that for a certain complexity of the meshes, the direct solvers will no longer be the most efficient solvers. Then, hierarchical solvers could to be used for the linear systems that need to solved in every iteration. For such an approach, it could be interesting to combine the hierachies used for HSIM and for the linear solves.

A benefit of HSIM is that it directly works for generalized eigenvalue problems, such as (4), and does not require to transform these to ordinary eigenvalue problems. This could be helpful when using the method for solving eigenproblems in which the mass matrix M is not a diagonal matrix, such as the discretization of the Laplace–Beltrami operator with higher-order elements [Reuter et al. 2006].

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their constructive feedback. This project is partly supported by the Indonesia Endowment Fund for Education (LPDP) through a doctoral scholarship for Ahmad Nasikun. For our experiments, we used models from the Aim@Shape repository, the Stanford Computer Graphics Laboratory (Stanford 3D Scanning Repository), Turbosquid, Al-Badri's and Nelles' Nefertiti, and INRIA.

REFERENCES

Burak Aksoylu, Andrei Khodakovsky, and Peter Schröder. 2005. Multilevel Solvers for Unstructured Surface Meshes. *SIAM J. Sci. Comput.* 26, 4 (2005), 1146–1165.

Peter Arbenz, Ulrich L. Hetmaniuk, Richard B. Lehoucq, and Raymond S. Tuminaro. 2005. A comparison of eigensolvers for large-scale 3D modal analysis using AMG-preconditioned iterative methods. *Internat. J. Numer. Methods Engrg.* 64, 2 (2005), 204–236.

M. Aubry, U. Schlickewei, and D. Cremers. 2011. The wave kernel signature: A quantum mechanical approach to shape analysis. In *ICCV*. 1626–1633.

Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Maks Ovsjanikov. 2013. An Operator Approach to Tangent Vector Field Processing. *Comp. Graph. Forum* 32, 5 (2013), 73–82.

Randolph E. Bank. 1982. Analysis of a Multilevel Inverse Iteration Procedure for Eigenvalue Problems. *SIAM J. Numer. Anal.* 19, 5 (1982), 886–898.

Klaus-Jürgen Bathe. 2013. The subspace iteration method–Revisited. *Computers & Structures* 126 (2013), 177–183.

Klaus-Jürgen Bathe. 2014. *Finite element procedures* (2nd edition ed.). Prentice Hall.

Klaus-Jürgen Bathe and Seshadri Ramaswamy. 1980. An accelerated subspace iteration method. *Computer Methods in Applied Mechanics and Engineering* 23, 3 (1980), 313–331.

D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst. 2015. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. *Comp. Graph. Forum* 34, 5 (2015), 13–23.

Mario Botsch, David Bommes, and Leif Kobbelt. 2005. Efficient Linear System Solvers for Mesh Processing. In *Mathematics of Surfaces (Lecture Notes in Computer Science, Vol. 3604)*, Ralph R. Martin, Helmut E. Bez, and Malcolm A. Sabin (Eds.). Springer, 62–83.

A. Brandt, S. McCormick, and J. Ruge. 1983. Multigrid Methods for Differential Eigenproblems. *SIAM J. Sci. Stat. Comput.* 4, 2 (June 1983), 244–260.

Christopher Brandt, Leonardo Scandolo, Elmar Eisemann, and Klaus Hildebrandt. 2017. Spectral Processing of Tangential Vector Fields. *Comp. Graph. Forum* 36, 6 (2017), 338–353.

James Brannick and Shuhao Cao. 2015. Bootstrap Multigrid for the Shifted Laplace-Beltrami Eigenvalue Problem. arXiv preprint arXiv:1511.07042.

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Process. Mag.* 34, 4 (2017), 18–42.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. *International Conference on Learning Representations* (2014).

Marcel Campen, Martin Heistermann, and Leif Kobbelt. 2013. Practical Anisotropic Geodesy. *Comp. Graph. Forum* 32, 5 (2013), 63–71.

Hongtao Chen, Hehu Xie, and Fei Xu. 2016. A full multigrid method for eigenvalue problems. *J. Comput. Phys.* 322 (2016), 747–759.

Ming Chuang, Linjie Luo, Benedict J. Brown, Szymon Rusinkiewicz, and Michael Kazhdan. 2009. Estimating the Laplace–Beltrami Operator by Restricting 3D Functions. *Comp. Graph. Forum* 28, 5 (2009), 1475–1484.

Luca Cosmo, Mikhail Panine, Arianna Rampini, Maks Ovsjanikov, Michael M. Bronstein, and Emanuele Rodolà. 2019. Isospectralization, or How to Hear Shape, Style, and Correspondence. In *IEEE CVPR*. 7529–7538.

Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. 2013a. Digital Geometry Processing with Discrete Exterior Calculus. In *ACM SIGGRAPH 2013 courses (SIGGRAPH '13)*.

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2013b. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)* 32, 5 (2013), 1–11.

Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. 2006. Spectral surface quadrangulation. *ACM Trans. Graph.* 25, 3 (2006), 1057–1066.

Jed A. Duersch, Meiyue Shao, Chao Yang, and Ming Gu. 2018. A Robust and Efficient Implementation of LOBPCG. *SIAM J. Sci. Comput.* 40, 5 (2018), C655–C676.

Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi. 1997. The Farthest Point Strategy for Progressive Image Sampling. *Trans. Img. Proc.* 6, 9 (1997), 1305—-1315.

Katarzyna Gebal, Jakob Andreas Bærentzen, Henrik Aanæs, and Rasmus Larsen. 2009. Shape Analysis Using the Auto Diffusion Function. *Comp. Graph. Forum* 28, 5 (2009), 1405–1413.

Yu-Cai Gong, Hong-Wei Zhou, Pu Chen, and Ming-Wu Yuan. 2005. Comparison of subspace iteration, iterative Ritz vector method and iterative Lanczos method. *Journal of Vibration Engineering* 18, 02 (2005), 227–232.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

W. Hackbusch. 1979. On the Computation of Approximate Eigenvalues and Eigenfunctions of Elliptic Operators by Means of a Multi-Grid Method. *SIAM J. Numer. Anal.* 16, 2 (1979), 201–215.

N. Halko, P. G. Martinsson, and J. A. Tropp. 2011. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Rev.* 53, 2 (2011), 217–288.

Klaus Hildebrandt, Konrad Polthier, and Max Wardetzky. 2006. On the convergence of metric and geometric properties of polyhedral surfaces. *Geometricae Dedicata* 123 (2006), 89–112.

Hugues Hoppe. 1996. Progressive meshes. In *ACM SIGGRAPH*. 99–108.

Xiaozhe Hu and Xiaoliang Cheng. 2011. Acceleration of a two-grid method for eigenvalue problems. *Math. Comp.* 80, 275 (2011), 1287–1301.

Jin Huang, Muyang Zhang, Jin Ma, Xinguo Liu, Leif Kobbelt, and Hujun Bao. 2008. Spectral quadrangulation with orientation and alignment control. *ACM Trans. Graph.* 27, 5 (2008), 1–9.

Qixing Huang, Martin Wicke, Bart Adams, and Leo Guibas. 2009. Shape Decomposition Using Modal Analysis. *Comp. Graph. Forum* 28, 2 (2009), 407–416.

Alec Jacobson, Daniele Panozzo, et al. 2016. libigl: A simple C++ geometry processing library. http://libigl.github.io/libigl/.

Zachi Karni and Craig Gotsman. 2000. Spectral Compression of Mesh Geometry. In *ACM SIGGRAPH*. 279–286.

Ki-Tae Kim and Klaus-Jürgen Bathe. 2017. The Bathe subspace iteration method enriched by turning vectors. *Computers & Structures* 186 (2017), 11–21.

Andrew V Knyazev. 2001. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM journal on scientific computing* 23, 2 (2001), 517–541.

Andrew V. Knyazev, Merico E. Argentati, Ilya Lashuk, and Evgueni E. Ovtchinnikov. 2007. Block Locally Optimal Preconditioned Eigenvalue Xolvers (BLOPEX) in Hypre and PETSc. *SIAM J. Sci. Comput.* 29, 5 (2007), 2224–2239.

Artiom Kovnatsky, Michael M. Bronstein, Alexander M. Bronstein, Klaus Glashoff, and Ron Kimmel. 2013. Coupled quasi-harmonic bases. *Comput. Graph. Forum* 32, 2 (2013), 439–448.

Dilip Krishnan, Raanan Fattal, and Richard Szeliski. 2013. Efficient preconditioning of laplacian matrices for computer graphics. *ACM Trans. Graph.* 32, 4 (2013), 142:1–142:15. https://doi.org/10.1145/2461912.2461992

R. Lehoucq, D. C. Sorensen, and C. Yang. 1998. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM.

Thibault Lescoat, Hsueh-Ti Derek Liu, Jean-Marc Thiery, Alec Jacobson, Tamy Boubekeur, and Maks Ovsjanikov. 2020. Spectral Mesh Simplification. *Computer Graphics Forum* 39, 2 (2020), 315–324.

Qun Lin and Hehu Xie. 2015. A multi-level correction scheme for eigenvalue problems. *Math. Comp.* 84 (2015), 71–88.

Ruotian Ling, Jin Huang, Bert Jüttler, Feng Sun, Hujun Bao, and Wenping Wang. 2014. Spectral Quadrangulation with Feature Curve Alignment and Element Size Control. *ACM Trans. Graph.* 34, 1 (2014), 11:1–11:11.

Or Litany, Emanuele Rodolà, Alexander M. Bronstein, and Michael M. Bronstein. 2017. Fully Spectral Partial Shape Matching. *Comput. Graph. Forum* 36, 2 (2017), 247–258.

Hsueh-Ti Derek Liu, Alec Jacobson, and Maks Ovsjanikov. 2019. Spectral coarsening of geometric operators. *ACM Trans. Graph.* 38, 4 (2019), 105:1–105:13.

Hsueh-Ti Derek Liu, Jiayi Eris Zhang, Mirela Ben-Chen, and Alec Jacobson. 2021. Surface Multigrid via Intrinsic Prolongation. *ACM Trans. Graph.* 40, 4 (2021).

Janne Martikainen, Tuomo Rossi, and Jari Toivanen. 2001. Computation of a few smallest eigenvalues of elliptic operators using fast elliptic solvers. *Communications in Numerical Methods in Engineering* 17, 8 (2001), 521–527.

Stephen F. McCormick. 1981. A Mesh Refinement Method for $Ax = \lambda Bx$. *Math. Comp.* 36, 154 (1981), 485–498.

Przemyslaw Musialski, Thomas Auzinger, Michael Birsak, Michael Wimmer, and Leif Kobbelt. 2015. Reduced-order Shape Optimization Using Offset Surfaces. *ACM Trans. Graph.* 34, 4 (2015), 102:1–102:9.

Boaz Nadler, Stéphane Lafon, Ronald R. Coifman, and Ioannis G. Kevrekidis. 2005. Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker–Planck Operators. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*. 955–962.

Ahmad Nasikun, Christopher Brandt, and Klaus Hildebrandt. 2018. Fast Approximation of Laplace–Beltrami Eigenproblems. *Comp. Graph. Forum* 37, 5 (2018).

Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. 2012. Functional Maps: A Flexible Representation of Maps Between Shapes. *ACM Trans. Graph.* 31, 4 (2012), 30:1–30:11.

Maks Ovsjanikov, Etienne Corman, Michael Bronstein, Emanuele Rodolà, Mirela Ben-Chen, Leonidas Guibas, Frederic Chazal, and Alex Bronstein. 2016. Computing and Processing Correspondences with Functional Maps. In *SIGGRAPH ASIA 2016 Courses*. ACM, 9:1–9:60.

Yixuan Qiu. 2015. SpectrA: C++ Library For Large Scale Eigenvalue Problems. https://spectralib.org/.

Arianna Rampini, Irene Tallini, Maks Ovsjanikov, Alexander M. Bronstein, and Emanuele Rodolà. 2019. Correspondence-Free Region Localization for Partial Shape Similarity via Hamiltonian Spectrum Alignment. In *IEEE 3D Vision*. 37–46.

Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. 2005. Laplace-Spectra as Fingerprints for Shape Matching. In *Proceedings of the ACM Symposium on Solid and Physical Modeling*. 101–106.

Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. 2006. Laplace-Beltrami spectra as "Shape-DNA" of surfaces and solids. *Computer-Aided Design* 38, 4 (2006), 342–366.

Emanuele Rodolà, Luca Cosmo, Michael M. Bronstein, Andrea Torsello, and Daniel Cremers. 2017. Partial Functional Correspondence. *Comput. Graph. Forum* 36, 1 (2017), 222–236.

Raif M. Rustamov. 2007. Laplace–Beltrami eigenfunctions for deformation invariant shape representation. In *Symposium on Geometry Processing*. 225–233.

Raif M. Rustamov, Maks Ovsjanikov, Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Leonidas Guibas. 2013. Map-based Exploration of Intrinsic Shape Differences and Variability. *ACM Trans. Graph.* 32, 4 (2013), 72:1–72:12.

Yousef Saad. 2011. *Numerical methods for large eigenvalue problems: revised edition.* Vol. 66. Siam.

Avinash Sharma, Radu Patrice Horaud, David Knossow, and Etienne von Lavante. 2009. Mesh Segmentation Using Laplacian Eigenvectors and Gaussian Mixtures. In *Manifold Learning and Its Applications*.

Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. 2020. Diffusion is All You Need for Learning on Surfaces. *CoRR* abs/2012.00888 (2020). https://arxiv.org/abs/2012.00888

Ran Song, Yonghuai Liu, Ralph R. Martin, and Paul L. Rosin. 2014. Mesh Saliency via Spectral Processing. *ACM Trans. Graph.* 33, 1 (2014), 6:1–6:17.

Klaus Stüben. 2001. A review of algebraic multigrid. *J. Comput. Appl. Math.* 128, 1 (2001), 281 – 309.

Jian Sun, Maks Ovsjanikov, and Leonidas J. Guibas. 2009. A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. *Comp. Graph. Forum* 28, 5 (2009), 1383–1392.

Bruno Vallet and Bruno Lévy. 2008. Spectral Geometry Processing with Manifold Harmonics. *Comp. Graph. Forum* 27, 2 (2008), 251–260.

Libor Váša, Stefano Marras, Kai Hormann, and Guido Brunnett. 2014. Compressing dynamic meshes with geometric Laplacians. *Comp. Graph. Forum* 33, 2 (2014), 145–154.

Amir Vaxman, Mirela Ben-Chen, and Craig Gotsman. 2010. A Multi-resolution Approach to Heat Kernels on Discrete Surfaces. *ACM Trans. Graph.* 29, 4 (2010), 121:1–121:10.

Yu Wang, Mirela Ben-Chen, Iosif Polterovich, and Justin Solomon. 2019. Steklov Spectral Geometry for Extrinsic Shape Analysis. *ACM Trans. Graph.* 38, 1 (2019), 7:1–7:21.

Max Wardetzky, Miklós Bergou, David Harmon, Denis Zorin, and Eitan Grinspun. 2007. Discrete quadratic curvature energies. *Computer Aided Geometric Design* 24, 8-9 (2007), 499–518.

Bryce Daniel Wilkins. 2019. *The E^2 Bathe subspace iteration method.* Ph. D. Dissertation. Massachusetts Institute of Technology.

Christopher K. I. Williams and Matthias Seeger. 2001. Using the Nyström Method to Speed Up Kernel Machines. In *Advances in Neural Information Processing Systems 13*. MIT Press, 682–688.

Edward L Wilson and Tetsuji Itoh. 1983. An eigensolution strategy for large systems. *Computers & Structures* 16, 1-4 (1983), 259–265.

Hehu Xie, Lei Zhang, and Houman Owhadi. 2019. Fast Eigenpairs Computation with Operator Adapted Wavelets and Hierarchical Subspace Correction. *SIAM J. Numer. Anal.* 57, 6 (2019), 2519–2550.

Jinchao Xu and Aihui Zhou. 2001. A Two-Grid Discretization Scheme for Eigenvalue Problems. *Math. Comp.* 70, 233 (2001), 17–25.

Yidu Yang and Hai Bi. 2011. Two-grid finite element discretization schemes based on shifted-inverse power method for elliptic eigenvalue problems. *SIAM J. Numer. Anal.* 49, 3/4 (2011), 1602–1624.

Ning Zhang, Xiaole Han, Yunhui He, Hehu Xie, and Chunguang You. 2015. An Algebraic Multigrid Method for Eigenvalue Problems in Some Different Cases. arXiv:1503.08462.

Qian-Cheng Zhao, Pu Chen, Wen-Bo Peng, Yu-Cai Gong, and Ming-Wu Yuan. 2007. Accelerated subspace iteration with aggressive shift. *Computers & structures* 85, 19-20 (2007), 1562–1578.