Efficient Multiscale Lanczos Eigenpair Extraction

THEO BRAUNE, Adobe Research, LIX, Ecole Polytechnique, France
JEREMIE DUMAS, Adobe Research, France
JEAN-MARC THIERY, Adobe Research, France

Eigenpair extractions are crucial for various applications in geometry processing and graphics. State of the Art libraries like ARPACK or Spectra rely on the implicitly restarted Lanczos iteration to extract eigenpairs efficiently. However for some large scale problems they lack convergence speed and robustness. In this paper we present a simple multigrid extension to accelerate the convergence and robustness of the implicitly restarted Lanczos method, and we demonstrate the efficiency of our method on a variety of problems commonly found in geometry processing and graphics.

CCS Concepts: • **Mathematics of computing → Computations on matrices**; **Solvers**; • **Applied computing** → *Physical sciences and engineering*; • **Computing methodologies → Computer graphics**.

Additional Key Words and Phrases: Linear Algebra, Eigenpair extraction, Multigrid, Lanczos

ACM Reference Format:
Theo Braune, Jeremie Dumas, and Jean-Marc Thiery. 2026. Efficient Multiscale Lanczos Eigenpair Extraction . *ACM Trans. Graph.* 45, 4 (July 2026), 14 pages. https://doi.org/10.1145/3811367

1 Introduction

Eigenvalue problems for large sparse symmetric positive definite (SPD) matrices are ubiquitous in computer graphics and geometry processing. They arise in applications ranging from shape analysis and spectral clustering to vibration analysis, surface parameterization, spectral descriptors and distances, or surface reconstruction. In many scenarios, one is interested in computing only a moderate number of extremal eigenpairs of a discretized differential operator, rather than a full spectral decomposition. State-of-the-art methods for large-scale eigenproblems typically rely on Krylov subspace techniques, such as the Lanczos or Arnoldi iterations [Lanczos 1950; Parlett 1980], often combined with implicit restarting to control memory usage and numerical stability [Saad 2011; Sorensen 1992]. These methods are attractive due to their generality and robustness, but their performance deteriorates rapidly as the problem size grows. In particular, each iteration requires solving a linear system involving the underlying SPD operator, and the cost of prefactorization or repeated backsubstitution becomes prohibitive for high-resolution meshes or volumetric discretizations.

In contrast, MultiGrid (MG) methods have long been recognized as optimal solvers for large sparse linear systems arising from elliptic partial differential equations [Adams and Demmel 1999; Brandt 1973; Hackbusch 1985; Trottenberg et al. 2001]. By exploiting the multiscale structure of discretized operators, MG techniques achieve convergence rates that are largely independent of problem size. As a result, multiscale ideas are deeply ingrained in geometry processing, where hierarchical representations are routinely used to accelerate linear solves, smooth signals, and process geometric data efficiently.

Despite this, the use of MG schemes in large-scale eigenvalue solvers remains limited. While MG methods are well understood for linear systems, it is not immediately clear how multiscale information can be leveraged effectively for eigenproblems, where the goal is to extract invariant subspaces rather than solve a single system.

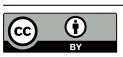
1.1 Related Work

MG methods for geometry processing have been extensively studied, including intrinsic MG constructions on surfaces [Aksoylu et al. 2005; Liu et al. 2021; Shi et al. 2009; Wiersma et al. 2023] and volumetric grids, as well as algebraic operator-dependent hierarchies [Ruge and Stüben 1987; Stüben 2001; Vaněk et al. 1996] with implementation in [Demidov 2020]. These approaches have proven highly effective for accelerating linear solves in applications such as smoothing, deformation, and simulation. However, they are typically employed either as standalone solvers or as preconditioners for Krylov methods applied to linear systems, rather than for eigenvalue extraction.

On the eigenvalue side, widely used numerical packages implement the implicitly restarted Lanczos (IRLM) or Arnoldi (IRAM) methods, which remain the standard tools for computing a small number of extremal eigenpairs of large sparse matrices [Lehoucq et al. 1997; Qiu 2021]. These methods are robust and well understood, but they fundamentally operate on a single resolution and rely on repeated applications of linear solves on the finest level.

Several works have explored reducing the cost of large-scale eigenvalue computations by exploiting coarse-level information. Hierarchical subspace iteration methods [Nasikun 2022; Nasikun and Hildebrandt 2022], for example, propagate approximate eigenvectors across a multilevel hierarchy and have been shown to significantly reduce the number of iterations, particularly for Laplacian-type operators. However, these approaches typically still rely on accurate solves or factorizations on the finest level, which can limit their applicability to more challenging operators. In particular, for problems involving nontrivial mass matrices or coupled systems arising in surface reconstruction and elasticity, the cost of a fine-level factorization can quickly become prohibitive. Related work has investigated preserving spectral properties under coarsening, with the goal of constructing reduced operators whose spectra approximate that of the original system [Chen et al. 2020; Liu et al. 2019]. While effective at maintaining spectral fidelity across levels, these approaches are not designed to yield a generic solver on the fine grid, nor do they directly integrate into a scalable eigensolver pipeline. Other multilevel eigenvalue algorithms, such as two-grid or coarse-grid correction schemes [Morgan and Yang 2018; Yang 2015], explicitly propagate

Authors' Contact Information: Theo Braune, Adobe Research, LIX, Ecole Polytechnique, France, theo.braune@polytechnique.edu; Jeremie Dumas, Adobe Research, France, jdumas@adobe.com; Jean-Marc Thiery, Adobe Research, France, jthiery@adobe.com.



ACM 1557-7368/2026/7-ART
https://doi.org/10.1145/3811367

Ritz vectors from coarse to fine levels and rely on repeated restarts to refine the solution. Although these methods can be effective in certain regimes and may reduce the number of restarts required, they are conceptually closer to explicitly restarted Krylov methods and do not fundamentally alter the fine-level computational bottleneck. Overall, existing approaches either employ MGs solely as an accelerator for linear solves—without modifying the eigenvalue algorithm itself—or propagate eigenvector information across levels while retaining expensive fine-level solves. This leaves a gap for methods that more tightly integrate multilevel structure into the eigensolver while avoiding costly fine-grid factorizations.

1.2 Contributions

We introduce a MG-accelerated framework for eigen extraction that directly integrates multiscale ideas into the core of the IRLM.

Our key contribution is a MG-aware Lanczos extension strategy that prolongates coarse-level Krylov subspaces and tridiagonal factorizations to finer levels, and refines them through warm-started power steps. This preserves the favorable convergence properties of IRLM while significantly reducing the cost of fine-level linear solves. By reusing multiscale spectral information, our method improves both performance and numerical stability, making more efficient the eigenvector extraction for large-scale and challenging operators.

2 Background

2.1 Multigrid Methods for Linear Solves

Large-scale geometry processing and physical simulation problems routinely give rise to sparse linear systems $Ax = b$, where A is SPD and typically originates from the discretization of elliptic partial differential operators (e.g. Laplace–Beltrami operators, linear elasticity, or diffusion processes). For such systems, classical Krylov methods such as Conjugate Gradient are attractive due to their low memory footprint, but their convergence rate deteriorates rapidly with increasing problem size and mesh resolution.

MG methods address this issue by exploiting a fundamental observation about the error spectrum of iterative solvers: simple relaxation schemes such as Jacobi or Gauss–Seidel eliminate highly-effectively *high-frequency* error components, while leaving *low-frequency* ones largely untouched. These smooth error modes become oscillatory when viewed on a coarser discretization, where they can again be efficiently damped. By recursively transferring the problem across nested levels, MG methods achieve convergence rates that are essentially independent of the problem size.

MG principle. Given a hierarchy of discretizations indexed from 0 (coarsest) to L (finest level), MG solvers (usually) alternate between:

- *Smoothing*: applying a small number of relaxation steps to damp high-frequency error components,
- *Restriction*: transferring the residual to a coarser level,
- *Coarse-grid correction*: approximately solving the error equation on the coarse level,
- *Prolongation*: interpolating it back to the finer level.

The most common realization of this strategy is the *V-cycle*, but many other cycles have been studied in the literature (eg, W/F/FMG-cycles, see Fig. 1) [Chen 2021].

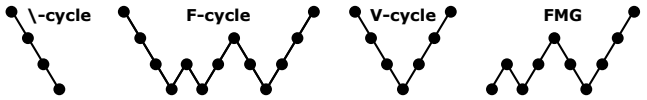


Fig. 1. Various MG schemes have been studied in the scientific literature, the V-cycle being prominent in Computer Graphics [Liu et al. 2021; Wiersma et al. 2023]. We propose in our work an adapted Full-MultiGrid (FMG) cycle to efficiently extract eigenpairs of a sparse SPD system.

Smoothers. A central component of any MG scheme is the choice of smoother. Gauss–Seidel or damped Jacobi relaxations are particularly effective for elliptic operators, as they strongly attenuate high-frequency error components and are cheap.

Multigrid preconditioners. While MGs can be used as a standalone solver, it is often most effective to employ them as preconditioners for Krylov methods: A MG cycle is then applied as an approximate inverse within each Krylov iteration, giving a preconditioned system with significantly improved eigenvalue distribution.

This combination preserves the robustness and global convergence properties of Krylov methods, while leveraging the near mesh-independent convergence behavior of MG schemes. In practice, MG-preconditioned CG or BiCGSTAB often outperforms either approach used in isolation, especially for large-scale problems arising in geometry processing and physical simulation.

2.2 Lanczos Methods for Eigenpair Extraction

We consider the problem of computing the k extremal generalized eigenpairs $(\lambda_i, x_i)_i$ of a SPD pair (A, M), i.e.,

$$Ax_i = \lambda_i M x_i, \tag{1}$$
$$x_i^T \cdot M \cdot x_j =: \langle x_i, x_j \rangle_M = \delta_i^j.$$

Such problems arise frequently in Geometry Processing, for instance when analyzing Laplace–Beltrami operators, modal vibrations of shells/volumes, or spectral decompositions of shapes.

In the following, we always search for dominant eigenpairs of an operator O (e.g., using the Power Method). To extract the largest (resp. smallest) eigenvalues of (A, M), one sets $O := M^{-1}A$ (resp. $O := A^{-1}M$). Shift-invert strategies allow extracting eigenpairs around a target σ eigenvalue, by setting $O := (A - \sigma M)^{-1}M$. In this setup, one recovers the original eigenvalues $\{\lambda_i\}$ from the extracted eigenvalues $\{\lambda_i'\}$ of the shifted problem through $\lambda_i = 1/\lambda_i' + \sigma$.

ALGORITHM 1: Power Method for Dominant Eigenvalue

Input: Power operator O, inner-product $\langle \cdot, \cdot \rangle_M$, initial vector x_0, tolerance tol, error measure $\epsilon(\lambda, x)$
Output: Approximate dominant eigenvalue λ of O, associated eigenvector x

$x \leftarrow x_0 / \|x_0\|_M$ (normalization);
repeat
  $w \leftarrow Ox$ (power step);
  $\lambda \leftarrow \langle x, w \rangle_M$;
  $x \leftarrow w / \|w\|_M$;
until $\epsilon(\lambda, v) <$ tol;

ALGORITHM 2: Extend Lanczos Basis (Generalized Setting)

Input: Power operator O, inner-product $\langle \cdot, \cdot \rangle_M$, basis vectors (v_{j-1}, v_j), tridiagonal matrix T_j
Output: Next vector v_{j+1} and updated T_{j+1}

$w = Ov_j$ (power step);
$t_{jj} \leftarrow \langle v_j, w \rangle_M, \quad t_{j,j-1} \leftarrow \langle v_{j-1}, w \rangle_M$;
$\tilde{v} \leftarrow w - t_{jj} v_j - t_{j,j-1} v_{j-1}$;
$t_{j+1,j} \leftarrow \|\tilde{v}\|_M, \quad v_{j+1} \leftarrow \tilde{v} / t_{j+1,j}$;

2.2.1 Lanczos iteration. To warm-up, we recall the classical Power Method (Algo. 1) to extract dominant general eigenpairs of Eq. (1). It allows computing a single eigenpair, and converges unfortunately quickly only in cases where the spectrum is well separated. To compute several eigenpairs at once, more evolved schemes such as the Lanczos method are needed.

Rather than discarding intermediate vectors, the Lanczos method builds an orthonormal basis V_k such that:

$$OV_k = V_k T_k + r_k e_k^T, \tag{2}$$

where T_k is symmetric and tridiagonal, r_k is the residual vector in iteration k and e_k denotes the $k-$th standard basis vector . Each vector v_{j+1} depends only on the previous two via the recurrence [Lanczos 1950]. The basis is grown iteratively by applying the operator to the last vector of the basis, and orthonormalizing the result against it to define the next basis vector (see Algo. 2).

This process builds a Krylov subspace whose tridiagonal projection T_k approximates the largest eigenvalues of the operator O. We can then compute the largest eigenpairs $\{\vartheta_i, \xi_i\}$ of T_k (the so-called Ritz pairs) of O, and reconstruct approximate eigenvectors in the original space through multiplication by V_k. For example, to extract largest eigenpairs, we use $O = M^{-1}A$, and see that $AV_k\xi_i \simeq MV_kT_k\xi_i = \lambda_i MV_k\xi_i$.

2.2.2 Termination Criterion. Given a Ritz pair (ξ_i, ϑ_i) with $T_k\xi_i = \vartheta_i\xi_i$, we follow [Arbenz 2016] and ARPACK [Lehoucq et al. 1997] to determine when an eigenpair has converged. Let $\hat{x}_i = V_k\xi_i$ denote the i-th approximate eigenvector. In this case we have

$$OV_k\xi_i = V_kT_k\xi_i + r_ke_k^t\xi_i = \vartheta_iV_k\xi_i + r_ke_k^t\xi_i = \vartheta_i\hat{x}_i + r_ke_k^t\xi_i,$$

thus

$$O\hat{x}_i - \vartheta_i\hat{x}_i = r_ke_k^T\xi_i.$$

Further, let $\beta_k = \|r_k\|_M$ and f_k such that $\beta_k f_k = r_k$. Following common practice, we consider an eigenpair as converged whenever

$$\beta_k|e_k^T\xi_i| \le \max(\mu, \mathrm{tol} \cdot |\vartheta_i|),$$

where $\mu > 0$ is a safety floor guard to prevent division by zero. We use $\mu = (\varepsilon_{\mathrm{mach}})^{\frac{2}{3}}$, where $\varepsilon_{\mathrm{mach}}$ denotes the machine precision.

2.2.3 Implicit Restarts. For brevity, we omit the explicitly restarted Lanczos method and focus instead on the IRLM, which we use as a baseline throughout this work. IRLM constructs a Krylov basis of size $M > m$, implicitly filters out unwanted spectral components via QR shifts, retains the m most relevant vectors, and subsequently expands the basis again while preserving the Lanczos recurrence. The overall procedure is summarized in Algo. 3. We refer the reader to [Arbenz 2016] for further background.

ALGORITHM 3: Implicitly Restarted Lanczos Method (IRLM)

Input: System (A, M), initial vector v_1, target size m, max size M
Output: Approximate m smallest generalized eigenpairs

Build a Lanczos basis (V_M, T_M) of size M (Algo. 2);
while *not all m eigenpairs converged (see Sec. 2.2.2)* **do**
  Compute Ritz pairs of T_M;
  Select $M - m$ unwanted Ritz values as shifts $\{\mu_i\}$ (Algo. 4);
  Apply QR shifts to filter the basis (Algo. 5);
  Truncate to size m: (V_m, T_m);
  Extend basis from size m to M (Algo. 2);

ALGORITHM 4: Adaptive shift selection for the implicit restart

Input: Ritz values $\{\theta_i\}_{i=1}^M$, Ritz residual estimates $\{\hat{r}_i\}_{i=1}^M$, target count m, max Krylov size M, number of converged pairs n_{conv}, near-convergence threshold τ_{lock}
Output: Adjusted subspace size m^+, shift set $\{\mu_i\}_{i=1}^p$

$m^+ \leftarrow m$;
// Protect near-converged Ritz pairs beyond the target set
for $i = m + 1$ **to** M **do**
  if $|\hat{r}_i| < \tau_{\mathrm{lock}}$ **then**
    $m^+ \leftarrow m^+ + 1$;
// Redistribute capacity freed by already-converged pairs
$m^+ \leftarrow m^+ + \min\Big(n_{\mathrm{conv}},\ \lfloor (M - m^+)/2 \rfloor\Big)$;
// Guard against degenerate cases
if $m^+ = 1$ ***and*** $M \ge 6$ **then**
  $m^+ \leftarrow \lfloor M/2 \rfloor$;
else if $m^+ = 1$ ***and*** $M > 2$ **then**
  $m^+ \leftarrow 2$;
$m^+ \leftarrow \min(m^+,\ M - 1)$;
$p \leftarrow M - m^+$;
Select the p unwanted Ritz values (those not among the m^+ best) as shifts $\{\mu_i\}_{i=1}^p$;
return m^+, $\{\mu_i\}_{i=1}^p$

In the simplest formulation of the implicit restart, one keeps m Ritz vectors and applies $p = M - m$ shifts at each restart cycle. In practice, however, a fixed split between kept and shifted subspaces is fragile. We observed that our solver performs in practice better, when we apply a heuristic strategy used in ARPACK [Lehoucq et al. 1997] to determine the number of QR shifts used in the implicitly restarted Lanczos method. We adaptively adjust the number of kept vectors m^+ at each restart according to Algo. 4. As convergence progresses, m^+ grows and fewer shifts are applied, so the restart transitions from aggressive polynomial filtering in the early iterations to gentle refinement in the final ones. The threshold τ_{lock} controls how aggressively near-converged pairs beyond the target set are protected from being shifted away. Following ARPACK, a typical choice is $\tau_{\mathrm{lock}} = 10\,\varepsilon_{\mathrm{mach}}$.

ALGORITHM 5: Multiple QR Shifts in the Lanczos Relation

Input: Tridiagonal matrix T_M, Lanczos basis V_M, shift values $\{\mu_i\}$
Output: Updated matrix T_M, updated basis V_M

Initialize: $Q_+ \leftarrow \mathrm{Id}$;
for i **do**
  Compute the QR decomposition: $T_M - \mu_i I = Q_{\mu_i} R_{\mu_i}$;
  Update the matrix: $T_M \leftarrow Q_{\mu_i}^T T_M Q_{\mu_i}$;
  Accumulate: $Q_+ \leftarrow Q_+ Q_{\mu_i}$;
Update the Lanczos basis: $V_M \leftarrow V_M Q_+$;

3 Our multigrid IRLM

IRLMs can suffer from various bottlenecks, depending on various factors such as number of requested eigenpairs, size of the problem, complexity of the operator (ie. number of non-zero entries per row, impacting performance of Cholesky factorizations and backsubstitutions), or spectral properties (how well are the eigenvalues spread across the spectrum, impacting convergence rate of the power step).

We tackle those issues by avoiding any expensive Cholesky factorization, relying on propagation of simpler-to-extract eigenpairs found at coarser levels, while benefiting from Lanczos bases sanitized in early stages, which we prolongate across levels similarly.

Our method allows computing eigenmodes that can be represented through multiscale decomposition, intuitively: smooth functions conceptually compatible with the coarsest level of our input multigrid. It builds upon those two observations:

(1) Smooth signals can be prolongated coarse-to-fine across common multigrid hierarchies. We adopt a FMG scheme and extract eigenvectors at the coarsest level, before prolongating them throughout the hierarchy.
(2) Frequencies added across levels are well identified and moderate. This allows us upsampling the Lanczos basis itself from one level to the next, and use it as warm start for the refinement of the basis through simple iterative solves.

3.1 Lanczos Basis Upsampling

We detail here our level-to-level Lanczos basis upsampling procedure, which is the key ingredient of our method.

Let $A_0 := A$ and $M_0 := M$ denote the finest-level stiffness and mass matrices. Given a prolongation matrix

$$P_i \colon \mathbb{R}^{n_{i+1}} \to \mathbb{R}^{n_i},$$

with $n_i > n_{i+1}$, we construct coarser matrices via *Galerkin projection*:

$$A_{i+1} = P_i^\top A_i P_i, \quad M_{i+1} = P_i^\top M_i P_i,$$

and build the associated operator O_i from (A_i, M_i)[1].

Suppose given a hierarchy of prolongation matrices $\{P_{L-1}, \dots, P_0\}$, and already computed Lanczos factorization for the k smallest generalized eigenpairs of O_{i+1} on level $i+1$:

$$O_{i+1} V_{i+1,k} = V_{i+1,k} T_{i+1,k} + r_{i+1,k} e_k^\top,$$

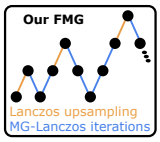
where $V_{i+1,k} \in \mathbb{R}^{n_{i+1} \times k}$ is the Lanczos basis and $T_{i+1,k}$ the associated tridiagonal matrix.

[1]For the shift-inverted operator ($O := (A - \sigma M)^{-1} M$), we apply the shift only on the finest operator: we effectively coarsen the shifted problem $(A - \sigma M; M)$.

To leverage this coarse-level information, we prolongate the Lanczos basis and residual to level i:

$$\tilde{V}_{i,k} = P_i V_{i+1,k}, \quad \tilde{r}_{i,k} = P_i r_{i+1,k}.$$

This yields an initial guess for a Lanczos factorization on level i, which we then refine. The key idea is to reuse the tridiagonal matrix $T_{i+1,k}$ found at the coarser level to initialize a warm-started solve on the finer level – effectively reversing the Lanczos recurrence to compute an initial power step vector consistent with the upsampled Krylov subspace.



The process is formalized in Algo. 6, which modifies the standard Lanczos extension (Algo. 2) by incorporating the upsampled vector from the coarser level as a spectral prior. This warm-started power step is then done with an iterative solver.

ALGORITHM 6: Multigrid-Accelerated Lanczos Extension

Input: Operator O, inner-product $\langle \cdot, \cdot \rangle_M$, basis vectors (v_{j-1}, v_j), prolongated estimate v'_{j+1}, coarse tridiagonal matrix $\tilde{T}_j$
Output: Next vector v_{j+1} and updated T_{j+1}

Scale the prolongated vector: $\tilde{v}'_{j+1} \leftarrow \tilde{t}_{j+1,j} \cdot v'_{j+1}$;
Reverse Lanczos step: $w \leftarrow \tilde{v}'_{j+1} + \tilde{t}_{jj} v_j + \tilde{t}_{j,j-1} v_{j-1}$;
$w = O v_j$ (linear solve done using warm-started iterative solver);
Compute projections: $t_{jj} \leftarrow \langle v_j, w \rangle_M, \quad t_{j,j-1} \leftarrow \langle v_{j-1}, w \rangle_M$;
Orthogonalize: $\tilde{v} \leftarrow w - t_{jj} v_j - t_{j,j-1} v_{j-1}$;
Normalize: $t_{j+1,j} \leftarrow \|\tilde{v}\|_M, \quad v_{j+1} \leftarrow \tilde{v} / t_{j+1,j}$;

The complete MG-accelerated Lanczos procedure proceeds in a coarse-to-fine manner. On the coarsest level, we compute a Lanczos factorization using a direct solver (eg, Cholesky):

$$O_\ell V_{\ell,k} = V_{\ell,k} T_{\ell,k} + r_{\ell,k} e_k^\top.$$

This provides both an initial Krylov basis and an associated tridiagonal matrix that capture the problem's dominant spectral information. The Lanczos basis is then prolongated successively to finer levels. At each level, we reuse the tridiagonal matrix obtained from the coarser level and employ the prolongated Lanczos basis to initialize the power steps of the Lanczos extension in the realm of Algo. 6.

3.2 Robust Multiscale Lanczos recursion

The algorithm described in Sec. 3.1 allows propagating Lanczos bases from coarse to fine levels, using reverse-engineered warm-starts for the iterative solves. In the best case (which happens often for well-conditioned systems), this approach is sufficient and leads to a simple multiscale eigenvector extraction strategy.

Unfortunately, the spectral properties of the systems might drift slightly throughout the hierarchy, and Lanczos bases allowing extracting n eigenvectors at a given level might only allow us to extract $n' < n$ eigenvectors at the next level after propagation. To cope with this issue, we devise simple multiscale schemes to sanitize the Krylov space throughout the hierarchy (see Algo. 7).

3.2.1 Multiscale Krylov enrichment. As increasing the Krylov space size leads generally to better spectral properties (making the extraction sometimes faster than when using a small Krylov space size), we change its size adaptively if need be.

ALGORITHM 7: Our MultiGrid IRL method (MG-IRLM)

Input: Levels $\ell = L, \dots, 0$ with SPD pairs (A^ℓ, M^ℓ) and prolongations $P^{\ell \to \ell-1}$;
number of requested eigenpairs n_{ev};
initial Krylov size n_K; maximal Krylov size $K_{\max}$; tolerance ε
Output: Converged eigenpairs on finest level $\ell = 0$

Coarsest-level initialization ($\ell = L$):;
Standard IRLM $(V^L, T^L) \leftarrow \text{IRLM}(A^L, M^L, n_K)$ for n_{ev}
Hierarchical refinement (coarse → fine):;
for $\ell = L - 1$ **down to** 0 **do**
  Prolongation:;
  $V^\ell \leftarrow P^{\ell+1 \to \ell} V^{\ell+1}, \quad T^\ell \leftarrow T^{\ell+1}$;
  Set target Krylov size $k^\ell_{\text{target}} \leftarrow \dim(V^{\ell+1})$;
  Warm-start refinement:;
  while *nb. of converged eigenpairs* $< n_{\text{ev}}$ ***and*** $\dim(V^\ell) < \dim(V^{\ell+1})$ **do**
    Extend (V^ℓ, T^ℓ) using Algorithm 6;
    $k^\ell \leftarrow k^\ell + 1$;
  A) Multiscale Krylov enrichment: (if necessary) // 3.2.1;
  while *nb. of converged eigenpairs* $< n_{\text{ev}}$ ***and*** $k^\ell < K_{\max}$ **do**
    Extend (V^L, T^L) by one standard Lanczos step;
    for $c = L$ **down to** $\ell + 1$ **do**
      Prolongate newly generated Lanczos vector: $v^{c-1}_{\text{new}} \leftarrow P^{c \to c-1} v^c_{\text{new}}$;
      Extend (V^{c-1}, T^{c-1}) by one warm-started Lanczos step;
    Extend (V^ℓ, T^ℓ) using the injected vector
  B) Multiscale QR-shift: (if necessary) // 3.2.2;
  if *nb. of converged eigenpairs* $< n_{\text{ev}}$ ***and*** $\dim(V^\ell) = K_{\max}$ **then**
    Perform implicit restart with QR shifts, retaining n_{ev} Ritz vectors, sanitizing lower-levels as well:;
    $\{(V^\ell, T^\ell), \{Q_i\}\} \leftarrow \text{IRLM\_Restart}(V^\ell, T^\ell, n_{\text{ev}})$;
    for $c = L$ **down to** $\ell + 1$ **do**
      ApplyQRShiftAndTruncate$(V_c, T_c, \{Q_i\})$;
    Go to A) after truncation.
return eigenpairs extracted from (V^0, T^0);

We keep the Lanczos bases $\{V_c, T_c\}$ for all levels c of the hierarchy. When extending the Krylov space, we simply extend the Lanczos basis at the coarsest level (Algo. 2) and we then run our multiscale propagation scheme to extend the extended bases coarse-to-fine. This simple strategy is detailed in point **A)** of Algo. 7.

3.2.2 Multiscale QR-shift. If the previous strategy failed after having extended the Krylov spaces to a maximum size, we perform QR-shifts to the Krylov basis at the current (intermediate) level l, and we apply them similarly to the levels below in order to keep our multiscale iteration consistent. This will make it possible to extend the bases if needed later on. This simple strategy is detailed in point **B)** of Algo. 7. Note that this operation is rendered possible by the fact that applying a QR-shift requires simple updates *from the right side* of V only (see Algo. 5), keeping the operation compatible with the performed prolongations done on their left side.

Table 1. **Laplace eigenvector extraction on triangle meshes.** V: vertex count. N: number of requested eigenvectors. We report timings for the preprocessing Pr (for us and MG: multigrid building; for Cholesky: factorization) and the total time Total. We compare the use for the GRAVO multigrid (Gr) [Wiersma et al. 2023] and an Algebraic multigrid (AMG) [Demidov 2020]. The last example (Sea shell) is representative of what can be processed with our technique on our standard laptop. Past this size, crashes are observed due to memory overflow.

Mesh	V↓	N	Ours (AMG) Pr/Total	Ours (Gr) Pr/Total	IRL-Chol Pr/Total
Femur	1.4M	5	0.9s/**9.8s**	13.7s / **31.6s**	16.5s / 23.2s
		50	_ /47.9s	_ / **87.1s**	_ / **46s**
		100	_ /86.9s	_ / **156.4s**	_ / **77.1s**
Spot	1.7M	5	1s/**14.4s**	14.8s/31.2s	27.0s/**36.1s**
		50	_ /**70.8s**	_ / **99.3s**	_ / 80.1s
		100	_ /**138.6s**	_ / **188.7s**	_ / 140.1s
Duck	2.5M	5	2.3s/**10.9s**	21.4s/38.5s	61.1s/**75.7s**
		50	_ /**41.6s**	_ / 86.9s	_ / **124.5s**
		100	_ /**91.3s**	_ / 160.2s	_ / **243.0s**
David	2.6M	5	2.8s/**43.1s**	27.1s/73.6s	95.4s/**112.2s**
		50	_ /201.8s	_ / **257.1s**	_ / **195.6s**
		100	_ /409.3s	_ / **493.9s**	_ / **301.9s**
letterA	3.8M	5	1.9s/**25.8s**	34.3s/73.4s	129.5s/**151.6s**
		50	_ /**163.3s**	_ / 254.7s	_ / **337.5s**
		100	_ /**285.8s**	_ / 386.4s	_ / **424.2s**
Seal	9.9M	5	17.1s/**117.5s**	117.7s/217.5s	820s/ **915.1s**
		50	_ /**601.2s**	_ / 844.2s	_ / **1454.6s**
		100	_ /**1213.1s**	_ / 1944.7s	_ / **3.2h**
Sea shell	12.5M	5	11.1s/**171.9s**	156.6s/362s	~~**1h**~~
		50	_ /**1237.5s**	_ / 1396s	_
		100	_ /**5370s**	_ / **crashed**	_
Sea shell	16.5M	5	19.8s/ **227.4s**	213.1s/415.5s	~~**1h**~~
		50	_ / **1133.5s**	_ / 1341s	_
		100	_ / **crashed**	_ / **crashed**	_

4 Applications

We focus on spectral problems common in Computer Graphics, and analyze our results on Laplace-Beltrami surface eigenvectors, isosurface extraction on tetrahedral meshes, elastic vibration modes on triangle and tetrahedral meshes, and standard spectral image clustering. We performed all tests on a Macbook Air M4 with 32GB of RAM, using 8 cores in OpenMP to accelerate matrix products.

Termination criteria. As discussed in Sec. 2.2.2, we use Ritz residuals to terminate our algorithm. We set tol = 10^{-9} as termination criterion for all our examples. As we use an iterative solver in place of the direct solver requiring a Cholesky decomposition, we have to specify an additional termination criterion for the inner loop solver. We set $\text{tol}_{\text{SOLVER}} = 10^{-6}$ for all our examples.

4.1 Laplace-Beltrami eigenvectors on triangle meshes

Many applications require extracting the Laplace-Beltrami's spectrum. Spectral distances (diffusion, biharmonic, commute-time, ...)

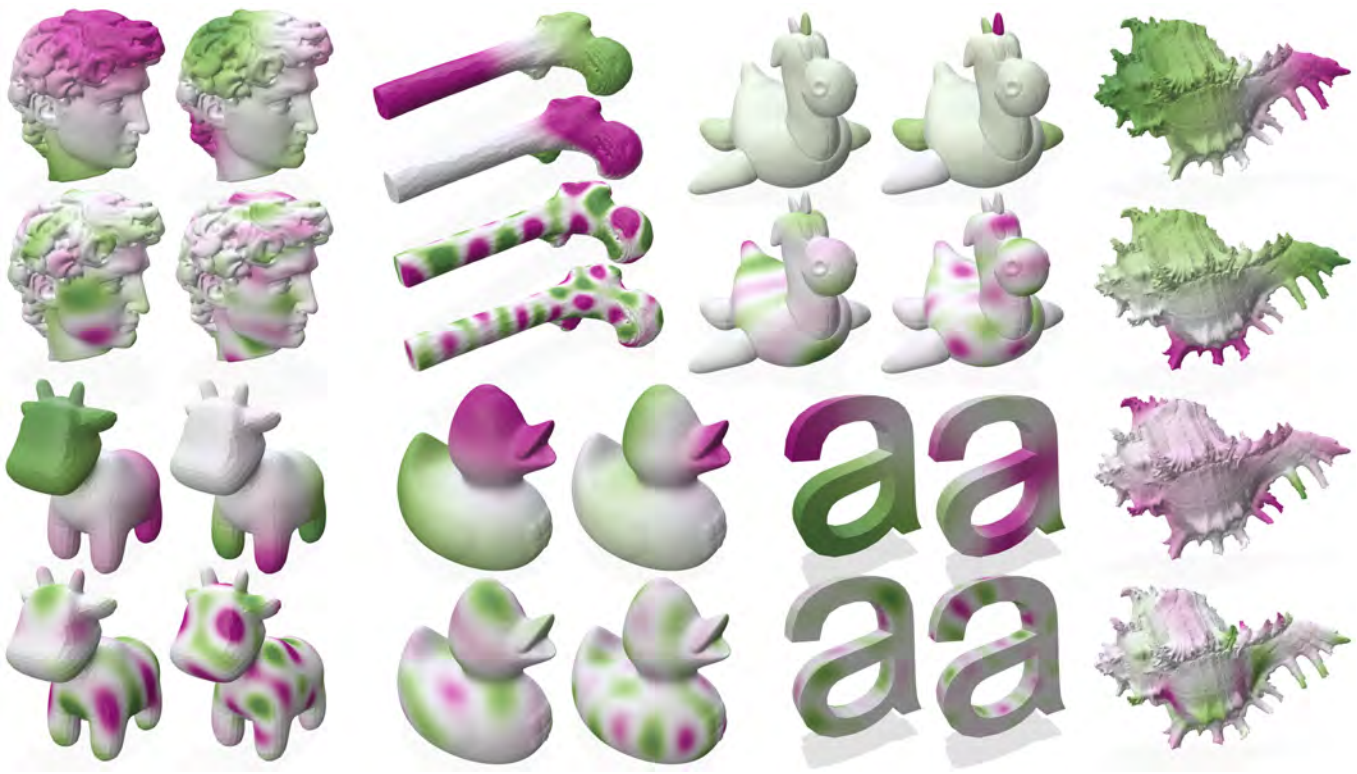


Fig. 2. Laplace-Beltrami eigenvectors computed with our multiscale solver. Note that the Seal example features several connected components.

are for example defined as (signing positively the eigenvalues λ_i):

$$d_{\text{spec}}(u,v,t) := \sum_i (\phi_i(u) - \phi_i(v))^2 \kappa_{\text{spec}}(\lambda_i, t), \tag{3}$$

with $\kappa_{\text{diff}} := \lambda_i^{2t}$, $\kappa_{\text{bih}} := \lambda_i^{-2}$, $\kappa_{\text{commute-time}} := \lambda_i^{-1}$.

Their computations and the one of other descriptors such as the Heat Kernel Signature [Sun et al. 2009] or Functional Maps [Ovsjanikov et al. 2012], require typically a few tens of eigenvectors, that capture smooth low-frequency signals of the input geometries.

We compute Laplacian eigenvectors on input triangle meshes of various complexity (Fig. 2), and report timings in Tab. 1.

Analysis. The Laplace-Beltrami operator on triangle meshes is remarkably simple, making Cholesky factorizations extremely competitive. Nevertheless, we observe in Tab. 1 that our approach is faster than the standard IRL-Cholesky baseline for input of moderate size when extracting a few eigenvectors (Femur, Spot, David), and it is always faster for very large inputs (Duck, LetterA, Seal). Though the AMG seems to give better results than the GMG on this example, those observations apply to both kinds.

4.2 Spectral surface extraction on tetrahedral meshes

We present an application to surface extraction [Alliez et al. 2007], using the formulation proposed in [Zhao et al. 2021] that discretizes the energy on a tetrahedral mesh with the Finite Element method. We maximize the following objective function:

$$\mathcal{E}_{Surf}(f) := \sum_p \nabla f^T \cdot C_p \cdot \nabla f \rightarrow \max, \text{ s.t.} \tag{4}$$
$$\frac{\alpha}{|\mathcal{X}|} \sum_p f(p)^2 + \beta \int_{\mathcal{T}} \|\nabla f(x)\|^2 dx + \gamma \int_{\mathcal{T}} \triangle f(x)^2 dx = 1$$

f being the scalar function whose 0-set is the surface to extract, $\mathcal{X}$ being the input pointset with unknown normals, $C_p = I + \sigma n_p \cdot n_p^T$ being an anisotropy matrix favoring alignment of ∇f with the estimated normal n_p at p, and $\mathcal{T}$ being a tetrahedral mesh on which $\mathcal{E}_{Surf}$ is discretized. Solving Eq. (4) is done through a (largest) general eigenvector extraction. We compare our method with standard IRL methods in Tab. 2, and we illustrate extracted surfaces in Fig. 3.

Analysis. For problems requiring very few eigenvectors, computing a Cholesky decomposition is in general not worthy, as the cost of the preprocessing itself is in general bigger than the time needed for iterative solvers to terminate. This is particularly true for spectral surface extraction (see Tab. 2), which requires **a single** eigenvector.

While the standard usage of MGs with a common IRL method may give reasonable performance, we find our method always faster and more *reliable* than this IRL-MG baseline. The overhead of IRL-MG over our method is variable across our tests: this is due to restarts triggered more randomly, while our approach allows sanitizing quickly Lanczos bases early at coarser scales, where restarts are much less expensive.

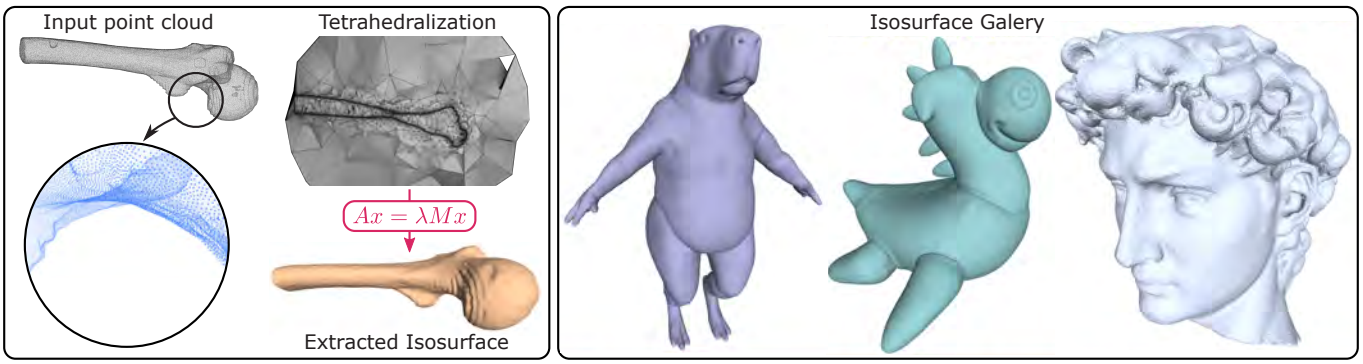


Fig. 3. **Left:** The spectral isosurface extraction method [Alliez et al. 2007] starts by assigning an (unoriented) normal per input point. A smooth implicit function whose gradient aligns with the normals is then optimized on a bounding tetrahedral mesh through a general eigenvector extraction. The surface is computed by contouring the resulting implicit field. **Right:** Additional surfaces extracted with our multiscale solver.

Table 2. **Isosurface extraction on tetrahedral meshes.** P/V: input points count / tetmesh vertex count. We report timings for the preprocessing Pr and total Total as in Tab. 1. We use an Algebraic Multigrid computed with [Demidov 2020]. We extract a single eigenvector in this case, with a Krylov space of size 10. We use for those examples a CG solver with a damped Jacobi V-cycle smoother as preconditioner.

Mesh	P/V↓	Ours (AMG) Pr./Total	IRL-Chol Pr./Total	IRL-MG Pr./Total
David	101k/189k	0.6s/**135s**	152s/157s	0.6s/**472s**
Femur	350k/553k	2s/**114s**	432s/**452s**	2s/132s
Seal	364k/558k	3s/**162s**	537s/**574s**	3s/418s
Spot	420k/665k	3.3s/ **153s**	890s/**981s**	3.3s/288s
Capybara	1.2M/1.9M	13s/**24m**	**crashed**	13s/36m

4.3 Surface vibration modes

Structural analysis is commonly performed by extracting eigenvectors of mechanical energy Hessians. Those eigenvectors are directions in which the energy is least/most changed, and help conveying structural weaknesses to engineers. In computer graphics, analogous vibration modes are widely used to extract meaningful deformation directions that minimize prescribed elastic energies. Such modes are obtained as generalized eigenvectors of

$$Hx = \lambda Mx, \tag{5}$$

H denoting the Hessian of the elastic energy, M the stacked mass matrix, and $x \in \mathbb{R}^{3V}$ the displacement vector the V vertices.

We consider surface meshes on which Dirichlet boundary conditions are enforced by pinning a few vertices. We showcase our solver on the elastic energies described in [Fröhlich and Botsch 2011] (Fig. 4) and report timings in Tab. 3. We used a CG solver preconditioned by a damped 3×3-block Jacobi V-cycle. The scalar prolongation operator is taken from [Wiersma et al. 2023] and is extended to 3D displacements via a Kronecker product with I_3.

Analysis. It is common to require several tens of vibration modes to "nail down" the space of suitable deformations to apply to a mesh during an interactive modeling session. It is thus more interesting to focus on the behaviors emerging for a large enough number of requested eigenmodes ($\geq$ 50). From our tests (Tab. 3), we see that the sweet spot for our approach appears to be between the Armadillo (709k vertices) and Bunny (1M vertices) models. For a big-enough mesh size, performing a Cholesky factorization becomes both too prohibitive (in terms of preprocessing times) and inefficient (in terms of solve performance at each iteration). Our multiscale approach shines in these regimes. Note again that our performance

Table 3. **Vibration modes extraction on triangle meshes.** V: (free) vertex count; N: eigenvectors requested; SK: size of the Krylov space used for the experiments. We report timings for the preprocessing Pr and total Total as in Tab. 1. We use in our experiments the Gravo multigrid [Wiersma et al. 2023]. The timings that are crossed out indicate that we terminated the experiments early. The last example crashed due to memory overflow.

Mesh	V↓	N	SK	Ours Pr./Total	IRL-Chol Pr./Total	IRL-MG Pr./Total
Oilpump	377k	10	30	4s/**207s**	382s/408s	4s/**1090s**
		50	110	_ /620s	_ /**488s**	_ /~~**30m**~~
		100	220	_ /1056s	_ /**586s**	_ /~~**n.a**~~
Blade	581k	10	30	6s/**342s**	721s/770s	6s/**1491s**
		50	110	_ /**978s**	_/998s	_ /~~**1h**~~
		100	220	_ /1726s	_/**1078s**	_ /~~**n.a**~~
Armadillo	709K	10	30	7s/**376s**	1336s/1662s	7s/~~**1h**~~..
		50	110	_ /**1115s**	_ /1616s	_ /~~**n.a**~~
		100	220	_ /2540s	_ /**1755s**	_ /~~**n.a**~~
Bunny	1.01M	10	30	12s/**377s**	2441s/**4121s**	12s/912s
		50	110	_ /**1281s**	_ /~~**2h**~~	_ /~~**5h**~~
		100	220	_ /**2408s**	_ /~~**n.a**~~	_ /~~**n.a**~~
Rockerarm	1.3M	10	30	15s/**632s**	**crashed**	15s/~~**1h**~~
		50	110	_ /**1977s**	_	_ /~~**n.a**~~
		100	220	_ /**3744s**	_	_ /~~**n.a**~~
Coral	1.6M	10	30	15.4s/**642s**	**crashed**	15.4s/ ~~**n.a**~~
		50	110	_ /**2974s**	_	_ / ~~**n.a**~~
		100	220	_ /**crashed**	_	_ / ~~**n.a**~~

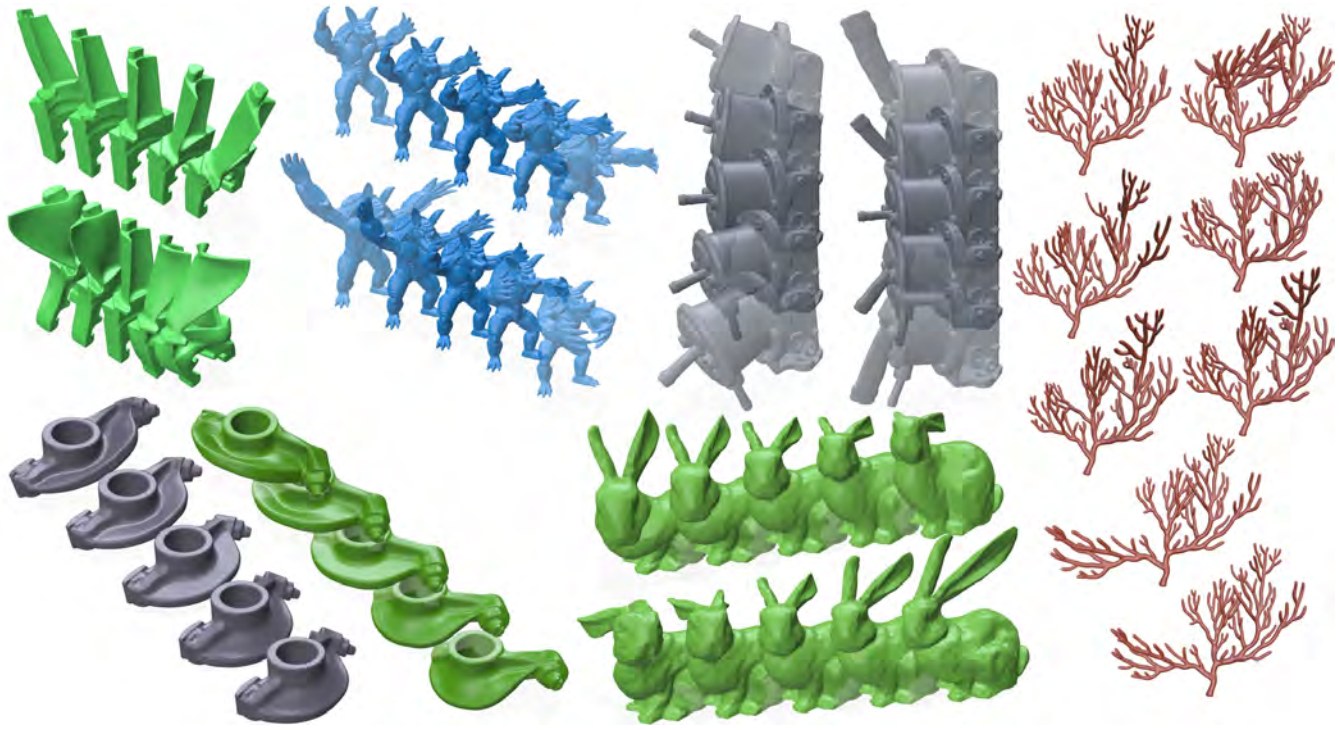


Fig. 4. Vibration modes of an elastic surface energy extracted with our multiscale solver.

Table 4. **Vibration modes extraction on tetrahedral meshes for a linear elasticity model.** V: (free) vertex count; N: eigenvectors requested; SK: size of the Krylov space. We report timings for the preprocessing Pr and total Total as in Tab. 1. We use in our experiments an AMG [Demidov 2020], as we do not have access to GMGs in the case of tetrahedral meshes. We used a BiCGStab iterative solver preconditioned with a block-GS V-cycle. Crossed-out timings indicate that we terminated the experiments early.

			$(\nu = 0.3, \mu = 10^6)$		$(\nu = 0.499, \mu = 10^6)$	
Mesh	N	SK	Ours Pr/Total	IRL-Chol Pr/Total	Ours Pr/Total	IRL-Chol Pr/Total
Wheel	10	30	<1s/**48s**	49s/56s	<1s/410s	62s/**71s**
V=65k	50	110	_ /**71s**	_ /79s	_ /1255s	_ /**96s**
	100	220	_ /**94s**	_/111s	_ /2042s	_/**133s**
Bull	10	30	2s/**318s**	1322s/1428s	2s/**1152s**	1321s/1366s
V=290k	50	110	_ /**606s**	_ /1675s	_ /2894s	_ /**1435s**
	100	220	_ /**807s**	_/1772s	_ /3898s	_/**1529s**
Rod	10	30	3s/**892s**	**crashed**	3s/**1421s**	**crashed**
V=393k	50	110	_ /**2621s**	_	_ /**3926s**	_
	100	220	_ /**3001s**	_	_ /~~2h~~	_
Heart	10	30	3s/**564s**	**crashed**	3s/**2655s**	**crashed**
V=433k	50	110	_ /**1119s**	_	_ /~~2h~~	_
	100	220	_ /**1935s**	_	_ /~~na~~	_

improvements do not come from the use of MGs for the solves only: the worst timings are obtained when using the standard IRLM with the MG solver in place of the Cholesky one (Tab. 3, IRL-MG column).

4.4 Tetrahedral mesh vibration modes

We demonstrate next our solver for the extraction of vibration modes on tetrahedral meshes, for standard mechanical energies (we use in our tests the Hessian of a linear elasticity energy, see [Hughes 2012; Sifakis and Barbič 2012]). We report in Tab. 4 results for two materials settings ($\nu = 0.499, \mu = 10^6$) and ($\nu = 0.3, \mu = 10^6$), and illustrate some in Fig. 5. As for Sec. 4.2, we only used an AMG, for lack of a well-established GMG on tetrahedral meshes. We used in those examples a BiCGStab solver preconditioned with a Block Gauss-Seidel V-cycle, as we noticed that our simpler CG solver was showing unstabilities, hinting at the increased complexity of the operator. As shown in Tab. 4, $\nu = 0.499$ is a challenging case for our approach (much more that the case $\nu = 0.3$): It is well documented (see [Smith et al. 2018]), that the case of nearly incompressible materials with Poisson ratio close to 0.5 is challenging for iterative solvers, due to the ill-conditioned nature of the Hessian.

4.5 Spectral clustering in images

Many spectral methods were developed for Image Processing applications such as clustering or anomaly detection [Ehret et al. 2019; Gula and Bertoldo 2023; Shi 2003]. We show here a simple application to spectral clustering [Zhang et al. 2021].

Spectral graph clustering methods require defining an affinity matrix W, $w_{ij} = w_{ji}$ denoting the affinity between nodes i and j (for images: two pixels), and extracting a normalized Laplacian as:

$$L_N := D^{-1/2}(D - W)D^{-1/2} \tag{6}$$

Fig. 5. Vibration modes on tetrahedral meshes for a linear elasticity model.

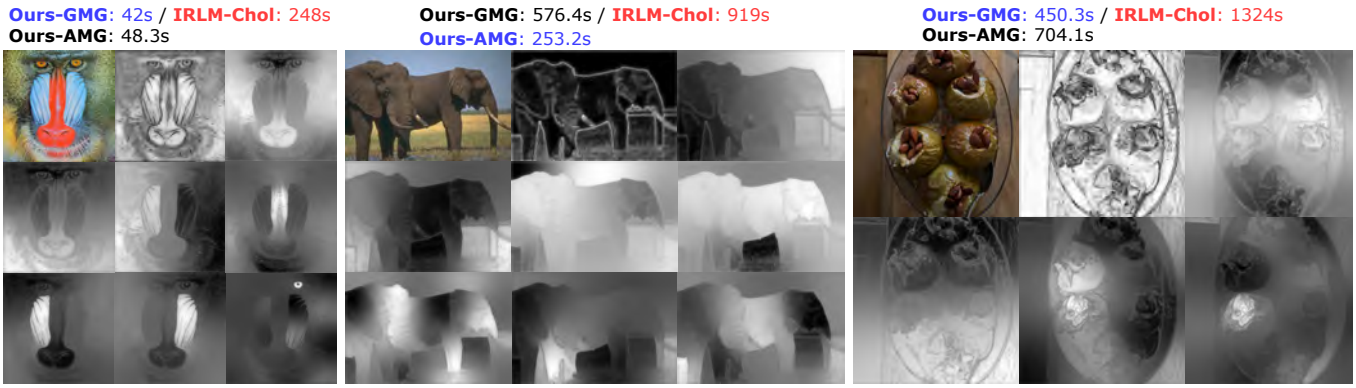


Fig. 6. Some of the 50 eigenvectors of a simple pixel affinity matrix extracted with our multiscale solver.

D being the diagonal mass matrix ($D_{ii} = \sum_j W_{ij}$), used to make the Laplacian row-normalized (akin to a stochastic walk matrix[2]). Low-frequency eigenvectors of L_N are then used as node descriptors, and clustering is obtained through k-means (or alternative) clustering on those descriptors. The choice of the affinity function impacts of course the clustering, and there is a vast literature on the choice of adequate affinity functions for Image Processing.

As explained in [Zhang et al. 2021], a main bottleneck for those approaches is their poor performance on large images, as extracting eigenvectors of very large affinity matrices is extremely costly. This has motivated the development of many approximation methods, among which Super-Pixels and Quad-Tree based approaches.

We present in Fig. 6 results of eigenvectors extracted for a simple affinity function, which is a simplified model of the version used by An & Pellacini [2008]. We use a simple Quad-Tree GMG as well as an AMG to demonstrate the usefulness of our approach on several examples. Surprisingly, while our affinity function results in a long-range operator, we found that our simple GMG was overall competitive with the AMG for this application.

[2]Equivalently, one can extract general eigenvectors of $(D - W)x = \lambda Dx$.

5 Discussion

5.1 Comparison with existing libraries.

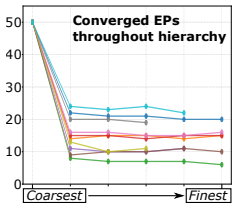
We compared our approach with *our* implementation of the IRLM with a Cholesky factorization, to make sure to identify the benefits of using our multiscale extension in a fair manner: all versions share the same codebase, the only difference is the usage of our MG scheme. We have run experiments with `Spectra` [Qiu 2021], which is based on Eigen and implements the same IRLM with a prefactorization. Our implementation is slightly faster, but similar asymptotic experimental complexities are observed.

5.2 Ablations

5.2.1 Multiscale vs. single-scale iterative solver. While the "natural" baseline we compare to is the commonly-found IRL method that uses a Cholesky factorization to compute direct linear solves, the IRL-MG variant we studied in several applications hints more directly at the efficiency of our multiscale approach used for the Lanczos iteration, as we employ the same solver for both our approach and this single-scale alternative. If very few eigenvectors need be extracted (e.g., Sec.4.2), IRL-MG might be a viable alternative, though still slower than our approach. For very complex problems (e.g., Sec. 4.3), IRL-MG becomes impractical for two reasons. First, less restarts are required using our approach because we filter spectral information efficiently throughout the hierarchy. Second, and very importantly,

the number of iterations per solve are dramatically reduced by our warm-start strategy. **Overall, IRL-MG was never faster than our approach in our tests (neither for few nor many eigenpairs).**

5.2.2 Multiscale Krylov enrichment. We study the importance of our multiscale Krylov enrichment (Secs. 3.2.2 and 3.2.1) . We request 50 eigenvectors of the Hamiltonian operator (Sec. 5.3), using a Krylov space of size 70, disabling steps A) and B) in Algo. 7. As shown in the inset, the number of converged eigenvectors diminishes throughout the hierarchy, resulting sometimes in a significant dropout at the finest level. Note that the likeliness of this behavior appearing is tied to the complexity of the operator.



5.3 Comparison with HSIM [2022]

Unifying the stopping criterion across solvers. A direct comparison between our Lanczos-based approach and HSIM or LOBPCG requires that all methods terminate at eigenpairs of comparable algebraic quality. We perform our comparisons on lowest eigenpairs extraction, which represents most of our presented applications. In this context, our Lanczos iteration runs –following the convention in ARPACK– on the shift-inverted operator $O = (A - \sigma M)^{-1} M$ in the M-inner product (ensuring that output eigenvectors $\{\phi_i\}$ are orthonormal w.r.t this product: $\langle \phi_i, \phi_j \rangle_M = \delta_i^j$), and the residual quantity $\beta_k |e_k^T \xi_i|$ monitored by our termination criterion (Sec. 2.2.2) corresponds to the M-norm of the generalized residual $A\hat{\phi}_i - \lambda_i M \hat{\phi}_i$, up to the standard shift-invert amplification factor. HSIM, in contrast, advocated measuring the M^{-1}-norm of the same quantity. On meshes with highly variable element sizes, the M-norm and M^{-1}-norm of a residual can differ by several orders of magnitude.

To eliminate this discrepancy, we use a simple change of variables $\psi = M\phi$, which transforms the generalized problem $A\phi = \lambda M\phi$ into the standard eigenproblem $AM^{-1}\psi = \lambda\psi$. This yields the shift-inverted operator $\tilde{O} = M(A - \sigma M)^{-1} (= MOM^{-1})$, which is self-adjoint for the M^{-1}-inner product, ensuring $\langle \psi_i, \psi_j \rangle_{M^{-1}} = \delta_i^j$. Running the Lanczos method on $\tilde{O}$ in $\langle \cdot, \cdot \rangle_{M^{-1}}$ preserves all algorithmic properties of the original formulation, but the residual quantity $\beta_k |e_k^T \xi_i|$ now corresponds natively to the M^{-1}-norm of the generalized residual (up to an operator-dependent multiplicative constant). The eigenvectors of the original problem are recovered via $\phi_i = M^{-1}\psi_i$. Note that $\langle \psi_i, \psi_j \rangle_{M^{-1}} = \langle \phi_i, \phi_j \rangle_M = \delta_i^j$.

With this reformulation, we run our Lanczos iteration to convergence and compute $\epsilon_i = \|A\phi_i - \lambda_i M\phi_i\|_{M^{-1}} / \|A\phi_i\|_{M^{-1}}$ for each converged eigenpair. We then set

$$\text{tol}_{\text{HSIM}} = \text{tol}_{\text{LOBPCG}} = \max_i \epsilon_i \tag{7}$$

as the termination threshold passed to HSIM and LOBPCG, ensuring that those must reach the same algebraic accuracy our method delivered, measured in its own native metric. The reformulation makes the comparison like-for-like: both solvers are judged by the same error functional, evaluated identically.

Improvements made to HSIM. Rather than comparing to the vanilla HSIM, which uses a specific GMG that under-performs for very large meshes, we modified the publicly-available implementation to take multigrids as input. Additional modifications include:

- We parallelized Step 4 of their Algorithm 1 (SIM).
- To make HSIM work with the Gravo MG, we added a Tychonov regularizer (10^{-8}) throughout the hierarchy to the Galerkin projections of the stiffness and mass matrices, to ensure numerical stability in the LDL^T decomposition.
- As our machine does not support CUDA, we replaced the dense $p \times p$ CUDA eigensolver (Step 7 of Algo. 1) with a Eigen::GeneralizedEigenSolver, and checked that this part remained negligible.

Table 5. **Comparison with HSIM for the Laplace-Beltrami Eigenproblem.** V: vertex count; N: eigenvectors requested; SK: size of the Krylov space used for the experiments. We report timings for the preprocessing Pr and total Total as in Tab. 1.

Mesh (V↓)	N	SK	Ours AMG Pr/Total	Ours GR Pr/Total	HSIM-GR Pr/Total
Blade	10	30	0.2s/**9.8s**	4.9s/12s	4.9s/**13.7s**
V=615k	50	110	_ /40.8s	_ /**35.6s**	_ /**57.4s**
$\tau = 3e{-}6$	150	330	_ /96.9s	_/**96.7s**	_/**190.9s**
	250	550	_ /**176.2s**	_/171.5s	_/**245.6s**
letter A	10	30	0.3/**29.7**	7.3s/28.4s	7.3s/**22.4s**
V=966k	50	110	_ /**99.8s**	_ /69.8s	_ /**67.1s**
$\tau = 3e{-}4$	150	330	_ /**241.6s**	_/208.6s	_/**182.9s**
	250	550	_ /**421.5s**	_/**349.1s**	_/413.7s
Femur	10	30	0.82s/**30.4s**	13.2/36.6	13.2s/**56.1s**
V=1.4M	50	110	_ /119.1s	_ /**102.8s**	_ /**249.4s**
$\tau = 2e{-}5$	150	330	_ /595s	_/**566.1s**	_/**1210.5s**
	250	550	_ /873s	_/**673s**	_/**crashed**
David	10	30	2.2s/**97.7s**	26.2s/121s	26.8s/**271.6**
V=2.6M	50	110	_ /350s	_ /**322s**	_ /**1105s**
$\tau = 4e{-}7$	150	330	_ /904s	_/**825s**	_/**2624s**
	250	550	_ /1531s	_/**1359s**	_/**crashed**

We report in Tab. 5 results on Laplace eigenmodes and in Tab. 6 (and Fig. 7) results on eigenmodes of the Hamiltonian[3] operator:

$$\left[\triangle + t_{\text{Pot}}(\kappa_1^2 + \kappa_2^2)M\right] \phi = \lambda M\phi. \tag{8}$$

Analysis. We observed that HSIM did not work in practice with AMGs. We suspect that the reason for this failure is that AMGs are built from a specific operator at hand, and using the AMG built from the non-shifted operator becomes inefficient in their setup, as HSIM requires shifting spectrally the operators throughout the hierarchy (Step 12 of their Algorithm 3). This prevented us from running successfully HSIM on several applications for which we only have AMGs available (e.g., Secs. 4.2 and 4.4). We also were not able to extract meaningful vibration modes on surface meshes (Sec. 4.3), for unclear reasons. One might be that the extracted signals are vector-valued, and the prolongation matrices we use are Kronecker products between the GMG prolongation matrices and the 3×3

[3] $(\kappa_1^2 + \kappa_2^2)M$ is the diagonal matrix with entry given by the *integrated* total curvature.

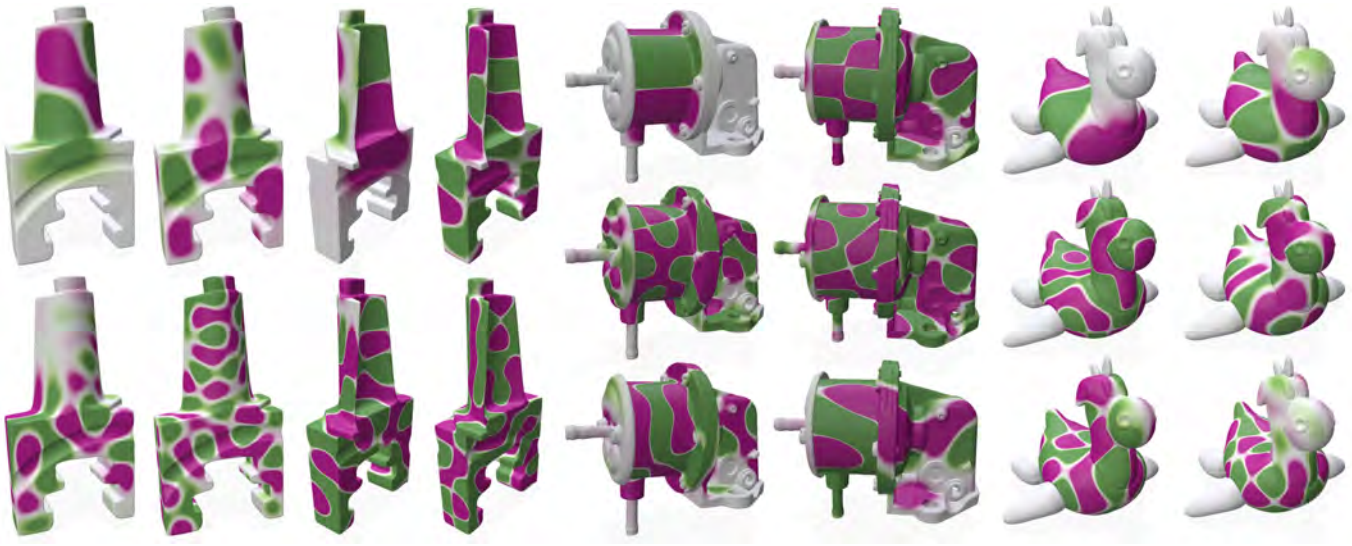


Fig. 7. Eigenvectors of the Hamiltonian operator, for $t_{\text{Pot}} = (0.3, 0.7)$.

Table 6. **Comparison with HSIM for the Hamiltonian Eigenproblem.** V: vertex count; N: eigenvectors requested; SK: size of the Krylov space used for the experiments. We report timings for the preprocessing Pr and total Total as in Tab. 1.

			$t_{\text{Pot}} = 0.3$			$t_{\text{Pot}} = 0.7$		
Mesh (V↓)	N	SK	Ours AMG Pr/Total	Ours GR Pr/Total	HSIM-GR Pr/Total	Ours AMG Pr/Total	Ours GR Pr/Total	HSIM-GR Pr/Total
Seal	10	30	0.06s/**8.5s**	2.9s/**crashed**	2.9s/14.8s	0.06s/**9.5s**	2.9s/**crashed**	2.9s/19.7s
V=364k	50	110	_ /**24.7s**	_ /_	_ /37.5s	_ /**31.4s**	_ /_	_ /34s
$\tau = 4e{-}5$	150	330	_ /**64.4s**	_/_	_/74.1s	_ /82.0s	_/_	_/**79.3s**
Oilpump	10	30	0.08s/11.0s	2.7s/**9.7s**	2.7s/**21.4s**	0.08s/8.6s	2.7s/**8.1s**	2.9s/**19.4s**
V=401k	50	110	_ /35.8s	_ /**25.7s**	_/**40.6s**	_ /32.1s	_ /**23.3s**	_/**39.1s**
$\tau = 8e{-}6$	150	330	_ /**147.7s**	_/**75.3s**	_/114.9s	_ /**157.8s**	_/**75.3s**	_/91.9s
Blade	10	30	0.14s/14.1s	4.9s/**14s**	4.9s/**28.9s**	0.14s/**13.6s**	4.9s/13.9s	4.9s/**29.6s**
V=615k	50	110	_ /**44.3s**	_ /**59.6s**	_ /**59.6s**	_ /49.0s	_ /**35.2s**	_ /**51.6s**
$\tau = 4.5e{-}6$	150	330	_ /**155.1s**	_/**88.4s**	_/173.3s	_ /**159s**	_/**100.1s**	_/140.6s

Identity matrix (differing from the setup presented in [Nasikun 2022]). This requires further investigation however.

Regarding Laplace eigenmode extraction, we see that both our approach and our modified HSIM are comparable for meshes up to 1M vertices, but our approach seems faster for larger meshes.

Regarding the Hamiltonian operator, we observe that HSIM is much more competitive with our method for the settings we tried ($t_{\text{Pot}} = (0.3, 0.7)$). Looking at the shape of the eigenvectors, we notice that those exhibit highly-localized sharp transitions, which may explain why our iterative solver requires more iterations to converge (as those signals are more complicated to decompose across geometric frequencies adapted to the multigrid levels). Note that our approach crashed on the Seal example when using the Gravo multigrid. We suspect, the reason is that the seal has different connected components and therefore the proximity clustering, that is done in Gravo based on a knn tree, might detect a proximity between nearby points even though they are in a different connected component. This would render the topology of the MG incompatible with the one captured in the operator built from the surface.

5.4 Comparison with LOBPCG [2001]

We also compare our method with Locally Block Optimal Preconditioned Conjugate Gradient [Knyazev 2001]. We use as a reference the implementation found in Scipy [Virtanen et al. 2020] and translated it into our C++ codebase. We modified the termination criterion to use also the M^{-1} norm, as in HSIM. As in the comparisons to HSIM, we ran our method and subsequently measured the relative error in the M^{-1} norm. We set this error as the tolerance value.

The comparative results presented in Tab. 7 highlight several key characteristics regarding the performance and scalability of our method versus the LOBPCG baseline.

Table 7. **Comparison with LOBPCG for the Laplace-Beltrami Eigenproblem.** V: vertex count; N: eigenvectors requested; SK: size of the Krylov space. Timings are reported as Pr/Total as in Tab. 1.

Mesh (V↓)	N	SK	Ours AMG Pr/Total	Ours GR Pr/Total	L-AMG Pr/Total	L-GR Pr/Total
Blade	5	15	0.2s / **7s**	4.9s / 10.9s	0.2s /7.1s	4.9s / **14.1s**
V=615k	10	30	_/**9.8s**	_/12s	_ / 11.7s	_ / **19.5s**
	15	40	_ / **13s**	_ / 19.7s	_ / **52.4**	_ / 38.7s
	20	50	_ / **17.4s**	_ / 24.4s	_ / **198s**	_ / 74.5s
letter A	5	15	_ / **12.5s**	_ / 14.3s	0.3s / 16.2s	6.6s / **18.4s**
V=966k	10	30	0.3s/29.7s	7.3s/28.4s	_ / **42.4s**	_ / **27.9s**
	15	40	_ / 36.8s	_ /**32.9s**	_ / **169s**	_ / 69.1s
	20	50	_ / 41.7s	_ / **38.3s**	_ / **1190s**	_ / 95.4s
Femur	5	15	_ / 22.8s	_ / 28.5s	0.8s / **21.4s**	12.5s / **30.4s**
V=1.4M	10	30	0.8s/**30.4s**	13.2s/36.6s	_ / **44.5s**	_ / 39.7s
	15	40	_ / 56.6s	_ / **53.2s**	_ / **202.9s**	_ / 95.9s
	20	50	_ / 70.2s	_ / **64.7s**	_ / ~~**30min**~~	_ / 149.5s
David	5	15	_ / 67.2s	_ / **68.9s**	2.3s / **49.2s**	25.8s / 66.7s
V=2.6M	10	30	2.2s/**97.7s**	26.2s/121s	_ / **182.3s**	_ / 113.4s
	15	40	_ / 137.3s	_ / **122.2s**	_ / **258s**	_ / 162s
	20	50	_ / 171s	_ / **152.2s**	_ / ~~**30min**~~	_ / 256s

Scalability. A primary observation regards the behavior of the solvers as the number of requested eigenvectors N increases. For a very small number of eigenpairs ($N = 5$), LOBPCG is highly competitive, often matching or slightly exceeding the performance of our method (e.g., on the David mesh). However, as N grows to 20, LOBPCG exhibits a non-linear increase in computation time. For instance, on the David mesh, our Gravo-based method scales from 68.9s to 152.2s (2.2× increase) when moving from $N = 5$ to $N = 20$, whereas LOBPCG-GR moves from 66.7s to 256s (3.8× increase), and LOBPCG-AMG fails to converge within the 30-minute limit.

Preconditioner Efficiency. The data confirms that the choice of preconditioner is critical for LOBPCG [Benner and Mach 2011]. Interestingly, while AMG provides a strong start for small N, the GMG demonstrates superior robustness for larger values of N. This is particularly evident on the *letter A* and *Femur* models, where LOBPCG-GR consistently outperforms LOBPCG-AMG for $N \geq 15$.

5.5 Failed experiments & Choice of solver.

We have tested several iterative solvers, to see which are better suited for the several problems we studied. We noticed that ill-conditioned problems featuring non-trivial kernels (dim > 1) proved particularly challenging. We report here three different failed attempts at using our multiscale solver:

- computing vibration modes for the *free* elasticity problem (ie, without any pinning – Sec. 4.3);
- computing eigenmodes of the Surface Hessian energy on triangle meshes with open boundaries [Stein et al. 2018];
- computing eigenmodes of the 1-form Laplacian [Fisher et al. 2007] for direction field design on surface triangle meshes.

The use of iterative solvers for the free elasticity problem has been largely documented in the scientific literature, and this problem is known to be difficult.

Regarding the second case, we were surprised by the instability shown by CG/BiCGStab solvers, considering that this operator's eigenvectors have the same smoothness properties as the Laplace-Beltrami operator, with only their natural boundary conditions differing. We believe these issues stem from the fact that this operator has an increasing number of non-trivial kernel vectors as the number of open boundaries in the mesh grows.

Regarding the third case, while we were able to compute eigenmodes of the 1-form Laplacian (see inset), our solver was consistently outperformed by our IRL-Cholesky baseline, due to the poor convergence rate obtained with the BiCGStab solver. A possible direction for future work would be to develop a more efficient preconditioner for the 1-form Laplacian based on *operator adapted wavelets*, cf. [Budninsky et al. 2019]. There is a rich literature on the use of iterative solvers for ill-conditioned problems [Chen et al. 2021; Saad 2003]. An important direction for future work is to study whether more stable solvers can be used off-the-shelf with our approach, or whether their use in a multiscale setup requires deeper analysis.

5.6 Choice of MG scheme.

We observe that the ideal choice of MG type (AMG, GMG) is highly application-dependent. While our work was originally designed upon the intuition that GMGs would be ideal candidates for propagating Lanczos signals, we observed that the use of AMGs led to competitive performance across a variety of applications in practice.

Our current elasticity hierarchy for GMGs is constructed via a Kronecker-product extension of scalar MG operators. While this approach is translation-preserving and easy to implement, it does not explicitly preserve rotational invariance and therefore fails to maintain the full elastic near-kernel on coarse levels. Many works address elasticity-aware coarsening and homogenization strategies [Chen et al. 2019; Kharevych et al. 2009], and building a MG hierarchy based on these ideas may lead to more robust elasticity solvers; investigating their effectiveness in our setting is left for future work.

5.7 GPU acceleration.

The core of our method should be possible to parallelize on the GPU for simple iterative solvers (eg., Conjugate Gradient with a damped Jacobi MG preconditioner), as it requires essentially:

(1) sparse matrix-vector products to prolongate signals,
(2) damped Jacobi iterations, which are embarrassingly parallel,
(3) dense vector-vector dot products at the core of the Conjugate Gradient iteration, which are not too difficult to parallelize.

The use of more complex solvers might prove necessary for very challenging problems, and it is unclear to us how those solvers could be parallelized on the GPU. Nevertheless, we believe that speeding-up significantly the eigenvector extraction of simpler problems such as the Laplace-Beltrami operator, spectral isosurfacing or the computation of pinned-down vibration modes, would find applications in interactive Geometry Processing of large meshes.

5.8 MG-accelerated Arnoldi iterations

We have targeted the case of SPD matrices, and have extended the Implicitly-Restarted Lanczos Method to a multiscale computation setup. To be able to handle general (in particular, non-symmetric) matrices, we would have to extend the Arnoldi iterative algorithm instead, which differs mainly from the Lanczos method in that the Arnolid recurrence gives rise to an upper Hessenberg matrix ($H_{ij} = 0\ \forall i > j + 1$) instead of a simpler tridiagonal matrix T as in Eq. (2).

Our strategy could be in principle used on Arnoldi iterations. Studying whether the resulting warm-start vectors can be of help to accelerate iterative solvers (e.g GMRES with a multigrid preconditioner) used in the Arnoldi iteration is an interesting future work, which would extend the applicability of our method to the efficient general eigenvector extraction of a larger class of matrix pairs.

5.9 Conclusion

We introduced a scalable multiscale algorithm for extracting eigenfunctions arising in large-scale geometric problems. Our experiments show that the method applies robustly across a range of applications, including operators with different sparsity patterns, discretizations, and conditioning regimes.

When a direct Cholesky factorization is feasible and inexpensive, such approaches remain competitive and are often preferable due to their simplicity and reliability. However, for problem sizes where direct methods become prohibitively costly in terms of memory or preprocessing time, our approach provides an effective alternative.

Compared to a naïve multiscale strategy in which a standard eigensolver is combined with a multigrid-preconditioned linear solver, our method exhibits more reliable convergence and improved numerical behavior in practice. By incorporating multiscale structure directly into the eigensolver rather than treating multigrid solely as a black-box accelerator, we obtain a more effective use of the hierarchy, particularly for challenging large-scale problems.

Our results indicate that multiscale eigensolvers are a promising direction for geometric processing tasks at scale. Further improvements—especially in the design of hierarchy construction and operator coarsening, remain an important avenue for future work.

Acknowledgments

The first author was supported by an IP Paris graduate fellowship and a Monge complement from École Polytechnique, and conducted part of this work during a research internship at Adobe Research. The surface meshes used in this work come from the dataset of [Myles et al. 2014], and the tetrahedral meshes from [Zhou and Jacobson 2016], tetrahedralized with TetGen [Si 2015]. The authors thank Tamy Boubekeur and Jiong Chen for helpful discussions.

References

Mark Adams and Jim Demmel. 1999. Parallel Multigrid Solver for 3D Unstructured Finite Element Problems. In *Proceedings of the ACM/IEEE Conference on Supercomputing (SC'99)*. ACM, Portland, Oregon, USA, 27.

Burak Aksoylu, Andrei Khodakovsky, and Peter Schröder. 2005. Multilevel Solvers for Unstructured Surface Meshes. *SIAM Journal on Scientific Computing* 26, 4 (2005), 1146–1165. doi:10.1137/S1064827503430138 arXiv:https://doi.org/10.1137/S1064827503430138

Pierre Alliez, David Cohen-Steiner, Yiying Tong, and Mathieu Desbrun. 2007. Voronoi-based variational reconstruction of unoriented point sets. In *Symposium on Geometry processing*, Vol. 7. 39–48.

Xiaobo An and Fabio Pellacini. 2008. Appprop: all-pairs appearance-space edit propagation. In *ACM SIGGRAPH 2008 papers*. 1–9.

Peter Arbenz. 2016. *Lecture Notes on Solving Large Scale Eigenvalue Problems*. https://people.inf.ethz.ch/arbenz/ewp/Lnotes/lsevp.pdf Lecture notes, ETH Zürich, Spring semester.

Peter Benner and Thomas Mach. 2011. Locally optimal block preconditioned conjugate gradient method for hierarchical matrices. *PAMM* 11, 1 (2011), 741–742.

Achi Brandt. 1973. Multi-Level Adaptive Technique (MLAT) for Fast Numerical Solution to Boundary Value Problems. In *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*. Springer, 82–89.

Max Budninsky, Houman Owhadi, and Mathieu Desbrun. 2019. Operator-adapted wavelets for finite-element differential forms. *J. Comput. Phys.* 388 (2019), 144–177.

Honglin Chen, Hsueh-Ti Derek Liu, Alec Jacobson, and David I.W. Levin. 2020. Chordal Decomposition for Spectral Coarsening. *ACM Trans. Graph.* 39, 6 (2020).

Jiong Chen, Max Budninskiy, Houman Owhadi, Hujun Bao, Jin Huang, and Mathieu Desbrun. 2019. Material-adapted refinable basis functions for elasticity simulation. *ACM Trans. Graph.* 38, 6, Article 161 (Nov. 2019), 15 pages. doi:10.1145/3355089.3356567

Jiong Chen, Florian Schäfer, Jin Huang, and Mathieu Desbrun. 2021. Multiscale Cholesky preconditioning for ill-conditioned problems. *ACM Trans. Graph.* 40, 4, Article 81 (July 2021), 13 pages. doi:10.1145/3450626.3459851

Yuxuan Chen. 2021. *Multigrid methods for complex engineering geometries and unstructured meshes*. Ph. D. Dissertation.

Denis Demidov. 2020. AMGCL – A C++ library for efficient solution of large sparse linear systems. *Software Impacts* 6 (2020), 100037. doi:10.1016/j.simpa.2020.100037

Thibaud Ehret, Axel Davy, Jean-Michel Morel, and Mauricio Delbracio. 2019. Image anomalies: A review and synthesis of detection methods. *Journal of Mathematical Imaging and Vision* 61, 5 (2019), 710–743.

Matthew Fisher, Peter Schröder, Mathieu Desbrun, and Hugues Hoppe. 2007. Design of tangent vector fields. *ACM transactions on graphics (TOG)* 26, 3 (2007), 56–es.

Stefan Fröhlich and Mario Botsch. 2011. Example-driven deformations based on discrete shells. In *Computer graphics forum*, Vol. 30. Wiley Online Library, 2246–2257.

Tetiana Gula and João PC Bertoldo. 2023. Gaussian Image Anomaly Detection with Greedy Eigencomponent Selection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4110–4118.

Wolfgang Hackbusch. 1985. *The Multi-Grid Method of the Second Kind*. Springer Berlin Heidelberg, Berlin, Heidelberg. doi:10.1007/978-3-662-02427-0_16

T.J.R. Hughes. 2012. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications. https://books.google.fr/books?id=cHH2n_qBK0IC

Lily Kharevych, Patrick Mullen, Houman Owhadi, and Mathieu Desbrun. 2009. Numerical coarsening of inhomogeneous elastic materials. *ACM Trans. Graph.* 28, 3, Article 51 (July 2009), 8 pages. doi:10.1145/1531326.1531357

Andrew V. Knyazev. 2001. Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method. *SIAM J. Sci. Comput.* 23, 2 (Jan. 2001), 517–541. doi:10.1137/S1064827500366124

Cornelius Lanczos. 1950. An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators. *J. Res. Nat. Bur. Standards* 45, 4 (1950), 255–282. doi:10.6028/jres.045.026

R. B. Lehoucq, D. C. Sorensen, and C. Yang. 1997. *ARPACK: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods*. Available from netlib@ornl.gov.

Hsueh-Ti Derek Liu, Alec Jacobson, and Maks Ovsjanikov. 2019. Spectral Coarsening of Geometric Operators. *ACM Transactions on Graphics* (2019).

Hsueh-Ti Derek Liu, Jiayi Eris Zhang, Mirela Ben-Chen, and Alec Jacobson. 2021. Surface multigrid via intrinsic prolongation. *arXiv preprint arXiv:2104.13755* (2021).

Ronald B. Morgan and Zhao Yang. 2018. Two-Grid and Multiple-Grid Arnoldi for Eigenvalues. *SIAM Journal on Scientific Computing* 40, 5 (2018), A3470–A3494. doi:10.1137/16M106226

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust field-aligned global parametrization. *ACM Trans. Graph.* 33, 4, Article 135 (July 2014), 14 pages. doi:10.1145/2601097.2601154

Ahmad Nasikun. 2022. *Efficient Methods for Spectral Geometry Processing*. Ph. D. Dissertation. TU Delft. doi:10.4233/uuid:2bd86c48-8b81-4a66-838f-c85bdb7db334

Ahmad Nasikun and Klaus Hildebrandt. 2022. The Hierarchical Subspace Iteration Method for Laplace–Beltrami Eigenproblems. *ACM Trans. Graph.* 41, 2, Article 17 (Jan. 2022), 14 pages. doi:10.1145/3495208

Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. 2012. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (ToG)* 31, 4 (2012), 1–11.

B.N. Parlett. 1980. *The Symmetric Eigenvalue Problem*. Society for Industrial and Applied Mathematics. https://books.google.fr/books?id=YAm9Ny6Z7PkC

Yixuan Qiu. 2021. Spectra: A Header-Only C++ Library for Large Scale Eigenvalue Problems. https://github.com/yixuan/spectra.

J. W. Ruge and K. Stüben. 1987. Algebraic Multigrid. (1987), 73–130. doi:10.1137/1.9781611971057.ch4 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611971057.ch4

Y. Saad. 2003. *Iterative Methods for Sparse Linear Systems* (2nd ed.). Society for Industrial and Applied Mathematics, USA.

Y. Saad. 2011. *Numerical Methods for Large Eigenvalue Problems: Revised Edition*. Society for Industrial and Applied Mathematics. https://books.google.fr/books?id=5gm8B4NwVRYC

Shi. 2003. Multiclass spectral clustering. In *Proceedings ninth IEEE international conference on computer vision*. IEEE, 313–319.

Xiaohan Shi, Hujun Bao, and Kun Zhou. 2009. Out-of-core multigrid solver for streaming meshes. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 1–7. doi:10.1145/1618452.1618519

Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2, Article 11 (Feb. 2015), 36 pages. doi:10.1145/2629697

Eftychios Sifakis and Jernej Barbič. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses*. ACM, New York, NY, USA. Course notes.

Breannan Smith, Fernando De Goes, and Theodore Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Trans. Graph.* 37, 2, Article 12 (March 2018), 15 pages. doi:10.1145/3180491

D. C. Sorensen. 1992. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.* 13, 1 (Jan. 1992), 357–385. doi:10.1137/0613025

Oded Stein, Eitan Grinspun, Max Wardetzky, and Alec Jacobson. 2018. Natural boundary conditions for smoothing in geometry processing. *ACM Transactions on Graphics (TOG)* 37, 2 (2018), 1–13.

K. Stüben. 2001. A review of algebraic multigrid. *J. Comput. Appl. Math.* 128, 1 (2001), 281–309. doi:10.1016/S0377-0427(00)00516-1 Numerical Analysis 2000. Vol. VII: Partial Differential Equations.

Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. 2009. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, Vol. 28. Wiley Online Library, 1383–1392.

U. Trottenberg, C.W. Oosterlee, and A. Schuller. 2001. *Multigrid Methods*. Elsevier Science. https://books.google.de/books?id=-og1wD-Nx_wC

Petr Vaněk, Jan Mandel, and Marian Brezina. 1996. Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Elliptic Problems. *Computing* 56, 3 (1996), 179–196. doi:10.1007/BF02238511

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. doi:10.1038/s41592-019-0686-2

Ruben Wiersma, Ahmad Nasikun, Elmar Eisemann, and Klaus Hildebrandt. 2023. A Fast Geometric Multigrid Method for Curved Surfaces. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–11.

Zhao Yang. 2015. *A Multigrid Krylov Method for Eigenvalue Problems*. Ph. D. Dissertation. Baylor University. Ph.D. dissertation.

Chongyang Zhang, Guofeng Zhu, Bobo Lian, Minxin Chen, Hong Chen, and Chenjian Wu. 2021. Image segmentation based on multiscale fast spectral clustering. *Multimedia Tools and Applications* 80, 16 (2021), 24969–24994.

Tong Zhao, Pierre Alliez, Tamy Boubekeur, Laurent Busé, and Jean-Marc Thiery. 2021. Progressive discrete domains for implicit surface reconstruction. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 143–156.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).