Mathematical Foundations of Data Sciences

Gabriel Peyré
CNRS & DMA
École Normale Supérieure
gabriel.peyre@ens.fr
https://mathematical-tours.github.io
www.numerical-tours.com

January 7, 2018
Chapter 5

Multiresolution Mesh Processing

This chapter shows how computations on a mesh can be performed in a multiscale manner, by considering meshes of increasing resolutions. This leads to the notion of subdivision surfaces and wavelet transform, which are two different tools to interpolate and decompose functions on meshes. Both methods rely on a special kind of meshes whose triangulations can be obtained by applying a regular refinement rule.

5.1 Semi-regular Meshes

5.1.1 Nested Multiscale Grids.

In order to perform multiscale mesh processing, one needs to pack the vertices \( V \) of a topological mesh \( M = (V, E, F) \) in sets of increasing resolution. As explained in section 3.1.2, it is important to remember that this construction is purely combinatorial, in that no geometrical information (such as actual positions of the vertices in \( \mathbb{R}^3 \)) is required to build the set of multi-resolution meshes. In fact these multiscale grids can be used to actually process the geometrical realization \( M \) of the mesh \( M \) as three real valued functions (the three coordinates of the points).

We thus consider a set of nested indexes

\[
V_0 \subset V_{-1} \subset \ldots \subset V_L = V
\]

which are split according to

\[
V_j = V_{j+1} \cup H_{j+1}.
\]

Next section describes how to actually compute this set of nested grids using a triangular split, but most of the mathematical tools are in fact valid for arbitrary set of indexes, as long as they are embedded in one each other through scales.

For mesh processing, an index \( \ell \in V_j \) corresponds to a vertex \( x_\ell \in \mathcal{V} \subset \mathbb{R}^3 \). The signals to be processed are vectors \( f \in \mathbb{R}^n \) of size \( n = |V_L| \) defined on the grid \( V_L \). We sometimes write \( f \in \ell^2(V_L) \) instead of \( f \in \mathbb{R}^n \) to emphasize the domain on which \( f \) is indexed. This chapter describes transforms for signals \( f \in \ell^2(V_L) \) sampled on the finest grid \( V_L \).

5.1.2 Semi-regular Triangulation.

The combinatorial structure of a triangular mesh is defined in section 3.1.2. This chapter considers only a certain class of meshes \( M = (V, E, F) \) that can be obtained by a regular split of faces, starting from an initial coarse triangulation. This splitting leads to a set of multiresolution meshes \( M_j = (V_j, E_j, F_j) \) for \( J \leq j \leq 0 \), where the full mesh is \( M_0 = M \).

Starting from this coarse triangulation, one defines by subdivision a multiscale triangulation \( (V_j, E_j, F_j)_{J \leq j \leq 0} \) where
Figure 5.1: Edge-splitting subdivision.

The subdivided set of edges is then
\[ \forall e = (a, b) \in E_j, \quad \sigma_1(e) = (a, \gamma(e)) \quad \text{and} \quad \sigma_2(e) = (b, \gamma(e)). \]

The subdivided set of edges is then
\[ E_{j-1} = \{ \sigma_i(e) : i = 1, 2 \quad \text{and} \quad e \in E_j \}. \]

Each face \( f = (a, b, c) \in F_j \) is subdivided into four faces
\[
\begin{aligned}
\mu_1(f) &= (a, \gamma(a, b), \gamma(a, c)), \\
\mu_2(f) &= (b, \gamma(b, a), \gamma(b, c)), \\
\mu_3(f) &= (c, \gamma(c, a), \gamma(c, b)), \\
\mu_4(f) &= (\gamma(a, b), \gamma(b, c), \gamma(c, a)).
\end{aligned}
\]

The subdivided set of faces is then
\[ F_{j-1} = \{ \mu_i(f) : i = 1, 2, 3, 4 \quad \text{and} \quad f \in F_j \}. \]
Figure 5.1 shows the notations related to the subdivision process. Figure 5.2 shows an example of recursive splitting of a triangle and a coarse triangulation. Figure 5.3 shows examples of semi-regular triangulation using a geometric realization (position of the vertices) to create a 3D surface.

The set of vertices can be classified as

- **Regular vertices** are those who belong neither to the coarse mesh $V_0$ nor to a boundary of a mesh $M_j$. These vertices have always 6 neighbors.

- **Extraordinary vertices** are the initial vertices of $V_0$. They exhibit arbitrary connectivity.

- **Boundary vertices** are those belonging to a mesh boundary. Boundary vertices not in $V_0$ always have 4 immediate neighbors.

Obviously not every meshes can be obtained from such a subdivision process. In practice, an arbitrary mesh, obtained from CAD design or range scanning usually does not have any multiscale structure. It is thus necessary to remesh it in order to modify the connectivity of the mesh. During this process, the position of the vertices in $\mathbb{R}^3$ is modified in order for the geometrical realization to stay close from the original piecewise linear surface. One can see [2] fur a survey of various semi-regular remeshing methods.

### 5.1.3 Spherical Geometry Images

Starting from some input surfaces $S \subset \mathbb{R}^3$, one typically wants to compute a semi-regular meshes $(M_j)_{j \geq L}$ that approximate $S$. In most case, the surface $S$ is actually given as an arbitrary triangulated mesh and this process corresponds to a semi-regular remeshing. Many algorithm have been devised for surface remeshing and we describe here a method [33] that works for surfaces that have the topology of a sphere. It means that the surface has genus 0, without boundary and without handles.

This methods works by computing several intermediate surface-wise parameterization.

- **Spherical parameterization:** each points of the original triangulation of $S$ is mapped onto the unit sphere.
This create a bijective parameterization
\[ \varphi_S : S^2 \rightarrow S. \]
This is a non-linear process that differ from the planar parameterization introduced in section 3.3.7. We do not give the details of such a process, but it requires minimizing the smoothness of the mapping \( \varphi^{-1}_S \) under the constraint that it maps points of \( S \) to unit length vectors (point on the sphere \( S^2 \)). The algorithm is explained in details in [33].
- **Spherical-tetraedron flattening:** one flatten each quadrant (1/8) of the sphere in order to have a mapping
\[ \varphi_T : \text{Octaedron} \rightarrow S^2. \]
One can use for instance a mapping between spherical barycentric coordinate on each quadrant and Euclidean barycentric coordinates on each face of the octahedron.
- **Tetraedron unfolding:** One maps each equilateral face of the octahedron on a rectangular triangle that corresponds to 1/8th of the square \([0, 1]^2\).
\[ \varphi_U : [0, 1]^2 \rightarrow \text{Octaedron}. \]
- **Regular sampling:** the geometry image is obtained by regularly sampling the square on a uniform grid
\[ x_\ell = \varphi_S \circ \varphi_T \circ \varphi_U (\ell/n), \quad \text{for} \quad \ell_i = 0, \ldots, n - 1. \]
The mapping \( \ell \mapsto x_\ell \in \mathbb{R}^3 \) is the geometry image, which can be stored as a 3-channel (color) image.
From such a geometry image \( x_\ell \), one can easily compute a semi-regular mesh by simply performing a regular 1:4 subdivision of the octahedron. Figure 5.4 shows the steps of the construction of a geometry image, and the resulting semi-regular mesh.

![Figure 5.4: Spherical geometry image construction, taken from [33].](image)

### 5.2 Subdivision Curves

Before getting into the detail of subdivision surfaces, we describe the subdivision process in the simpler setting of 1D signals. This leads to the construction of subdivision of 1D functions and subdivision curves.

In this 1D setting, the grid point indexes are dyadic sub-grids of \( \mathbb{Z} \)
\[ \forall j \geq L, \quad V_j = \{ \ell 2^{j-L} : 0 \leq \ell < s_0 2^{-j} \}, \]
where \( s_0 = |V_0| \) is the size of the initial vector \( f_0 \) to be subdivided.

Each subdivision steps computes, from a set \( f_j(\ell) \in \ell^2(V_j) \) of coarse values, a refined vector \( f_{j-1} \in \ell^2(V_{j-1}) \) defined by
\[
\begin{cases}
\forall k \in H_j, & f_{j-1}(k) = \sum_i f_j((k-1)/2 + i)h(t), \\
\forall \ell \in V_j, & f_{j-1}(\ell) = \sum_i f_j(\ell + t)\tilde{h}(t).
\end{cases}
\]
where the set of weights \( h \) and \( \tilde{h} \) acts as local averaging operators. This averaging should be corrected at the boundary, and we use here cyclic boundary conditions which identifies 0 and \( s_0 2^{-j} \) in \( V_j \). Figure 5.5 shows a graphical display of these averaging operators.

One can write this subdivision steps as convolution by introducing the global set of weights
\[
g = [\ldots, \tilde{h}(-1), h(0), \tilde{h}(0), h(1), \tilde{h}(1), \ldots]
\]
since one has
\[
f_{j-1} = (f_j \uparrow 2) \ast g \quad \text{where} \quad a \uparrow 2 = [\ldots, a(-1), 0, a(0), 0, a(1), 0, \ldots].
\]
This corresponds to the traditional description of the wavelet low-pass filtering [27].
An interesting property is that this curve is included in the convex hull of the control polygon

\[(X(t), Y(t))_t \subset \text{Conv}(X_0, Y_0).\]

Figure 5.7 shows examples of subdivision curves.

Figure 5.7: Two examples of subdivision curves. The red curve represent the original curve \((X^0, Y^0)\).

5.3 Subdivision Surfaces

Subdivision schemes allows to compute a set of progressively refined vectors on a semi-regular mesh. More precisely, from an initial vector \(f_0 \in \mathbb{R}^{|V_0|}\) defined on the coarse mesh \(M_0\), local interpolation kernels computes iteratively vectors \(f_j \in \mathbb{R}^{|V_j|}\) of finer resolution. When applied to 3 function \((f_i^0)_{i=1,2,3}\) defining the geometrical position of points in \(\mathbb{R}^3\), this hierarchical construction defines a subdivision surface. These subdivisions surfaces are used extensively in computer aided geometry and computer graphics. One can see [15] for a survey of subdivision surfaces and their applications.

5.3.1 Interpolation Operators

In order to refine a vector \(f_j \in \mathbb{R}^{|V_j|}\) defined on the vertex \(V_j\) of the mesh \(M_j\), one uses two interpolators

\[P_j : \ell^2(V_j) \rightarrow \ell^2(H_j) \quad \text{and} \quad \tilde{P}_j : \ell^2(V_j) \rightarrow \ell^2(V_j).\]  

A new refined function \(f_{j-1} \in \mathbb{R}^{|V_{j-1}|}\) defined on the vertices \(V_{j-1} = V_j \cup H_j\) of \(M_{j-1}\) is defined by applying these two refinement operators:

\[
\forall \ell \in V_{j-1}, \quad f_{j-1}(\ell) = \begin{cases} 
(P_j f_j)(\ell) & \text{if } \ell \in V_j, \\
(\tilde{P}_j f_j)(\ell) & \text{if } \ell \in H_j.
\end{cases}
\]

Since \(V_j \subset V_{j-1}\), the operator \(\tilde{P}_j\) only modify slightly the value at vertex in \(V_j\). On the other hand, the operator \(P_j\) creates new value at the vertices of \(H_j\) that are inserted between \(V_j\) and \(V_{j-1}\).
In practical applications, these interpolating operators are local, meaning that the value of \( (P_j f_j)(\ell) \) and \( (\tilde{P}_j f_j)(\ell) \) depends only on values \( f_j(\ell') \) for \( \ell' \in V_j \) being close to \( \ell \in V_{j-1} \), typically in the 1-ring or 2-ring vertex neighborhood.

A particularly important setting for subdivision scheme is when one apply the subdivision steps in parallel to three vectors \((X_j, Y_j, Z_j)\) starting from three initial vectors describing the position in 3D space of a coarse mesh \( M_0 \). This allows to defines finer and finer spacial localization for the vertex of the refined meshes \( M_j \). Figure 5.9 shows an example of such a subdivision surface. In order for the resulting infinitely refined surface to have good properties such as being continuous and even smooth, one needs to design carefully the interpolation operators. Next section gives examples of such operators.

### 5.3.2 Some Classical Subdivision Stencils

In order to define the interpolation operators \( P_j \) and \( \tilde{P}_j \) of equation (5.1), one needs to use a naming convention for the neighborhoods of vertices.

For a vertex \( \ell \in V_j \), the one ring neighborhood \( V_{\ell} \) has already been defined in equation (3.1). It is the set of vertices adjacent to \( \ell \). In a regular point (that does not belongs to \( V_0 \) and not on a boundary of the mesh), its size is \(|V_{\ell}| = 6\) since a point has 6 neighbors. This 1-ring is used to define \( \tilde{P}_j \).

For a vertex \( k \in H_j \subset V_{j-1} \), the butterfly neighborhood is a set of vertices in \( V_j \) close to \( k \). This neighborhood is used to define \( P_j \). The two immediate neighbors are

\[
\{(v_1^k, v_2^k) = \{v \in V_j : (v, k) \in E_{j-1}\} \}
\]

Two other vertices \((w_1^k, w_2^k)\) are defined using the two faces adjacent to edge \((v_1^k, v_2^k) \in E_j\)

\[
f_1^k = (v_1^k, v_2^k, w_1^k) \in F_j \quad \text{and} \quad f_2^k = (v_1^k, v_2^k, w_2^k) \in F_j.
\]

For edges \( E_{j} \) on the boundary of \( M_{j} \), one one face is available, in which case we implicitly assume that \( f_1 = f_2 \) (reflecting boundary conditions). The four last vertices are defined using faces adjacent to \( f_1 \) and \( f_2 \):

\[
\forall i, j = 1, 2, \quad f_{1}^{i,j} = (z^{i,j}_k, v_1^k, w_1^k) \in F_j \quad \text{with} \quad f_{1}^{i,j} \neq f_j.
\]

Once again, reflecting boundary condition are applied for faces on the boundary of the mesh. The butterfly neighborhood is depicted on figure 5.8.

**Linear Interpolating Scheme** The simplest subdivision rule compute values along edge mid point using a simple linear interpolation as follow

\[
\begin{cases}
\forall k \in H_j, & (P_j f_j)(k) = \frac{1}{2}(f(v_1^k) + f(v_2^k)), \\
\forall \ell \in V_j, & (\tilde{P}_j f_j)(\ell) = f_j(\ell).
\end{cases}
\tag{5.2}
\]

Since \( \tilde{P}_j \) is the identity operator, this scheme is called interpolating. It means that value of \( f_0 \) on points of the coarse triangulation are kept during iteration of the subdivision.

**Butterfly Interpolating Scheme** The linear scheme creates function that are piecewise linear on each face of the coarse triangulation \( F_0 \). In order to create smooth surface, one needs to use more points in the butterfly neighborhood as follow

\[
\begin{cases}
\forall k \in H_j, & (P_j f_j)(k) = \frac{1}{2} \sum_{i=1}^2 f(v_i^k) + \frac{1}{8} \sum_{i=1}^2 f(w_i^k) - \frac{1}{16} \sum_{i,j=1}^2 f(z_{k}^{i,j}), \\
\forall \ell \in V_j, & (\tilde{P}_j f_j)(\ell) = f_j(\ell).
\end{cases}
\tag{5.3}
\]
**Loop Approximating Scheme** In order to gain flexibility in the subdivision design, one can also modify points in $V_j$ during the iterations. This means that $P_j$ is not any more the identity, and that all the values will evolve during the iterations. The question of whether these iterated modifications actually converge to a limit value is studied in the next section.

The Loop subdivision rule is defined as

$$
\forall k \in H_j, \quad (P_j f_j)(k) = \frac{3}{8} \sum_{i=1}^{2} f(v^i_k) + \frac{1}{8} \sum_{i=1}^{2} f(w^i_k),
$$
$$
\forall \ell \in V_j, \quad (\tilde{P}_j f_j)(\ell) = (1 - |V_{\ell}| \beta_{|V_{\ell}|}) f_j(\ell) + \beta_{|V_{\ell}|} \sum_{\ell' \in V_{\ell}} f_j(\ell').
$$

where the weights depend on the number of neighbors and are defined as

$$
\beta_m \overset{\text{def.}}{=} \frac{1}{m} \left( \frac{5}{8} - \left( \frac{3}{8} + \frac{1}{4 \cos(2\pi/m)} \right)^2 \right).
$$

**Other schemes.** It is possible to define subdivision schemes using rules that do not involve a regular 1:4 splitting of each coarse face. For instance, in dual schemes such as the one depicted in figure 5.11, the faces of $F_j$ are not included in $F_{j-1}$ but only in $F_{j-2}$.

### 5.3.3 Invariant Neighborhoods

In order to study the convergence of subdivision schemes, one needs to consider independently each vertex $x \in V_{j_0(x)}$, where $j_0(x)$ is the coarser scale at which $x$ appears

$$
j_0(x) = \max \{ j ; x \in V_j \}.
$$

Original vertices satisfy $j_0(x) = 0$ and are the only one (except boundary vertices) that have a non-regular connectivity.

The vertex $x$ belongs to the mesh $M_{j_0(x)}$ which is going to be refined through scales $j < j_0(x)$. In order to analyze this refinement, one needs to define an invariant neighborhood $V^x_j \subset V_j$ of $x$ for each scale $j \leq j_0(x)$. These neighborhoods are the set of points that are required to compute the operators $P_j$ and $\tilde{P}_j$. 

---

Figure 5.8: The butterfly neighborhood of a vertex $k \in H_j$. 

---
Figure 5.9: Examples of iterative subdivision using Loop scheme. The points \((X_0, Y_0, Z_0)\) of the initial coarse mesh \(M_0\) are shown in red.

More precisely, given a vector \(f \in \ell^2(V_{j-1})\), the neighborhoods are required to satisfy

\[
\begin{align*}
\forall \ell \in V_{j-1}^x \cap V_j, & \quad (\hat{P}_j f)(\ell) \text{ depends only on } V_j^x \\
\forall k \in V_{j-1}^x \cap H_j, & \quad (P_j f)(k) \text{ depends only on } \hat{V}_j^x.
\end{align*}
\]

We further impose that all the invariant neighborhoods have the same size

\[
\forall j \leq j_0(x), \quad \#V_j^x = m_x.
\]

Figure 5.12 shows an example of invariant neighborhood which corresponds to the 2-ring \(V^{(2)}_x\), as defined in (3.2).

Thanks to the invariance of these neighborhood systems, one can restrict the predictors around \(x\) and define

\[
P_j^x : V_j^x \rightarrow V_{j-1}^x \cap V_j \quad \text{and} \quad \hat{P}_j^x : V_j^x \rightarrow V_{j-1}^x \cap H_j.
\]

The subdivision matrix \(S_j^x \in \mathbb{R}^{m_x \times m_x}\) is then defined as matrix of the following mapping

\[
(\hat{P}_j^x, P_j^x) : V_j^x \rightarrow V_{j-1}^x.
\]

All the subdivision schemes studied in this chapter are invariant, meaning that the subdivision rule does not change through the scales \(j\). This impose that the subdivision matrices are constant \(S_j^x = S^x\). In fact, in all the examples given in the previous section, they only depends on the number \(|V_x|\) of neighbors in the one ring of \(x\).

### 5.3.4 Convergence of Subdivisions

The value at \(x \in V_{j_0(x)}\) of a function \(f_j \in \ell^2(V_j)\) obtained by subdividing at scale \(j \leq j_0(x)\) an initial vector \(f_0 \in \ell^2(V_0)\) can be computed as

\[
f_j(x) = (S^x f_{j+1}^x)(x) = \left((S^x)^{j_0(x)-j} f_{j_0(x)}^x\right)(x),
\]

where the vector \(f_i^x \in \mathbb{R}^{m_x}\) is the restriction of \(f_i\) to the set \(V_j^x\).
In order to analyze the limiting function resulting from an infinite number of subdivision, one can use the eigen vector decomposition of the matrix $S^x$

$$S^x = \Phi V \Lambda \Phi^T$$

where

$$\Phi^T = \Phi^{-1}, \quad \Lambda = \text{diag}(\lambda_i), \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{m_x}.$$ 

Since the subdivision matrix $S_x$ is not symmetric, some of the eigenvalues might be complex, and we shall ignore this difficulty here. The fact that $P_j$ and $\tilde{P}_j$ are predictor implies that the subdivision matrix has to satisfy $S^x 1 = 1$, meaning that $\tilde{\varphi}_1 = 1$ is an eigenvector associated to the eigenvalue 1. In the following we further makes the following assumption

$$1 = \lambda_1 < \lambda_2 = \lambda_3 < \lambda_4. \quad (5.4)$$

This hypothesis is satisfied by all the subdivision rules introduced in the previous section.

If one write $\Phi = (\varphi_i)_{i=1}^m$ and $\Phi = (\varphi_i)_{i=1}^m$, one has the following decomposition of a vector $f \in \mathbb{R}^{m_x}$

$$f = \sum_{i=1}^{m_x} (f, \varphi_i) \tilde{\varphi}_i \quad \text{and} \quad (S^x)^k(x) = \sum_{i=1}^{m_x} \lambda_i^k (f, \varphi_i) \tilde{\varphi}_i.$$ 

One thus has the following asymptotic expansion

$$\frac{1}{\lambda_k} \left( f - (f, \varphi_1) 1 \right) = (f, \varphi_2) \tilde{\varphi}_2 + (f, \varphi_3) \tilde{\varphi}_3 + o(1). \quad (5.5)$$

This expression describes the asymptotic behavior of the subdivision scheme at zero order (position) and first order (tangents).

**Theorem 25** (Convergence of the subdivision scheme). *If the subdivision matrix $S^x$ of a point $x$ satisfies (5.4) then the subdivision process converges at $x$ to the value*

$$f^1(x) \xrightarrow{j \to \infty} (f_{j=0}(x), \varphi_1).$$

The smoothness of the resulting function is more difficult to analyze. A particularly important setting is when one computes the subdivision of 3 function $p_0 = (X_0, Y_0, Z_0) \in \ell^2(V_0)^3$ corresponding to the position
in $\mathbb{R}^3$ (geometrical realization) of a coarse mesh $M_0$. In this case, the subdivided functions $p_j = (X_j, Y_j, Z_j)$ gives refined 3D meshes that converge uniformly to a continuous surfaces

$$p(x) = (X(x), Y(x), Z(x)) = (\langle X_j^x, \varphi_1 \rangle, \langle Y_j^x, \varphi_1 \rangle, \langle Z_j^x, \varphi_1 \rangle).$$

Condition (5.4) nearly implies that the resulting surface is smooth. Indeed, the asymptotic expansion (5.5) shows that for a point $x'$ near $x$ in the subdivision domain, the differential vector can be well approximated as a projection on a 2D plane

$$p(x) - p(x') + o(1) \in \text{Span}(\tau^x_2, \tau^x_3) \quad \text{where} \quad \tau^i(x) \overset{\text{def}}{=} (\langle X_j^x, \varphi_i \rangle, \langle Y_j^x, \varphi_i \rangle, \langle Z_j^x, \varphi_i \rangle).$$

If the vectors $\tau^x_2$ and $\tau^x_3$ are linearly independent, they form a basis of the tangent plane at $p(x)$.

**Example of the Loop subdivision.** For the Loop interpolation operators defined in equation (5.3.2), the invariant neighborhood $V^x_j$ correspond to the 2-ring of $x$ in the triangulation $G_j$, as shown in figure 5.12. For a vertex with $k$ neighbors, $|V^x_j| = k$, the size of these invariant neighborhood is $m_x = 3k + 1$. A particular neighboring for $k = 3$ is depicted in figure 5.12, together with an indexing in $\{0, \ldots, 3k = 9\}$ of the points in $V^x_j$ and $V^{x-1}_j$. For this indexing, the subdivision matrix reads

$$
\begin{pmatrix}
7 & 3 & 3 & 3 \\
1 & 1 & 1 & 10 & 1 & 1 \\
1 & 1 & 1 & 1 & 10 & 1 \\
1 & 1 & 1 & 1 & 1 & 10 \\
1 & 1 & 3 & 3 \\
1 & 1 & 3 & 3 \\
1 & 1 & 3 & 3 \\
1 & 1 & 3 & 3 \\
1 & 3 & 1 & 1 \\
1 & 1 & 3 & 1 \\
1 & 2 & 1 & 3
\end{pmatrix}
$$

where the 0’s have been omitted and where the rows should be rescaled to sum to 1. The eigenvalues of this matrix satisfy $\lambda_1 = 1$ and $\lambda_2 = \lambda_3 = 1/3 > \lambda_4$. 

89
Figure 5.12: Invariant neighborhood $V^x_j$ and $V^x_{j-1}$ (indexing with red circles) of the Loop subdivision scheme for a vertex of valence $|V^x_i| = 0$. The number in $\{0, \ldots, 9\}$ refers to the numbering of the vertices in $V^x_j$ and $V^x_{j-1}$.

5.4 Wavelets on Meshes

5.4.1 Multiscale Biorthogonal Bases on Meshes

The transforms considered in this section are multiscale and indexed by the set of nested grids $(V^x_j)_{L<j \leq J}$. This corresponds to computing a set of coefficients $(d_j)_{L<j \leq J} \cup f_J$ from an initial input signal $f$. These coefficients correspond to inner products with basis vectors $\{d_j \in \ell^2(H^x_j)\}$ where $\forall k \in H^x_j$, $d_j(k) = \langle f, \psi^x_j,k \rangle$, $f_J \in \ell^2(V^x_J)$ where $\forall \ell \in V^x_J$, $f_J(\ell) = \langle f, \phi^x_J,\ell \rangle$.

By analogy with the wavelet setting, the vectors $\psi^x_j,k \in \mathbb{R}^n$ corresponds to primal wavelets and are intended to capture the details present in the signal $f$ at a scale $j$, whereas the scaling vectors $\phi^x_J,\ell \in \mathbb{R}^n$ capture the missing coarse approximation of $f$ at scale $J$. This decomposition is stopped at any coarse scale $L < J \leq 0$.

In order to reconstruct the function $f$ from this set of transformed coefficients, one needs to use a set of bi-orthogonal basis vectors

$$f = \sum_{L<j \leq J, k \in H^x_j} d_j(k)\tilde{\psi}^x_j,k + \sum_{\ell \in V^x_J} f_J(\ell)\tilde{\phi}^x_J,\ell.$$ 

If this reconstruction formula holds for any scale $L < J \leq 0$, the set of vectors $(\psi^x_j,k,\phi^x_J,\ell)_{k \in H^x_j, \ell \in V^x_J}$ and $(\tilde{\psi}^x_j,k,\tilde{\phi}^x_J,\ell)_{k \in H^x_j, \ell \in V^x_J}$ is said to be a pair of primal and dual multiscale bases (together with their scaling functions).

The following paragraph shows how one can modify such a pair of multiscale bases while still maintaining the biorthogonality property. This lifting process is useful to design multiscale bases with various properties on complicated domains.

5.4.2 The Lifting Scheme

The lifting scheme is a construction of multiscale biorthogonal bases introduced by Sweldens [41, 42]. It extends the traditional construction of wavelets in two main directions:
- As explained in [17], it allows to implement already existing filter banks more efficiency by splitting the computation into elementary blocks. This computational gain is described at the end of the section together with the factorization of wavelets into lifting steps.
- It allows to define multiscale transforms over domains that are not translation invariant. This section gives two examples of such transforms: a non-separable 2D wavelet transform and wavelets on triangulated meshes.

In order to build wavelets on triangulation, one can specialize the lifting scheme to a particular setting where only two lifting steps are applied.

**Forward lifting scheme.** The forward algorithm performs the transform

\[(f_{j-1}(\ell))_{\ell \in V_{j-1}} \rightarrow (d_j(k))_{k \in H_j} \cup (f_j(\ell))_{\ell \in V_j}\]

by applying the following steps
- **Splitting:** this corresponds selecting the coefficient of \(f_{j-1}(\ell)\) that are in \(V_j\) or in \(H_j\)

\[(f_{j-1}(\ell))_{\ell \in V_{j-1}} = (f_j(\ell))_{\ell \in V_j} \cup (f_j(\ell))_{\ell \in H_j}.\]

These two sets of coefficients are treated differently in the two remaining steps of the transform.
- **Predict step:** creates wavelets coefficients \(d_j\) by computing local differences between each coefficient in \(V_j\) and its neighbors in \(H_j\)

\[\forall k \in H_j, \quad d_j(k) = f_{j-1}(k) - \sum_{\ell \in V_j} p_j(k, \ell) f_{j-1}(\ell).\]

The coefficients \(p_j(k, \ell)\) are weights that determine the predict operator

\[P_j : \begin{cases} \ell^2(V_j) & \rightarrow \ell^2(H_j) \\ g & \mapsto h = P_j g \end{cases}, \quad \text{where} \quad h(k) = \sum_{k \in H_j} p_j(k, \ell) g(\ell).\]

- **Update step:** enhances the properties of each remaining low pass coefficients \(f_{j-1}(\ell)\) for \(\ell \in V_j\) by pooling locally the wavelets coefficients \(d_j(k)\) for \(k\) around \(\ell\)

\[\forall \ell \in V_j, \quad f_j(\ell) = f_{j-1}(\ell) + \sum_{k \in H_j} u_j(\ell, k) d_j(k).\]

The coefficients \(u_j(\ell, k)\) are weights that determine the update operator

\[U_j : \begin{cases} \ell^2(H_j) & \rightarrow \ell^2(V_j) \\ h & \mapsto g = U_j h \end{cases}, \quad \text{where} \quad g(\ell) = \sum_{k \in H_j} u_j(\ell, k) h(k).\]

Figure 5.13, top row, shows the block diagram associated to this forward lifting wavelet transform.

The iterations of the forward lifting transform can also be written in vector and operator format

\[
\begin{cases}
  d_j = f^{H_j}_{V_j} - P_j f^{V_j}_{V_{j-1}}, \\
  f_j = f_{j-1}^{V_j} + U_j d_j = (\text{Id}_{V_j} - U_j P_j) f_{j-1}^{V_j} + U_j f_{j-1}^{H_j},
\end{cases}
\]

where \(g^A\) is the restriction of some vector \(g\) to the set \(A\).

**Backward lifting scheme.** The backward transform algorithm does the reverse computation

\[(d_j(k))_{k \in H_j} \cup (f_j(\ell))_{\ell \in V_j} \rightarrow (f_{j-1}(\ell))_{\ell \in V_{j-1}}\]

One of the main feature of the lifting scheme is that this is achieved by simply reversing the order of the lifting steps and interchanging +/- signs.
The lifting scheme is more general than the algorithm described in this section since several passes of predict/update steps can be applied to further enhance the properties of the resulting transform. However, the steps beyond the two initial ones are difficult to analyze, except in the notable exception of points sampled evenly on a 1D axes, where a factorization algorithm \[17\] allows to recover traditional wavelet filters.

5.4.3 Imposing vanishing moments.

The operator $P_j$ is called a predictor since the values of $P_j f_{j-1}^{V_j}$ should typically be close to $f_{j-1}^{H_{j-1}}$ for the wavelet coefficients $d_j$ to be small. Such predictors have already been constructed in equations (5.2), (5.3) and (5.3.2).

The operator $U_j$ is called an update operator since the additional term $U_j d_j$ should enhance the properties of $f_{j-1}^{V_j}$. This update step does not appear in the theory of subdivision surface and this section considers a local update operator which guarantees the conservation of the mean value when switching from $f_{j-1}$ to $f_j$.

**Polynomial vectors.** In order to select predict and update operator that have good properties, one follows the insight gained from the analysis of the wavelet approximation of signal on the real line. In order to do
so, one need analyze the effect of a lifting wavelet transform on polynomials. The most basic constraint enforces one vanishing moment by imposing orthogonality with the constant vector $\Phi_0 = 1$. This constraint does not require to known the spacial location $x_\ell$ of each index $\ell \in V_L$. In order to impose higher order vanishing moments, one needs to assume some sampling pattern, for instance

$$\forall \ell \in V_L, \; f(\ell) = \bar{f}(x_\ell) \quad \text{where} \quad x_\ell \in \mathbb{R}^q$$

and where $\bar{f}$ is a function defined on $\mathbb{R}^q$. For instance, the points $x_\ell$ might corresponds to a regular sampling of the line (this is the traditional wavelet setting) or to an irregular sampling of a 2D surface embedded in $\mathbb{R}^3$. The next paragraphs describe several situations with different sampling grids. Once the precise locations of the samples are known, one can for instance select $\Phi_s$ as some monomials of degree $(s_1, \ldots, s_q)$ over $\mathbb{R}^q$.

### Vanishing moment and polynomials reproduction.

Having defined these polynomial vectors, one requires that the following constraints are fulfilled.

- **Vanishing moments**: the wavelet coefficients of a low order polynomial should be 0, which implies that

$$\forall k \in H_j, \; \langle \Phi_s, \psi_{j,\ell} \rangle = 0. \quad (5.7)$$

- **Polynomial reproduction**: coarse coefficients $f_j$ computed from a polynomial $f_{j-1}$ should also be polynomials, which implies that

$$\forall \ell \in V_j, \; \langle \Phi_s, \varphi_{j,\ell} \rangle = \Phi_s(\ell) \quad (5.8)$$

In order for the wavelets and scaling function to satisfy conditions (5.7) and (5.8), the predict operator $P_j$ and update operator $U_j$ should be designed carefully. One can impose these constraint from the fine scale $j = L$ until the coarse scale $j = 0$. Indeed, if $(\varphi_{j-1,\ell}, \psi_{j-1,\ell})_{k,\ell}$ satisfy conditions (5.7) and (5.8), then, for the scale $j$

$$\forall s \in S, \quad \begin{cases} (5.7) \iff P_j \Phi_s^V_j = \Phi_s^H_j, \\ (5.8) \iff U_j^T (\Phi_s^V_j + P_j^T \Phi_s^H_j) = \Phi_s^H_j. \end{cases}$$

where $\Phi_s^A \in \ell^2(A)$ is the restriction of $\Phi_s$ to $A$.

In contrast, the constraint (5.8) on the update operator $P_j$ is more involved and the next section shows how to handle it on a triangulation situations for only one vanishing moment $|S| = 1$.

### 5.4.4 Lifted Wavelets on Meshes

The lifted wavelet bases can be used to process signals $f \in \ell^2(V_L)$ where $\ell \in V_L$ index a sampling $x_\ell$ of an arbitrary surface. The construction of biorthogonal wavelets on triangulated mesh has been first proposed by Lounsbery et al. [26] and re-casted into the lifting scheme framework by Schroeder and Sweldens [36, 37].

#### Designing predict operators.

The constraints (5.7) on the predictor $P_j$ is easily solved. For instance, for each $k$, one selects only $|S|$ non vanishing weights $(p_j(k, \ell))_\ell$ and solves a small $|S| \times |S|$ linear system. Furthermore, in the case of a regular triangulation with edges of constant length, predictors with several vanishing moments have been already defined in (5.2), (5.3) and (5.3.2). Figure 5.14 shows the weights for these predictors.

One can choose any of these operators, and creates respectively linear, butterfly and Loop wavelets bases. All these predictors have one vanishing moment since they satisfy $P_j 1^{H_j} = 1^V$. In fact they have more vanishing moments if one consider polynomials $\Phi_s$ sampled at points $x_\ell \in \mathbb{R}^2$ of an hexagonal tiling with constant edge length. In practice, if the triangulation under consideration have edges with smoothly varying length, the resulting predictor are efficient to predict the value of smooth functions on the triangulation.
Designing update operators. In order to ensure the reproduction of constant polynomials, we design the update operator so that it depends only on the direct neighbors in $H_j$ of each point in $V_j$

$$
\forall \ell \in V_j, \quad V_\ell = \{\gamma(\ell, \ell') : (\ell, \ell') \in E_j\}.
$$

One then looks for a valid update operator in the following form

$$
\forall h \in L^2(H_j), \forall \ell \in V_j, \quad (U_j h)(\ell) = \lambda_\ell \sum_{k \in V_\ell} h(k), \quad (5.9)
$$

where each $\lambda_\ell$ should be fixed in order for condition $(5.8)$ to be satisfied.

In an semi-regular triangulation, $|V_\ell| = 6$ excepted maybe for some points in the coarse grid $\ell \in V_0$. In this setting, the values of $\lambda_\ell$ can be computed by a recursion through the scales. In an ideal triangulation where $|V_\ell| = 6$ for all $\ell$, one can use a constant weight $\lambda_\ell = \lambda$.

For the predictors defined in (5.2), (5.3) and (5.3.2), one has

$$
P_j^T 1^{H_j} = 3 \times 1^{V_j} \quad \text{and} \quad U_j^T 1^{V_j} = 6 \lambda 1^{H_j}
$$

so setting $\lambda_\ell = 1/24$ solves equation (5.8). Figure 5.15 shows examples of butterfly wavelets on a planar semi-regular triangulation.

![Example of wavelets $\psi_{j,k}$ on a semi-regular triangulation. The height over the triangle (together with the color) indicates the value of the wavelet vector.](image)

5.4.5 Non-linear Mesh Compression

These wavelets can be used to perform an approximation of a function $f \in L^2(V_L)$ defined on the fine triangulation. For instance a wavelet approximation can be applied to each coordinate $f_i, i = 1, 2, 3$ of the
actual position $x_\ell = (f_1(\ell), f_2(\ell), f_3(\ell)) \in \mathbb{R}^3$ of the surface points, as done in [23, 24]. This leads to a scheme to approximate and compress a 3D surface using the lifted biorthogonal wavelets associated to the semi-regular triangulation. This is possible because these wavelets depend only on the combinatorial grids $V_j$ and not on the precise position of the samples $x_\ell$ in 3D.

In order to perform a wavelet approximation in this biorthogonal basis, one uses a non-linear thresholding at $T > 0$

$$f = \sum_{(j,k) \in I_T} \langle f, \psi_{j,k} \rangle \tilde{\psi}_{j,k},$$

where $I_T = \left\{ (j,k) : k \in H_j \text{ and } |\langle f, \psi_{j,k} \rangle| > T|\text{supp}(\psi_{j,k})|^{-1/2} \right\}$.

Note that for each coefficient the threshold $T$ is scaled according to the size of the support of the wavelet in order to approximately normalize the wavelets in $\ell^2(V_L)$ norm.

Figure 5.16 shows an example of compression of the position of a vertex in 3D spaces as 3 functions defined on a semi-regular mesh. Figure 5.17 shows an example of compression of a spherical texture map which is a single function defined at each vertex of a semi-regular mesh obtained by subdividing an icosahedron.

Figure 5.16: Non-linear wavelet mesh compression with a decreasing number of coefficients.
Figure 5.17: Non-linear spherical wavelet compression with a decreasing number of coefficients.
Bibliography


