Eurographics Symposium on Geometry Processing 2015
Mirela Ben-Chen and Ligang Liu
(Guest Editors)

*Volume 34 (2015), Number 5*

# Fast and Exact (Poisson) Solvers on Symmetric Geometries

M. Kazhdan

Johns Hopkins University, USA

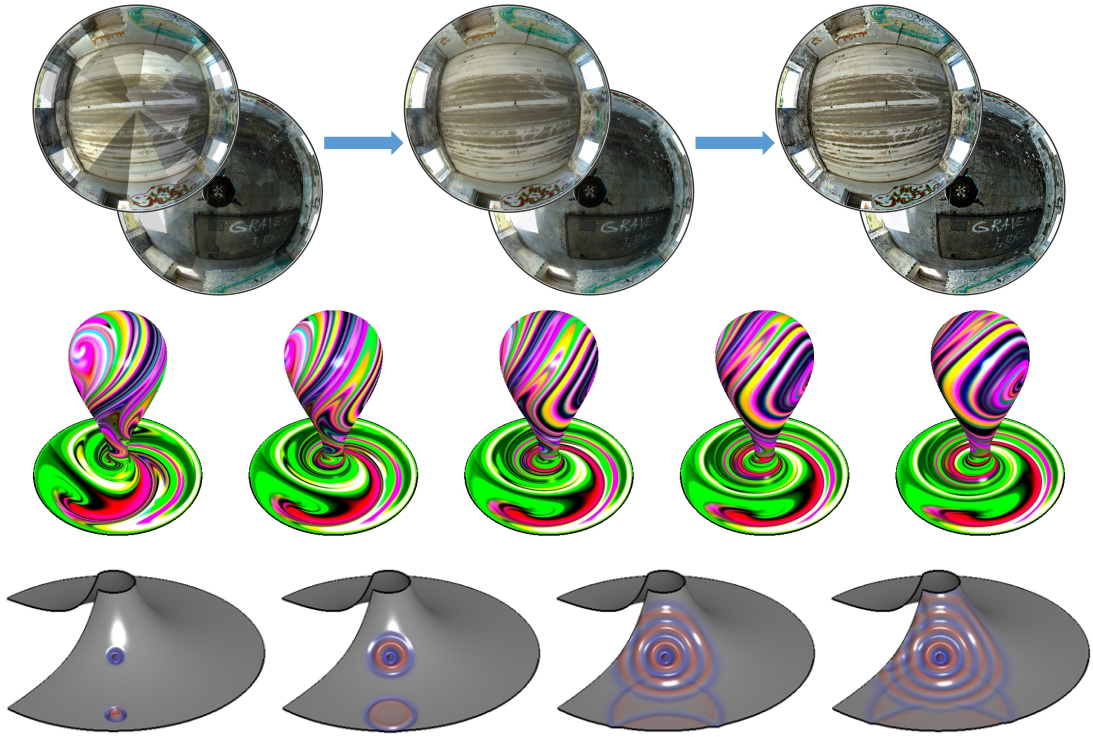

**Figure 1:** *Applications of our solver: (top) spherical image stitching, followed by Laplacian sharpening, (middle) successive frames of incompressible fluid simulation on a surface of revolution, and (bottom) successive frames of wave propagation on a surface of revolution with angular boundary.*

**Abstract**

*In computer graphics, numerous geometry processing applications reduce to the solution of a Poisson equation. When considering geometries with symmetry, a natural question to consider is whether and how the symmetry can be leveraged to derive an efficient solver for the underlying system of linear equations. In this work we provide a simple representation-theoretic analysis that demonstrates how symmetries of the geometry translate into block diagonalization of the linear operators and we show how this results in efficient linear solvers for surfaces of revolution with and without angular boundaries.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Geometric algorithms, languages, and systems—Fluid Simulation

## 1. Introduction

Solving the Poisson equation is an essential step in many graphics applications. For example, it enables gradient domain image processing, it provides the Hodge decomposition that supports fluid simulation, and it describes the oscillation of a membrane governed by the wave equation (Figure 1). As such, there has been a large body of research focused on efficiently finding exact or approximate solutions.

In this work, we consider the problem of solving Poisson-like systems on symmetric geometries. On the one hand, these geometries contain sufficient regularity to enable the design of an efficient exact solver. On the other, they include surfaces with varying curvature – so an efficient solver makes it possible to interactively explore how the surface metric interacts with the Laplace-Beltrami operator.

The key to our approach derives from the observation that when a linear operator commutes with the symmetry group of the geometry, the linear operator becomes block diagonal in the irreducible representations of the group, with (at least) one block associated with each class of irreducible representations. Thus, rather than having to solve one large linear system of equations, we can obtain the solution to the linear system by solving a set of smaller ones.

Taking a representation-theoretic perspective allows us to generalize earlier work in the area. Considering dihedral symmetry, our work explains Hockney's [Hoc65] FFT/tridiagonal solver for Poisson equations on rectangular domains with Neumann/Dirichlet boundaries and provides an extension to surfaces of revolution. And, using the spherical harmonic decomposition, we show how existing approaches for processing surfaces of revolution using the structure of circulant matrices [Dav79, BB06] extends to 3D volumes with full rotational symmetry.

We begin our discussion with a brief description of earlier work on Poisson solvers on Euclidean and non-Euclidean domains (§2). Next, we review some basic results from representation theory and show how these imply block-diagonalizability (§3). We then look in detail at the implications for surfaces of revolution (§4) and also consider the case of volumes with rotational symmetry (§5). We motivate the utility of our approach by considering applications in spherical image processing, as well as stable fluid simulation, and wave propagation on surfaces of revolution (§6) and conclude by summarizing our work and discussing potential directions for future research (§7).

## 2. Related Work

The ubiquity of the Poisson equation has made it well-studied across numerous communities. While a comprehensive survey of related methods is beyond the scope of the paper, we briefly review some of the key approaches for solving the system of equations.

In general, approaches for solving the Poisson equation can be categorized as either *iterative* or *direct* depending on whether an approximate or exact solution is returned.

### Iterative Solvers

This class of approaches is categorized by algorithms that iteratively improve an estimate of the solution. For general formulations of the Poisson equation, greedy techniques like Jacobi and Gauss-Seidel [Saa03] relaxation have been used. While these methods are only guaranteed to improve the solution, other methods like conjugate gradients [She94] return the exact solution in a finite number of iterations.

On their own, these methods are often inefficient because they take too many iterations to produce a reasonable solution. However, these approaches can be integrated with a hierarchical multigrid solver to produce accurate answers in linear time [BHM00]. Though multigrid was initially used for planar lattices, where a hierarchical representation is naturally obtained by coarsening the grid, extensions to other regular geometries [KH10], as well as extensions to triangle meshes [KCVS98, AKS05, SYBF06, CLB*09] and graphs in general [RS87, CFH*00] have also been proposed.

### Direct Solvers

Leveraging the fact that the Laplacian is a (semi-)definite systems makes it possible to apply methods like Cholesky factorization [GL96] to express the system as the product of upper and lower triangular matrices. Then a solution can be obtained using forward and backward substitution. A challenge of using this type of approach is that even though the discretization of the Laplacian is often sparse, its inverse does not need to be, and computing/storing the Cholesky factorization can become prohibitively expensive. For a recent survey of direct solvers in geometry-processing applications, we refer the reader to [BBK05].

Spectral solvers the fact that the cosine and sine functions are eigenvectors of the planar Laplace operator. Thus, a solution to the planar Poisson problem can be obtained by computing the Fourier transform, scaling the Fourier coefficients by the reciprocal of their associated frequency, and then applying the inverse Fourier transform [SS88]. As the Fourier decomposition on a planar grid can be obtained quickly using the FFT [CT65], this algorithm is quite efficient in practice. Although spectral decompositions of the Laplace operator has also been used for signal processing on more general geometries [VL08], this is not a practical method for solving the Poisson system, as computing the spectral decomposition is significantly more expensive than solving the system (e.g. the shift-invert implementation requires solving the Poisson equation to compute the spectral decomposition).

In [Hoc65], Hockney describes an approach that is a hybrid of spectral and direct solvers: Computing the Fourier

transform of along just the rows of a planar grid, the Poisson system reduces to a set of tridiagonal systems of equations – one for each column of Fourier coefficients. This is generalized in [Dav79] which describes how the FFT can be used to diagonalize (block-)circulant matrices and has been used for finding the modes of vibration of surfaces of revolution [BB06]. Taking a representation theoretic perspective, we extend this approach to general symmetry groups.

## 3. Representation Theory and Block Diagonalization

We begin by briefly reviewing some basic representation theory ([Ser77, FH91]). Using this theory, we show that if a linear operator commutes with the symmetry group, the linear operator can be expressed in block-diagonal form.

### 3.1. Representation Theory Review

**Definition** Given a finite/compact group $G$ and a complex vector space $V$, a *representation of $G$ on $V$* is a map $\rho$ from $G$ into the automorphisms of $V$ satisfying:

$$\rho(e) = \mathrm{Id}.$$
$$\rho(g\cdot h) = \rho(g)\cdot\rho(h) \qquad \forall g,h\in G$$

where $e$ is the identity element in $G$.

**Definition** Given a representation $(\rho,V)$ of $G$, a subspace $W\subset V$ is a *sub-representation* if $\rho(g)(W)\subset W$ for all $g\in G$.
**Theorem** (Maschke)**.** *Given a representation $(\rho,V)$, if $W_1\subset V$ is a sub-representation, then there exists $W_2\subset V$, with $V = W_1\bigoplus W_2$ such that $W_2$ is also a sub-representation.*

**Definition** A representation $(\rho,V)$ is *irreducible* if the only sub-representations of $V$ are $\{0\}$ and $V$ itself.

**Definition** Given two representations $(\rho_1,V_1)$ and $(\rho_2,V_2)$, a linear map $L:V_1\to V_2$ is said to be *$G$-linear* if:

$$L\circ\rho_1(g) = \rho_2(g)\circ L \qquad \forall g\in G.$$

If $L$ is an isomorphism, then the irreducible representations are said to be *isomorphic*.
**Lemma** (Schur)**.** *If $L$ is a $G$-linear map between irreducible representations then $L=0$ or $L$ is an isomorphism.*
**Corollary** (Schur)**.** *The only $G$-linear maps from an irreducible representation into itself are multiples of the identity.*
**Theorem** (Canonical Decomposition)**.** *Given a representation $(\rho,V)$, $V$ can be decomposed as the direct sum of irreducible representations (with multiplicity):*

$$V = \bigoplus_{\omega\in\Omega} V^\omega \qquad \textit{with} \qquad V^\omega = \bigoplus_{k=1}^{m_\omega} V_k^\omega$$

*where $V_k^\omega$ are irreducible representations and $V_k^\omega$ is isomorphic to $V_k^{\tilde\omega}$ if and only if $\omega=\tilde\omega$.*

**Remark** While the isomorphism between $V^\omega$ and the direct sum of $V_k^\omega$ is not unique, the decomposition of $V$ into the direct sum of $V^\omega$ is.

### 3.2. Block Diagonalization

**Lemma 3.1.** *Given a representation $(\rho,V)$, if $L$ is a $G$-linear map then $L(V^\omega)\subset V^\omega$ for all $\omega$.*

**Proof** Let $W\subset V^\omega$ be an irreducible sub-representations. By Schur's Lemma it follows that $L(W)$ must also be an irreducible sub-representation. However, if $L(W)\not\subset V^\omega$ then, using Mashke's Theorem, we can obtain a different decomposition of $V$ into irreducible components, contradicting the uniqueness of the decomposition. □

**Corollary 3.2.** *If $L$ is $G$-linear and $\{v_1^\omega,\dots,v_{d_\omega}^\omega\}$ is a basis for $V^\omega$, then $L$ is block diagonal in the basis $\{v_1^\omega,\dots,v_{d_\omega}^\omega\}_{\omega\in\Omega}$. In particular, solving the system $Lx=b$ can be done by solving $|\Omega|$ systems each of size $d_\omega\times d_\omega$, rather than solving a single system of size $\sum d_\omega\times\sum d_\omega$.*

At first glance, Corollary 3.2 seems counter-intuitive. On the one hand, it suggests that as the symmetry group gets more complicated and the dimensions of the irreducible representations grow, the diagonal blocks become larger and the solver becomes less efficient. On the other hand, we would expect that as the linear system has more symmetries, there should be more opportunity for designing an efficient solver. We show that this is in fact the case.

**Notation** For simplicity, we assume that $V=V^\omega$ (as a $G$-linear map must send each $V^\omega$ back into itself). We write:

$$V = \bigoplus_{k=1}^{m} V_k$$

where the $V_k$ are isomorphic irreducible representations. We set $n$ to be the dimension of $V_k$. And we set $\iota_k:V_1\to V_k$ to be the isomorphism between the irreducible representations.

**Definition** We say that a basis $\{v_1^1,\cdots,v_n^1,\cdots,v_1^m,\cdots,v_n^m\}$, with $v_i^k\in V_k$ for all $1\le i\le n$ and $1\le k\le m$, is *consistent* if:

$$v_i^k = \iota_k(v_i^1).$$

**Lemma 3.3.** *Given a $G$-linear map $L:V\to V$ and given a consistent basis for $V$, the matrix representation of $L$ in this basis consists of $m\times m$ blocks where each block is a multiple of the identity.*

**Proof** It suffices to show that if $\pi_j:V\to V_j$ is the projection onto $V_j$ then in a consistent basis the matrix expression for:

$$\pi_j\circ L : V_i\to V_j$$

is a multiple of the identity.

Using Maschke's theorem, it follows that $\pi_j$ is $G$-linear. Since $L$ and $\{\iota_k\}$ are also $G$-linear, and since $G$-linearity is preserved under composition and inversion, the map:

$$\iota_j^{-1}\circ\pi_j\circ L\circ\iota_i : V_1\to V_1$$

must be $G$-linear as well.

Finally, since $V_1$ is irreducible, the corollary to Schur's Lemma implies that the map is a multiple of the identity. Thus, $\pi_j\circ L:V_i\to V_j$ is a multiple of the identity matrix when expressed in the consistent basis. □

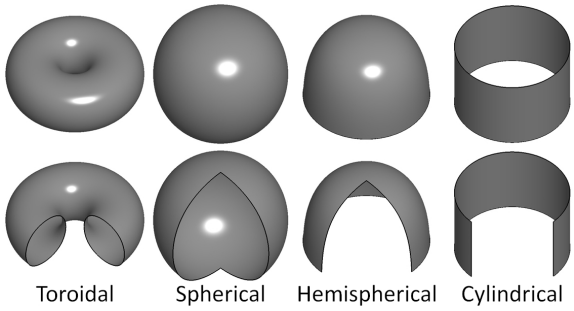


**Figure 2:** *The four topologies our method supports (top), and the corresponding surfaces obtained by sweeping the generating curve by less than* 360° *(bottom).*

**Corollary 3.4.** *If we choose a consistent basis, then each* $dim(V^\omega) \times dim(V^\omega)$ *block of the matrix representation of* $L$ *further decomposes into a block diagonal matrix with* $dim(V_1^\omega)$ identical *blocks, each of size* $m_\omega \times m_\omega$.

## 4. Surfaces of Revolution

We now consider the specific case of solving the Poisson equation on surfaces of revolution.

Our approach applies to four different topologies of surfaces of revolution, as well as "incomplete" surfaces of revolution obtained by rotating a generating curve about the axis of revolution by an angle smaller than 360°. These are visualized in Figure 2 and include geometries that are:

- **Toroidal**: Obtained by sweeping a closed curve that does not intersect the $y$-axis.
- **Spherical**: Obtained by sweeping an open curve, both of whose end-points lie on the $y$-axis.
- **Hemispherical**: Obtained by sweeping an open curve, one of whose end-points lies on the $y$-axis.
- **Cylindrical**: Obtained by sweeping an open curve, both of whose end-points lie off the $y$-axis.

### 4.1. Preliminaries

We assume that we are given a surface of revolution, represented by a 2D generating curve, comprised of the $N$ points $(\mathbf{u}_k, \mathbf{v}_k) \subset \mathbb{R}^2$, with $\mathbf{v}_k \geq 0$ and equality only at the poles of spherical topologies. The surface is then represented by the $M \times N$ grid of points:

$$\mathbf{V}_{j,k} = \left( \mathbf{u}_k \cdot \cos\left(\frac{2j\pi}{M}\right), \mathbf{v}_k, \mathbf{u}_k \cdot \sin\left(\frac{2j\pi}{M}\right) \right).$$

[For spheres the parallels at the poles collapse to a single point and the dimension is $(N-2) \times M + 2$.]

In our indexing, we assume that that the first index (corresponding to the angle of revolution) is taken modulo $M$, (e.g. $\mathbf{V}_{j,k} = \mathbf{V}_{j \pm M,k}$). [For spheres, the indexing at the poles is taken modulo 1 (e.g. $\mathbf{V}_{0,0} = \mathbf{V}_{j,0}$ and $\mathbf{V}_{0,N-1} = \mathbf{V}_{j,N-1}$).]

**Notation** We denote the vertex set of the grid by $\mathscr{V}$ and the edge set by $\mathscr{E}$.

**Notation** We denote the space of discrete functions on the surface by $\mathscr{F} \approx \mathbb{C}^{|\mathscr{V}|}$, which associates a value to every vertex of the grid. We will write elements of the function space in bold capitals, $\mathbf{F}, \mathbf{G} \in \mathscr{F}$.

**Notation** We set $\theta_M = 2\pi/M$ and set $\zeta_M$ to be the $M$-th complex root of unity, $\zeta_M = \cos(\theta_M) + i\sin(\theta_M)$.

### 4.2. Cyclic Symmetry

Taking $G = C_M$ to be the cyclic group with $M$ elements, we obtain a decomposition of $\mathscr{F}$ with respect to the irreducible representations of $G$:

$$\mathscr{F} = \bigoplus_{\omega=-M/2}^{M/2-1} \mathscr{F}_C^\omega$$

where $\mathbf{F}^\omega \in \mathscr{F}_C^\omega$ is a function which, restricted to any parallel, is a complex exponential of frequency $\omega$:

$$(\mathbf{F}^\omega)_{j,k} = \hat{\mathbf{f}}_k^\omega \cdot \zeta_M^{j\omega}.$$

Here, $\hat{\mathbf{f}}^\omega$ is the vector in $\mathbb{C}^N$ giving the $\omega$-th Fourier coefficient of $\mathbf{F}$ along each of the $N$ parallels and the multiplicity $m_\omega$ is equal to $N$ – the number of parallels. [For spheres we have $m_0 = N$ and $m_\omega = N-2$ for $\omega \neq 0$.]

By Corollary 3.2, taking $\{\mathbf{E}^{\omega,1}, \cdots, \mathbf{E}^{\omega,N}\}$ as a basis with:

$$(\mathbf{E}^{\omega,n})_{j,k} = \delta_{n,k} \cdot \zeta_M^{j\omega} \qquad \text{with} \quad \delta_{n,k} = \begin{cases} 1 & \text{if } n = k \\ 0 & \text{otherwise} \end{cases},$$

if $L$ is a linear operator commuting with discrete rotation of the surface about the axis of revolution, then the matrix expression for $L$ in this basis becomes block-diagonal, with an $N \times N$ block for each frequency $\omega$.

This reproduces the approach of [BB06] which uses the FFT to diagonalize systems of linear equations defined by circulant matrices [Dav79], providing an efficient solver by:

1. Computing the Fourier Transform along each row of $\mathbf{G}$ to get the frequency decomposition of $\mathbf{G}$ along the parallels:
$$\mathbf{G} = \sum_{n=1}^{N} \sum_{\omega=-M/2}^{M/2-1} \widehat{\mathbf{G}}_{\omega,n} \cdot \mathbf{E}^{\omega,n}.$$
2. Solving $M$ distinct linear systems of size $N \times N$ to get the coefficients of $\mathbf{F}$ in the basis $\{\mathbf{E}^{\omega,n}\}$.
3. Computing the inverse Fourier Transform to get the values of $\mathbf{F}$ at the grid points.

In addition, if $L$ is local then each $N \times N$ linear system will be diagonally banded and can be solved in $O(N)$ time, giving an overall running time of $O(NM \log M)$ for solving the system $L(\mathbf{F}) = \mathbf{G}$.

### 4.3. Dihedral Symmetry

Taking $G = D_{2M}$ to be the dihedral group with $2M$ elements, we obtain a decomposition of $\mathscr{F}$ with respect to the irreducible representations of $G$:

$$\mathscr{F} = \bigoplus_{\omega=0}^{M/2} \mathscr{F}_D^\omega \quad \text{with} \quad \mathscr{F}_D^\omega = \begin{cases} \mathscr{F}_C^0 & \text{if } \omega = 0 \\ \mathscr{F}_C^{-M/2} & \text{if } \omega = M/2 \\ \mathscr{F}_C^\omega \oplus \mathscr{F}_C^{-\omega} & \text{otherwise.} \end{cases}$$

As above, taking $\{\mathbf{C}^{\omega,1},\cdots,\mathbf{C}^{\omega,N},\mathbf{S}^{\omega,1},\cdots,\mathbf{S}^{\omega,N}\}$ to be a basis for $\mathscr{F}_D^\omega$, with:

$$(\mathbf{C}^{\omega,n})_{j,k} = \delta_{n,k} \cdot \cos(j\omega\theta_M)$$
$$(\mathbf{S}^{\omega,n})_{j,k} = \delta_{n,k} \cdot \sin(j\omega\theta_M),$$

if $L$ is a linear operator that commutes with discrete rotation of the surface about the axis of revolution and reflection through the $xy$-plane, then $L$ is block-diagonal in this basis.

Furthermore, because this basis is consistent, Corollary 3.4 guarantees that the $2N \times 2N$ block is itself block-diagonal, composed of two $N \times N$ blocks, and the linear map $L$ must take the span of $\{\mathbf{C}^{\omega,1},\cdots,\mathbf{C}^{\omega,N}\}$ (respectively $\{\mathbf{S}^{\omega,1},\cdots,\mathbf{S}^{\omega,N}\}$) back into itself.

#### Surfaces with Angular Boundaries

One significant distinction between the analysis we describe above and the original formulation of Hockney [Hoc65] is that our description is targeted at surfaces of revolution, which have a significant amount of symmetry, while the method of Hockney (though restricted to planar signals) also applies to domains with boundaries.

Interestingly, the formulation we propose for surface of revolution can be extended to support surfaces of revolution with angular boundaries – obtained by sweeping the generating curve around the axis of revolution by an angle smaller than 360°. (Boundaries resulting from sweeping an open generating curve by 360° do not require special treatment as they do not affect the symmetry of the surface.) To do this, we imbue the domain with the requisite symmetry by considering the periodic extension. Specifically, to support angular boundaries, we first make the domain periodic by taking a double-covering of the surface where we attach a second copy of the domain at the angular boundaries. This takes us back to the scenario of surfaces of revolution without angular boundaries.

We can then decompose the space of functions on this double-covering into even/odd functions (functions which have the same/negative values on the two pre-images of a point). These spaces correspond to the spaces of functions on the original surface satisfying Neumann/Dirichlet boundary constraints. Although the dihedral group does not map these functions back into themselves, we know that if a linear operator $L$ on the double-covering commutes with the dihedral symmetry group, then it will map the even/odd subspaces back into themselves, giving a block diagonalization of the corresponding linear operator on the surface with boundary.

In particular, since the Laplace operator is isometry invariant, *any* discretization of the operator over a surface of revolution that preserves the (discrete) cyclic/dihedral symmetry will necessarily be block-diagonal in the Fourier basis. The explicit characterization of the condition that needs to be satisfied for a linear operator commute with the action of the cyclic/dihedral group, as well as the equations giving the entries in the diagonal blocks, are given in Appendix A

## 5. Volumes of Rotation

Though this work is motivated by applications to surfaces of revolution, this section considers the example of solving a Poisson equation over the unit ball. This application highlights the generality of the representation-theoretic formulation by demonstrating how $SO(3)$ symmetry can be used to design an efficient solver.

To this end, we consider the case when we are given a function $g : B^3 \to \mathbb{C}$ and we would like to solve for the function $f : B^3 \to \mathbb{C}$ such that:

$$\Delta f = g.$$

For simplicity, we assume trivial Dirichlet boundaries and a homogeneous metric, though the discussion applies to general Dirichlet/Neumann boundary constraints and applies to any radially parameterized metric.

Using the fact that the space of spherical harmonics of frequency $l$ are an irreducible representation for $SO(3)$, we decompose $\mathscr{F}$ as:

$$\mathscr{F} = \bigoplus_{l=0}^{\infty} \mathscr{F}^l$$

where $\mathscr{F}^l$ is the space of functions whose restriction to a fixed radius is a linear sum of degree-$l$ spherical harmonics:

$$\mathscr{F}^l = \left\{ f \in \mathscr{F} \middle| f(\theta,\phi,r) = \sum_{m=-l}^{l} \hat{f}_l^m(r) \cdot Y_l^m(\theta,\phi) \right\}$$

(with $Y_l^m(\theta,\phi)$ the spherical harmonic of frequency $l$ and index $m$).

Furthermore, by Corollary 3.4, for each frequency $l$ there exists a linear operator $\mathscr{L}_l : B^1 \to B^1$ such that:

$$f(\theta,\phi,r) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \hat{f}_l^m(r) \cdot Y_l^m(\theta,\phi)$$
$$\Downarrow$$
$$\Delta f(\theta,\phi,r) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \mathscr{L}_l\left(\hat{f}_l^m(r)\right) \cdot Y_l^m(\theta,\phi).$$

Discretizing over an $O(N \times N \times N)$ grid, this gives a block diagonalization of the Laplace operator into $O(N^2)$ blocks

where the $(l,m)$-th block is of size $O(N \times N)$. As with surfaces of revolution, the locality of the Laplace operator implies that the linear operator defined on each block will be diagonally banded, so the Poisson system can be solved by:

1. Computing the spherical harmonic decomposition for each of $O(N)$ radii.
2. Solving a diagonally banded $N \times N$ system of equations for each of $O(N^2)$ pairs of spherical frequency and index.
3. Computing the inverse spherical harmonic decomposition for each radius.

Since the discretized Laplacian is diagonally banded, the complexity of solving a Poisson system on the unit ball, discretized on an $N \times N \times N$ grid is $O(N^3 \log^2 N)$ using the fast spherical harmonic transform [HRKM03].

**Remark** When the systems is not diagonally banded, Corollary 3.4 still guarantees that a consistent basis results in only $O(N)$ different linear systems that need to be solver. Thus, each $N \times N$ system will be solved $O(N)$ times and the cost of Cholesky pre-factorization can be amortized.

## 6. Results and Discussion

We implement our approach using the FFTW [FJ05] to perform the forward and backward Fourier transforms. (When extending to surfaces with angular boundary, we use the DCT-I transform for Neumann boundary conditions and DST-I transform for Dirichlet boundary conditions.) The tridiagonal solve is implemented using the Thomas Algorithm (extended with the Sherman-Morrison Formula for toroidal geometry) [Saa03]. Additionally, the implementation is trivially parallelized using OpenMP [DM98] as the Fourier transforms of the parallels can be performed simultaneously, as can the solution of the tridiagonal systems for the different frequency components. All experiments were obtained using an Intel Core i7-4710MQ processor with 16 GB of RAM, parallelized across eight (hyper) threads.

Our implementation requires the discretization of the space of functions and associated differential operators. These include: $M^0$ – the 0-form mass matrix; $M^1$ – the 1-form mass matrix; $S$ – the stiffness matrix; $D$ – the divergence operator; $G$ – the finite-difference gradient operator; $\{B_v\}_{v \in \mathcal{V}}$ – the 0-form basis; $\{\vec{B}_e\}_{e \in \mathcal{E}}$ – the 1-form basis; and $\{\vec{\mathbf{H}}_1, \vec{\mathbf{H}}_2\}$ – the harmonic vector field basis, with $\vec{\mathbf{H}}_1$ the gradient of the function giving the angle about the axis of revolution and $\vec{\mathbf{H}}_2$ its rotation by 90°. Details of the discretization are provided in Appendix B. For the results in this section, we used the conical interpretation to define the metric, though we found the visual quality and running time for the trapezoidal interpretation to be similar.

### 6.1. Image Stitching and Contrast Enhancement

#### Image Stitching

In image stitching, the problem is to composite multiple registered images so as to remove discontinuity artifacts.

| | Running Time | | Memory | |
|---|---|---|---|---|
| | Stitching | Sharpening | Stitching | Sharpening |
| [KH10] (out-of-core) | 148.6 (s) | 150.0 (s) | 106 (MB) | 106 (MB) |
| [KH10] (in-core) | 35.0 (s) | 35.1 (s) | 10987 (MB) | 10299 (MB) |
| Our Solver | 4.6 (s) | 4.7 (s) | 5402 (MB) | 5402 (MB) |

**Table 1:** *Solver time and memory usage for processing the spherical* $16K \times 8K$ *panorama in Figure 1(top), comparing the proposed hybrid FFT-tridiagonal solver with the streaming multigrid solver of Kazhdan* et al. *[KH10].*

The general approach for solving this problem [PGB03, ADA*04, LZPW04] proceeds in two steps. First, a target gradient field $\vec{\mathbf{G}}$ is generated by compositing the gradients from the individual images and setting seam-crossing gradients to zero. Then a least-squares system is solved to obtain the function $\mathbf{F}$ whose gradients best match the target. This corresponds to the linear system:

$$S(\mathbf{F}) = D(\vec{\mathbf{G}})$$

with $S$ and $D$ the stiffness and divergence matrices.

#### Contrast Enhancement

In contrast enhancement, the problem is to accentuate the edges in an image while preserving the overall content. As shown by Bhat *et al.* [BCCZ08] the solution can be characterized by the system:

$$(\alpha M^0 + S)\left(\mathbf{F}^{\text{new}}\right) = (\alpha M^0 + \beta S)\left(\mathbf{F}^{\text{old}}\right)$$

with $M^0$ and $S$ the (0-form) mass and stiffness matrices, $\alpha > 0$ the fidelity term, characterizing how similar the processed image should be to the original, and $\beta \geq 0$ the gradient modulation term which indicates how strongly edges should be enhanced. For $\beta < 1$ edges are dampened and a smoothed image is obtained. For $\beta > 1$ edges are amplified and a sharpened image is obtained.

**Results** Figure 1 (top) shows an example of first stitching a $16384 \times 8192$ spherical panorama and then performing contrast enhancement to exaggerate the detail ($\beta > 1$). (Visualizations are obtained by looking down at the North pole, with back/front-face culling enabled.) We compare the performance of our approach to both the in-core and out-of-core streaming multigrid spherical solvers [KH10] in Table 1. (For the streaming multigrid solver we used a single V-cycle with 5 Gauss-Seidel iterations per level.)

As the table shows, our approach outperforms the in-core implementation of the streaming multigrid solver, providing a solution in less time and comparable memory. As our solver is strictly in-core, its memory usage is proportional to

the size of the image. In contrast, the out-of-core implementation of the multigrid solver only requires that a few rows of the image reside in memory at any time.

### 6.2. Fluid Simulation

#### Unconditionally Stable Fluids

In simulating incompressible fluids, the goal is to solve for the temporally evolving vector field describing the velocity of the fluid. As demonstrated by Stam [Sta99, Sta01, Sta03] a stable solution can be implemented as follows:

1. **Adding** external forces.
2. **Advecting** the velocity by back-tracing along flow lines.
3. **Diffusing** the flow field to capture the effects of viscosity.
4. **Projecting** onto the divergence-free vector fields.

We represent the velocity field in the basis $\{\vec{B}_e^{\perp}\}_{e\in\mathscr{E}}$, obtained by rotating the 1-form basis $\{\vec{B}_e\}_{e\in\mathscr{E}}$ by 90°.

**Projection** Given the coefficients, $\vec{\mathbf{W}}$, of the velocity field in this basis we obtain (most of) the divergence-free component by computing the curl of the velocity field, solving for the scalar potential $\mathbf{P}$, computing the gradient, and rotating by 90° in the tangent plane:

$$\pi_{\text{div-free}}(\vec{\mathbf{W}}) = G(\mathbf{P}) \qquad \text{with} \qquad S(\mathbf{P}) = D(\vec{\mathbf{W}}).$$

(Since we are working with the rotated basis, $D(\vec{\mathbf{W}})$ becomes the curl and $G(\mathbf{P})$ becomes the 90° rotation of the gradient.)

**Diffusion** Since diffusion commutes with the gradient operator we implement this step by diffusing the scalar potential $\mathbf{P}$ prior to taking its gradient. This is accomplished by solving the (screened) Poisson equation:

$$\left(\frac{1}{\delta\nu}M^0 + S\right)\left(\mathbf{P}^{\text{diffused}}\right) = \frac{1}{\delta\nu}M^0(\mathbf{P})$$

where $\delta$ is the step-size for the time integration and $\nu$ is the viscosity coefficient. (This is the same equation as for contrast enhancement, with $\alpha = 1/(\delta\cdot\nu)$ and $\beta = 0$.)

Because diffusion and projection are performed in sequence, our implementation only performs one set of forward Fourier transforms before the diffusion phase and one set of inverse Fourier transforms after the projection phase.

**Harmonic Vector Fields** While the above discussion can be used to simulate vector fields on simply connected domains, it ignores the harmonic vector fields – vector fields for which both the divergence and curl operator vanish (since taking the curl annihilates the harmonic component.)

To address this we first project the vector field $\vec{\mathbf{W}}$ onto the (at most) two-dimensional space of harmonic vector fields using the 1-form mass-matrix, $M^1$, and then add the projected component back into the divergence-free vector field after projection. (Harmonic vector fields are fixed under the diffusion operator, so we do not need to incorporate them in the diffusion phase.)

**Boundaries** For surfaces of revolution with boundaries, we use free-slip boundary conditions. In general, this corresponds to using a basis satisfying Neumann boundary constraints (so that the normal derivative vanishes at the boundary). However, since we are rotating the vector field by 90°, we obtain a free-slip boundary by using a basis satisfying Dirichlet boundary constraints.

In this case the space of harmonic vector fields is spanned by $\vec{\mathbf{H}}_2$ when sweeping an open curve completely around the axis of revolution and $\vec{\mathbf{H}}_1$ when sweeping a closed curve by less than 360°. Note that these vectors fields are normal to the boundary, so their rotation by 90° becomes tangent, conforming to the desired free-slip boundary conditions.

**Results** Figure 1(middle) shows an example of our interactive fluid simulation for a surface of revolution with hemispherical topology, sampled on a grid with $M = N = 1024$. The figure shows successive snapshots of the fluid flow, visualized by seeding "ink" in the fluid and visualizing the advection of the ink under the flow.

We compare the performance of our solver to the performance of a direct solver using CHOLMOD's, MKL-backed Cholesky factorization [CDHR08, Int13] with default reordering. (Though we do not pursue it in this work, we believe that it would also be valuable to evaluate the sparsity and run-time performance for more advanced, geometry-driven sparsity-preserving reorderings [BBK05].)

The results of these experiments are shown in Table 2. Our solver is two orders of magnitude faster than Cholesky factorization. Furthermore, using a Cholesky factorization incurs a significant cost in pre-processing time and memory, due to the computational complexity of LU factorization. In contrast, our solver requires very little pre-processing (dominated by setting up the FFTW planner) and has a smaller memory footprint.

As illustrated by the table, our Poisson solver is sufficiently fast that it is no longer the bottleneck in the stable fluid simulation. Indeed, in our interactive simulation we found that most of the computational cost was incurred in the advection phase, where the tracing of flow-lines over the geometry requires repeated sampling of the vector field and unfolding of adjacent faces into a common plane.

For the specific case of spherical geometry, we also compare our solver to the fully spectral solution obtained using the fast spherical harmonic decomposition [HRKM03]. For similar experiments, we found that our solver was roughly five times faster on a $1024\times1024$ grid. This conforms with the theoretical complexities – for an $N\times N$ grid our solver has run-time complexity $O(N^2\log N)$ while the spherical harmonic decomposition has complexity $O(N^2\log^2 N)$.

#### Vorticity Advection

An alternate stable fluid formulation was proposed by Elcott *et al*. [ETK*07] which describes an implementation that

| Resolution | Solver | Initialization | Projection + Diffusion | Advection | Memory | Frame-Rate |
|---|---|---|---|---|---|---|
| $256 \times 256$ | Direct | 0.2 (s) | 0.03 (s) | <0.01 | 200 (MB) | 26 (fps) |
| | Ours | 0.2 (s) | <0.01 (s) | | 46 (MB) | 145 (fps) |
| $512 \times 512$ | Direct | 1.8 (s) | 0.13 (s) | 0.02 | 588 (MB) | 7 (fps) |
| | Ours | 0.6 (s) | <0.01 (s) | | 96 (MB) | 38 (fps) |
| $1024 \times 1024$ | Direct | 8.0 (s) | 0.50 (s) | 0.07 | 2436 (MB) | 1.7 (fps) |
| | Ours | 1.2 (s) | 0.01 (s) | | 293 (MB) | 10 (fps) |
| $2048 \times 2048$ | Direct | 37.0 (s) | 1.79 (s) | 0.26 | 10474 (MB) | 0.5 (fps) |
| | Ours | 2.4 (s) | 0.05 (s) | | 1177 (MB) | 2.1 (fps) |

**Table 2:** *Initialization time, projection time, advection time, memory usage, and frame-rate for fluid simulation on a surface of revolution at different resolutions, comparing the proposed hybrid FFT-tridiagonal solver with a direct solver.*

advects vorticity rather than velocity. A Poisson equation is then solved to compute the flux, and the (rotated by 90°) gradient of the flux gives the velocity.

**Results** Figure 3 shows results for flow on a $1024 \times 1024$ surface of revolution (with angular boundaries). The figure shows vorticity, with counter-clockwise vorticity drawn in blue and clockwise vorticity in red. The top row shows results for similarly oriented vortices placed next to each other – the vortices orbit each other and then merge. The bottom row shows results when the vortices are oppositely oriented – the vortices move in parallel until they reach the boundary and then traverse the boundary in opposite directions.

For these examples, the frame-rate was similar to the frame-rate achieved in implementing Stam's method, and again the computational bottle-neck is advection. (In our implementation we used a simple version of the advection, transforming the vorticity into a 0-form, advecting the 0-form by back-sampling, and then transforming back to a 2-form. This does not guarantee exact circulation preservation as in [ETK*07], but provides the correct limit behavior as the flow is assumed to be divergence-free, so that areas are preserved under advection.)

### 6.3. Wave Propagation

Lastly, we consider the simulation of waves propagating on a surface of revolution, governed by the PDE:

$$\frac{\partial^2 u}{\partial t^2} = a\Delta u - b\frac{\partial u}{\partial t} + f$$

where $1/\sqrt{a}$ is the speed of the wave, $b$ is the damping coefficient, and $f$ is the (time-varying) external force. Using implicit integration gives a sequence of solutions $\{\mathbf{U}^t\}$ with:

$$\left((1+b\delta)M^0 + a\delta^2 S\right)\left(\mathbf{U}^{t+1}\right) = M^0\left((2-b\delta)\mathbf{U}^t - \mathbf{U}^{t-1} + \delta^2\mathbf{F}^{t+1}\right)$$

with $\delta$ the step-size of the time integration and $\mathbf{F}^{t+1}$ the discretized external forces at time $t+1$.

**Results** Figure 1(bottom) shows a visualization of the wave propagation enabled by our system, showing successive time-frames of a simulation with two point sources, on a domain with Dirichlet boundary constraints along the meridians and Neumann boundary constraints along the parallels. The visualization is obtained by rendering the normal-offset surface, with $\mathbf{U}$ taken as the height of the offset. At the bottom we use an instantaneous source, achieved by adding an impulse in both time and space. At the top, we use a spatial impulse, modulated temporally by a cosine function to create a temporally oscillating source.

Table 3 gives the performance of our solver for surfaces of revolution at different resolutions, with different boundary conditions. The solver has similar performance as the fluid simulation solver, though the frame-rate is improved because we no longer need to perform advection and the simulation now bottle-necks on the rendering of the offset surface.

With Neumann/Dirichlet angular boundaries, running time and memory usage deteriorate slightly. We believe that this is due to the slower implementation of the DCT-I and DST-I transforms in FFTW, as well as the fact that when using boundary conditions we have twice as many tridiagonal systems to solve. (For periodic boundaries we solve $M/2$ different systems for the complex coefficients of the Fourier decomposition. For Neumann/Dirichlet boundaries we solve $M$ different systems for the real coefficients of the sine/cosine decomposition.)

## 7. Conclusion

We presented a representation theoretic analysis of how symmetry can be leveraged in defining efficient linear solvers. We use this to define an efficient Poisson solver on surfaces of revolution, with and without angular boundaries, demonstrating applications in image processing, fluid simulation, and wave propagation.

In the future, we would like to explore several extensions.

From a practical perspective, we would like to explore both out-of-core and GPU implementations of the solver. For planar domains, an out-of-core solver has already been

| Resolution | Angular Boundary | Solver | Memory | Frame-Rate |
|---|---|---|---|---|
| $256 \times 256$ | Periodic | <0.01 (s) | 55 (MB) | 162.6 (fps) |
| | Neumann | <0.01 (s) | 56 (MB) | 157.5 (fps) |
| | Dirichlet | <0.01 (s) | 56 (MB) | 155.0 (fps) |
| $512 \times 512$ | Periodic | <0.01 (s) | 135 (MB) | 89.7 (fps) |
| | Neumann | <0.01 (s) | 143 (MB) | 76.3 (fps) |
| | Dirichlet | <0.01 (s) | 144 (MB) | 80.3 (fps) |
| $1024 \times 1024$ | Periodic | 0.01 (s) | 444 (MB) | 23.0 (fps) |
| | Neumann | 0.02 (s) | 464 (MB) | 20.6 (fps) |
| | Dirichlet | 0.02 (s) | 463 (MB) | 20.8 (fps) |
| $2048 \times 2048$ | Periodic | 0.15 (s) | 1682 (MB) | 2.4 (fps) |
| | Neumann | 0.25 (s) | 1767 (MB) | 2.0 (fps) |
| | Dirichlet | 0.29 (s) | 1764 (MB) | 1.9 (fps) |

**Table 3:** *Solver time, memory usage, and frame-rate for wave-propagation on a surface of revolution discretized at different resolutions, with and without boundaries.*

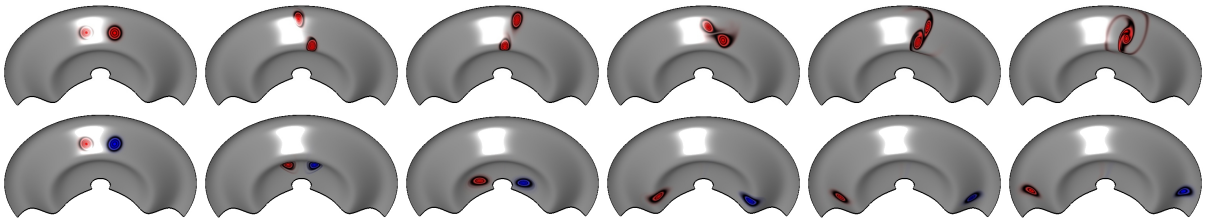

**Figure 3:** *Vorticity evolving under incompressible flow: Showing the results when starting with two adjacent vortices that are consistently oriented (top) and oppositely oriented (bottom). Regions with counter-clockwise vorticity are drawn in blue and regions with negative clockwise vorticity are drawn in red.*

proposed by McCann [McC08] and we believe a similar approach would work for surfaces of revolution. For a GPU implementation we would like to leverage the CUDA implementation of the FFTW [NBGS08].

From a theoretical perspective, we would also like to extend our implementation to the dual formulation, where basis functions are associated with the faces of the discretization, rather than the vertices. This would enable the use of more accurate, second-order, B-splines for discretization and should be realizable using a basis analogous to the one used in the spherical multigrid implementation [KH10].

Finally, we would also like to explore extensions of the approach to other symmetries, taking advantage of efficient implementations of the FFT for other groups [MR96].

**Acknowledgements** The author would like to express sincere gratitude to the members of the "Discrete Differential Geometry" workshop in Oberwolfach 2015 for feedback that was invaluable in shaping the final version of this paper.

## References

[ADA*04] AGARWALA A., DONTCHEVA M., AGRAWALA M., DRUCKER S., COLBURN A., CURLESS B., SALESIN D., COHEN M.: Interactive digital photomontage. *ACM Transactions on Graphics (SIGGRAPH '04)* (2004), 294–302. 6

[AKS05] AKSOYLU B., KHODAKOVSKY A., SCHRÖDER P.: Multilevel solvers for unstructured surface meshes. *SIAM Journal of Scientific Computing 26*, 4 (2005), 1146–1165. 2

[BB06] BINDEL D., BRUYNS C.: Shape-changing symmetric objects for sound synthesis. In *Audio Engineering Society 121* (2006), pp. 875–881. 2, 3, 4

[BBK05] BOTSCH M., BOMMES D., KOBBELT L.: Efficient linear system solvers for mesh processing. In *In IMA Conference on the Mathematics of Surfaces* (2005), Springer, pp. 62–83. 2, 7

[BCCZ08] BHAT P., CURLESS B., COHEN M., ZITNICK L.: Fourier analysis of the 2D screened Poisson equation for gradient domain problems. In *European Conference on Computer Vision* (2008), pp. 114–128. 6

[BHM00] BRIGGS W., HENSON V., MCCORMICK S.: *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, 2000. 2

[CDHR08] CHEN Y., DAVIS T., HAGER W., RAJAMANICKAM S.: Algorithm 887: Cholmod, supernodal sparse Cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software 35*, 3 (2008), 22:1–22:14. 7

[CFH*00] CLEARY A., FALGOUT R., HENSON V., JONES J., MANTEUFFEL T., MCCORMICK S., MIRANDA G., RUGE J.: Robustness and scalability of algebraic multigrid. *SIAM Journal of Scientific Computing 21*, 5 (2000), 1886–1908. 2

[CLB*09] CHUANG M., LUO L., BROWN B., RUSINKIEWICZ

S., KAZHDAN M.: Estimating the Laplace-Beltrami operator by restricting 3d functions. In *Proceedings of the Symposium on Geometry Processing* (2009), pp. 1475–1484. 2

[CT65] COOLEY J., TUKEY J.: An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation 19* (1965), 297–301. 2

[Dav79] DAVIS P.: *Circulant matrices*. Pure and applied mathematics. Wiley, 1979. 2, 3, 4

[DM98] DAGUM L., MENON R.: OpenMP: An industry-standard API for shared-memory programming. *IEEE Computational Science and Engineering 5*, 1 (1998), 46–55. 6

[ETK*07] ELCOTT S., TONG T., KANSO E., SCHRÖDER P., DESBRUN M.: Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph. 26*, 1 (2007). 7, 8

[FH91] FULTON W., HARRIS J.: *Representation Theory: A First Course*. Springer-Verlag, New York, 1991. 3

[FJ05] FRIGO M., JOHNSON S.: The design and implementation of FFTW3. *Proceedings of the IEEE 93*, 2 (2005), 216–231. Special issue on "Program Generation, Optimization, and Platform Adaptation". 6

[GL96] GOLUB G., LOAN C. V.: *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996. 2

[Hoc65] HOCKNEY R.: A fast direct solution of Poisson's equation using Fourier analysis. *Journal of the ACM 12*, 1 (1965), 95–113. 2, 5, 11

[Höl03] HÖLLIG K.: *Finite Element Methods with B-Splines*. Society for Industrial and Applied Mathematics, 2003. 11

[HRKM03] HEALY D., ROCKMORE D., KOSTELEC P., MOORE S.: FFTs on the 2-sphere – improvements and variations. *Journal of Fourier Analysis and Applications 9* (2003), 341–385. 6, 7

[Int13] INTEL: MKL. https://software.intel.com/en-us/intel-mkl, 2013. 7

[KCVS98] KOBBELT L., CAMPAGNA S., VORSATZ J., SEIDEL H.-P.: Interactive multi-resolution modeling on arbitrary meshes. In *ACM SIGGRAPH* (1998), pp. 105–114. 2

[KH10] KAZHDAN M., HOPPE H.: Metric-aware processing of spherical imagery. *ACM Transactions on Graphics 29*, 6 (2010), 149:1–149:10. 2, 6, 9

[LZPW04] LEVIN A., ZOMET A., PELEG S., WEISS Y.: Seamless image stitching in the gradient domain. In *European Conference on Computer Vision* (2004), pp. 377–389. 6

[McC08] MCCANN J.: Recalling the single-FFT direct Poisson solve. In *ACM SIGGRAPH 2008 Posters* (2008), SIGGRAPH '08, pp. 71:1–71:1. 9

[MR96] MASLEN D., ROCKMORE D.: *Generalized FFTS - A Survey of Some Recent Results*. Tech. rep., Dartmouth College, Hanover, NH, USA, 1996. 9

[NBGS08] NICKOLLS J., BUCK I., GARLAND M., SKADRON K.: Scalable parallel programming with CUDA. *Queue 6*, 2 (2008), 40–53. 9

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. *ACM Transactions on Graphics (SIGGRAPH '03)* (2003), 313–318. 6

[RS87] RUGE J., STUEBEN K.: Algebraic multigrid. In *Multigrid Methods* (1987), SIAM, pp. 73–130. 2

[Saa03] SAAD Y.: *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003. 2, 6

[Ser77] SERRE J.: *Linear representations of finite groups*. Springer-Verlag, New York, 1977. 3

[She94] SHEWCHUK J.: *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Tech. rep., Carnegie Mellon University, Pittsburgh, PA, USA, 1994. 2

[SS88] SCHUMANN U., SWEET R.: Fast Fourier transforms for direct solution of Poisson's equation with staggered boundary conditions. *Journal of Computational Physics 75*, 1 (1988), 123–137. 2

[Sta99] STAM J.: Stable fluids. In *ACM SIGGRAPH Conference Proceedings* (1999), pp. 121–128. 7

[Sta01] STAM J.: A simple fluid solver based on the FFT. *Journal of Graphics Tools 6* (2001), 43–52. 7

[Sta03] STAM J.: Flows on surfaces of arbitrary topology. *ACM Transactions on Graphics (SIGGRAPH '03) 22* (2003), 724–731. 7

[SYBF06] SHI L., YU Y., BELL N., FENG W.: A fast multigrid algorithm for mesh deformation. *ACM Transactions on Graph 25*, 3 (2006), 1108–1117. 2

[VL08] VALLET B., LÃL'VY B.: Spectral geometry processing with manifold harmonics. *Computer Graphics Forum (Proceedings Eurographics) 27*, 2 (2008), 251–260. 2

## Appendix A: Characterizing Commuting Linear Operators

We describe the conditions that need to be satisfied for a linear operator to commute with the cyclic/dihedral group on a surface of revolution and provide an explicit formula for the block diagonal matrices. As in Section 4, we assume that the surface is sampled on an $M \times N$ grid, with $M$ representing the number of meridians and $N$ the number of parallels.

**Notation** We denote by $\langle \cdot, \cdot \rangle$ Hermitian inner-product:

$$\langle \mathbf{F}, \mathbf{G} \rangle = \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} \mathbf{F}_{j,k} \cdot \overline{\mathbf{G}}_{j,k}.$$

**Notation** We denote by $R_m$ the (unitary) operator shifting the signal by $m$ positions along the parallel (i.e. rotation by $m\theta_M$ around the $y$-axis) and by $R$ the operator negating the angular index (i.e. reflection through the $xy$-plane):

$$\begin{array}{l} (R_m(\mathbf{F}))_{j,k} = \mathbf{F}_{j-m,k} \\ (R(\mathbf{F}))_{j,k} = \mathbf{F}_{-j,k} \end{array} \qquad \forall \mathbf{F} \in \mathscr{F}.$$

**Proposition A.1.** *A linear operator $L$ commutes with the action of the cyclic group if and only if there exist functions (stencils) $\boldsymbol{L}^0, \ldots, \boldsymbol{L}^{N-1} \in \mathscr{F}$ such that for all $\boldsymbol{F} \in \mathscr{F}$:*

$$(L(\boldsymbol{F}))_{j,k} = \langle \boldsymbol{F}, R_j(\boldsymbol{L}^k) \rangle.$$

*That is, restricted to the $k$-th parallel $L$ can be implemented as a circular correlation with the filter $\boldsymbol{L}^k$.*

**Proof** Define $\mathbf{L}^k \in \mathscr{F}$ to be the function such that:

$$(L(\mathbf{F}))_{0,k} = \langle \mathbf{F}, \mathbf{L}^k \rangle.$$

On the one hand, we have:

$$(L(\mathbf{F}))_{j,k} = \big(R_{-j}(L(\mathbf{F}))\big)_{0,k}.$$

On the other, we have:

$$\langle \mathbf{F}, R_j(\mathbf{L}^k) \rangle = \langle \mathbf{F}, R_{-j}(\mathbf{L}^k) \rangle = \big(L(R_{-j}(\mathbf{F}))\big)_{0,k}.$$

Since this is true for all $\mathbf{F}$, we have $(L(\mathbf{F}))_{j,k} = \langle \mathbf{F}, R_j(\mathbf{L}^k)\rangle$ if and only if $R_m \circ L = L \circ R_m$ for all $m$. That is, if and only if $L$ commutes with rotation. □

[For spheres, this implies that the stencils have the property that $\mathbf{L}^0_{j,k} = \mathbf{L}^0_{0,k}$ and $\mathbf{L}^{N-1}_{j,k} = \mathbf{L}^{N-1}_{0,k}$.]

**Proposition A.2.** *The linear operator $L$ commutes with the action of the dihedral group if and only if the stencils satisfy $R(\boldsymbol{L}^k) = \boldsymbol{L}^k$ for all $k \in [0,N)$.*

**Proof** Since $L$ commutes with rotation, we have:

$$\langle \mathbf{F}, R_j(\mathbf{L}^k)\rangle = (L(\mathbf{F}))_{j,k} = R(L(\mathbf{F}))_{-j,k}.$$

On the other hand, we also have:

$$\begin{aligned}\langle \mathbf{F}, R_j(R(\mathbf{L}^k))\rangle &= \langle \mathbf{F}, R(R_{-j}(\mathbf{L}^k))\rangle \\ &= \langle R(\mathbf{F}, R_{-j}(\mathbf{L}^k))\rangle = (L(R(\mathbf{F})))_{-j,k}.\end{aligned}$$

Since this is true for all $\mathbf{F}$ it follows that $R(\mathbf{L}^k) = \mathbf{L}^k$ if and only if $L \circ R = R \circ L$. □

### Defining the Diagonal Blocks

Representing a linear operator $L$ that commutes with the cyclic group in terms of the stencils $\mathbf{L}^0,\dots,\mathbf{L}^{N-1} \in \mathscr{F}$ and considering each frequency in turn, for $\mathbf{F}^\omega \in \mathscr{F}^\omega$ we get:

$$\begin{aligned}(L(\mathbf{F}^\omega))_{j,k} &= \langle \mathbf{F}^\omega, R_j(\mathbf{L}^k)\rangle = \zeta_M^{j\omega}\langle \mathbf{F}^\omega, \mathbf{L}^k\rangle \\ &= \zeta_M^{j\omega}\sum_{\tilde{j}=0}^{M-1}\sum_{\tilde{k}=0}^{N-1}\overline{\mathbf{L}^k}_{\tilde{j},\tilde{k}}\left(\zeta_M^{\tilde{j}\omega}\cdot\hat{\mathbf{f}}^\omega_{\tilde{k}}\right)\end{aligned}$$

That is, if $\hat{\mathbf{f}}^\omega$ is the set of $\omega$-th Fourier coefficients along the different parallels, then the linear operator $L$ takes these to a new set of $\omega$-th Fourier coefficients, $\hat{\mathbf{g}}^\omega$, defined by:

$$\hat{\mathbf{g}}^\omega = \hat{L}^\omega\hat{\mathbf{f}}^\omega \qquad \text{with} \qquad \hat{L}^\omega_{k,\tilde{k}} = \sum_{\tilde{j}=0}^{M-1}\overline{\mathbf{L}^k}_{\tilde{j},\tilde{k}}\zeta_M^{\tilde{j}\omega}. \quad (1)$$

In general, the entries of $\hat{L}^\omega$ need not be real, as they are the weighted sum of (complex) roots of unity. However, if $L$ also commutes with the action of the dihedral group then Equation 1 becomes:

$$\begin{aligned}\hat{L}^\omega_{k,\tilde{k}} &= \mathbf{L}^k_{0,\tilde{k}} + \frac{1}{2}\sum_{\tilde{j}=1}^{M-1}\mathbf{L}^k_{\tilde{j},\tilde{k}}\left(\zeta_M^{\tilde{j}\omega} + \zeta_M^{-\tilde{j}\omega}\right) \\ &= \mathbf{L}^k_{0,\tilde{k}} + \sum_{\tilde{j}=1}^{M-1}\mathbf{L}^k_{\tilde{j},\beta}\cos\left(\tilde{j}\omega\theta_M\right). \quad (2)\end{aligned}$$

Thus, when the stencil is reflectively symmetric, the coefficients of $\hat{L}^\omega$ are real, and the operator $L$ preserves not only the frequency decomposition, but also the real/imaginary decomposition. Hence the cosine (respectively sine) components map to themselves. (As expected, this agrees with the formula derived by Hockney [Hoc65] for the specific case of the finite-differences Laplacian in the plane.)

### Domains with Angular Boundaries

Given stencils $\mathbf{L}^0,\dots,\mathbf{L}^{N-1} \in \mathscr{F}$ satisfying the property that $R(\mathbf{L}^k) = \mathbf{L}^k$, we can use these to define a linear operator $L$ that satisfies Neumann/Dirichlet angular boundary constraints and also supports a fast solver.

For both boundary types we proceed by treating the $M \times N$ grid with angular boundaries as a $(2M-2) \times N$ grid with periodic boundaries.

**Neumann Boundary Constraints** In this case, the linear operator $L^{\mathscr{N}}$ is defined by reflection across the boundaries:

$$\left(L^{\mathscr{N}}(\mathbf{F})\right)_{j,k} = \sum_{\tilde{j}=-M/2}^{M/2}\sum_{\tilde{k}=0}^{N-1}\mathbf{L}^k_{\tilde{j},\tilde{k}}\mathbf{F}_{\rho_M(j+\tilde{j}),\tilde{k}}$$

where the reflection map, $\rho_M$, is defined as:

$$\rho_M(m) = \begin{cases} m \bmod (2M-2) & \text{if} \quad m < M \\ (2M-m) \bmod (2M-2) & \text{otherwise}\end{cases}.$$

**Dirichlet Boundary Constraints** In this case, the linear operator $L^{\mathscr{D}}$ is defined by anti-reflection across the boundaries:

$$\left(L^{\mathscr{D}}(\mathbf{F})\right)_{j,k} = \sum_{\tilde{j}=-M/2}^{M/2}\sum_{\tilde{k}=0}^{N-1}\sigma(j+\tilde{j})\mathbf{L}^k_{\tilde{j},\tilde{j}}\mathbf{F}_{\pi_M(j+\tilde{j}),\tilde{k}}$$

where the sign function, $\sigma_M$, is defined as:

$$\sigma_M(m) = \begin{cases} 1 & \text{if} \quad 0 < m \bmod (2M-2) < M-1 \\ 0 & \text{if} \quad m \bmod (M-1) = 0 \\ -1 & \text{otherwise}\end{cases}.$$

**Remark** As expected, the values of $L^{\mathscr{D}}(\mathbf{F})$ does not depend on the value of $\mathbf{F}_{j,k}$ when $j = 0$ or $j = M$. This reflects the fact that in implementing Dirichlet boundary conditions the values on the angular boundaries are fixed at zero.

## Appendix B: Discretizing the Poisson System

To define the Laplacian system, we need to make two choices. First, we need to choose a function basis. Second, we need to define a metric on the surface.

In what follows, the functions and the metric will take values in the domain $\mathscr{R} = [0,M) \times [0,N) \subset \mathbb{R}^2$ for toroidal geometry and $\mathscr{R} = [0,M) \times [0,N-1]$ otherwise.

### Choosing a Basis

To define the space of functions, we use the standard finite-elements discretization, representing functions as a linear combination of B-splines [Höl03].

### The Space of Scalar-Valued Functions

To span the space of scalar-valued function, we use first order B-splines, denoting by $B^1_i(x)$ the piecewise linear "hat"

function at position $i$:

$$B_j^1(x) = \begin{cases} (x-j+1) & \text{if } x \in [j-1, j] \\ (j+1-x) & \text{if } x \in [j, j+1] \\ 0 & \text{otherwise.} \end{cases}$$

There are three types of vertices to consider.

**Away From Poles** Most vertices of the grid will be adjacent to four quadrilateral faces. For such a vertex $v = (j,k)$, we use the bilinear hat function:

$$B_v(x,y) = B_j^1(x) \cdot B_k^1(y).$$

Indexing is cyclic, with the first index taken modulo $M$ and the second modulo $N$.

**Near Poles** For spherical domains, we have vertices in parallels $k = 1$ and $k = N-2$. These vertices are adjacent to two quadrilateral faces and two triangular faces. Treating the triangles as degenerate quads, we can use the same bilinear basis functions as for vertices that are away from the poles.

**At Poles** For spherical domains, we also have pole vertices $v = (j,k)$ with $k = 0$ and $k = N-1$. For these we use basis functions that are linear "hat" functions in $y$ alone:

$$B_v(x,y) = B_k^1(y).$$

### The Space of (Cotangent) Vector Fields

The choice of B-spline basis makes it possible to represent the gradient of a function by assigning a scalar value to each edge in the grid. In particular, if we let $B_j^0(x)$ denote the zero-th order B-spline:

$$B_j^0(x) = \begin{cases} 1 & \text{if } x \in [j, j+1) \\ 0 & \text{otherwise} \end{cases}$$

we can express the gradient of $B_v$, with $v = (j,k)$, as the mixed tensor-product of zero-th and first order B-splines:

$$\nabla B_v(x,y) = \\ = \left( \left( B_{j-1}^0 - B_j^0 \right)(x) \cdot B_k^1(y), B_j^1(x) \cdot \left( B_{k-1}^0 - B_k^0 \right)(y) \right).$$

Thus, the gradient resides in the span:

$$\text{Span} \left\{ \left( B_j^0(x) \cdot B_k^1(y), 0 \right), \left( 0, B_j^1(x) \cdot B_k^0(y) \right) \right\}.$$

These functions can be indexed by the edges of the grid and for each edge $e$ we set:

$$\vec{B}_e = \begin{cases} \left( B_j^0(x) \cdot B_k^1(y), 0 \right) & \text{if } e = (j,k) \to (j+1,k) \\ \left( 0, B_j^1(x) \cdot B_k^0(y) \right) & \text{if } e = (j,k) \to (j,k+1) \end{cases}$$

In this basis, the gradient operator can be represented by the finite-differencing matrix, $G \in \mathbb{R}^{|\mathscr{V}| \times |\mathscr{E}|}$ with:

$$G_{v,e} = \begin{cases} 1 & \text{if } v = \text{end}(e) \\ -1 & \text{if } v = \text{start}(e) \\ 0 & \text{otherwise} \end{cases}$$

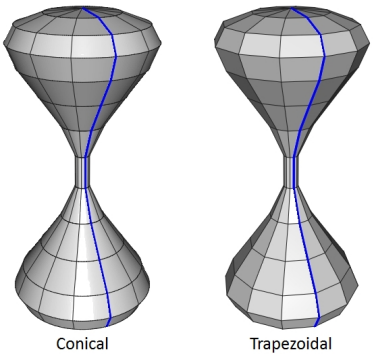


**Figure 4:** *Conical (left) and trapezoidal (right) parameterizations for surfaces of revolution defined by the same generating curve (highlighted in blue).*

### Choosing a Metric

To define differentiation and integration, we need to define a metric over $\mathscr{R}$. This can be thought of as a $2 \times 2$ symmetric positive definite matrix associated to each point in the domain that describes how area is stretched as it is mapped from $\mathscr{R}$ to the surface of revolution. Using the metric, we obtain discretizations of the standard differential operators, allowing us to define the discrete mass, Laplace, and divergence operators, as well as compute harmonic vector fields.

In practice, the metric is defined by choosing a parameterization $\Phi : \mathscr{R} \to \mathbb{R}^3$ that maps the 2D domain to the surface of revolution. Then, the metric is defined as the product of the differential of the map and its transpose:

$$\mu(x,y) = \left( d\Phi\big|_{(x,y)} \right)^t \cdot \left( d\Phi\big|_{(x,y)} \right).$$

### Conical

We define this parameterization so that $[0,M) \times [k,k+1]$ maps to a section of a cone. We do this by trigonometrically interpolating across parallels and linearly interpolating between them. Specifically, denoting $\delta_y = y - \lfloor y \rfloor$, we set:

$$\Phi(x,y) = \left( \mathbf{u}(y) \cdot \cos\left( \frac{2x\pi}{M} \right), \mathbf{v}(y), \mathbf{u}(y) \cdot \sin\left( \frac{2x\pi}{M} \right) \right)$$

where $\mathbf{u}(y)$ and $\mathbf{v}(y)$ are the linear interpolants of the points on the generating curve of the surface of revolution:

$$\mathbf{u}(y) = \mathbf{u}_{\lfloor y \rfloor} \cdot (1 - \delta_y) + \mathbf{u}_{\lceil y \rceil} \cdot \delta_y$$
$$\mathbf{v}(y) = \mathbf{u}_{\lfloor y \rfloor} \cdot (1 - \delta_y) + \mathbf{v}_{\lceil y \rceil} \cdot \delta_y.$$

Computing the differential of this map, we get:

$$\mu(x,y) = \left( \mu^{1/2}(x,y) \right)^t \cdot \left( \mu^{1/2}(x,y) \right)$$

where

$$\mu^{1/2}(x,y) = \begin{pmatrix} r_1 \cdot \theta + (r_2 - r_1) \cdot \theta \cdot \delta_y & 0 \\ 0 & r_2 - r_1 \end{pmatrix}$$

with $r_i$ is the distance from a point on the parallel at $\lfloor y \rfloor + i$ to the apex of the cone and $\theta$ is the angle of the cone.

### Trapezoidal

We define this parameterization so that $[j, j+1] \times [k, k+1]$ maps to an isosceles trapezoid. We do this by linearly interpolating across and between parallels. Specifically, denoting $\delta_x = x - \lfloor x \rfloor$ and $\delta_y = y - \lfloor y \rfloor$, we set:

$$\begin{aligned}\Phi(x,y) = &\left(\mathbf{V}_{\lfloor x \rfloor, \lfloor y \rfloor} \cdot (1-\delta_x) + \mathbf{V}_{\lceil x \rceil, \lfloor y \rfloor} \cdot \delta_x\right) \cdot (\quad \delta_y) \\ &+ \left(\mathbf{V}_{\lfloor x \rfloor, \lceil y \rceil} \cdot (1-\delta_x) + \mathbf{V}_{\lceil x \rceil, \lceil y \rceil} \cdot \delta_x\right) \cdot (1-\delta_y).\end{aligned}$$

Computing the differential of this map, we get:

$$\mu(x,y) = \left(\mu^{1/2}(x,y)\right)^t \cdot \left(\mu^{1/2}(x,y)\right)$$

where

$$\mu^{1/2}(x,y) = \begin{pmatrix} b_0 + d_b \cdot \delta_y & -\frac{1}{2} \cdot d_b + d_b \cdot \delta_x \\ 0 & h \end{pmatrix}$$

with $b_i$ the widths of the bases, $h$ the height, and $d_b$ the difference in base widths:

$$\begin{aligned} b_i &= |\Phi(\lceil x \rceil, \lfloor y \rfloor + i) - \Phi(\lfloor x \rfloor, \lfloor y \rfloor + i)| \\ h &= |\Phi(\lfloor x \rfloor + 0.5, \lceil y \rceil) - \Phi(\lfloor x \rfloor + 0.5, \lfloor y \rfloor)| \\ d_b &= b_1 - b_0. \end{aligned}$$

Figure 4 shows the two surfaces of revolution obtained by using conical (left) and trapezoidal (right) parameterizations defined by the same generating curve (shown in blue). Note that the trapezoidal interpretation of a surface of revolution is only discretely rotationally symmetric, while the conical interpretation is continuously rotationally symmetric. And, the metric defined by the trapezoidal parameterization is not diagonal (visualized by the fact that the network of curves is not orthogonal) while the metric defined by the conical parameterization is. None-the-less, both define Poisson systems that can be efficiently solved with our approach.

### Computing the System

To discretize the operators, we compute the entries of the 0-form and 1-form mass matrices $M^0$ and $M^1$, as well as the stiffness $S$, and (negative) divergence $D$, matrices:

$$\begin{aligned} M^0_{v,w} &= \int_{\mathscr{R}} B_v \cdot B_w \cdot |\mu|^{1/2} \; dxdy \\ S_{v,w} &= \int_{\mathscr{R}} (\nabla B_v)^t \, \mu^{-1} \, (\nabla B_w) \cdot |\mu|^{1/2} \; dxdy \\ D_{e,v} &= \int_{\mathscr{R}} (\nabla B_v)^t \, \mu^{-1} \left(\vec{B}_e\right) \cdot |\mu|^{1/2} \; dxdy \\ M^1_{e,f} &= \int_{\mathscr{R}} \left(\vec{B}_e\right)^t \mu^{-1} \left(\vec{B}_f\right) \cdot |\mu|^{1/2} \; dxdy \end{aligned}$$

where $|\mu|$ is the determinant of $\mu$.

For the 0-form mass matrix, we observe that the integrand is the product of piecewise polynomial terms, and hence is integrable in closed form over the rectangle $\mathscr{R}$.

For the 1-form mass, divergence, and stiffness matrices, the integrand is the quotient of a piecewise polynomial and the determinant of $\mu^{1/2}$. (The inversion of $\mu$ introduces the square of the determinant in the denominator, one of which is canceled out.) However, since the determinant of $\mu^{1/2}$ is a piecewise linear function in $y$, we can still compute the integral in closed form, requiring the evaluation of the log function in the definite integral.

One can show that this formulation works even at the poles of spherical domains. The concern could be that the the metric $\mu$ is singular at the poles and hence its inverse is not well-defined. However, because the pole basis functions are purely functions of $y$ and the basis functions in the adjacent parallels vanish at the poles, the integral remains well-defined.

### Harmonic Vector Fields

Using the discrete Laplacian, divergence, and gradient, we can also compute a basis for the space of harmonic (cotangent) vector fields – vector fields that are divergence-free and can be locally defined as the gradient of a scalar function.

**Toroidal Domains** For toroidal domains, the space of harmonic vector fields is two-dimensional, which we express as the span of vector fields $\vec{\mathbf{H}}_1$ and $\vec{\mathbf{H}}_2$, with $\vec{\mathbf{H}}_1$ the gradient of the function giving the angle about the axis of revolution and $\vec{\mathbf{H}}_2$ its rotation by 90°.

To define $\vec{\mathbf{H}}_1$ we start with the vector field $\vec{\mathbf{G}}_1$ that has a coefficient of one along parallel edges and a coefficient of zero along meridian edges. Locally, this is the gradient of the angle of revolution (up to a constant) and is therefore curl-free. To make it divergence-free, we project out the component of $\vec{\mathbf{G}}_1$ that diverges by solving a Poisson equation:

$$\vec{\mathbf{H}}_1 = \vec{\mathbf{G}}_1 - G(S^{-1}(D(\vec{\mathbf{G}}_1))).$$

$\vec{\mathbf{H}}_2$ is defined similarly, this time starting with the vector field $\vec{\mathbf{G}}_2$ that has a coefficient of one along meridian edges and a coefficient of zero along parallel edges.

We make these orthonormal using Gram-Schmidt orthogonalization with respect to the 1-form mass-matrix.

**Cylindrical Domains** When the surface of revolution is a topological cylinder, either because we are sweeping an open curve completely around the axis of revolution, or because we are sweeping a closed curve by an angle smaller than 360°, the space of harmonic vector fields is one-dimensional. The choice between $\vec{\mathbf{H}}_1$ and $\vec{\mathbf{H}}_2$ as the spanner of the space of harmonic vector fields depends on the type of boundary (open curve vs. angular boundary) and the type of boundary constraints (Neumann vs. Dirichlet):

$$\vec{\mathbf{H}} = \begin{cases} \vec{\mathbf{H}}_1 & \text{if NEUMANN xor ANGULAR} \\ \vec{\mathbf{H}}_2 & \text{otherwise.} \end{cases}$$