Laplace-Beltrami: The Swiss Army Knife of Geometry Processing

(SGP 2014 Tutorial—July 7 2014)
Justin Solomon / Stanford University
Keenan Crane / Columbia University
Etienne Vouga / Harvard University
Introduction
• Laplace-Beltrami operator ("Laplacian") provides a basis for a diverse variety of geometry processing tasks.
• Laplace-Beltrami operator ("Laplacian") provides a basis for a diverse variety of geometry processing tasks.
• Remarkably common pipeline:
Introduction

- *Laplace-Beltrami operator* ("Laplacian") provides a basis for a diverse variety of geometry processing tasks.
- Remarkably common pipeline:
  1. simple pre-processing (build $f$)

Expressing tasks in terms of Laplacian/smooth PDEs makes life easier at code/implementation level.
Lots of existing theory to help understand/interpret algorithms, provide analysis/guarantees.
Also makes it easy to work with a broad range of geometric data structures (meshes, point clouds, etc.)
• *Laplace-Beltrami operator* ("Laplacian") provides a basis for a diverse variety of geometry processing tasks.

• Remarkably common pipeline:
  1. simple pre-processing (build $f$)
  2. solve a PDE involving the Laplacian (e.g., $\Delta u = f$)

• Expressing tasks in terms of Laplacian/smooth PDEs makes life easier at code/implementation level.

• Lots of existing theory to help understand/interpret algorithms, provide analysis/guarantees.

• Also makes it easy to work with a broad range of geometric data structures (meshes, point clouds, etc.)
• Laplace-Beltrami operator ("Laplacian") provides a basis for a diverse variety of geometry processing tasks.

• Remarkably common pipeline:
  1. simple pre-processing (build $f$)
  2. solve a PDE involving the Laplacian (e.g., $\Delta u = f$)
  3. simple post-processing (do something with $u$)

• Expressing tasks in terms of Laplacian/smooth PDEs makes life easier at code/implementation level.

• Lots of existing theory to help understand/interpret algorithms, provide analysis/guarantees.

• Also makes it easy to work with a broad range of geometric data structures (meshes, point clouds, etc.)
Introduction

• *Laplace-Beltrami operator* ("Laplacian") provides a basis for a diverse variety of geometry processing tasks.

• Remarkably common pipeline:
  1. simple pre-processing (build $f$)
  2. solve a PDE involving the Laplacian (e.g., $\Delta u = f$)
  3. simple post-processing (do something with $u$)

• Expressing tasks in terms of Laplacian/smooth PDEs makes life easier at code/implementation level.
Introduction

- Laplace-Beltrami operator ("Laplacian") provides a basis for a diverse variety of geometry processing tasks.
- Remarkably common pipeline:
  1. simple pre-processing (build $f$)
  2. solve a PDE involving the Laplacian (e.g., $\Delta u = f$)
  3. simple post-processing (do something with $u$)
- Expressing tasks in terms of Laplacian/smooth PDEs makes life easier at code/implementation level.
- Lots of existing theory to help understand/interpret algorithms, provide analysis/guarantees.
• *Laplace-Beltrami operator* (“Laplacian”) provides a basis for a diverse variety of geometry processing tasks.
• Remarkably common pipeline:
  1. simple pre-processing (build $f$)
  2. solve a PDE involving the Laplacian (e.g., $\Delta u = f$)
  3. simple post-processing (do something with $u$)
• Expressing tasks in terms of Laplacian/smooth PDEs makes life easier at code/implementation level.
• Lots of existing theory to help understand/interpret algorithms, provide analysis/guarantees.
• Also makes it easy to work with a broad range of geometric data structures (meshes, point clouds, etc.)
Introduction

- Goals of this tutorial:
Introduction

- Goals of this tutorial:
  - Understand the Laplacian in the smooth setting. (Etienne)
Introduction

• Goals of this tutorial:
  • Understand the Laplacian in the smooth setting. *(Etienne)*
  • Build the Laplacian in the discrete setting. *(Keenan)*
Goals of this tutorial:

- Understand the Laplacian in the smooth setting. *(Etienne)*

- Build the Laplacian in the discrete setting. *(Keenan)*

- Use Laplacian to implement a variety of methods. *(Justin)*
SMOOTH THEORY
The Interpolation Problem

- given:
  - region $\Omega \subset \mathbb{R}^2$ with boundary $\partial \Omega$
  - function $f$ on $\partial \Omega$

fill in $f$ “as smoothly as possible”
The Interpolation Problem

• given:
  • region $\Omega \subset \mathbb{R}^2$ with boundary $\partial \Omega$
  • function $f$ on $\partial \Omega$

fill in $f$ “as smoothly as possible”

• (what does this even mean?)
The Interpolation Problem

- given:
  - region $\Omega \subset \mathbb{R}^2$ with boundary $\partial \Omega$
  - function $f$ on $\partial \Omega$

fill in $f$ “as smoothly as possible”

- (what does this even mean?)
- smooth:
  - constant functions
  - linear functions
The Interpolation Problem

\[ f = \begin{cases} 1 & \text{given:} \\ -1 & \text{region } \Omega \subset \mathbb{R}^2 \text{ with boundary } \partial \Omega \\ \text{function } f \text{ on } \partial \Omega \\ \text{fill in } f \text{ “as smoothly as possible”} \\ (\text{what does this even mean?}) \\ \text{smooth:} \\ \text{constant functions} \\ \text{linear functions} \\ \text{not smooth:} \\ \text{f not continuous} \end{cases} \]
The Interpolation Problem

given:
- region $\Omega \subset \mathbb{R}^2$ with boundary $\partial \Omega$
- function $f$ on $\partial \Omega$

fill in $f$ “as smoothly as possible”

(what does this even mean?)

smooth:
- constant functions
- linear functions

not smooth:
- $f$ not continuous
- large variations over short distances
- $(\| \nabla f \| \text{ large})$
Dirichlet Energy

- \[ E(f) = \int_{\Omega} \| \nabla f \|^2 \, dA \]
- properties:
  - nonnegative
  - zero for constant functions
  - measures smoothness

non-smooth \( f(x) \)
Dirichlet Energy

\[ E(f) = \int_{\Omega} \| \nabla f \|^2 \, dA \]

properties:
- nonnegative
- zero for constant functions
- measures smoothness

solution to interpolation problem is minimizer of \( E \)
Dirichlet Energy

- $E(f) = \int_{\Omega} \|\nabla f\|^2 \, dA$
- properties:
  - nonnegative
  - zero for constant functions
  - measures smoothness
- solution to interpolation problem is minimizer of $E$
- how do we find minimum?
Dirichlet Energy

- \( E(f) = \int_{\Omega} \| \nabla f \|^2 \, dA \)
- it can be shown that:
  - \( E(f) = C - \int_{\Omega} f \Delta f \, dA \)

non-smooth \( f(x) \)

\( \| \nabla f \|^2 \)
Dirichlet Energy

• $E(f) = \int_{\Omega} \|\nabla f\|^2 dA$

• it can be shown that:
  • $E(f) = C - \int_{\Omega} f \Delta f \, dA$
  • $-2\Delta f$ is the gradient of Dirichlet energy

non-smooth $f(x)$

$\Delta f$
Dirichlet Energy

\[ E(f) = \int_{\Omega} \|\nabla f\|^2 \, dA \]

it can be shown that:

\[ E(f) = C - \int_{\Omega} f \Delta f \, dA \]

\(-2\Delta f\) is the gradient of Dirichlet energy

\(f\) minimizes \(E\) if \(\Delta f = 0\)
Dirichlet Energy

- \( E(f) = \int_{\Omega} \|\nabla f\|^2 dA \)
- it can be shown that:
  - \( E(f) = C - \int_{\Omega} f \Delta f \, dA \)
  - \(-2\Delta f\) is the gradient of Dirichlet energy
  - \( f\) minimizes \( E\) if \( \Delta f = 0 \)
- PDE form (Laplace’s Equation):
  \[
  \Delta f(x) = 0 \quad x \in \Omega \\
  f(x) = f_0(x) \quad x \in \partial \Omega
  \]
Dirichlet Energy

- \( E(f) = \int_{\Omega} \| \nabla f \|^2 \, dA \)
- it can be shown that:
  - \( E(f) = C - \int_{\Omega} f \Delta f \, dA \)
  - \(-2\Delta f\) is the gradient of Dirichlet energy
  - \( f \) minimizes \( E \) if \( \Delta f = 0 \)
- PDE form (Laplace’s Equation):
  \[
  \Delta f(x) = 0 \quad x \in \Omega \\
  f(x) = f_0(x) \quad x \in \partial \Omega
  \]
- physical interpretation: temperature at steady state
On a Surface

\[ f = -1 \]

boundary conditions

\[ f = 1 \]

nonsmooth \( f(x) \)
On a Surface

- can still define Dirichlet energy $E(f) = \int_M \|\nabla f\|^2$
On a Surface

- can still define Dirichlet energy $E(f) = \int_M \|\nabla f\|^2$
- $\nabla E(f) = -\Delta f$, now $\Delta$ is the Laplace-Beltrami operator of $M$
On a Surface

- can still define Dirichlet energy $E(f) = \int_M \|\nabla f\|^2$
- $\nabla E(f) = -\Delta f$, now $\Delta$ is the Laplace-Beltrami operator of $M$
- also works in higher dimensions, on discrete graphs/point clouds, …
Existence and Uniqueness

• Laplace’s equation

\[ \Delta f(x) = 0 \quad x \in M \]
\[ f(x) = f_0(x) \quad x \in \partial M \]

has a unique solution for all reasonable\(^1\) surfaces \(M\)

\(^1\)e.g. compact, smooth, with piecewise smooth boundary
Existence and Uniqueness

• Laplace’s equation

\[ \Delta f(x) = 0 \quad x \in M \]
\[ f(x) = f_0(x) \quad x \in \partial M \]

has a unique solution for all reasonable\(^1\) surfaces \(M\)

• physical interpretation: apply heating/cooling \(f_0\) to the boundary of a metal plate. Interior temperature will reach some steady state

\(^1\)e.g. compact, smooth, with piecewise smooth boundary
Existence and Uniqueness

- Laplace’s equation

\[ \Delta f(x) = 0 \quad x \in M \]
\[ f(x) = f_0(x) \quad x \in \partial M \]

has a unique solution for all reasonable\(^1\) surfaces \(M\)

- physical interpretation: apply heating/cooling \(f_0\) to the boundary of a metal plate. Interior temperature will reach some steady state

- gradient descent is exactly the heat or diffusion equation
\[ \frac{df}{dt}(x) = \Delta f(x). \]

\(^1\)e.g. compact, smooth, with piecewise smooth boundary
Heat Equation Illustrated

time
Boundary Conditions

- can specify $\nabla f \cdot \hat{n}$ on boundary instead of $f$:

  \[
  \Delta f(x) = 0 \quad x \in \Omega \\
  f(x) = f_0(x) \quad x \in \partial \Omega_D \quad \text{(Dirichlet bdry)} \\
  \nabla f \cdot \hat{n} = g_0(x) \quad x \in \partial \Omega_N \quad \text{(Neumann bdry)}
  \]
Boundary Conditions

- can specify $\nabla f \cdot \hat{n}$ on boundary instead of $f$:
  
  $\Delta f(x) = 0 \quad x \in \Omega$
  
  $f(x) = f_0(x) \quad x \in \partial \Omega_D \ (\text{Dirichlet bdry})$
  
  $\nabla f \cdot \hat{n} = g_0(x) \quad x \in \partial \Omega_N \ (\text{Neumann bdry})$

- usually: $g_0 = 0 \ (\text{natural bdry conds})$
Boundary Conditions

\[ g_0 = 0 \]

\[ f_0 = -1 \]

\[ f_0 = 1 \]

- can specify \( \nabla f \cdot \hat{n} \) on boundary instead of \( f \):

\[
\Delta f(x) = 0 \quad x \in \Omega \\
f(x) = f_0(x) \quad x \in \partial\Omega_D \quad (\text{Dirichlet bdry}) \\
\nabla f \cdot \hat{n} = g_0(x) \quad x \in \partial\Omega_N \quad (\text{Neumann bdry})
\]

- usually: \( g_0 = 0 \) (natural bdry conds)

- physical interpretation: free boundary through which heat cannot flow
Interpolation with $\Delta$ in Practice

in geometry processing:
- positions
- displacements
- vector fields
- parameterizations
- … you name it

Joshi et al
Eck et al
Sorkine and Cohen-Or
Heat Equation with Source

- what if you add heat sources inside $\Omega$?

$\frac{df}{dt}(x) = g(x) + Df(x)$

PDE form: Poisson's equation

$Df(x) = g(x)x^2$ for $f(x) = f_0(x)x^2$ for $\partial W$.

common variational problem:

$\min \int_M kr f^kv^2dA$ becomes Poisson problem, $g = r \cdot v$. 
Heat Equation with Source

- what if you add heat sources inside $\Omega$?

$$\frac{df}{dt}(x) = g(x) + \Delta f(x)$$
Heat Equation with Source

• what if you add heat sources inside $\Omega$?

\[
\frac{df}{dt}(x) = g(x) + \Delta f(x)
\]

• PDE form: Poisson’s equation

\[
\Delta f(x) = g(x) \quad x \in \Omega
\]
\[
f(x) = f_0(x) \quad x \in \partial \Omega
\]
Heat Equation with Source

- what if you add heat sources inside $\Omega$?

$$\frac{df}{dt}(x) = g(x) + \Delta f(x)$$

- PDE form: Poisson’s equation

$$\Delta f(x) = g(x) \quad x \in \Omega$$

$$f(x) = f_0(x) \quad x \in \partial\Omega$$

- common variational problem:

$$\min_f \int_M \| \nabla f - v \|^2 dA$$
Heat Equation with Source

- what if you add heat sources inside $\Omega$?
  $$\frac{df}{dt}(x) = g(x) + \Delta f(x)$$

- PDE form: Poisson’s equation
  $$\Delta f(x) = g(x) \quad x \in \Omega$$
  $$f(x) = f_0(x) \quad x \in \partial \Omega$$

- common variational problem:
  $$\min_f \int_M \| \nabla f - \mathbf{v} \|^2 dA$$
  becomes Poisson problem, $g = \nabla \cdot \mathbf{v}$
Essential Algebraic Properties I

- **linearity:**
  \[ \Delta (f(x) + \alpha g(x)) = \Delta f(x) + \alpha \Delta g(x) \]
Essential Algebraic Properties I

- **linearity:** \( \Delta (f(x) + \alpha g(x)) = \Delta f(x) + \alpha \Delta g(x) \)
- **constants in kernel:** \( \Delta \alpha = 0 \)
Essential Algebraic Properties I

• **linearity:** \[ \Delta (f(x) + \alpha g(x)) = \Delta f(x) + \alpha \Delta g(x) \]

• **constants in kernel:** \[ \Delta \alpha = 0 \]

for functions that vanish on \( \partial M \):

• **self-adjoint:** \[ \int_M f \Delta g \, dA = - \int_M \langle \nabla f, \nabla g \rangle \, dA = \int_M g \Delta f \, dA \]

• **negative:** \[ \int_M f \Delta f \, dA \leq 0 \]
Essential Algebraic Properties I

- **linearity:** \( \Delta (f(x) + \alpha g(x)) = \Delta f(x) + \alpha \Delta g(x) \)
- **constants in kernel:** \( \Delta \alpha = 0 \)

for functions that vanish on \( \partial M \):
- **self-adjoint:** \( \int_M f \Delta g \, dA = - \int_M \langle \nabla f, \nabla g \rangle \, dA = \int_M g \Delta f \, dA \)
- **negative:** \( \int_M f \Delta f \, dA \leq 0 \)

(intuition: \( \Delta \approx \) an \( \infty \)-dimensional negative-semidefinite matrix)
Solving Poisson’s Equation with Green’s Functions

- the Green’s function $G$ on $\mathbb{R}^2$ solves $\Delta f = g$ for $g = \delta$
Solving Poisson’s Equation with Green’s Functions

- the Green’s function $G$ on $\mathbb{R}^2$ solves $\Delta f = g$ for $g = \delta$
- linearity: if $g = \sum \alpha_i \delta(x - x_i)$, $f = \sum \alpha_i G(x - x_i)$
Solving Poisson’s Equation with Green’s Functions

- the Green’s function $G$ on $\mathbb{R}^2$ solves $\Delta f = g$ for $g = \delta$
- linearity: if $g = \sum \alpha_i \delta(x - x_i)$, $f = \sum \alpha_i G(x - x_i)$
- for any $g$, $f = G \ast g$
Essential Algebraic Properties II

A function $f : M \to \mathbb{R}$ with $\Delta f = 0$ is called harmonic. Properties:

- $f$ is smooth and analytic

Some harmonic $f(x, y)$
Essential Algebraic Properties II

A function \( f : M \rightarrow \mathbb{R} \) with \( \Delta f = 0 \) is called \textit{harmonic}. Properties:

- \( f \) is smooth and analytic
- \( f(x) \) is the \textit{average} of \( f \) over any disk around \( x \):

\[
 f(x) = \frac{1}{\pi r^2} \int_{B(x,r)} f(y) \, dA
\]
a function $f : M \to \mathbb{R}$ with $\Delta f = 0$ is called \textit{harmonic}. Properties:

- $f$ is smooth and analytic
- $f(x)$ is the \textit{average} of $f$ over any disk around $x$:

$$f(x) = \frac{1}{\pi r^2} \int_{B(x,r)} f(y) \, dA$$

- \textit{maximum principle}: $f$ has no local maxima or minima in $M$
Essential Algebraic Properties II

A function \( f : M \rightarrow \mathbb{R} \) with \( \Delta f = 0 \) is called *harmonic*. Properties:

- \( f \) is smooth and analytic
- \( f(x) \) is the average of \( f \) over any disk around \( x \):
  \[
  f(x) = \frac{1}{\pi r^2} \int_{B(x,r)} f(y) \, dA
  \]

- *maximum principle*: \( f \) has no local maxima or minima in \( M \)
- (can have saddle points)
for a curve $\gamma(u) = (x[u], y[u]) : \mathbb{R} \to \mathbb{R}^2$

- total Dirichlet energy $\int \|\nabla x\|^2 + \|\nabla y\|^2$ is arc length
for a curve $\gamma(u) = (x[u], y[u]) : \mathbb{R} \rightarrow \mathbb{R}^2$

- total Dirichlet energy $\int \| \nabla x \|^2 + \| \nabla y \|^2$ is arc length
- $\Delta \gamma = (\Delta x, \Delta y)$ is gradient of arc length
for a curve $\gamma(u) = (x[u], y[u]) : \mathbb{R} \to \mathbb{R}^2$

- total Dirichlet energy $\int \| \nabla x \|^2 + \| \nabla y \|^2$ is arc length
- $\Delta \gamma = (\Delta x, \Delta y)$ is gradient of arc length
- $\Delta \gamma$ is the curvature normal $\kappa \hat{n}$
for a curve $\gamma(u) = (x[u], y[u]) : \mathbb{R} \to \mathbb{R}^2$

- total Dirichlet energy $\int \|\nabla x\|^2 + \|\nabla y\|^2$ is arc length
- $\Delta \gamma = (\Delta x, \Delta y)$ is gradient of arc length
- $\Delta \gamma$ is the curvature normal $\kappa \hat{n}$
- minimal curves are harmonic
for a curve $\gamma(u) = (x[u], y[u]) : \mathbb{R} \rightarrow \mathbb{R}^2$

- total Dirichlet energy $\int \| \nabla x \|^2 + \| \nabla y \|^2$ is arc length
- $\Delta \gamma = (\Delta x, \Delta y)$ is gradient of arc length
- $\Delta \gamma$ is the curvature normal $\kappa \mathbf{n}$
- minimal curves are harmonic (straight lines)
Essential Geometric Properties II

for a surface \( r(u, v) = (x[u, v], y[u, v], z[u, v]) : \mathbb{R} \rightarrow \mathbb{R}^3 \)

- total Dirichlet energy is surface area
- \( \Delta r = (\Delta x, \Delta y, \Delta z) \) is gradient of surface area
for a surface \( r(u, v) = (x[u, v], y[u, v], z[u, v]) : \mathbb{R} \to \mathbb{R}^3 \)

- total Dirichlet energy is surface area
- \( \Delta r = (\Delta x, \Delta y, \Delta z) \) is gradient of surface area
- \( \Delta r \) is the mean curvature normal \( 2H\hat{n} \)
Essential Geometric Properties II

for a surface $r(u,v) = (x[u,v], y[u,v], z[u,v]) : \mathbb{R} \to \mathbb{R}^3$

- total Dirichlet energy is surface area
- $\Delta r = (\Delta x, \Delta y, \Delta z)$ is gradient of surface area
- $\Delta r$ is the mean curvature normal $2H\hat{n}$
- minimal surfaces are harmonic!

Images: Paul Nylander
• $\Delta$ is intrinsic
Essential Geometric Properties III

- $\Delta$ is intrinsic
- for $\Omega \subset \mathbb{R}^2$, rigid motions of $\Omega$ don’t change $\Delta$
• $\Delta$ is intrinsic
• for $\Omega \subset \mathbb{R}^2$, rigid motions of $\Omega$ don’t change $\Delta$
• for a surface $\Omega$, isometric deformations of $\Omega$ don’t change $\Delta$
on line segment $[0, 1]$:

- recall Fourier basis: $\phi_i(x) = \cos(ix)$
on line segment $[0, 1]$:

- recall Fourier basis: $\phi_i(x) = \cos(ix)$
- can decompose $f = \sum \alpha_i \phi_i$
on line segment $[0, 1]$:

- recall Fourier basis: $\phi_i(x) = \cos(ix)$
- can decompose $f = \sum \alpha_i \phi_i$
- $\phi_i$ satisfies $\Delta \phi_i = -i^2 \phi_i$
Signal Processing on a Line

on line segment $[0, 1]$:

- recall Fourier basis: $\phi_i(x) = \cos(ix)$
- can decompose $f = \sum \alpha_i \phi_i$
- $\phi_i$ satisfies $\Delta \phi_i = -i^2 \phi_i$
- Dirichlet energy of $f$: $\sum i^2 \alpha_i$
on line segment $[0, 1]$:  

- recall Fourier basis: $\phi_i(x) = \cos(ix)$  
- can decompose $f = \sum \alpha_i \phi_i$  
- $\phi_i$ satisfies $\Delta \phi_i = -i^2 \phi_i$  
- Dirichlet energy of $f$: $\sum i^2 \alpha_i$  

$$f(x) = \sum_{i=1}^{N} \alpha_i \phi_i(x) + \sum_{i=N+1}^{\infty} \alpha_i \phi_i(x)$$

low-frequency base  
high-frequency detail
• \( \phi \) is a (Dirichlet) eigenfunction of \( \Delta \) on \( M \) w/ eigenvalue \( \lambda \):

\[
\Delta \phi(x) = \lambda \phi(x), \quad x \in M
\]

\[
0 = \phi(x), \quad x \in \partial M
\]

\[
1 = \int_M \|\phi\| \, dA.
\]
• \( \phi \) is a (Dirichlet) eigenfunction of \( \Delta \) on \( M \) w/ eigenvalue \( \lambda \):

\[
\Delta \phi(x) = \lambda \phi(x), \quad x \in M \\
0 = \phi(x), \quad x \in \partial M \\
1 = \int_M \|\phi\| \, dA.
\]

• recall intuition: \( \Delta \) as \( \infty \)-dim negative-semidefinite matrix
Laplacian Spectrum

• $\phi$ is a (Dirichlet) eigenfunction of $\Delta$ on $M$ w/ eigenvalue $\lambda$:

\[
\Delta \phi(x) = \lambda \phi(x), \quad x \in M
\]

\[
0 = \phi(x), \quad x \in \partial M
\]

\[
1 = \int_M \|\phi\| \, dA.
\]

• recall intuition: $\Delta$ as $\infty$-dim negative-semidefinite matrix
• expect orthogonal eigenfunctions with negative eigenvalue
Laplacian Spectrum

• \( \phi \) is a (Dirichlet) eigenfunction of \( \Delta \) on \( M \) w/ eigenvalue \( \lambda \):

\[
\Delta \phi(x) = \lambda \phi(x), \quad x \in M
\]
\[
0 = \phi(x), \quad x \in \partial M
\]
\[
1 = \int_M \|\phi\| \, dA.
\]

• recall intuition: \( \Delta \) as \( \infty \)-dim negative-semidefinite matrix
• expect orthogonal eigenfunctions with negative eigenvalue
• spectrum is discrete: countably many eigenfunctions,

\[
0 \geq \lambda_1 \geq \lambda_2 \geq \lambda_3 \ldots
\]
Laplacian Spectrum of Bunny

$\phi_2$

$\phi_3$

$\phi_6$

$\phi_{18}$
Laplacian Spectrum: Signal Processing

- expand function $f$ in eigenbasis:
  \[ f(x) = \sum_i \alpha_i \phi_i(x) \]

- Dirichlet energy of $f$:
  \[ E(f) = \int_M \| \nabla f \|^2 dA = -\int_M f \Delta f dA = \sum_i \alpha_i^2 (-\lambda_i) \]
Laplacian Spectrum: Signal Processing

- expand function $f$ in eigenbasis:

$$f(x) = \sum_{i} \alpha_i \phi_i(x)$$

- Dirichlet energy of $f$:

$$E(f) = \int_{M} \|\nabla f\|^2 dA = - \int_{M} f \Delta f \ dA = \sum_{i} \alpha_i^2 (-\lambda_i)$$

- large $\lambda_i$ terms dominate
Laplacian Spectrum: Signal Processing

- large $\lambda_i$ terms dominate

$$f(x) = \sum_{i=1}^{N} \alpha_i \phi_i(x) + \sum_{i=N+1}^{\infty} \alpha_i \phi_i(x)$$

- low-frequency base
- high-frequency detail
Laplacian Spectrum: Special Cases

perhaps you’ve heard of

- Fourier basis: \( M = \mathbb{R}^n \)
- spherical harmonics: \( M = \text{sphere} \)
perhaps you’ve heard of

- Fourier basis: \( M = \mathbb{R}^n \)
- spherical harmonics: \( M = \text{sphere} \)

Laplacian spectrum generalizes these to any surface
Discretization
Discrete Geometry
Triangle Meshes

- approximate surface by *triangles*
Triangle Meshes

- approximate surface by *triangles*
- “glued together” along edges
Triangle Meshes

- approximate surface by *triangles*
- “glued together” along edges
- many possible data structures
Triangle Meshes

- approximate surface by *triangles*
- “glued together” along edges
- many possible data structures
- *half edge, quad edge, corner table, …*
Triangle Meshes

- approximate surface by *triangles*
- “glued together” along edges
- many possible data structures
  - *half edge, quad edge, corner table, …*
- for simplicity: *vertex-face adjacency list*
Triangle Meshes

- approximate surface by triangles
- “glued together” along edges
- many possible data structures
- half edge, quad edge, corner table, …
- for simplicity: vertex-face adjacency list
- (will be enough for our applications!)
Vertex-Face Adjacency List—Example

# xyz-coordinates of vertices
v 0 0 0
v 1 0 0
v .5 .866 0
v .5 -.866 0

# vertex-face adjacency info
f 1 2 3
f 1 4 2
Manifold
Nonmanifold
Manifold Triangle Mesh

- manifold
  - "locally disk-like"

- Which triangle meshes are manifold?
- Two triangles per edge (no "fins")
- Every vertex looks like a "fan"

Why?

Simplicity.

(Sometimes not necessary...)
Manifold Triangle Mesh

- manifold
  \(\text{locally disk-like}\)
- Two triangles per edge (no "fins")
- Every vertex looks like a "fan"
- Why?
  Simplicity.
- (Sometimes not necessary...)

\(\text{manifold} \iff \text{"locally disk-like"}\)

Which triangle meshes are manifold?
Manifold Triangle Mesh

- manifold \iff "locally disk-like"
- Which triangle meshes are manifold?
- Two triangles per edge (no "fins")
Manifold Triangle Mesh

- manifold $\iff$ “locally disk-like”
- Which triangle meshes are manifold?
- Two triangles per edge (no “fins”)
- Every vertex looks like a “fan”
Manifold Triangle Mesh

- **manifold** \(\iff\) “locally disk-like”
- Which triangle meshes are manifold?
- Two triangles per edge (no “fins”)
- Every vertex looks like a “fan”
Manifold Triangle Mesh

- manifold $\iff$ “locally disk-like”
- Which triangle meshes are manifold?
- Two triangles per edge (no “fins”)
- Every vertex looks like a “fan”
- Why? Simplicity.
- (Sometimes not necessary...)
Manifold Triangle Mesh

- manifold $\iff$ "locally disk-like"
- Which triangle meshes are manifold?
- Two triangles per edge (no "fins")
- Every vertex looks like a "fan"
- (Sometimes not necessary...)
The Cotangent Laplacian

(Assuming a manifold triangle mesh...)

\[(\Delta u)_i \approx \frac{1}{2A_i} \sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(u_i - u_j)\]
The Cotangent Laplacian

(Assuming a manifold triangle mesh . . .)

\[(\Delta u)_i \approx \frac{1}{2A_i} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(u_i - u_j)\]

The set \(\mathcal{N}(i)\) contains the immediate neighbors of vertex \(i\).
The Cotangent Laplacian

(Assuming a manifold triangle mesh...)

\[(\Delta u)_i \approx \frac{1}{2A_i} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(u_i - u_j)\]

The set \(\mathcal{N}(i)\) contains the immediate neighbors of vertex \(i\).

The quantity \(A_i\) is vertex area—for now: 1/3rd of triangle areas.
Origin of the Cotan Formula?

- Many different ways to derive it
Origin of the Cotan Formula?

- Many different ways to derive it
  - piecewise linear finite elements (FEM)
Origin of the Cotan Formula?

• Many different ways to derive it
  • piecewise linear finite elements (FEM)
  • finite volumes
Many different ways to derive it
- piecewise linear finite elements (FEM)
- finite volumes
- discrete exterior calculus (DEC)
Origin of the Cotan Formula?

• Many different ways to derive it
  - piecewise linear finite elements (FEM)
  - finite volumes
  - discrete exterior calculus (DEC)
  - …
Origin of the Cotan Formula?

- Many different ways to derive it
  - piecewise linear finite elements (FEM)
  - finite volumes
  - discrete exterior calculus (DEC)
  - …

- Re-derived in many different contexts:
  - mean curvature flow [Desbrun et al., 1999]
Many different ways to derive it
  • piecewise linear finite elements (FEM)
  • finite volumes
  • discrete exterior calculus (DEC)
  • …

Re-derived in many different contexts:
  • mean curvature flow [Desbrun et al., 1999]
  • minimal surfaces [Pinkall and Polthier, 1993]
Origin of the Cotan Formula?

- Many different ways to derive it
  - piecewise linear finite elements (FEM)
  - finite volumes
  - discrete exterior calculus (DEC)
  - …
- Re-derived in many different contexts:
  - mean curvature flow [Desbrun et al., 1999]
  - minimal surfaces [Pinkall and Polthier, 1993]
  - electrical networks [Duffin, 1959]
Origin of the Cotan Formula?

- Many different ways to derive it
  - piecewise linear finite elements (FEM)
  - finite volumes
  - discrete exterior calculus (DEC)
  - …

- Re-derived in many different contexts:
  - mean curvature flow [Desbrun et al., 1999]
  - minimal surfaces [Pinkall and Polthier, 1993]
  - electrical networks [Duffin, 1959]
  - Poisson equation [MacNeal, 1949]
Origin of the Cotan Formula?

- Many different ways to derive it
  - piecewise linear finite elements (FEM)
  - finite volumes
  - discrete exterior calculus (DEC)
  - ...

- Re-derived in many different contexts:
  - mean curvature flow [Desbrun et al., 1999]
  - minimal surfaces [Pinkall and Polthier, 1993]
  - electrical networks [Duffin, 1959]
  - Poisson equation [MacNeal, 1949]
  - (Courant? Frankel? Manhattan Project?)
Origin of the Cotan Formula?

- Many different ways to derive it
  - piecewise linear finite elements (FEM)
  - finite volumes
  - discrete exterior calculus (DEC)
  - ...

- Re-derived in many different contexts:
  - mean curvature flow [Desbrun et al., 1999]
  - minimal surfaces [Pinkall and Polthier, 1993]
  - electrical networks [Duffin, 1959]
  - Poisson equation [MacNeal, 1949]
  - (Courant? Frankel? Manhattan Project?)

- All these different viewpoints yield **exact same** cotan formula
Origin of the Cotan Formula?

- Many different ways to derive it
  - piecewise linear finite elements (FEM)
  - finite volumes
  - discrete exterior calculus (DEC)
  - …
- Re-derived in many different contexts:
  - mean curvature flow [Desbrun et al., 1999]
  - minimal surfaces [Pinkall and Polthier, 1993]
  - electrical networks [Duffin, 1959]
  - Poisson equation [MacNeal, 1949]
  - (Courant? Frankel? Manhattan Project?)

- All these different viewpoints yield exact same cotan formula
- For three different derivations, see [Crane et al., 2013a]
If the network is first laid out on a large sheet of drawing paper, the angles can be measured with a protractor and the distances scaled off with sufficient accuracy in a short time.

"If the mesh is sufficiently fine, this will not lead to a large error. It indicates, however, that an attempt should be made to keep the triangles as nearly regular as possible."
• Integrate over each dual cell $C_i$
Integrate over each dual cell $C_i$

$$\int_{C_i} \Delta u = \int_{C_i} f \quad ("weak")$$
• Integrate over each dual cell $C_i$
• $\int_{C_i} \Delta u = \int_{C_i} f$ (“weak”)
• Right-hand side approximated as $A_i f_i$
Cotan-Laplacian via Finite Volumes

- Integrate over each dual cell $C_i$
- $\int_{C_i} \Delta u = \int_{C_i} f$ (“weak”)
- Right-hand side approximated as $A_i f_i$
- Left-hand side becomes $\int_{C_i} \nabla \cdot \nabla u = \int_{\partial C_i} n \cdot \nabla u$ (Stokes’)
Integrate over each dual cell $C_i$

$$\int_{C_i} \Delta u = \int_{C_i} f \quad \text{("weak")}$$

Right-hand side approximated as $A_i f_i$

Left-hand side becomes $\int_{C_i} \nabla \cdot \nabla u = \int_{\partial C_i} n \cdot \nabla u \quad \text{(Stokes')}$

Get piecewise integral over boundary $\sum_{e_j \in \partial C_i} \int_{e_j} n_j \cdot \nabla u$
Cotan-Laplacian via Finite Volumes

- Integrate over each dual cell $C_i$
- $\int_{C_i} \Delta u = \int_{C_i} f$ ("weak")
- Right-hand side approximated as $A_i f_i$
- Left-hand side becomes $\int_{C_i} \nabla \cdot \nabla u = \int_{\partial C_i} n \cdot \nabla u$ (Stokes')
- Get piecewise integral over boundary $\sum_{e_j \in \partial C_i} \int_{e_j} n_j \cdot \nabla u$
- After some trigonometry: $\frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) (u_i - u_j)$
Cotan-Laplacian via Finite Volumes

- Integrate over each dual cell $C_i$
- $\int_{C_i} \Delta u = \int_{C_i} f$ ("weak")
- Right-hand side approximated as $A_i f_i$
- Left-hand side becomes $\int_{C_i} \nabla \cdot \nabla u = \int_{\partial C_i} n \cdot \nabla u$ (Stokes’)
- Get piecewise integral over boundary $\sum_{e_j \in \partial C_i} \int_{e_j} n_j \cdot \nabla u$
- After some trigonometry: $\frac{1}{2} \sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) (u_i - u_j)$
- (Can divide by $A_i$ to approximate pointwise value)
Triangle Quality—Rule of Thumb

(For further discussion see Shewchuk, “What Is a Good Linear Finite Element?”)
Triangle Quality—Delaunay Property

Delaunay

Not Delaunay
Some simple ways to improve quality of Laplacian

- If \( a + b > p \), "flip" the edge; after enough flips, mesh will be Delaunay [Bobenko and Springborn, 2005]
- Other ways to improve mesh (edge collapse, edge split, ...)
- Particular interest recently in interface tracking
- For more, see [Dunyach et al., 2013, Wojtan et al., 2011]
Local Mesh Improvement

Some simple ways to improve quality of Laplacian

E.g., if $\alpha + \beta > \pi$, “flip” the edge; after enough flips, mesh will be Delaunay [Bobenko and Springborn, 2005]
Some simple ways to improve quality of Laplacian

E.g., if $\alpha + \beta > \pi$, “flip” the edge; after enough flips, mesh will be Delaunay [Bobenko and Springborn, 2005]

Other ways to improve mesh (edge collapse, edge split, …)
Local Mesh Improvement

- Some simple ways to improve quality of Laplacian
- E.g., if $\alpha + \beta > \pi$, “flip” the edge; after enough flips, mesh will be Delaunay [Bobenko and Springborn, 2005]
- Other ways to improve mesh (edge collapse, edge split, …)
- Particular interest recently in interface tracking
Local Mesh Improvement

- Some simple ways to improve quality of Laplacian
  - E.g., if $\alpha + \beta > \pi$, “flip” the edge; after enough flips, mesh will be Delaunay [Bobenko and Springborn, 2005]
- Other ways to improve mesh (edge collapse, edge split, …)
- Particular interest recently in interface tracking
- For more, see [Dunyach et al., 2013, Wojtan et al., 2011].
Meshes and Matrices

- So far, Laplacian expressed as a sum:

(Laplace matrix, ignoring weights!)
• So far, Laplacian expressed as a sum:
• $\frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) (u_j - u_i)$

(Laplace matrix, ignoring weights!)
So far, Laplacian expressed as a sum:
• \( \frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) (u_j - u_i) \)
• For computation, encode using matrices

(Laplace matrix, ignoring weights!)
Meshes and Matrices

- So far, Laplacian expressed as a sum:
  \[ \frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(u_j - u_i) \]
- For computation, encode using matrices
- First, give each vertex an index 1, \ldots, |V|

(Laplace matrix, ignoring weights!)
Meshes and Matrices

- So far, Laplacian expressed as a sum:
- \( \frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(u_j - u_i) \)
- For computation, encode using matrices
- First, give each vertex an index 1, \ldots, |V|
- Weak Laplacian is matrix \( C \in \mathbb{R}^{|V| \times |V|} \)

(Laplace matrix, ignoring weights!)
Meshes and Matrices

- So far, Laplacian expressed as a sum:
- \( \frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) (u_j - u_i) \)
- For computation, encode using matrices
- First, give each vertex an index 1, \ldots, |V|
- *Weak Laplacian* is matrix \( C \in \mathbb{R}^{|V| \times |V|} \)
- Row \( i \) represents sum for \( i \)th vertex

(Laplace matrix, *ignoring weights!*)
So far, Laplacian expressed as a sum:
$$\frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) (u_j - u_i)$$

For computation, encode using matrices

First, give each vertex an index $1, \ldots, |V|$

*Weak Laplacian* is matrix $C \in \mathbb{R}^{|V| \times |V|}$

Row $i$ represents sum for $i$th vertex

- $C_{ij} = \frac{1}{2} \cot \alpha_{ij} + \cot \beta_{ij}$ for $j \in \mathcal{N}(i)$

(Laplace matrix, *ignoring weights!*)
So far, Laplacian expressed as a sum:
\[ \frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) (u_j - u_i) \]
For computation, encode using matrices
First, give each vertex an index
1, \ldots, |V|
Weak Laplacian is matrix \( C \in \mathbb{R}^{|V| \times |V|} \)
Row \( i \) represents sum for \( i \)th vertex
\[ C_{ij} = \frac{1}{2} \cot \alpha_{ij} + \cot \beta_{ij} \text{ for } j \in \mathcal{N}(i) \]
\[ C_{ii} = -\sum_{j \in \mathcal{N}(i)} C_{ij} \]
(Laplace matrix, ignoring weights!)
Meshes and Matrices

- So far, Laplacian expressed as a sum:
  \[ \frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(u_j - u_i) \]
- For computation, encode using matrices
- First, give each vertex an index 1, \ldots, |V|
- **Weak Laplacian** is matrix \( C \in \mathbb{R}^{|V| \times |V|} \)
  - Row \( i \) represents sum for \( i \)th vertex
  - \( C_{ij} = \frac{1}{2} \cot \alpha_{ij} + \cot \beta_{ij} \) for \( j \in \mathcal{N}(i) \)
  - \( C_{ii} = -\sum_{j \in \mathcal{N}(i)} C_{ij} \)
- All other entries are zero

(Laplace matrix, *ignoring weights!*)

(Laplace matrix, *ignoring weights!*)
Meshes and Matrices

- So far, Laplacian expressed as a sum:
  
  \[ \frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(u_j - u_i) \]

- For computation, encode using matrices

- First, give each vertex an index 1, \ldots, |V|

- Weak Laplacian is matrix \( C \in \mathbb{R}^{|V| \times |V|} \)

- Row \( i \) represents sum for \( i \)th vertex

  \[ C_{ij} = \frac{1}{2} \cot \alpha_{ij} + \cot \beta_{ij} \text{ for } j \in \mathcal{N}(i) \]

  \[ C_{ii} = - \sum_{j \in \mathcal{N}(i)} C_{ij} \]

- All other entries are zero

- Use \textit{sparse} matrices!
Meshes and Matrices

- So far, Laplacian expressed as a sum:
  $$\frac{1}{2} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) (u_j - u_i)$$

- For computation, encode using matrices
  - First, give each vertex an index $1, \ldots, |V|$
  - Weak Laplacian is matrix $C \in \mathbb{R}^{|V| \times |V|}$
    - Row $i$ represents sum for $i$th vertex
      - $C_{ij} = \frac{1}{2} \cot \alpha_{ij} + \cot \beta_{ij}$ for $j \in \mathcal{N}(i)$
      - $C_{ii} = -\sum_{j \in \mathcal{N}(i)} C_{ij}$
  - All other entries are zero
  - Use sparse matrices!

  (MATLAB: sparse, SuiteSparse: cholmod_sparse, Eigen: SparseMatrix)
Mass Matrix

- Matrix C encodes only part of Laplacian—recall that

\[(\Delta u)_i = \frac{1}{2A_i} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(u_j - u_i)\]
Mass Matrix

- Matrix C encodes only part of Laplacian—recall that
  \[(\Delta u)_i = \frac{1}{2A_i} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(u_j - u_i)\]
- Still need to incorporate vertex areas $A_i$
Mass Matrix

- Matrix $C$ encodes only part of Laplacian—recall that

$$\Delta u_i = \frac{1}{2A_i} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(u_j - u_i)$$

- Still need to incorporate vertex areas $A_i$
- For convenience, build diagonal mass matrix $M \in \mathbb{R}^{\lvert V \rvert \times \lvert V \rvert}$:

$$M = \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_{\lvert V \rvert} \end{bmatrix}$$
Mass Matrix

- Matrix $C$ encodes only part of Laplacian—recall that
  \[
  (\Delta u)_i = \frac{1}{2A_i} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(u_j - u_i)
  \]

- Still need to incorporate vertex areas $A_i$
- For convenience, build diagonal mass matrix $M \in \mathbb{R}^{|V| \times |V|}$:
  \[
  M = \begin{bmatrix}
  A_1 \\
  \vdots \\
  A_{|V|}
  \end{bmatrix}
  \]

- Entries are just $M_{ii} = A_i$ (all other entries are zero)
• Matrix $C$ encodes only part of Laplacian—recall that

$$(\Delta u)_i = \frac{1}{2A_i} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(u_j - u_i)$$

• Still need to incorporate vertex areas $A_i$
• For convenience, build diagonal mass matrix $M \in \mathbb{R}^{|V| \times |V|}$:

$$M = \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_{|V|} \end{bmatrix}$$

• Entries are just $M_{ii} = A_i$ (all other entries are zero)
• Laplace operator is then $L := M^{-1}C$
Mass Matrix

- Matrix C encodes only part of Laplacian—recall that

\[(\Delta u)_i = \frac{1}{2A_i} \sum_{j \in \mathcal{N}(i)} \left( \cot \alpha_{ij} + \cot \beta_{ij} \right) (u_j - u_i)\]

- Still need to incorporate vertex areas \(A_i\)
- For convenience, build diagonal mass matrix \(M \in \mathbb{R}^{|V| \times |V|}:\)

\[
M = \begin{bmatrix}
A_1 \\
\vdots \\
A_{|V|}
\end{bmatrix}
\]

- Entries are just \(M_{ii} = A_i\) (all other entries are zero)
- Laplace operator is then \(L := M^{-1}C\)
- Applying \(L\) to a column vector \(u \in \mathbb{R}^{|V|}\) “implements” the cotan formula shown above
Discrete Poisson / Laplace Equation

- Poisson equation $\Delta u = f$ becomes linear algebra problem:
  
  $$Lu = f$$

- Discrete approximation $u$ approaches smooth solution $u$ as mesh is refined (for smooth data, "good" meshes...). 

- Laplace is just Poisson with "zero" on right hand side!
• Poisson equation $\Delta u = f$ becomes linear algebra problem:

$$Lu = f$$

• Vector $f \in \mathbb{R}^{|V|}$ is given data; $u \in \mathbb{R}^{|V|}$ is unknown.
Poisson equation $\Delta u = f$ becomes linear algebra problem:

$$Lu = f$$

- Vector $f \in \mathbb{R}^{|V|}$ is given data; $u \in \mathbb{R}^{|V|}$ is unknown.
- Discrete approximation $u$ approaches smooth solution $u$ as mesh is refined (for smooth data, “good” meshes...).
Discrete Poisson / Laplace Equation

- Poisson equation $\Delta u = f$ becomes linear algebra problem:
  \[ Lu = f \]

- Vector $f \in \mathbb{R}^{|V|}$ is given data; $u \in \mathbb{R}^{|V|}$ is unknown.

- Discrete approximation $u$ approaches smooth solution $u$ as mesh is refined (for smooth data, “good” meshes…).

- Laplace is just Poisson with “zero” on right hand side!
Discrete Heat Equation

- Heat equation $\frac{du}{dt} = \Delta u$ must also be discretized in time

Explicit:

$$\left( u_{k+1} - u_k \right) / h = Lu_k$$ (cheaper to compute)

Implicit:

$$\left( u_{k+1} - u_k \right) / h = Lu_k + u_{k+1}$$ (more stable)

Implicit update becomes linear system

$$ \left( I + hL \right) u_{k+1} = u_k $$
Discrete Heat Equation

- Heat equation \( \frac{du}{dt} = \Delta u \) must also be discretized in time
- Replace time derivative with finite difference:

\[
\frac{du}{dt} \Rightarrow \frac{u_{k+1} - u_k}{h}, \quad h > 0
\]

“time step”
Discrete Heat Equation

- Heat equation \( \frac{du}{dt} = \Delta u \) must also be discretized in time.
- Replace time derivative with finite difference:

  \[
  \frac{du}{dt} \Rightarrow \frac{u_{k+1} - u_k}{h}, \quad h > 0
  \]

  "time step"

- How (or really, "when") do we approximate \( \Delta u \)?
Discrete Heat Equation

- Heat equation $\frac{du}{dt} = \Delta u$ must also be discretized in time.
- Replace time derivative with finite difference:
  \[
  \frac{du}{dt} \Rightarrow \frac{u_{k+1} - u_k}{h}, \quad h > 0
  \]
  “time step”

- How (or really, “when”) do we approximate $\Delta u$?
- *Explicit*: $(u_{k+1} - u_k)/h = Lu_k$ (cheaper to compute)
Discrete Heat Equation

- Heat equation $\frac{du}{dt} = \Delta u$ must also be discretized in time.
- Replace time derivative with finite difference:
  \[
  \frac{du}{dt} \Rightarrow \frac{u_{k+1} - u_k}{h}, \quad h > 0
  \]

  "time step"

- How (or really, “when”) do we approximate $\Delta u$?
  - **Explicit**: $(u_{k+1} - u_k)/h = Lu_k$ (cheaper to compute)
  - **Implicit**: $(u_{k+1} - u_k)/h = Lu_{k+1}$ (more stable)
Discrete Heat Equation

- Heat equation $\frac{du}{dt} = \Delta u$ must also be discretized in time.
- Replace time derivative with finite difference:

$$
\frac{du}{dt} \Rightarrow \frac{u_{k+1} - u_k}{h}, \quad h > 0
$$

“time step”

- How (or really, “when”) do we approximate $\Delta u$?
- **Explicit:** $(u_{k+1} - u_k) / h = Lu_k$ (cheaper to compute)
- **Implicit:** $(u_{k+1} - u_k) / h = Lu_{k+1}$ (more stable)
- Implicit update becomes linear system $(I - hL)u_{k+1} = u_k$
Discrete Eigenvalue Problem

- Smallest eigenvalue problem $\Delta u = \lambda u$ becomes

  $$Lu = \lambda u$$

  for smallest nonzero eigenvalue $\lambda$. 
Discrete Eigenvalue Problem

- Smallest eigenvalue problem $\Delta u = \lambda u$ becomes $Lu = \lambda u$
  for smallest nonzero eigenvalue $\lambda$.
- Can be solved using (inverse) power method:
  - Pick random $u_0$
  - Until convergence:
    - Solve $Lu_{k+1} = u_k$
    - Remove mean value from $u_{k+1}$
    - $u_{k+1} \leftarrow u_{k+1} / |u_{k+1}|$
Discrete Eigenvalue Problem

- Smallest eigenvalue problem $\Delta u = \lambda u$ becomes $Lu = \lambda u$

  for smallest nonzero eigenvalue $\lambda$.

- Can be solved using \textit{(inverse) power method}:
  
  - Pick random $u_0$
  
  - Until convergence:
    
    - Solve $Lu_{k+1} = u_k$
    
    - Remove mean value from $u_{k+1}$
    
    - $u_{k+1} \leftarrow u_{k+1} / |u_{k+1}|$

- By \textit{prefactoring} $L$, overall cost is nearly identical to solving a single Poisson equation!
Properties of cotan-Laplace

- *Always, always, always* positive-semidefinite $f^T C f \geq 0$
  (even if cotan weights are negative!)

- No boundary constant vector in the kernel / cokernel

- Why does it matter? E.g., for Poisson equation:
  - solution is unique only up to constant shift
  - if RHS has nonzero mean, cannot be solved!

- Exhibits maximum principle on Delaunay mesh

  - Delaunay: triangle circumcircles are empty
  - Maximum principle: solution to Laplace equation has no interior extrema (local max or min)

- NOTE: non-Delaunay meshes can also exhibit max principle! (And often do.) Delaunay sufficient but not necessary. Currently no nice, simple necessary condition on mesh geometry.

- For more, see [Wardetzky et al., 2007]
Properties of cotan-Laplace

- *Always, always, always* positive-semidefinite $f^T C f \geq 0$ (even if cotan weights are negative!)
- Why? $f^T C f$ is *identical* to summing $||\nabla f||^2$!

No boundary constant vector in the kernel / cokernel

- Why does it matter? E.g., for Poisson equation:
  - solution is unique only up to constant shift
  - if RHS has nonzero mean, cannot be solved!
- Exhibits maximum principle on Delaunay mesh
  - Delaunay: triangle circumcircles are empty
  - Maximum principle: solution to Laplace equation has no interior extrema (local max or min)
- **NOTE**: non-Delaunay meshes can also exhibit max principle! (And often do.) Delaunay sufficient but not necessary. Currently no nice, simple necessary condition on mesh geometry.
- For more, see [Wardetzky et al., 2007]
Properties of cotan-Laplace

- *Always, always, always* positive-semidefinite $f^T C f \geq 0$ (even if cotan weights are negative!)
- Why? $f^T C f$ is *identical* to summing $\| \nabla f \|^2$!
- No boundary $\Rightarrow$ constant vector in the kernel / cokernel
Properties of cotan-Laplace

- *Always, always, always* positive-semidefinite $f^T Cf \geq 0$ (even if cotan weights are negative!)
- Why? $f^T Cf$ is *identical* to summing $||\nabla f||^2$!
- No boundary $\Rightarrow$ constant vector in the kernel / cokernel
- Why does it matter? E.g., for Poisson equation:

---

**NOTE**: non-Delaunay meshes can also exhibit max principle! (And often do.) Delaunay sufficient but not necessary. Currently no nice, simple necessary condition on mesh geometry.

- For more, see [Wardetzky et al., 2007]
Properties of cotan-Laplace

- Always, always, always positive-semidefinite $f^T C f \geq 0$
  (even if cotan weights are negative!)
- Why? $f^T C f$ is identical to summing $||\nabla f||^2$!
- No boundary $\Rightarrow$ constant vector in the kernel / cokernel
- Why does it matter? E.g., for Poisson equation:
  - solution is unique only up to constant shift
Properties of cotan-Laplace

- *Always, always, always* positive-semidefinite $f^T C f \geq 0$ (even if cotan weights are negative!)
- Why? $f^T C f$ is *identical* to summing $||\nabla f||^2$!
- No boundary $\Rightarrow$ constant vector in the kernel / cokernel
- Why does it matter? E.g., for Poisson equation:
  - solution is unique only up to constant shift
  - if RHS has nonzero mean, cannot be solved!

Note: non-Delaunay meshes can also exhibit max principle! (And often do.) Delaunay sufficient but not necessary. Currently no nice, simple necessary condition on mesh geometry.

For more, see [Wardetzky et al., 2007]
Properties of cotan-Laplace

- *Always, always, always* positive-semidefinite $f^T Cf \geq 0$  
  (even if cotan weights are negative!)
- Why? $f^T Cf$ is *identical* to summing $||\nabla f||^2$!
- No boundary $\Rightarrow$ constant vector in the kernel / cokernel
- Why does it matter? E.g., for Poisson equation:
  - solution is unique only up to constant shift
  - if RHS has nonzero mean, cannot be solved!
- Exhibits *maximum principle* on Delaunay mesh

\[ \text{NOTE: non-Delaunay meshes can also exhibit max principle! (And often do.) Delaunay sufficient but not necessary. Currently no nice, simple necessary condition on mesh geometry.} \]

For more, see [Wardetzky et al., 2007]
Properties of cotan-Laplace

- *Always, always, always* positive-semidefinite $f^T Cf \geq 0$ (even if cotan weights are negative!)
- Why? $f^T Cf$ is *identical* to summing $||\nabla f||^2$!
- No boundary $\Rightarrow$ constant vector in the kernel / cokernel
- Why does it matter? E.g., for Poisson equation:
  - solution is unique only up to constant shift
  - if RHS has nonzero mean, cannot be solved!
- Exhibits *maximum principle* on Delaunay mesh
  - Delaunay: triangle circumcircles are empty

- For more, see [Wardetzky et al., 2007]
Properties of cotan-Laplace

- *Always, always, always* positive-semidefinite $f^\top C f \geq 0$ (even if cotan weights are negative!)
- **Why?** $f^\top C f$ is *identical* to summing $||\nabla f||^2$!
- No boundary $\Rightarrow$ constant vector in the kernel / cokernel
- Why does it matter? E.g., for Poisson equation:
  - solution is unique only up to constant shift
  - if RHS has nonzero mean, cannot be solved!
- Exhibits *maximum principle* on Delaunay mesh
  - Delaunay: triangle circumcircles are empty
  - Maximum principle: solution to Laplace equation has no interior extrema (local max or min)

*NOTE*: non-Delaunay meshes can also exhibit max principle! (And often do.) Delaunay sufficient but not necessary. Currently no nice, simple necessary condition on mesh geometry.

*For more, see [Wardetzky et al., 2007]*
Properties of cotan-Laplace

- *Always, always, always* positive-semidefinite $f^T C f \geq 0$ (even if cotan weights are negative!)
- Why? $f^T C f$ is *identical* to summing $||\nabla f||^2$!
- No boundary $\Rightarrow$ constant vector in the kernel / cokernel
- Why does it matter? E.g., for Poisson equation:
  - solution is unique only up to constant shift
  - if RHS has nonzero mean, cannot be solved!
- Exhibits *maximum principle* on Delaunay mesh
  - Delaunay: triangle circumcircles are empty
  - Maximum principle: solution to Laplace equation has no interior extrema (local max or min)
- **NOTE**: non-Delaunay meshes can also exhibit max principle! (And often do.) Delaunay *sufficient* but not *necessary*. Currently no nice, simple *necessary* condition on mesh geometry.

• For more, see [Wardetzky et al., 2007]
Properties of cotan-Laplace

- *Always, always, always* positive-semidefinite $f^T C f \geq 0$ (even if cotan weights are negative!)
- Why? $f^T C f$ is *identical* to summing $||\nabla f||^2$!
- No boundary $\Rightarrow$ constant vector in the kernel / cokernel
- Why does it matter? E.g., for Poisson equation:
  - solution is unique only up to constant shift
  - if RHS has nonzero mean, cannot be solved!
- Exhibits *maximum principle* on Delaunay mesh
  - Delaunay: triangle circumcircles are empty
  - Maximum principle: solution to Laplace equation has no interior extrema (local max or min)
- **NOTE**: non-Delaunay meshes can also exhibit max principle! (And often do.) Delaunay *sufficient* but not *necessary*. Currently no nice, simple *necessary* condition on mesh geometry.
- For more, see [Wardetzky et al., 2007]
Numerical Issues—Symmetry

• “Best” case for sparse linear systems:
  
  \[
  \text{symmetric positive-(semi)definite} \quad (A^T = A, x^TAx \geq 0 \ \forall x)
  \]
Numerical Issues—Symmetry

- “Best” case for sparse linear systems:
  \[ \text{symmetric positive-(semi)definite} \quad (A^T = A, x^T A x \geq 0 \ \forall x) \]
- Many good solvers (Cholesky, conjugate gradient, \ldots )

\[ C u = M f \]
\[ \text{In other words: just multiply by vertex areas!} \]
\[ \text{Seemingly superficial change} \ldots \]
\[ \text{but makes computation simpler / more efficient} \]
Numerical Issues—Symmetry

• “Best” case for sparse linear systems:
  
  \[ \text{symmetric positive-(semi)definite} \ (A^T = A, \ x^T Ax \geq 0 \ \forall x) \]

• Many good solvers (Cholesky, conjugate gradient, …)

• Discrete Poisson equation looks like \( M^{-1} Cu = f \)
Numerical Issues—Symmetry

• “Best” case for sparse linear systems:
  \[ \text{symmetric positive-}(semi)\text{definite} \ (A^T = A, x^T A x \geq 0 \ \forall x) \]

• Many good solvers (Cholesky, conjugate gradient, . . . )

• Discrete Poisson equation looks like \( M^{-1} C u = f \)

• \( C \) is symmetric, but \( M^{-1} C \) is not!

• Can easily be made symmetric:
  \[ C u = M f \]
Numerical Issues—Symmetry

• “Best” case for sparse linear systems: 
  \( \text{symmetric positive-(semi)definite} \ (A^T = A, x^T Ax \geq 0 \ \forall x) \)

• Many good solvers (Cholesky, conjugate gradient, . . .)

• Discrete Poisson equation looks like \( M^{-1}Cu = f \)

• \( C \) is symmetric, but \( M^{-1}C \) is not!

• Can easily be made symmetric:

  \[ Cu = Mf \]

• In other words: just multiply by vertex areas!
Numerical Issues—Symmetry

• “Best” case for sparse linear systems:
  
  \( A^T = A, \ x^T A x \geq 0 \ \forall x \)

• Many good solvers (Cholesky, conjugate gradient, …)

• Discrete Poisson equation looks like \( M^{-1} C u = f \)

• \( C \) is symmetric, but \( M^{-1} C \) is not!

• Can easily be made symmetric:
  
  \( C u = M f \)

• In other words: just multiply by vertex areas!

• Seemingly superficial change…
Numerical Issues—Symmetry

- “Best” case for sparse linear systems: *symmetric positive-(semi)definite* \((A^T = A, x^T A x \geq 0 \ \forall x)\)
- Many good solvers (Cholesky, conjugate gradient, …)
- Discrete Poisson equation looks like \(M^{-1} C u = f\)
- \(C\) is symmetric, but \(M^{-1} C\) is not!
- Can easily be made symmetric:
  \[
  Cu = Mf
  \]

- In other words: just multiply by vertex areas!
- Seemingly superficial change…
- …but makes computation simpler / more efficient
Numerical Issues—Symmetry

- “Best” case for sparse linear systems:
  \[
  \text{symmetric positive-(semi)definite} \quad (A^T = A, \ x^T A x \geq 0 \ \forall x)
  \]
- Many good solvers (Cholesky, conjugate gradient, . . .)
- Discrete Poisson equation looks like \( M^{-1} C u = f \)
- \( C \) is symmetric, but \( M^{-1} C \) is not!
- Can easily be made symmetric:
  \[
  C u = M f
  \]
- In other words: just multiply by vertex areas!
- Seemingly superficial change . . .
- . . . but makes computation simpler / more efficient
Numerical Issues—Symmetry, continued

- Can also make heat equation symmetric
Numerical Issues—Symmetry, continued

- Can also make heat equation symmetric
- Instead of \((1 - hL)u_{k+1} = u_k\), use

\[
(M - hC)u_{k+1} = Mu_k
\]

- What about smallest eigenvalue problem \(Lu = \lambda u\)?
  - Two options:
    1. Solve generalized eigenvalue problem \(Cu = \lambda Mu\)
    2. Solve \(M^{1/2}CM^{1/2} \tilde{u} = \lambda \tilde{u}\), recover \(u = M^{1/2} \tilde{u}\)

Note: \(M^{1/2}\) just means "put \(1/p\) on the diagonal!"
• Can also make heat equation symmetric
• Instead of \((I - hL)u_{k+1} = u_k\), use

\[(M - hC)u_{k+1} = Mu_k\]

• What about smallest eigenvalue problem \(Lu = \lambda u\)?
• Can also make heat equation symmetric
• Instead of \((l - hL)u_{k+1} = u_k\), use
\[
(M - hC)u_{k+1} = Mu_k
\]
• What about smallest eigenvalue problem \(Lu = \lambda u\)?
• Two options:
Numerical Issues—Symmetry, continued

- Can also make heat equation symmetric
- Instead of \((I - hL)u_{k+1} = u_k\), use
  \[(M - hC)u_{k+1} = Mu_k\]

- What about smallest eigenvalue problem \(Lu = \lambda u\)?
- Two options:
  1. Solve generalized eigenvalue problem \(Cu = \lambda Mu\)
• Can also make heat equation symmetric
• Instead of $\left( I - hL \right) u_{k+1} = u_k$, use

\[
\left( M - hC \right) u_{k+1} = M u_k
\]

• What about smallest eigenvalue problem $Lu = \lambda u$?
• Two options:
  1. Solve \textit{generalized} eigenvalue problem $Cu = \lambda Mu$
  2. Solve $M^{-1/2}CM^{-1/2} \tilde{u} = \lambda \tilde{u}$, recover $u = M^{-1/2} \tilde{u}$
Numerical Issues—Symmetry, continued

- Can also make heat equation symmetric
- Instead of \((I - hL)u_{k+1} = u_k\), use
  \[(M - hC)u_{k+1} = Mu_k\]
- What about smallest eigenvalue problem \(Lu = \lambda u\)?
- Two options:
  1. Solve generalized eigenvalue problem \(Cu = \lambda Mu\)
  2. Solve \(M^{-1/2}CM^{-1/2}\tilde{u} = \lambda \tilde{u}\), recover \(u = M^{-1/2}\tilde{u}\)
- Note: \(M^{-1/2}\) just means “put \(1/\sqrt{A_i}\) on the diagonal!”
Numerical Issues—Symmetry, continued

- Can also make heat equation symmetric
- Instead of $(1 - hL)u_{k+1} = u_k$, use
  \[(M - hC)u_{k+1} = Mu_k\]

- What about smallest eigenvalue problem $Lu = \lambda u$?
- Two options:
  1. Solve generalized eigenvalue problem $Cu = \lambda Mu$
  2. Solve $M^{-1/2}CM^{-1/2}\tilde{u} = \lambda\tilde{u}$, recover $u = M^{-1/2}\tilde{u}$

- Note: $M^{-1/2}$ just means “put $1/\sqrt{A_i}$ on the diagonal!”
Numerical Issues—Direct vs. Iterative Solvers

- Direct (e.g., $LL^T$, $LU$, $QR$, ...)
Numerical Issues—Direct vs. Iterative Solvers

- **Direct** (e.g., $LL^T$, $LU$, $QR$, …)
  - **pros**: great for multiple right-hand sides; (can be) less sensitive to numerical instability; solve many types of problems, under/overdetermined systems.
Numerical Issues—Direct vs. Iterative Solvers

- Direct (e.g., $LL^T$, $LU$, QR, ...)
  - **pros:** great for multiple right-hand sides; (can be) less sensitive to numerical instability; solve many types of problems, under/overdetermined systems.
  - **cons:** prohibitively expensive for large problems; factors are quite dense for 3D (volumetric) problems
Numerical Issues—Direct vs. Iterative Solvers

- **Direct** (e.g., $LL^T$, $LU$, $QR$, …)
  - **pros**: great for multiple right-hand sides; (can be) less sensitive to numerical instability; solve many types of problems, under/overdetermined systems.
  - **cons**: prohibitively expensive for large problems; factors are quite dense for 3D (volumetric) problems

- **Iterative** (e.g., conjugate gradient, multigrid, …)
Numerical Issues—Direct vs. Iterative Solvers

- **Direct (e.g., \(LL^T, LU, QR, \ldots\))**
  - **pros**: great for multiple right-hand sides; (can be) less sensitive to numerical instability; solve many types of problems, under/overdetermined systems.
  - **cons**: prohibitively expensive for large problems; factors are quite dense for 3D (volumetric) problems

- **Iterative (e.g., conjugate gradient, multigrid, \ldots)**
  - **pros**: can handle very large problems; can be implemented via *callback* (instead of matrix); asymptotic running times approaching linear time (in theory...)

- No perfect solution! Each problem is different.
Numerical Issues—Direct vs. Iterative Solvers

- **Direct (e.g., $LL^T$, $LU$, $QR$, …)**
  - **pros:** great for multiple right-hand sides; (can be) less sensitive to numerical instability; solve many types of problems, under/overdetermined systems.
  - **cons:** prohibitively expensive for large problems; factors are quite dense for 3D (volumetric) problems

- **Iterative (e.g., conjugate gradient, multigrid, …)**
  - **pros:** can handle very large problems; can be implemented via "callback" (instead of matrix); asymptotic running times approaching linear time (in theory…)
  - **cons:** poor performance without good preconditioners; less benefit for multiple right-hand sides; best-in-class methods may handle only symmetric positive-(semi)definite systems

No perfect solution! Each problem is different.
Numerical Issues—Direct vs. Iterative Solvers

• Direct (e.g., $LL^T$, $LU$, $QR$, …)
  • **Pros**: great for multiple right-hand sides; (can be) less sensitive to numerical instability; solve many types of problems, under/overdetermined systems.
  • **Cons**: prohibitively expensive for large problems; factors are quite dense for 3D (volumetric) problems

• Iterative (e.g., conjugate gradient, multigrid, …)
  • **Pros**: can handle very large problems; can be implemented via callback (instead of matrix); asymptotic running times approaching linear time (in theory…)
  • **Cons**: poor performance without good preconditioners; less benefit for multiple right-hand sides; best-in-class methods may handle only symmetric positive-(semi)definite systems

• No perfect solution! Each problem is different.
Solving Equations in Linear Time

- Is solving Poisson, Laplace, etc., *truly* linear time in 2D?
Solving Equations in Linear Time

• Is solving Poisson, Laplace, etc., truly linear time in 2D?
• Jury is still out, but keep inching forward:
  • [Vaidya, 1991]—use spanning tree as preconditioner
  • [Alon et al., 1995]—use low-stretch spanning trees
  • [Spielman and Teng, 2004]—first “nearly linear time” solver
  • [Krishnan et al., 2013]—practical solver for graphics
• Lots of recent activity in both preconditioners and direct solvers (e.g., [Koutis et al., 2011], [Gillman and Martinsson, 2013])
Solving Equations in Linear Time

- Is solving Poisson, Laplace, etc., *truly* linear time in 2D?
- Jury is still out, but keep inching forward:
  - [Vaidya, 1991]—use spanning tree as preconditioner
  - [Alon et al., 1995]—use low-stretch spanning trees
  - [Spielman and Teng, 2004]—first “nearly linear time” solver
  - [Krishnan et al., 2013]—practical solver for graphics
  - Lots of recent activity in both preconditioners and direct solvers (e.g., [Koutis et al., 2011], [Gillman and Martinsson, 2013])

- *Best theoretical results may lack practical implementations!*
Solving Equations in Linear Time

• Is solving Poisson, Laplace, etc., truly linear time in 2D?
• Jury is still out, but keep inching forward:
  • [Vaidya, 1991]—use spanning tree as preconditioner
  • [Alon et al., 1995]—use low-stretch spanning trees
  • [Spielman and Teng, 2004]—first “nearly linear time” solver
  • [Krishnan et al., 2013]—practical solver for graphics
  • Lots of recent activity in both preconditioners and direct solvers (e.g., [Koutis et al., 2011], [Gillman and Martinsson, 2013])

• Best theoretical results may lack practical implementations!
• Older codes benefit from extensive low-level optimization
• Is solving Poisson, Laplace, etc., truly linear time in 2D?
• Jury is still out, but keep inching forward:
  • [Vaidya, 1991]—use spanning tree as preconditioner
  • [Alon et al., 1995]—use low-stretch spanning trees
  • [Spielman and Teng, 2004]—first “nearly linear time” solver
  • [Krishnan et al., 2013]—practical solver for graphics
  • Lots of recent activity in both preconditioners and direct solvers (e.g., [Koutis et al., 2011], [Gillman and Martinsson, 2013])

  • Best theoretical results may lack practical implementations!
  • Older codes benefit from extensive low-level optimization
  • Long term: probably indistinguishable from $O(n)$
Boundary Conditions

- PDE (Laplace, Poisson, heat equation, etc.) determines behavior “inside” domain $\Omega$

$\Omega$

$\partial \Omega$
Boundary Conditions

- PDE (Laplace, Poisson, heat equation, etc.) determines behavior “inside” domain $\Omega$
- Also need to say how solution behaves on boundary $\partial \Omega$
Boundary Conditions

- PDE (Laplace, Poisson, heat equation, etc.) determines behavior “inside” domain $\Omega$
- Also need to say how solution behaves on boundary $\partial \Omega$
- Often trickiest part (both mathematically & numerically)
Boundary Conditions

- PDE (Laplace, Poisson, heat equation, etc.) determines behavior “inside” domain $\Omega$
- Also need to say how solution behaves on boundary $\partial \Omega$
- Often trickiest part (both mathematically & numerically)
- Very easy to get wrong!
Dirichlet Boundary Conditions

• “Dirichlet” \( \iff \) prescribe values
“Dirichlet” \(\iff\) prescribe \textit{values}

E.g., \(\phi(0) = a, \phi(1) = b\)
Dirichlet Boundary Conditions

- “Dirichlet” \(\iff\) prescribe \textit{values}
- E.g., \(\phi(0) = a, \phi(1) = b\)
- (Many possible functions “in between!”)
Neumann Boundary Conditions

- "Neumann" \iff prescribe derivatives
Neumann Boundary Conditions

- "Neumann" $\iff$ prescribe derivatives
- E.g., $\phi'(0) = u, \phi'(1) = v$
Neumann Boundary Conditions

- “Neumann” \(\iff\) prescribe derivatives
- E.g., \(\phi'(0) = u, \phi'(1) = v\)
- (Again, many possible solutions.)
Both Neumann & Dirichlet

- Or: prescribe some values, some derivatives

\[ f(0) = u, \quad f(1) = b \]

- (What about \( f(0) = v, \quad f(1) = b \)?)
Both Neumann & Dirichlet

- Or: prescribe some values, some derivatives
- E.g., $\phi'(0) = u$, $\phi(1) = b$
Both Neumann & Dirichlet

- Or: prescribe some values, some derivatives
- E.g., $\phi'(0) = u$, $\phi(1) = b$
- (What about $\phi'(1) = v$, $\phi(1) = b$?)
Laplace w/ Dirichlet Boundary Conditions (1D)

- 1D Laplace: $\frac{\partial^2 \phi}{\partial x^2} = 0$

- Can we always satisfy Dirichlet boundary conditions?
  - Yes: a line can interpolate any two points
Laplace w/ Dirichlet Boundary Conditions (1D)

- 1D Laplace: $\frac{\partial^2 \phi}{\partial x^2} = 0$
- Solutions: $\phi(x) = cx + d$ (linear functions)
Laplace w/ Dirichlet Boundary Conditions (1D)

- 1D Laplace: $\frac{\partial^2 \phi}{\partial x^2} = 0$
- Solutions: $\phi(x) = cx + d$ (linear functions)
- Can we always satisfy Dirichlet boundary conditions?
Laplace w/ Dirichlet Boundary Conditions (1D)

- 1D Laplace: $\frac{\partial^2 \phi}{\partial x^2} = 0$
- Solutions: $\phi(x) = cx + d$ (linear functions)
- Can we always satisfy Dirichlet boundary conditions?

- Yes: a line can interpolate any two points
Laplace w/ Neumann Boundary Conditions (1D)

- What about Neumann boundary conditions?

Solution must still be a line: \( f(x) = cx + d \)

Can we prescribe the derivative at both ends?

No! A line can have only one slope!

In general: solutions to PDE may not exist for given BCs.
Laplace w/ Neumann Boundary Conditions (1D)

- What about Neumann boundary conditions?
- Solution must still be a line: $\phi(x) = cx + d$
Laplace w/ Neumann Boundary Conditions (1D)

- What about Neumann boundary conditions?
- Solution must still be a line: $\phi(x) = cx + d$
- Can we prescribe the derivative at both ends?
Laplace w/ Neumann Boundary Conditions (1D)

- What about Neumann boundary conditions?
- Solution must still be a line: $\phi(x) = cx + d$
- Can we prescribe the derivative at both ends?

- No! A line can have only one slope!
Laplace w/ Neumann Boundary Conditions (1D)

- What about Neumann boundary conditions?
- Solution must still be a line: \( \phi(x) = cx + d \)
- Can we prescribe the derivative at both ends?

\[ \begin{align*}
\phi(x) & \quad \text{at} \\
u & \quad \text{at} \\
v & \quad \text{at}
\end{align*} \]

- No! A line can have only one slope!
- In general: solutions to PDE may not exist for given BCs
Laplace w/ Dirichlet Boundary Conditions (2D)

- 2D Laplace: $\Delta \phi = 0$
Laplace w/ Dirichlet Boundary Conditions (2D)

- 2D Laplace: $\Delta \phi = 0$
- Can we always satisfy Dirichlet boundary conditions?
Laplace w/ Dirichlet Boundary Conditions (2D)

- 2D Laplace: $\Delta \phi = 0$
- Can we always satisfy Dirichlet boundary conditions?
- Yes: Laplace is steady-state solution to heat flow $\frac{d}{dt} \phi = \Delta \phi$
- Dirichlet data is just “heat” along boundary
Laplace w/ Neumann Boundary Conditions (2D)

- What about Neumann boundary conditions?

\[ Df = 0 \]

Want to prescribe normal derivative \( n \cdot r_{f} \)

Wasn't always possible in 1D . . .

In 2D, we have divergence theorem:

\[ \int_{W} 0 = \int_{W} Df = \int_{W} r \cdot r_{f} = \int_{\partial W} n \cdot r_{f} \]

Conclusion: can only solve \( Df = 0 \) if Neumann BCs have zero mean!
Laplace w/ Neumann Boundary Conditions (2D)

- What about Neumann boundary conditions?
- Still want to solve $\Delta \phi = 0$
- Want to prescribe normal derivative $n \cdot \nabla \phi$

Wasn't always possible in 1D... In 2D, we have divergence theorem:

$$\int_{\Omega} Df = \int_{\partial \Omega} n \cdot \nabla f$$

Conclusion: can only solve $\Delta \phi = 0$ if Neumann BCs have zero mean!
• What about Neumann boundary conditions?
• Still want to solve $\Delta \phi = 0$
• Want to prescribe normal derivative $n \cdot \nabla \phi$
• Wasn’t always possible in 1D…
Laplace w/ Neumann Boundary Conditions (2D)

- What about Neumann boundary conditions?
- Still want to solve $\Delta \phi = 0$
- Want to prescribe normal derivative $n \cdot \nabla \phi$
- Wasn’t always possible in 1D…
- In 2D, we have divergence theorem:

$$
\int_{\Omega} 0 = \int_{\Omega} \Delta \phi = \int_{\Omega} \nabla \cdot \nabla \phi = \int_{\partial \Omega} n \cdot \nabla \phi
$$
Laplace w/ Neumann Boundary Conditions (2D)

- What about Neumann boundary conditions?
- Still want to solve $\Delta \phi = 0$
- Want to prescribe normal derivative $n \cdot \nabla \phi$
- Wasn’t always possible in 1D…
- In 2D, we have divergence theorem:

$$\int_\Omega 0 \overset{!}{=} \int_\Omega \Delta \phi = \int_\Omega \nabla \cdot \nabla \phi = \int_{\partial \Omega} n \cdot \nabla \phi$$

- Conclusion: can only solve $\Delta \phi = 0$ if Neumann BCs have zero mean!
Discrete Boundary Conditions - Dirichlet

- Suppose we want to solve $\Delta u = f$ s.t. $u|_{\partial \Omega} = g$ (Poisson equation w/ Dirichlet boundary conditions)
Suppose we want to solve $\Delta u = f$ s.t. $u|_{\partial \Omega} = g$ (Poisson equation w/ Dirichlet boundary conditions)

Discretized Poisson equation as $Cu = Mf$

Here $N_{\partial \Omega}(i)$ denotes neighbors of $i$ on the boundary
Discrete Boundary Conditions - Dirichlet

- Suppose we want to solve \( \Delta u = f \) s.t. \( u|_{\partial \Omega} = g \) (Poisson equation w/ Dirichlet boundary conditions).
- Discretized Poisson equation as \( Cu = Mf \)
- Let \( I, B \) denote interior, boundary vertices, respectively. Get

\[
\begin{bmatrix}
C_{II} & C_{IB} \\
C_{BI} & C_{BB}
\end{bmatrix}
\begin{bmatrix}
u_I \\
u_B
\end{bmatrix}
= \begin{bmatrix}
M_{II} & 0 \\
0 & M_{BB}
\end{bmatrix}
\begin{bmatrix}
f_I \\
f_B
\end{bmatrix}
\]

- Since \( u_B \) is known (boundary values), solve just \( C_{II}u_I = M_{II}f_I \) for \( u_I \) (right-hand side is known).
- Can skip matrix multiply and compute entries of RHS directly:

\[
A_i f_i = \sum_{j \in N_{\partial}(i)} \left( \cot \alpha_{ij} + \cot \beta_{ij} \right) u_j
\]

Here \( N_{\partial}(i) \) denotes neighbors of \( i \) on the boundary.
Discrete Boundary Conditions - Dirichlet

- Suppose we want to solve $\Delta u = f$ s.t. $u|_{\partial \Omega} = g$ (Poisson equation w/ Dirichlet boundary conditions)
- Discretized Poisson equation as $Cu = Mf$
- Let $I, B$ denote interior, boundary vertices, respectively. Get

$$
\begin{bmatrix}
C_{II} & C_{IB} \\
C_{BI} & C_{BB}
\end{bmatrix}
\begin{bmatrix}
u_I \\
u_B
\end{bmatrix}
= 
\begin{bmatrix}
M_{II} & 0 \\
0 & M_{BB}
\end{bmatrix}
\begin{bmatrix}
f_I \\
f_B
\end{bmatrix}
$$

- Since $u_B$ is known (boundary values), solve just $C_{II}u_I = M_{II}f_I - C_{IB}u_B$ for $u_I$ (right-hand side is known).
Discrete Boundary Conditions - Dirichlet

- Suppose we want to solve $\Delta u = f$ s.t. $u|_{\partial \Omega} = g$ (Poisson equation w/ Dirichlet boundary conditions)
- Discretized Poisson equation as $Cu = Mf$
- Let $I, B$ denote interior, boundary vertices, respectively. Get

$$\begin{bmatrix}
  C_{II} & C_{IB} \\
  C_{BI} & C_{BB}
\end{bmatrix}
\begin{bmatrix}
  u_I \\
  u_B
\end{bmatrix}
= \begin{bmatrix}
  M_{II} & 0 \\
  0 & M_{BB}
\end{bmatrix}
\begin{bmatrix}
  f_I \\
  f_B
\end{bmatrix}$$

- Since $u_B$ is known (boundary values), solve just $C_{II}u_I = M_{II}f_I - C_{IB}u_B$ for $u_I$ (right-hand side is known).
- Can skip matrix multiply and compute entries of RHS directly: $A_if_i - \sum_{j \in N_\partial(i)} (\cot \alpha_{ij} + \cot \beta_{ij})u_j$
- Here $N_\partial(i)$ denotes neighbors of $i$ on the boundary
Discrete Boundary Conditions - Neummann

- Integrate both sides of $\Delta u = f$ over cell $C_i$ ("finite volume")

\[
\int_{C_i} f = \int_{C_i} \Delta u = \int_{C_i} \nabla \cdot \nabla u = \int_{\partial C_i} n \cdot \nabla u
\]

- Gives usual cotangent formula for interior vertices; for boundary vertex $i$, yields

\[
A_{ii} = \frac{1}{2} (g_a + g_b) + \frac{1}{2} \sum_{j \in \mathcal{N}_{\text{int}}} (\cot \alpha_{ij} + \cot \beta_{ij}) (u_j - u_i)
\]

- Here $g_a, g_b$ are prescribed normal derivatives; just subtract from RHS and solve $Cu = Mf$ as usual
Other possible boundary conditions (e.g., Robin)
**Discrete Boundary Conditions - Neumann**

- Other possible boundary conditions (e.g., Robin)
- Dirichlet, Neumann most common—implementation of other BCs will be similar

Easy test? Compute the residual $r = Ax - b$. If the relative residual $||r||/||b||$ is far from zero (e.g., greater than $10^{-14}$ in double precision), you did not actually solve your problem!
Discrete Boundary Conditions - Neumann

- Other possible boundary conditions (e.g., Robin)
- Dirichlet, Neumann most common—implementation of other BCs will be similar
- When in doubt, return to smooth equations and integrate!
- Easy test? Compute the residual $r = Ax - b$. If the relative residual $\frac{|r|}{|b|}$ is far from zero (e.g., greater than $10^{-14}$ in double precision), you did not actually solve your problem!
Discrete Boundary Conditions - Neumann

- Other possible boundary conditions (e.g., Robin)
- Dirichlet, Neumann most common—implementation of other BCs will be similar
- When in doubt, return to smooth equations and integrate!
- …and make sure your equation has a solution!
Discrete Boundary Conditions - Neumann

- Other possible boundary conditions (e.g., Robin)
- Dirichlet, Neumann most common—implementation of other BCs will be similar
- When in doubt, return to smooth equations and integrate!
- …and make sure your equation has a solution!
- Solver will NOT always tell you if there’s a problem!

\[ r = Ax - b \]

If the relative residual \(|r|/|b|\) is far from zero (e.g., greater than \(10^{-14}\) in double precision), you did not actually solve your problem!
Discrete Boundary Conditions - Neumann

• Other possible boundary conditions (e.g., Robin)
• Dirichlet, Neumann most common—implementation of other BCs will be similar
• When in doubt, return to smooth equations and integrate!
• …and make sure your equation has a solution!
• Solver will NOT always tell you if there’s a problem!
• Easy test? Compute the residual $r := Ax - b$. If the relative residual $\|r\|_\infty / \|b\|_\infty$ is far from zero (e.g., greater than $10^{-14}$ in double precision), you did not actually solve your problem!
Discrete Boundary Conditions - Neumann

- Other possible boundary conditions (e.g., Robin)
- Dirichlet, Neumann most common—implementation of other BCs will be similar
- When in doubt, return to smooth equations and integrate!
- …and make sure your equation has a solution!
- Solver will NOT always tell you if there’s a problem!
- Easy test? Compute the residual $r := Ax - b$. If the relative residual $\|r\|_\infty / \|b\|_\infty$ is far from zero (e.g., greater than $10^{-14}$ in double precision), you did not actually solve your problem!
• Have spent a lot of time on triangle meshes…
• Have spent a lot of time on triangle meshes…
• …plenty of other ways to describe a surface!
Alternative Discretizations

- Have spent a lot of time on triangle meshes...
- ...plenty of other ways to describe a surface!
- E.g., *points* are increasingly popular (due to 3D scanning)
Alternative Discretizations

- Have spent a lot of time on triangle meshes...
- ...plenty of other ways to describe a surface!
- E.g., *points* are increasingly popular (due to 3D scanning)
- Also: more accurate discretization on triangle meshes
• **Quads** popular alternative to triangles. Why?
Quad, Polygon Meshes

- **Quads** popular alternative to triangles. Why?
  - capture *principal curvatures* of a surface

See [Bommes et al., 2013] for further discussion.
• **Quads** popular alternative to triangles. Why?
  - capture *principal curvatures* of a surface
  - nice bases can be built via *tensor products*

---

**Quad, Polygon Meshes**

- Quad, Polygon Meshes
- Quads popular alternative to triangles. Why?
  - capture *principal curvatures* of a surface
  - nice bases can be built via *tensor products*
Quad, Polygon Meshes

- **Quads** popular alternative to triangles. Why?
  - capture *principal curvatures* of a surface
  - nice bases can be built via tensor products
  - see [Bommes et al., 2013] for further discussion
Quad, Polygon Meshes

- **Quads** popular alternative to triangles. Why?
  - capture *principal curvatures* of a surface
  - nice bases can be built via *tensor products*
  - see [Bommes et al., 2013] for further discussion
- More generally: meshes with quads *and* triangles *and* ...
Quad, Polygon Meshes

- **Quads** popular alternative to triangles. Why?
  - capture *principal curvatures* of a surface
  - nice bases can be built via *tensor products*
  - see [Bommes et al., 2013] for further discussion
- More generally: meshes with quads *and* triangles and ... 
- Nice discretization: [Alexa and Wardetzky, 2011]
Quad, Polygon Meshes

- **Quads** popular alternative to triangles. Why?
  - capture *principal curvatures* of a surface
  - nice bases can be built via *tensor products*
  - see [Bommes et al., 2013] for further discussion

- More generally: meshes with quads *and* triangles *and* ...

- Nice discretization:
  [Alexa and Wardetzky, 2011]

- Can then solve all the same problems (Laplace, Poisson, heat, ...)
• Real data often *point cloud* with no connectivity (plus noise, holes…)

\[
d\frac{du}{dt} = \Delta u = \nabla^2 u \Rightarrow \nabla^2 u \equiv \frac{u(T) - u(0)}{T}
\]

• How do we get \(u(T)\)? Convolve \(u\) with (Euclidean) heat kernel

\[
\frac{1}{4\pi T} e^{-\frac{r^2}{4T}}
\]

• Converges with more samples, \(T\) goes to zero (under certain conditions!)

• Details: [Belkin et al., 2009, Liu et al., 2012]

• From there, solve all the same problems! (Again.)
Point Clouds

- Real data often *point cloud* with no connectivity (plus noise, holes…)
- Can still build Laplace operator!
Point Clouds

- Real data often *point cloud* with no connectivity (plus noise, holes…)
- Can still build Laplace operator!
- Rough idea: use heat flow to discretize $\Delta$

\[ \frac{\partial u}{\partial t} = \Delta u \sim \frac{u(T) - u(0)}{T} \]

- How do we get $u(T)$? Convolve $u$ with (Euclidean) heat kernel $\frac{1}{4\pi T} e^{-\frac{x^2}{4T}}$
- Converges with more samples, $T$ goes to zero (under certain conditions!)
- Details: [Belkin et al., 2009, Liu et al., 2012]
- From there, solve all the same problems! (Again.)
Point Clouds

- Real data often *point cloud* with no connectivity (plus noise, holes…)
- Can still build Laplace operator!
- Rough idea: use heat flow to discretize $\Delta$
- $\frac{d}{dt} u = \Delta u \implies \Delta u \approx (u(T) - u(0))/T$

Details: [Belkin et al., 2009, Liu et al., 2012]
Point Clouds

- Real data often *point cloud* with no connectivity (plus noise, holes…)
- Can still build Laplace operator!
- Rough idea: use heat flow to discretize $\Delta$
- $\frac{d}{dt} u = \Delta u \implies \Delta u \approx (u(T) - u(0))/T$
- How do we get $u(T)$? Convolve $u$ with (Euclidean) heat kernel $\frac{1}{4\pi T}e^{r^2/4T}$
Point Clouds

- Real data often *point cloud* with no connectivity (plus noise, holes...)
- Can still build Laplace operator!
- Rough idea: use heat flow to discretize $\Delta$
  \[ \frac{d}{dt} u = \Delta u \implies \Delta u \approx (u(T) - u(0))/T \]
- How do we get $u(T)$? Convolve $u$ with (Euclidean) heat kernel $\frac{1}{4\pi T} e^{-r^2/4T}$
- Converges with more samples, $T$ goes to zero (under certain conditions!)

Details: [Belkin et al., 2009, Liu et al., 2012]

From there, solve all the same problems! (Again.)
Point Clouds

- Real data often *point cloud* with no connectivity (plus noise, holes…)
- Can still build Laplace operator!
- Rough idea: use heat flow to discretize $\Delta$
  
  \[
  \frac{d}{dt} u = \Delta u \implies \Delta u \approx (u(T) - u(0))/T
  \]
- How do we get $u(T)$? Convolve $u$ with (Euclidean) heat kernel $\frac{1}{4\pi T}e^{-r^2/4T}$
- Converges with more samples, $T$ goes to zero (under certain conditions!)
- Details: [Belkin et al., 2009, Liu et al., 2012]
Point Clouds

- Real data often point cloud with no connectivity (plus noise, holes…)
- Can still build Laplace operator!
- Rough idea: use heat flow to discretize $\Delta$
  \[ \frac{d}{dt} u = \Delta u \implies \Delta u \approx (u(T) - u(0)) / T \]
- How do we get $u(T)$? Convolve $u$ with (Euclidean) heat kernel $\frac{1}{4\pi T} e^{-r^2/4T}$
- Converges with more samples, $T$ goes to zero (under certain conditions!)
- Details: [Belkin et al., 2009, Liu et al., 2012]
- From there, solve all the same problems! (Again.)
Earlier saw Laplacian discretized via dual mesh

- Space of orthogonal duals explored by [Mullen et al., 2011]
- Leads to many applications in geometry processing [de Goes et al., 2012, de Goes et al., 2013, de Goes et al., 2014]
• Earlier saw Laplacian discretized via *dual mesh*
• Different duals lead to operators with different accuracy
• Earlier saw Laplacian discretized via *dual mesh*
• Different duals lead to operators with different accuracy
• Space of *orthogonal duals* explored by [Mullen et al., 2011]
• Earlier saw Laplacian discretized via dual mesh
• Different duals lead to operators with different accuracy
• Space of orthogonal duals explored by [Mullen et al., 2011]
• Leads to many applications in geometry processing
  [de Goes et al., 2012, de Goes et al., 2013, de Goes et al., 2014]
Volumes / Tetrahedral Meshes

- Same problems (Poisson, Laplace, etc.) can also be solved on volumes

- Popular choice: tetrahedral meshes (graded, conform to boundary, ...)

- Many ways to get Laplace matrix
  - One nice way: discrete exterior calculus (DEC) [Hirani, 2003, Desbrun et al., 2005]
  - Just incidence matrices (e.g., which tets contain which triangles?) & primal / dual volumes (area, length, etc.).
  - Added bonus: play with definition of dual to improve accuracy [Mullen et al., 2011].
Volumes / Tetrahedral Meshes

- Same problems (Poisson, Laplace, etc.) can also be solved on volumes
- Popular choice: tetrahedral meshes (graded, conform to boundary, ...)

- Many ways to get Laplace matrix
  - One nice way: discrete exterior calculus (DEC) [Hirani, 2003, Desbrun et al., 2005]
  - Just incidence matrices (e.g., which tets contain which triangles?) & primal / dual volumes (area, length, etc.).
  - Added bonus: play with definition of dual to improve accuracy [Mullen et al., 2011].
Volumes / Tetrahedral Meshes

- Same problems (Poisson, Laplace, etc.) can also be solved on volumes
- Popular choice: *tetrahedral* meshes (graded, conform to boundary, ...)
- Many ways to get Laplace matrix
Volumes / Tetrahedral Meshes

- Same problems (Poisson, Laplace, etc.) can also be solved on volumes
- Popular choice: *tetrahedral* meshes (graded, conform to boundary, …)
- Many ways to get Laplace matrix
- One nice way: discrete exterior calculus (DEC) [Hirani, 2003, Desbrun et al., 2005]
Volumes / Tetrahedral Meshes

- Same problems (Poisson, Laplace, etc.) can also be solved on volumes
- Popular choice: tetrahedral meshes (graded, conform to boundary, ...)
- Many ways to get Laplace matrix
- One nice way: discrete exterior calculus (DEC) [Hirani, 2003, Desbrun et al., 2005]
- Just incidence matrices (e.g., which tets contain which triangles?) & primal / dual volumes (area, length, etc.).
Volumes / Tetrahedral Meshes

- Same problems (Poisson, Laplace, etc.) can also be solved on volumes
- Popular choice: tetrahedral meshes (graded, conform to boundary, ...)
- Many ways to get Laplace matrix
- One nice way: discrete exterior calculus (DEC) [Hirani, 2003, Desbrun et al., 2005]
- Just incidence matrices (e.g., which tets contain which triangles?) & primal / dual volumes (area, length, etc.).
- Added bonus: play with definition of dual to improve accuracy [Mullen et al., 2011].
- Covered some standard discretizations

...and More!
• Covered some standard discretizations
• Many possibilities (level sets, hex meshes…)

...and More!
...and More!

- Covered some standard discretizations
- Many possibilities (level sets, hex meshes…)
- Often enough to have gradient $G$ and inner product $W$. 
• Covered some standard discretizations
• Many possibilities (level sets, hex meshes…)
• Often enough to have gradient $G$ and inner product $W$.
• (weak!) Laplacian is then $C = G^T W G$ (think Dirichlet energy)
...and More!

- Covered some standard discretizations
- Many possibilities (level sets, hex meshes…)
- Often enough to have gradient $G$ and inner product $W$.
- (weak!) Laplacian is then $C = G^T W G$ (think Dirichlet energy)
- Key message: build Laplace; do lots of cool stuff.
APPLICATIONS
Remarkably Common Pipeline

\[
\{\text{simple pre-processing}\} \rightarrow (-1) \rightarrow \{\text{simple post-processing}\}
\]
“Our method boils down to ‘backslash’ in Matlab!”
Reminder: Model Equations

\[ \Delta f = 0 \] \quad \textit{Laplace equation}  
\text{Linear solve}

\[ \Delta f = g \] \quad \textit{Poisson equation}  
\text{Linear solve}

\[ f_t = \Delta f \] \quad \textit{Heat equation}  
\text{ODE time-step}

\[ \Delta \phi_i = \lambda_i \phi_i \] \quad \textit{Vibration modes}  
\text{Eigenproblem}
Reminder: Model Equations

\[ \Delta f = 0 \quad \text{Laplace equation} \]
Linear solve

\[ \Delta f = g \quad \text{Poisson equation} \]
Linear solve

\[ f_t = \Delta f \quad \text{Heat equation} \]
ODE time-step

\[ \Delta \phi_i = \lambda_i \phi_i \quad \text{Vibration modes} \]
Eigenproblem
Reminder: Model Equations

\[ \Delta f = 0 \] \text{ Laplace equation} \\

\[ \Delta f = g \] \text{ Poisson equation} \\

\[ f_t = \Delta f \] \text{ Heat equation} \\

\[ \Delta \phi_i = \lambda_i \phi_i \] \text{ Vibration modes}
Reminder: Variational Interpretation

\[ \min_{f(x)} \int_{\Sigma} \| \nabla f(x) \|^2 \, dA \]

\[ \Delta f (x) = 0 \]
Reminder: Variational Interpretation

$$\min_f \int_\Sigma \| \nabla f(x) \|^2 \, dA$$

†<calculus>

$$\Delta f(x) = 0$$

The (inverse) Laplacian wants to make functions smooth.

“Elliptic regularity”
Want smooth $f : M \to \mathbb{R}^2$. 

Application: Mesh Parameterization
\[ \min_{f : M \to \mathbb{R}^2} \int \| \nabla f \|^2 \]

Does this work?
Variational Approach

\[ \Delta f = 0 \]

\[ \min_{f: M \to \mathbb{R}^2} \int \| \nabla f \|^2 \]

Does this work?

\[ f(x) \equiv \text{const.} \]
Harmonic Parameterization

$$\Delta f = 0$$

$$\min_{f:M \to \mathbb{R}^2} \int_{\partial M \text{ fixed}} \| \nabla f \|^2$$

[Eck et al., 1995]
Harmonic Parameterization

\[ \min_{f: M \to \mathbb{R}^2} \int \| \nabla f \|^2 \]

subject to

\[ \Delta f = 0 \text{ in } M \setminus \partial M, \text{ with } f|_{\partial M} \text{ fixed} \]

[Ek et al., 1995]
Reminder: Model Equations

\[ \Delta f = 0 \]
Laplace equation
Linear solve

\[ \Delta f = g \]
Poisson equation
Linear solve

\[ f_t = \Delta f \]
Heat equation
ODE time-step

\[ \Delta \phi_i = \lambda_i \phi_i \]
Vibration modes
Eigenproblem
Recall: Green’s Function

\[ \Delta f = g \]

\[ \Delta g_p = \delta_p \text{ for } p \in M \]
Application: Biharmonic Distances

$$d_b(p, q) \equiv \|g_p - g_q\|_2$$

[Lipman et al., 2010], formula in [Solomon et al., 2014]
Hodge Decomposition

\[ \Delta f = g \]

\[ \vec{v}(x) = R^{90^\circ} \nabla g + \nabla f + \vec{h}(x) \]

- Divergence-free part: \( R^{90^\circ} \nabla g \)
- Curl-free part: \( \nabla f \)
- Harmonic part: \( \vec{h}(x) (= 0 \text{ if surface has no holes}) \)
Computing the Curl-Free Part

\[ \min_{f(x)} \int_{\Sigma} \| \nabla f(x) - \vec{v}(x) \|^2 \, dA \]

\[ \Leftrightarrow \langle \text{calculus} \rangle \]

\[ \Delta f(x) = \nabla \cdot \vec{v}(x) \]

Get divergence-free part as \( \vec{v}(x) - \nabla f(x) \) (when \( \vec{h} \equiv \vec{0} \))
Application: Vector Field Design

\[ \Delta f = g \]

\[ \Delta f = -\bar{K} \implies \vec{v}(x) = \nabla f(x) \]

[Crane et al., 2010, de Goes and Crane, 2010]
Application: Earth Mover’s Distance

\[
\Delta f = g
\]

\[
\min_{\vec{J}(x)} \int_M \| \vec{J}(x) \|
\]

such that \( \vec{J} = R^{90^\circ} \nabla g + \nabla f + \vec{h}(x) \)

\[
\Delta f = \rho_1 - \rho_0
\]

[Solomon et al., 2014]
Reminder: Model Equations

\[ \Delta f = 0 \quad \text{Laplace equation} \]
Linear solve

\[ \Delta f = g \quad \text{Poisson equation} \]
Linear solve

\[ f_t = \Delta f \quad \text{Heat equation} \]
ODE time-step

\[ \Delta \phi_i = \lambda_i \phi_i \quad \text{Vibration modes} \]
Eigenproblem
Generalizing Gaussian Blurs

Gradient descent on $\int \| \nabla f(x) \|^2 \, dx$:

$$\frac{\partial f(x,t)}{\partial t} = \Delta_x f(x,t)$$

with $f(\cdot,0) \equiv f_0(\cdot)$.

Image by M. Bottazzi
Idea: Take $f_0(x)$ to be the coordinate function.
Application: Implicit Fairing

Idea: Take $f_0(x)$ to be the coordinate function.

Detail: $\Delta$ changes over time.

[Desbrun et al., 1999]
**Alternative: Screened Poisson Smoothing**

Simplest incarnation of [Chuang and Kazhdan, 2011]:

$$\min_{f(x)} \alpha^2 \|f - f_0\|^2 + \|\nabla f\|^2$$

\[\iff\]

$$(\alpha^2 I - \Delta)f = \alpha^2 f_0$$
Interesting Connection

\[ f_t = \Delta f \rightarrow \Delta f = g \]

(Semi-)Implicit Euler:

\[ (I - hL)u_{k+1} = u_k \]

Screened Poisson:

\[ (\alpha^2 I - \Delta)f = \alpha^2 f_0 \]
(Semi-)Implicit Euler:

\[(I - hL)u_{k+1} = u_k\]

Screened Poisson:

\[(\alpha^2 I - \Delta)f = \alpha^2 f_0\]

One time step of *implicit Euler* is *screened Poisson*. 
(Semi-)Implicit Euler:

\[(I - hL)u_{k+1} = u_k\]

Screened Poisson:

\[(\alpha^2 I - \Delta)f = \alpha^2 f_0\]

One time step of implicit Euler is screened Poisson.

Accidentally replaced one PDE with another!
Application: The “Heat Method”

Eikonal equation for geodesics:

\[ \| \nabla \phi \|_2 = 1 \]

\[ \implies \text{Need direction of } \nabla \phi. \]
Application: The “Heat Method”

Eikonal equation for geodesics:
\[ \| \nabla \phi \|_2 = 1 \]
\[ \implies \text{Need direction of } \nabla \phi. \]

Idea:
Find \( u \) such that \( \nabla u \) is parallel to geodesic.
Application: The “Heat Method”

1. Integrate $u' = \nabla u$ (heat equation) to time $t \ll 1$.
2. Define vector field $X \equiv -\frac{\nabla u}{\|\nabla u\|_2}$.
3. Solve least-squares problem $\nabla \phi \approx X \iff \Delta \phi = \nabla \cdot X$.

$\nabla u$

$X$

$\phi$

Blazingly fast!

[Crane et al., 2013b]
Reminder: Model Equations

\[ \Delta f = 0 \] \text{ Laplace equation} \newline \text{Linear solve}

\[ \Delta f = g \] \text{ Poisson equation} \newline \text{Linear solve}

\[ f_t = \Delta f \] \text{ Heat equation} \newline \text{ODE time-step}

\[ \Delta \phi_i = \lambda_i \phi_i \] \text{ Vibration modes} \newline \text{Eigenproblem}
Laplace-Beltrami Eigenfunctions

\[ \Delta \phi_i = \lambda_i \phi_i \]

Use eigenvalues and eigenfunctions to characterize shape.

Image by B. Vallet and B. Lévy
Intrinsic Laplacian-Based Descriptors

All computable from eigenfunctions!

- \( \text{HKS}(x; t) = \sum_i e^{\lambda_i t} \phi_i(x)^2 \) [Sun et al., 2009]
- \( \text{GPS}(x) = \left( \frac{\phi_1(x)}{\sqrt{-\lambda_1}}, \frac{\phi_2(x)}{\sqrt{-\lambda_2}}, \ldots \right) \) [Rustamov, 2007]
- \( \text{WKS}(x; e) = C_e \sum_i \phi_i(x)^2 \exp \left( -\frac{1}{2\sigma^2} (e - \log(-\lambda_i)) \right) \) [Aubry et al., 2011]

Many others—or learn a function of eigenvalues!

[Litman and Bronstein, 2014]
$f_t = \Delta f$

**Example: Heat Kernel Signature**

Heat diffusion encodes geometry for all times $t \geq 0$!

$$\text{HKS}(x; t) \equiv k_t(x, x)$$

“Amount of heat diffused from $x$ to itself over at time $t$.”

- Signature of point $x$ is a function of $t \geq 0$
- *Intrinsic* descriptor

[Sun et al., 2009]
\[ \Delta \phi_i = \lambda_i \phi_i \]

\[ \Delta \phi_i = \lambda_i \phi_i, f_0(x) = \sum_i a_i \phi_i(x) \]

\[ \frac{\partial f(x,t)}{\partial t} = \Delta f \text{ with } f(x,0) \equiv f_0(x) \]
HKS via Laplacian Eigenfunctions

\[ \Delta \phi_i = \lambda_i \phi_i \]

\[ \Delta \phi_i = \lambda_i \phi_i, f_0(x) = \sum_i a_i \phi_i(x) \]

\[ \frac{\partial f(x, t)}{\partial t} = \Delta f \text{ with } f(x, 0) \equiv f_0(x) \]

\[ \implies f(x, t) = \sum_i a_i e^{\lambda_i t} \phi_i(x) \]
\[ \Delta \phi_i = \lambda_i \phi_i \]

\[ \Delta \phi_i = \lambda_i \phi_i, f_0(x) = \sum_i a_i \phi_i(x) \]

\[ \frac{\partial f(x,t)}{\partial t} = \Delta f \text{ with } f(x,0) \equiv f_0(x) \]

\[ \implies f(x,t) = \sum_i a_i e^{\lambda_i t} \phi_i(x) \]

\[ \implies \text{HKS}(x; t) \equiv k_t(x,x) = \sum_i e^{\lambda_i t} \phi_i(x)^2 \]
Δφᵢ = λᵢφᵢ

**Application: Shape Retrieval**

Solve problems like *shape similarity search*.

“**Shape DNA**” [Reuter et al., 2006]: Identify a shape by its vector of Laplacian eigenvalues.

![2d MDS plot of mesh Shape-DNAs.](image)
Connect critical points (well-spaced) of $\phi_i$ in Morse-Smale complex.

[Dong et al., 2006]
Other Ideas I

- **Mesh editing**: Displacement of vertices and parameters of a deformation should be *smooth* functions along a surface [Sorkine et al., 2004, Sorkine and Alexa, 2007] (and many others)
• **Surface reconstruction:** Poisson equation helps distinguish inside and outside [Kazhdan et al., 2006]

• **Regularization for mapping:** To compute $\phi : M_1 \to M_2$, ask that $\phi \circ \Delta_1 \approx \Delta_2 \circ \phi$ [Ovsjanikov et al., 2012]
Discrete laplacians on general polygonal meshes.
*ACM Trans. Graph.*, 30(4).

A graph-theoretic game and its application to the k-server problem.

The wave kernel signature: A quantum mechanical approach to shape analysis.
In *Proc. ICCV Workshops*, pages 1626–1633.

Constructing laplace operator from point clouds in rd.

A discrete Laplace-Beltrami operator for simplicial surfaces.
*ArXiv Mathematics e-prints*.

Quad-mesh generation and processing: A survey.

Interactive and anisotropic geometry processing using the screened Poisson equation.

Digital geometry processing with discrete exterior calculus.
In *ACM SIGGRAPH 2013 Courses, SIGGRAPH ’13*, pages 7:1–7:126, New York, NY, USA. ACM.

Trivial connections on discrete surfaces.
Geodesics in heat: A new approach to computing distance based on heat flow.
**ACM Trans. Graph., 32.**

On the equilibrium of simplicial masonry structures.
**ACM Trans. Graph., 32(4):93:1–93:10.**

Blue noise through optimal transport.
**ACM Trans. Graph., 31.**

Trivial connections on discrete surfaces revisited: A simplified algorithm for simply-connected surfaces.

Discrete 2-tensor fields on triangulations.
**Symposium on Geometry Processing.**

Discrete differential forms for computational modeling.
In *ACM SIGGRAPH 2005 Courses, SIGGRAPH ’05, New York, NY, USA*. ACM.

Implicit fairing of irregular meshes using diffusion and curvature flow.

Spectral surface quadrangulation.
**ACM Trans. Graph., 25(3):1057–1066.**

Distributed and lumped networks.
*Journal of Mathematics and Mechanics, 8:793–826.*

Adaptive Remeshing for Real-Time Mesh Deformation.
In *Proceedings of Eurographics Short Papers*, pages 29–32.

Multiresolution analysis of arbitrary meshes.

A direct solver with O(N) complexity for variable coefficient elliptic PDEs discretized via a high-order composite spectral collocation method.
*SIAM Journal on Scientific Computation.*

**Hirani, A.** (2003).
Discrete exterior calculus.

Poisson surface reconstruction.

A nearly $m \log n$ time solver for sdd linear systems.
pages 590–598.

Efficient preconditioning of laplacian matrices for computer graphics.

Biharmonic distance.

Learning spectral descriptors for deformable shape correspondence.

Point-based manifold harmonics.

MacNeal, R. (1949).
The solution of partial differential equations by means of electrical networks.

Hot: Hodge-optimized triangulations.

Functional maps: A flexible representation of maps between shapes.

Computing discrete minimal surfaces and their conjugates.

Laplace–Beltrami spectra as ‘shape-dna’ of surfaces and solids.

Laplace-Beltrami eigenfunctions for deformation invariant shape representation.

Earth mover’s distances on discrete surfaces.
in *Proc. SIGGRAPH, to appear*.

As-rigid-as-possible surface modeling.

Laplacian surface editing.
In Proc. SGP, pages 175–184. ACM.

Nearly linear time algorithms for graph partitioning, graph sparsification, and solving linear systems.
pages 81–90.

A concise and provably informative multi-scale signature based on heat diffusion.

Solving linear equations with symmetric diagonally dominant matrices by constructing good
preconditioners.
Workshop Talk at the IMA Workshop on Graph Theory and Sparse Matrix Computation.

Discrete laplace operators: No free lunch.
In Proceedings of the Fifth Eurographics Symposium on Geometry Processing, SGP ’07, pages 33–37, Aire-la-Ville,
Switzerland, Switzerland. Eurographics Association.

Liquid simulation with mesh-based surface tracking.
In ACM SIGGRAPH 2011 Courses, SIGGRAPH ’11, pages 8:1–8:84, New York, NY, USA. ACM.