CSE 270 (WI 2024) Homework 2

Exterior Product

Exercise 2.1 — 5 pts. Let \( dx, dy, dz, dt \) denote a basis for \((\mathbb{R}^4)^*\). Define the 2-form \( \alpha = u_{12} \, dx \wedge dy + u_{24} \, dy \wedge dt + u_{34} \, dz \wedge dt \) and the 1-form \( \beta = u_2 \, dy + u_3 \, dz \). Compute \( \alpha \wedge \beta \) and \( \alpha \wedge \alpha \).

Vector Identities in 3D

Consider the Euclidean vector space \( \mathbb{R}^3 \) with an orthonormal basis \( e_1, e_2, e_3 \). In this case \( e_1^b, e_2^b, e_3^b \) form the dual basis for \((\mathbb{R}^3)^*\). Each vector \( u = u_1 e_1 + u_2 e_2 + u_3 e_3 \) can be arranged into a 1-form:

\[
\mathbf{u}^b = u_1 e_1^b + u_2 e_2^b + u_3 e_3^b \in \left(\mathbb{R}^3\right)^*
\]

or a 2-form

\[
\star \mathbf{u}^b = u_1 (e_2^b \wedge e_3^b) + u_2 (e_3^b \wedge e_1^b) + u_3 (e_1^b \wedge e_2^b) \in \wedge^2 \mathbb{R}^3^*.
\]

Let \( a, b, w \in \mathbb{R}^3 \), \( \alpha = a^b \), \( \beta = b^b \), and \( \omega = \star w^b \); let \( \det e_1^b \wedge e_2^b \wedge e_3^b \). Then:

- Wedge between two 1-forms corresponds to a cross product: \( \alpha \wedge \beta = \star (a \times b)^b \).
- Wedge between a 1-form and a 2-form corresponds to a dot product: \( \alpha \wedge \omega = \omega \wedge \alpha = \langle a, w \rangle \) det.
- Interior product applied to a 1-form corresponds to a dot product: \( i_a \beta = \langle a, b \rangle \).
- Interior product applied to a 2-form corresponds to a cross product: \( i_{a} \omega = (w \times a)^b \).
- Interior product applied to the volume form gives the correspondence between vectors and 2-forms: \( i_w \det = \omega \).

Important properties of the interior product \( i_a \), an operator that sends \( k \)-forms to \( (k-1) \)-forms, include:

- \( i_a \beta = \beta(a) \) for 1-forms \( \beta \).
- Leibniz rule. If \( \eta \) is a \( k \)-form then \( i_a (\eta \wedge \sigma) = (i_a \eta) \wedge \sigma + (-1)^k \eta \wedge (i_a \eta) \).
- \( i_a i_a \eta = 0 \) for any \( k \)-form \( \eta \).
- In practical terms, \( i_a \eta \beta (b_1, \ldots, b_{k-1}) = \eta(a, b_1, \ldots, b_{k-1}) \) is the insertion of the vector \( a \) to the first slot of a \( k \)-form.

Exercise 2.2 — 10 pts. Use the Leibniz rule, and the vector-form correspondence in 3D, to show the BAC-CAB formula:

\[
\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = \mathbf{b} \langle \mathbf{a}, \mathbf{c} \rangle - \mathbf{c} \langle \mathbf{a}, \mathbf{b} \rangle, \quad \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^3
\]

and the Binet–Cauchy identity

\[
\langle \mathbf{a} \times \mathbf{b}, \mathbf{c} \times \mathbf{d} \rangle = \langle \mathbf{a}, \mathbf{c} \rangle \langle \mathbf{b}, \mathbf{d} \rangle - \langle \mathbf{b}, \mathbf{c} \rangle \langle \mathbf{a}, \mathbf{d} \rangle, \quad \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{R}^3.
\]

Hint: Treat one of the cross products as the wedge product and treat the other cross product as the interior product.

Vector Calculus Operators

Let \( f : \mathbb{R}^3 \to \mathbb{R} \) be a 0-form and \( \mathbf{v} : \mathbb{R}^3 \to \mathbb{R}^3 \) a vector field. We can arrange \( \mathbf{v} = v_1 e_1 + v_2 e_2 + v_3 e_3 \) into a 1-form \( \mathbf{v}^b = v_1 \, dx + v_2 \, dy + v_3 \, dz \) or a 2-form \( \star \mathbf{v}^b = i_{\mathbf{v}}(dx \wedge dy \wedge dz) = v_1 \, dy \wedge dz + v_2 \, dz \wedge dx + v_3 \, dx \wedge dy \). Then
• $d$ on 0-form gives gradient.
• $d$ on 1-form gives curl.
• $d$ on 2-form gives divergence.

\[
\text{grad } f = \nabla f = (df)^lat
\]
\[
\text{curl } v = (\nabla \times v)^lat = \star d v^lat
\]
\[
\text{div } v \det = (\nabla \cdot v) \det = d \star v^lat = d \star \det.
\]

In fact, the relation between $\nabla$, $\nabla \times$, $\nabla \cdot$ and $d$’s are not limited to $\mathbb{R}^3$, but any $\mathbb{R}^n$-dimensional Riemannian manifold. One defines grad, curl and div through Hodge stars and exterior derivatives.

A diagram summarizing the vector calculus operators on a 3-dimensional manifold is:

\[
\begin{array}{ccc}
\Gamma(TM) & \xrightarrow{\sharp} & \Omega^0(M) \\
d_0 \downarrow & & \downarrow d_0 \\
(\text{grad}) & & \Omega^1(M) \\
\downarrow \star_0 & & \downarrow \star_1 \\
\Omega^0(M) & \xrightarrow{(\text{curl})} & \Omega^2(M) \\
\downarrow d_1 & & \downarrow d_2 \\
\star_1 & & \star_2 \\
\Omega^2(M) & \xrightarrow{(\text{div})} & \Omega^3(M) \\
\downarrow d_2 & & \downarrow d_0 \\
\star_2 & & \star_3 \\
\end{array}
\]

Recall that $d \circ d = 0$, which corresponds to the vector calculus identities:

\[
\text{curl} \circ \text{grad} = 0, \quad \text{div} \circ \text{curl} = 0.
\]

**Exercise 2.3 — 20 pts.** Let $f, g: \mathbb{R}^3 \to \mathbb{R}$ be a scalar field and $a, b: \mathbb{R}^3 \to \mathbb{R}^3$ be vector fields in 3D. Use Leibniz rules (for $d$) and the text before Exercise 2.2 to show

(a) $\nabla \cdot (a \times b) = (\nabla \times a) \cdot b - a \cdot (\nabla \times b)$.

(b) $\nabla \cdot (f a) = (\nabla f) \cdot a + f \nabla \cdot a$.

(c) $\nabla \times (f a) = \nabla f \times a + f \nabla \times a$.

(d) $\nabla \times (f \nabla g) = \nabla f \times \nabla g$.

**Non-Cartesian Coordinate**

Two common coordinate systems for the 2D Euclidean space $M = \mathbb{R}^2$ are the Cartesian coordinate $x, y$ and the polar coordinate $r, \theta$. Each of $x, y, r, \theta$ is viewed as a function $M \to \mathbb{R}$ that takes a position and returns the coordinate value as a real number. (Remove the origin if necessary.) Their differentials $dx, dy, dr, d\theta$ are 1-form fields defined over $M$, possibly with the origin removed.

Assume we know the inner products between these 1-forms

\[
\langle dx, dx \rangle = \langle dy, dy \rangle = 1, \quad \langle dx, dy \rangle = 0, \tag{10a}
\]
\[
\langle dr, dr \rangle = 1, \quad \langle dr, d\theta \rangle = 0, \quad \langle d\theta, d\theta \rangle = \frac{1}{r^2}. \tag{10b}
\]
Figure 1 Each discrete function \((u_i)_{i \in P}\) is uniquely extended to a piecewise linear function \(U(x)\) on the triangle mesh such that \(U = u_i\) at each vertex \(i\). This linear interpolation (b) can be written as a linear combination of the hat function (a).

We also know that the Euclidean area form is given by
\[
\det_{\mathbb{R}^2} = \star 1 = dx \wedge dy = r\, dr \wedge d\theta.
\] (11)

Exercise 2.4 — 5 pts. Using (10) and (11), compute \(\star dr\) and \(\star d\theta\). (Write them in the form of \(g(r, \theta)\, dr + h(r, \theta)\, d\theta\).)

Show that the Laplacian \(\Delta f = \star d \star df\) in the polar coordinate is given by
\[
\star d \star df = \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial f}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 f}{\partial \theta^2}.
\] (12)

Hint You don’t need to know how \(r, \theta\) are related to \(x, y\). The definition of \(\star\) is that \(\alpha \wedge \star \beta = \langle \alpha, \beta \rangle \det_{\mathbb{R}^2}\). The exterior derivative of a 0-form is \(df = \frac{\partial f}{\partial r} \, dr + \frac{\partial f}{\partial \theta} \, d\theta\); the exterior derivative of a 1-form is \(d(g\, dr + h\, d\theta) = dg \wedge dr + dh \wedge d\theta\), where \(dg, dh\) are exterior derivatives of 0-forms \(g, h\).

Piecewise Linear Finite Element

In the last homework, we compute the cotangent-weight discrete Laplacian. Here we explore a different approach—Finite Element Method (FEM).¹

Let \(M = (P, E, F)\) be a triangle mesh. In FEM, each discrete 0-form data set \(u = (u_j)_{j \in P}\) describes a piecewise linear function \(U\) by interpolation. That is, \(U\) is linear in each triangle, and \(U = u_i\) at vertex \(i\). This interpolation can be written as
\[
U(x) = \sum_{i \in P} u_i \phi_i(x)
\] (13)

where \(\phi_i(x), i \in P\), are the so called hat functions that are piecewise linear and \(\phi_i(p_j) = \delta_{ij}\) on vertex \(p_j\) (Figure 1).

¹Read more in Lecture Note Chapter 8.
Now that we have a continuous and almost-everywhere differentiable function $U$, we can measure its continuous Dirichlet energy:

$$E_{D}^{\text{discrete}}(u) := E_{D}^{\text{cont}}(U) = \frac{1}{2} \int_{M} \langle \nabla U, \nabla U \rangle \, dA$$

(14)

$$= \frac{1}{2} \sum_{i \in P} u_i \nabla i \sum_{j \in P} u_j \nabla j \int_{M} \langle \nabla i, \nabla j \rangle \, dA$$

(15)

$$= \frac{1}{2} \sum_{i \in P} u_i \nabla i \int_{M} \langle \nabla i, \nabla j \rangle \, dA$$

(16)

$$= \frac{1}{2} \sum_{c \in E} u_i \nabla j \sum_{t_{ijk} \ni c} \langle \nabla i, \nabla j \rangle A_{ijk} = \frac{1}{2} u^{T} L u. \quad (17)$$

where the line uses the fact that each $\nabla i$ is constant per triangle, and $\nabla i, \nabla j$ have nonzero overlaps only when $i, j$ share an edge. Here, $A_{ijk}$ is the area of triangle $t_{ijk}$.

Exercise 2.5 — 10 pts. As mentioned above, on a discrete triangular mesh $M$, the stiffness matrix $L$ is given by

$$L_{ij} = \int_{M} \nabla i \nabla j \wedge \star d\varphi_j = \int_{M} \langle \nabla i, \nabla j \rangle \, dA = \sum_{t_{ijk} \ni ij} \langle \nabla i, \nabla j \rangle A_{ijk}. \quad (a)$$

Show that the aspect ratio of a triangle can be expressed as the sum of the cotangents of the interior angles at its base, i.e.,

$$\frac{w}{b} = \cot \alpha + \cot \beta.$$ 

(b) Show that the gradient of the hat function on triangle $t_{ijk}$ is given by

$$\nabla i = \frac{c_{jk}^{\perp}}{2A_{ijk}}$$

where $c_{jk}^{\perp}$ is the edge vector $c_{jk}$ rotated $90^\circ$ counterclockwise within $t_{ijk}$.

(c) Show that for any hat function $\varphi_i$ associated with vertex $p_i$ of triangle $t_{ijk}$,

$$\langle \nabla i, \nabla j \rangle A_{ijk} = \frac{1}{2} (\cot \alpha + \cot \beta).$$
Show that for the hat functions $\varphi_i$ and $\varphi_j$ associated with vertices $p_i$ and $p_j$ of triangle $t_{ijk}$, we have

$$\langle \text{grad} \varphi_i, \text{grad} \varphi_j \rangle A_{ijk} = \frac{1}{2} \cot \theta$$

where $\theta$ is the angle between the opposite edge vectors.

Putting all these facts together, we have the cotan formula

$$(Lu)_i = \frac{1}{2} \sum_{e_{ij} \ni p_i} (\cot \alpha_{ij} + \cot \beta_{ij})(u_i - u_j).$$

where $\alpha_{ij}$ and $\beta_{ij}$ are the angles of opposite vertices across from $e_{ij}$ in the two adjacent triangles. This FEM stiffness matrix agrees with the discrete exterior calculus (DEC) Laplace matrix using circumcentric Hodge star (cf. HW1).

**Implementation (25 pts): Geodesic Distance using Heat Equation**

In this implementation exercise, we will construct the distance function on a given surface. This function gives us the distance to a given point (or a given subset) on the surface. The distance is the intrinsic geodesic distance within the surface (the length of the shortest path along the surface), not the extrinsic $\mathbb{R}^3$ distance.

The method we adopt is the heat method.\(^2\)

The heat method uses a few techniques we learned in the lecture, including the heat equation and the Poisson reconstruction. Heat method allows us to quickly find the distance function across the entire surface. This is much simpler than those path finding algorithms (e.g. Dijkstra algorithm).

Let $M = (P, E, F)$ be the triangle mesh. Select a subset $S \subset P$ (e.g. a point) of the point set $P$, representing the source from which we compute the distance. Our goal is to find $\phi \in C^0(M) = \mathbb{R}^{|P|}$ that approximates the shortest distance from each given point to $S$.

The observation is that the gradient $\nabla \phi$ of the distance field $\phi$ should satisfy $|\nabla \phi| = 1$ for its magnitude and should radiate away from $S$ for its direction (except at the cut locus). We will synthesize a vector field that radiates away from $S$. And then we normalize it so that it becomes a candidate for $\nabla \phi$. Finally, all we need is to find $\phi$ whose gradient best fits this candidate gradient. This final part is achieved by the Poisson reconstruction method we discussed in the lecture.

For the first step, we need to synthesize some vector field that radiates away from the epicenter $S$. To do so, we simulate the heat equation. First set temperature $u$ to an impulse ($\delta$ function) at $S$. Next, let the heat equation evolve $u$ for a short time. The resulting temperature

\(^2\)This paper is a good read. https://www.cs.cmu.edu/~kmcrane/Projects/HeatMethod/paper.pdf
distribution will have a (negative) gradient $-\mathbf{g} = -\nabla u$ radiating away from $S$, which is all we need. Please read the article to learn more about the method.

In conclusion, you will implement the following algorithm:

**Algorithm 2.1 Heat method for geodesic distance**

**Input:** $S \subset P$
1. Set $u = \delta_S$;  
2. Evolve $u$ by the heat equation $\star_0 \frac{\partial u}{\partial t} = -d_0^T \star_1 d_0 u$ for a short time;
3. Compute the gradient $\mathbf{g}$ of $u$ on each triangle;
4. Normalize and add a minus sign $X = -\mathbf{g}/|\mathbf{g}|$;
5. Convert the face-wise vector field $X$ to a 1-form $\alpha \in C^1(M)$;
6. Find $\phi \in C^0(M)$ such that $\phi|_S = 0$ and $\|d_0 \phi - \alpha\|^2$ is minimized;

**Output:** $\phi$.

**Result**
Show a rendering that shows the set $S$ (can be just a point) and the resulting distance function $\phi$ (Figure 2).

The remainder of this documents are the tips for each of the steps of the algorithm.

### 2.0.1 Initial temperature

Set $u_i = 0$ for all $i \in P \setminus S$. For $i \in S$, set

$$u_i = \frac{1}{A_i}$$

where $A_i$ is the point area of point $i$. This setup is a discrete Dirac \(\delta\)-function that describes the initial impulse.
2.0.2 Heat equation

The \textit{backward Euler method} is one of the standard algorithms to solve the time evolution

\[ \star_0 \frac{\partial u}{\partial t} = -d_0^T \star_1 d_0 u. \]  

(19)

We discretize the time domain uniformly by a time step \( h \). Let \( u(0), u(1), u(2), \ldots \) denote the solution at each time point. Discretize the time derivative in (19) and evaluate the right-hand side of (19) at the “future-end” of the time interval:

\[ \star_0 \frac{u(n+1) - u(n)}{b} = -Lu(n+1). \]  

(20)

After some rearrangement, the update rule from a previous time frame \( u(n) \) to a new time frame \( u(n+1) \) is to solve the following linear system

\[ (\star_0 + bL) u(n+1) = \star_0 u(n). \]  

(21)

For Algorithm 2.1 we only need to solve the heat equation for one small time step. That is, just solve (21) once to obtain \( u(1) \) given \( u(0) \).

\textbf{Sidenote}

If you just want to get an animation of heat diffusion, you just need to call (21) repeatedly. In Houdini, you can drop a “Solver” node. In the solver node, drop an “output” node. Build a network starting from the existing node named “Previous frame” and finalize at the “output” node. This stream of nodes will be repeatedly called when you press the play button.

2.0.3 Compute gradient \( g = \nabla u \)

In a triangle, suppose you know the value of \( u \) on each of the three vertices, what is the gradient of \( u \) if you linearly interpolate \( u \) across the triangle?

You can apply Exercise 2.5 (b) to evaluate \( g = \nabla u = u_i \nabla \varphi_i + u_j \nabla \varphi_j + u_k \nabla \varphi_k \) on triangle \( ijk \).

You can also use the Houdini node “measure” to measure the gradient of a point attribute. It will produce vector \( g \) per face.

2.0.4 \( X = -g/|g| \)

Run it per triangle.

2.0.5 Convert \( X \) to 1-form \( \alpha \)

We want the discrete 1-form \( \alpha \) on (half)edge \( ij \) to approximate the line integral

\[ \alpha_{ij} = \int_j^i X \cdot d\vec{e} = \int_j^i X^e \]  

(22)

along edge \( ij \). To do so, we use a simple averaging. For each (half)edge \( ij \) that interfaces face \( ijk \) and \( ij\ell \), set

\[ \alpha_{ij} = \left\langle \vec{e}_{ij}, \frac{X_{ijk} + X_{ij\ell}}{2} \right\rangle \]  

(23)

where \( \vec{e}_{ij} = p_j - p_i \) is the edge vector.

---

3This choice makes the Euler method “backward.” The forward Euler method is less stable. We will have some stability analysis exercise in the next homework.
2.0.6  Poisson reconstruction

The least squares solution to $d_0 \phi \approx \alpha$ is given by (recall lecture)

$$d_0^\top \star_1 d_0 \phi = d_0^\top \star_1 \alpha$$  \hspace{2cm} (24)

which should be as straightforward as HW1. One thing that is different from HW1 is that this
time there is an additional constraint $\phi|_S = 0$. Here is the technique to impose this kind of
point constraints (Dirichlet boundary condition) to a linear system. For linear system such as

$$A \phi = b$$  \hspace{2cm} (25)

categorize the indices in $P$ into $S$ and $I = P \setminus S$:

$$
\begin{bmatrix}
  A_{II} & A_{IS} \\
  A_{SI} & A_{SS}
\end{bmatrix}
\begin{bmatrix}
  \phi_I \\
  \phi_S
\end{bmatrix}
= 
\begin{bmatrix}
  b_I \\
  b_S
\end{bmatrix}
$$  \hspace{2cm} (26)

Now, if we constrain $\phi_S$ to some known value, then the problem reduces to only the first row
block:

$$A_{II} \phi_I = b_I - A_{IS} \phi_S; \quad \text{in our case } \phi_S = 0.$$  \hspace{2cm} (27)

The following is some tip about slicing matrices in numpy/scipy.

```python
# Suppose A is a scipy sparse matrix
# Suppose there is a point attribute is_S taking value 0 or 1 indicating whether the
# point is in S
# Read it into python, and write ==1 to turn it into boolean value
is_S = np.array( geo.pointIntAttribValues("is_S")) == 1
is_I = ~is_S

# you can slice sparse matrix using boolean array
A_I = A[is_I, :]
A_IS = A_I[:, is_S]
A_II = A_I[:, is_I]

# Suppose phi_I is the solution to A_II phi_I = rhs_I
phi = np.zeros( number_of_points )
phi[is_I] = phi_I # fill in the solution using boolean slicing
```

2.0.7  Visualization

The contour plot shown in Figure 2 can be achieved using a houdini digital asset we provide.
The digital asset is called “quantize.” It quantizes the values of a given point attribute, and cut
the mesh into strips of levelsets for those quantized values. On the top menu, select Assets
→ Install Asset Library, and install quantize.hda. Then you will have a new node type called
quantize. On the quantize node interface, set the “Point Attribute” entry as the name of your
point attribute that you want to quantize. Check “generate vals” with an appropriate min, max
and partitions. These vals are the list of values into which the point attribute is quantized. The
first output of the node is the cut mesh; in each strip the point attribute is replaced by a constant
value. You can put a color node and color the mesh according to the attribute value. The second
output is the contour curve of the mesh. One may use a polywire or wireframe node to turn the
curves into some wire with thickness for rendering.