Discrete Hodge Theory on Graphs: a Tutorial

James L. Johnson, Western Washington University
Tom Goldring, National Security Agency

June 5, 2012

Abstract

Hodge theory provides a unifying view of the various line, surface, and volume integrals that appear in physics and engineering applications. Hodge theory on graphs develops discrete versions of the differential forms found in the continuous theory and enables a graph decomposition into gradient, solenoidal, and harmonic components. Interpreted via similarity to gradient, curl, and Laplacian operators on vector fields, these components are useful in solving certain ranking and approximations problems that arise naturally in a graph context. This tutorial develops the rudiments of discrete Hodge theory and provides several example applications.

1 A motivating example

In academic life there arise from time to time certain amenities, such as sabbaticals, research grants, and teaching load reductions. Because candidates inevitably outnumber prizes, a lollipop committee is appointed to rank the applicants. In its first meeting, the committee finds a large stack of dossiers, each proclaiming a candidate's merits, each written in a different style, each emphasizing different accomplishments. Committee members, alone or in subgroups, examine the evidence and construct rankings. The process consumes several weeks and, of course, produces conflicting judgments. Reconciliation meetings then sort out the conflicts in an ad hoc fashion. The process is reasonably satisfactory and is certainly sanctified by tradition. Given variations in human judgment, conflicts are expected, and the faculty committee serves to extract a group consensus from the individual voices.

Improvised human judgment, informed by a professional appreciation of the context, suffices to rank short lists, say of length two or three. We typically assemble longer rankings by repeatedly positioning a new entry in an already ranked sublist, a process that frequently involves reassessments of neighboring candidates. An algorithm to assemble a global ranking from a collection of independent short subrankings is clearly needed.

Specifically, we propose that the committee generate a random but representative collection of three-candidate triads. On receiving his triad allotment, a committee member treats each triad as an independent problem. That is, considering a particular triad, a judge need not consider nor even remember judgments that he or any other member has pronounced on other triads. He simply rates the triad candidates as top, middle, and bottom. Triad overlaps are expected, and therefore judgment conflicts persist. However, an algorithm can reconcile these conflicts into a global ranking that best approximates the triad judgments.

The reconciliation algorithm is, of course, our major topic here, and the following sections clarify the sense in which the computed global ranking best approximates the triad rankings. Before considering these details, however, we construct some artificial data that illustrate the judgment conflicts of a typical committee. We envision a 20-candidate field. Candidate 1 is the strongest, candidate 2 is next, and so forth, with candidate 20 being the weakest. We then generate typical committee triad data that somewhat confuses the true ranking and evaluate the algorithm on how well it approximates the correct ranking from this data.

On triad $(i < j < k)$, we simulate a judgment by independently sampling normal distributions $\mathcal{N}(i, \sigma)$, $\mathcal{N}(j, \sigma)$, and $\mathcal{N}(k, \sigma)$, obtaining values X_i, X_j, and X_k. We note the permutation (u, v, w) of (i, j, k) such that $X_u < X_v < X_w$ and declare the local ranking: u is strongest, v is in the middle, and w is weakest. If $u < v < w$, this assignment discovers the true relative strengths of (i, j, k). In this case, $u = i$, $v = j$, $w = k$, and the samples are sufficiently close to their means that the relative order $X_i < X_j < X_k$ preserves the order $i < j < k$. The variance parameter, σ^2, increases or decreases the probability that samples constitute a different order than the normal centers.

The probability of an erroneous judgment on pair $i < j$ is $P(X_i \geq X_j)$. Noting that $X_i - X_j$ is normally distributed with mean $(i - j)$ and variance $2\sigma^2$, we obtain

$$P(X_i \geq X_j) \quad = \quad P(X_i - X_j \geq 0) = P\left(\frac{X_i - X_j - (i-j)}{\sigma\sqrt{2}} \geq \frac{-(i-j)}{\sigma\sqrt{2}}\right) = 1 - \Phi\left(\frac{j-i}{\sigma\sqrt{2}}\right),$$

where Φ is the cumulative distribution function of the standard normal $\mathcal{N}(0, 1)$. Table (1a) exhibits error probabilities for various strength differences and σ values. Clearly, candidates with large strength differences

$j-i$	$P(X_i - X_j) \geq 0$		
	$\sigma = 2$	$\sigma = 5$	$\sigma = 10$
1	0.3618	0.4438	0.4718
2	0.2398	0.3886	0.4438
4	0.0786	0.2858	0.3886
6	0.0169	0.1981	0.3357
8	0.0023	0.1289	0.2858
10	0.0002	0.0786	0.2398
12	0.0000	0.0448	0.1981
14	0.0000	0.0239	0.1611
16	0.0000	0.0118	0.1289
18	0.0000	0.0055	0.1015
20	0.0000	0.0023	0.0786

(a)

Candidate	Score	Candidate	Score
1	6.100	12	2.709
2	4.867	13	2.561
3	4.850	16	2.042
4	4.676	14	1.794
5	3.875	11	1.793
8	3.874	19	1.737
6	3.631	15	1.501
9	3.384	17	1.029
7	3.295	18	0.354
10	2.956	20	0.000

(b)

Table 1: (a) Misjudgment probabilities for strength differences $j - i$ and varying σ
(b) Global ranking from simulated comittee data with $\sigma = 4$

can nevertheless be misjudged with significant probability. For a given strength difference, misjudgments are more probable with larger σ, which corresponds to greater overlap of the distributions and therefore simulates a less competent committee.

We propose to rank candidate triads rather than pairs. In this case, a misjudgment occurs when $i < j < k$ holds but $X_i < X_j < X_k$ fails. Since either $X_i \geq X_j$ or $X_j \geq X_k$ implies $X_i < X_j < X_k$ fails, we conclude that the probability of a misjudgment is at least $\max\{P(X_i \geq X_j), P(X_j \geq X_k)\}$. Therefore if we substitute $\max\{j - i, k - j\}$ for $j - i$ in Table (1a), the corresponding line gives a *lower* bound for the misjudgment probability of triad $(i < j < k)$.

Returning to our 20-candidate context, we maintain that a random 10% of the $\binom{20}{3}$ triads, provided they cover the candidate field, can provide a good consensus ranking. The committee tasks are then: (a) choose a random selection of 114 triads, ensuring that some chain of triad entries connects each candidate pair; (b) rank the triads independently; (c) algorithmically obtain the global ranking that is most consistent with the independent judgments. Table (1b) reports the algorithm's performance on simulated data. While a perfect ranking would rank candidates 1 through 20 in increasing order, the average algorithm misplacement is 1.10 positions. No candidate is misplaced more than 4 positions, and 8 of the 20 are placed exactly. The list's top quarter contains all candidates who should be in that quarter. Other quarters also exhibit near perfect selection. In general, separated list segments are in perfect order. That is, for example, each candidate in the highest ranking quarter is accurately judged of higher merit than any candidate in the third quarter.

The following sections elaborate the theory behind Hodge decomposition of graphs and illustrate its applicability, returning to the lollipop problem in the last section. The tutorial exposition here assumes no familiarity with continuous Hodge theory. For a deeper mathematical treatment, the reader is referred to [1]. For similar approaches but different derivations and application areas, see [2] and [3].

2 Graphs as simplicial complexes

Traditionally, an undirected graph is a pair (V, E), where V is a vertex set and E is an edge set. In this paper, we deal only with finite graphs and fix $|V| = n > 0$ throughout the discussion. Common practice denotes the vertices as $v_1, v_2, \ldots, v_n$ and an edge between v_i and v_j as e_{ij}. However, we wish to consider higher dimensional structures and therefore need a more general notation. Consequently, we denote the vertex set by $(1), (2), \ldots, (n)$ and use the term 0-dimensional simplex for a vertex. In general, a k-dimensional *simplex* (k-simplex) of a graph is fully connected subgraph on $k + 1$ of the vertices.

A vertex is a 0-simplex; an edge is a 1-simplex; a triangle is a 2-simplex, and so forth. The graphs of Figure (2) all exhibit the same simplex collections. Specifically, each contains 8 0-simplices, the 8 vertices. Each contains 13 of the possible $\binom{8}{2}$ 1-simplices, the edges, with names $(12), (18), (23)$ and so forth. We find 5 2-simplices, the triangles $(236), (345), (346), (356), (456)$, and one 3-simplex (3456). These graphs contain no higher dimensional simplices because none admits a fully connected subgraph on 5 or more vertices.

Clearly, each boundary component of a k-simplex constitutes a $(k-1)$-simplex. The four boundary faces

of the 3-simplex (3456) are the 2-simplices (345), (346), (356), (456). However, it is possible to have additional 2-simplices that are not boundary components of a 3-simplex. In Figure (2), for example, triangle (236) is not the boundary of any 3-simplex. Similarly, triangle boundaries constitute 1-simplices (edges) and there may exist additional edges that are not boundary components of any triangle.

When we specify a graph as $\mathcal{G} = (V, E)$, we emphasize only to the lowest dimensional simplices. This specification determines the higher dimensional components, so a definition need not explicitly reference them. However, our purpose here is better served by specifying $\mathcal{G} = (\Sigma_0, \Sigma_1, \ldots, \Sigma_K)$, where $\Sigma_0 = V$, the collection of vertices, $\Sigma_1 = E$, the collection of edges, Σ_2 is the collection of triangles, and so forth. In general, Σ_k contains fully connected vertex subsets of size $k+1$. For any given graph on n vertices, there exists an integer $K < n$ for which fully connected subgraphs over $K+1$ vertices exists, but no larger vertex set participates in a fully connected subgraph. The collection $\mathcal{S} = \cup_{k=0}^{K} \Sigma_k$ is called a *simplicial complex*.

We used ascending vertex order to name the k-simplices of Figure (2). However, triangle (326) contains the same vertices and is therefore the same triangle as (236). In our computations, a given simplex may appear under any of its aliases, and we will be concerned whether or not the alias $(i_1 i_2 \ldots i_k)$ is an even or odd permutation of the corresponding increasing sequence. We adopt the notation $i_1 \ldots i_k$ for an index sequence that may or may not be in ascending order, whereas $\overline{i_1 \ldots i_k}$ means $1 \le i_1 < \ldots < i_k \le n$. To specify that $\overline{j_1 \ldots j_k}$ is the increasing permutation of $i_1 \ldots i_k$, we use the abbreviation $\overline{j_1 \ldots j_k} \sim i_1 \ldots i_k$. The *canonical name* of a k-simplex is the increasing sequence $\overline{j_1 \ldots j_{k+1}}$ of its vertices. When context suffices, we drop parentheses, as for example, in a functional expression where $f(i_1 \ldots i_k)$ replaces $f((i_1 \ldots i_k))$.

An *oriented k-simplex* is a k-simplex together with one of two possible orientations: positive or negative. We associate the positive orientation with the sequence $\overline{i_1 \ldots i_{k+1}}$ and with any even index permutation of that canonical name. We associate the negative orientation with odd permutations. These orientations will play a role in the mappings from simplices to real numbers that we now introduce.

With $\mathcal{G} = (\Sigma_0, \Sigma_1, \ldots, \Sigma_K)$, let $<\Sigma_k>$ denote the real vector space spanned by Σ_k. Thus,

$$<\Sigma_k> \;=\; \left\{ \sum_{(\overline{i_1 \ldots i_{k+1}}) \in \Sigma_k} a_{\overline{i_1 \ldots i_{k+1}}} \, (\overline{i_1 \ldots i_{k+1}}) \;:\; a_{\overline{i_1 \ldots i_{k+1}}} \in \mathcal{R} \right\}. \tag{1}$$

We will refer to the elements of Σ_k, each indexed in ascending order, as the canonical basis for $<\Sigma_k>$. In Figure (2), $<\Sigma_2>$ contains vectors of the form $a_{236}(236) + a_{345}(345) + a_{346}(346) + a_{356}(356) + a_{456}(456)$. The vector space $<\Sigma_2>$ has five dimensions, corresponding to the graph's five triangles.

A vector $f \in <\Sigma_k>$ defines a function $f : \Sigma_k \to \mathcal{R}$ via $f(\overline{i_1 \ldots i_{k+1}}) =$ the coefficient of $(\overline{i_1 \ldots i_{k+1}})$. In $<\Sigma_k>$, the vector representation is called a $(k+1)$-form and is traditionally written with wedge-products. As a 3-form, the above example from Σ_2 is $a_{236}\, dx_2 \wedge dx_3 \wedge dx_6 + a_{346}\, dx_3 \wedge dx_4 \wedge dx_6 + \ldots$. Such k-forms derive from the differential geometry context of continuous Hodge theory. The functional representation is called a *cochain*, a term deriving from cohomology studies. As a cochain $f : \Sigma_2 \to \mathcal{R}$, the example maps simplex (236) to real number a_{236} and so forth.

Like Σ_k-simplices, $<\Sigma_k>$-vectors have alternative names. Specifically, vector manipulation induces index permutations in the underlying simplices, which can flip their orientations and thereby introduce sign changes in the vector. For example,

$$\begin{array}{rcrcrcrcr} 47.2\,(236) & - & 62.8\,(345) & + & 23.0\,(346) & + & 28.1\,(356) & - & 34.0\,(456) \\ 47.2\,(236) & + & 62.8\,(354) & - & 23.0\,(436) & + & 28.1\,(563) & - & 34.0\,(645) \end{array}$$

represent the same vector. That is, $(i_1 \ldots i_k) \equiv \epsilon_{i_1 \ldots i_k} (\overline{j_1 \ldots j_k})$, where $\overline{j_1 \ldots j_k} \sim i_1 \ldots i_k$ and $\epsilon_{i_1 \ldots i_k} = \pm 1$ as $i_1 \ldots i_k$ is an even or odd permutation of $\overline{j_1 \ldots j_k}$.

For $\omega, \eta \in <\Sigma_k>$, we define the inner product $<\omega, \eta>$ as the linear extension of

$$<(i_1 \ldots i_{k+1}), (j_1 \ldots j_{k+1})> \;=\; \begin{cases} \epsilon_{i_1 \ldots i_{k+1}} \cdot \epsilon_{j_1 \ldots j_{k+1}}, & \text{if } j_1 \ldots j_{k+1} \text{ is a permutation of } i_1 \ldots i_{k+1} \\ 0, & \text{otherwise,} \end{cases}$$

which is the traditional dot-product of the coefficient vectors after adjusting each simplex to its representation as an increasing sequence. Clearly $<\omega, \eta> = <\eta, \omega>$.

Returning now to the simplicial complex $\mathcal{G} = (\Sigma_0, \Sigma_1, \ldots, \Sigma_K)$, we define mappings from $<\Sigma_k>$ to $<\Sigma_{k+1}>$ and vice versa. For this purpose, two further notations are convenient: $\overline{i_1 \ldots \check{j} \ldots i_k}$ denotes $\overline{i_1 \ldots i_k}$ with index j inserted in the proper position and $\overline{i_1 \ldots \widehat{i_j} \ldots i_k}$ is $\overline{i_1 \ldots i_k}$ with element i_j removed.

For $0 \le k < K$, define $\delta_k : <\Sigma_k> \to <\Sigma_{k+1}>$ as the linear extension of

$$\delta_k(\overline{i_1 \ldots i_{k+1}}) \;=\; \sum_{(j\overline{i_1 \ldots i_{k+1}}) \in \Sigma_{k+1}} (j\overline{i_1 \ldots i_{k+1}}) = \sum_{\overline{(i_1 \ldots \check{j} \ldots i_{k+1})} \in \Sigma_{k+1}} \epsilon_{j\overline{i_1 \ldots i_{k+1}}} \, (\overline{i_1 \ldots \check{j} \ldots i_{k+1}}). \tag{2}$$

As an example of this computation, consider the vector $\omega = a(36) + b(34) \in <\Sigma_1>$ of Figure (2). We find

$$\begin{aligned}\delta_1(\omega) &= a\delta_1(36) + b\delta_1(34) \\ &= a[(236) + (536) + (436)] + b[(534) + (634)] = a[(236) - (356) - (346)] + b[(345) + (346)] \\ &= a(236) + b(345) + (b-a)(346) - a(356) \in <\Sigma_2> . \end{aligned} \tag{3}$$

For $\overline{j_1 \dots j_{k+1}} \sim i_1 \dots i_{k+1}$, linearity enables the following computation. Hence, we need not adjust the vector $(i_1 \dots i_{k+1})$ to an increasing sequence to apply the first construction in Equation (2).

$$\begin{aligned}\delta_k(i_1 \dots i_{k+1}) &= \epsilon_{i_1 \dots i_{k+1}} \delta_k(\overline{j_1 \dots j_{k+1}}) = \sum_{(p\overline{j_1 \dots j_{k+1}}) \in \Sigma_{k+1}} \epsilon_{i_1 \dots i_{k+1}} (p\overline{j_1 \dots j_{k+1}}) \\ &= \sum_{(pi_1 \dots i_{k+1}) \in \Sigma_{k+1}} \left(\epsilon_{i_1 \dots i_{k+1}}\right)^2 (pi_1 \dots i_{k+1}) = \sum_{(pi_1 \dots i_{k+1}) \in \Sigma_{k+1}} (pi_1 \dots i_{k+1}).\end{aligned}$$

Equation (2) obtains δ_k of a particular k-simplex as an algebraic combination of $(k+1)$-simplices. By linearity, (2) suffices to compute $\delta_k(\omega)$ for any $\omega \in <\Sigma_k>$. A computational formula for the functional representation is also useful; it gives the value assigned by $\delta_k(\omega)$ to a particular $(k+1)$ simplex as a combination of the values assigned by ω to k-simplices. From (2), a $(k+1)$-simplex $(\overline{i_1 \dots i_{k+2}})$ appears as one of the summands in each $\delta_k(\overline{i_1 \dots \widehat{i_j} \dots i_{k+2}})$ expansion, in which case it appears with the sign $(-1)^{j-1}$:

$$[\delta_k(\omega)](\overline{i_1 \dots i_{k+2}}) = \sum_{j=1}^{k+2} (-1)^{j-1} \omega(\overline{i_1 \dots \widehat{i_j} \dots i_{k+2}}). \tag{4}$$

Consider again $\omega = a(36) + b(34) \in <\Sigma_1>$ of Figure (2). As a function, $\omega : \Sigma_1 \to \mathcal{R}$ is

$$\omega(\overline{i_1 i_2}) = \begin{cases} a, & (\overline{i_1 i_2}) = (36) \\ b, & (\overline{i_1 i_2}) = (34) \\ 0, & \text{otherwise.} \end{cases}$$

Using (4), we check that $\delta_1(\omega)$ assigns $(b-a)$ to the 2-simplex (346), in agreement with (3):

$$[\delta_1(\omega)](346) = \omega(46) - \omega(36) + \omega(34) = b - a.$$

For $0 \le k < K$, define $\delta_k^* : <\Sigma_{k+1}> \to <\Sigma_k>$ via the inner product: $\delta^*(\eta)$ is the linear map such that

$$<\omega, \delta_k^*(\eta)> = <\delta_k(\omega), \eta> \tag{5}$$

for all $\omega \in \Sigma_k$. The discussion below shows how this expression unambiguously defines δ_k^*.

A linear map is determined by its action on each of the individual simplices in Σ_{k+1}. So, assuming

$$\delta_k^*(\overline{i_1 \dots i_{k+2}}) = \sum_{(\overline{j_1 \dots j_{k+1}}) \in \Sigma_k} a_{\overline{j_1 \dots j_{k+1}}} (\overline{j_1 \dots j_{k+1}}), \tag{6}$$

a formula for δ_k^* must calculate the unknown coefficients $a_{\overline{j_1 \dots j_{k+1}}}$. For a fixed $\overline{j_1 \dots j_{k+1}}$, (6) gives

$$\begin{aligned} a_{\overline{j_1 \dots j_{k+1}}} &= <(\overline{j_1 \dots j_{k+1}}), \delta_k^*(\overline{i_1 \dots i_{k+2}})> = <\delta_k(\overline{j_1 \dots j_{k+1}}), (\overline{i_1 \dots i_{k+2}})> \\ &= \sum_{(p\overline{j_1 \dots j_{k+1}}) \in \Sigma_{k+1}} \epsilon_{p\overline{j_1 \dots j_{k+1}}} <(\overline{j_1 \dots \check{p} \dots j_{k+1}}), (\overline{i_1 \dots i_{k+2}})> . \end{aligned}$$

The last inner product is one if $p = i_q$ and $\overline{j_1 \dots j_{k+1}} = \overline{i_1 \dots \widehat{i_q} \dots i_{k+2}}$; otherwise it is zero. Therefore,

$$a_{\overline{j_1 \dots j_{k+1}}} = \begin{cases} \epsilon_{\overline{i_q i_1 \dots \widehat{i_q} \dots i_{k+2}}} = (-1)^{q-1}, & \overline{j_1 \dots j_{k+1}} = \overline{i_1 \dots \widehat{i_q} \dots i_{k+2}} \\ 0, & \text{otherwise.} \end{cases}$$

That is, $\delta_k^*(\overline{i_1 \dots i_{k+2}})$ involves only the boundary elements of $(\overline{i_1 \dots i_{k+2}})$:

$$\delta_k^*(\overline{i_1 \dots i_{k+2}}) = \sum_{q=1}^{k+2} (-1)^{q-1} (\overline{i_1 \dots \widehat{i_q} \dots i_{k+2}}), \tag{7}$$

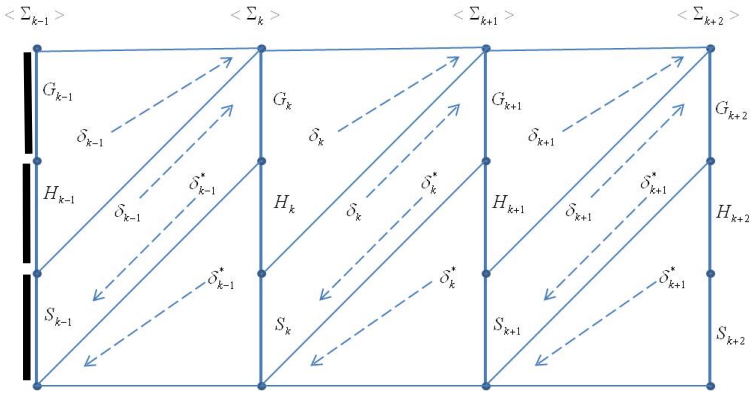


Figure 1: Hodge decompositions

For example, consider $\eta = a(346) + b(356) \in <\Sigma_2>$ of Figure (2):

$$\begin{aligned} \delta_1^*(\eta) &= a\delta_1^*(346) + b\delta_1^*(356) = a[(46) - (36) + (34)] + b[(56) - (36) + (35)] \\ &= a(34) + b(35) - (a+b)(36) + a(46) + b(56) \in <\Sigma_1> . \end{aligned} \tag{8}$$

As we did with δ_k, we can deduce a functional form for δ_k^*. From (7), we see that a term $(\overline{i_1 \dots i_{k+1}})$ occurs in the δ_k^* expansion of each $(\overline{i_1 \dots \check{p} \dots i_{k+1}}) \in \Sigma_{k+1}$. Consequently, for $\eta \in <\Sigma_{k+1}>$,

$$[\delta_k^*(\eta)](\overline{i_1 \dots i_{k+1}}) = \sum_{(p\overline{i_1 \dots i_{k+1}}) \in \Sigma_{k+1}} \eta(p\overline{i_1 \dots i_{k+1}}). \tag{9}$$

In the preceding example, η assumes the functional form

$$\eta(\overline{i_1 i_2 i_3}) = \begin{cases} a, & (\overline{i_1 i_2 i_3}) = (346) \\ b, & (\overline{i_1 i_2 i_3}) = (356) \\ 0, & \text{otherwise.} \end{cases}$$

We check that (9) assigns $(-a-b)$ to (36), in agreement with (8):

$$[\delta_1^*(\eta)](36) = \eta(236) + \eta(536) + \eta(436) = \eta(236) - \eta(356) - \eta(346) = -b - a.$$

3 The Hodge Decomposition

Let $\mathcal{G} = (\Sigma_0, \Sigma_1, \dots, \Sigma_K)$ as in the previous section. For $0 < k < K$, the Hodge decomposition exhibits $<\Sigma_k>$ as a direct sum of three orthogonal subspaces: $<\Sigma_k> = G_k \oplus H_k \oplus S_k$. G_k will be the image of $<\Sigma_{k-1}>$ under δ_{k-1}, S_k will be the image of $<\Sigma_{k+1}>$ under δ_k^*, and H_k will contain those elements of $<\Sigma_k>$ that are orthogonal to both G_k and S_k. Figure (1) visualizes the mappings among the three components of each $<\Sigma_k>$. The three-segmented vertical lines suggest the three orthogonal components of each space. The disks represent the zero-vector. There is, of course, just one zero-vector, so all disks in a given vertical line represent the unique zero vector; the multiple copies facilitate a cleaner graphic. We now derive the details of these decompositions.

G (k-1) deltk-1
S(k+1) delt* k+1
H perp S,G

Lemma 1 *For any* $\omega \in <\Sigma_k>$, $\delta_{k+1}(\delta_k(\omega)) = 0$. *For any* $\eta \in <\Sigma_{k+1}>$, $\delta_{k-1}^*(\delta_k^*(\eta)) = 0$.

Proof. By linearity, we need only establish the claim on basis vectors. For $\omega = (\overline{i_1 \dots i_{k+1}}) \in \Sigma_k$,

$$\begin{aligned} \delta_{k+1}(\delta_k(\omega)) &= \delta_{k+1}(\delta_k(\overline{i_1 \dots i_{k+1}})) = \sum_{(p\overline{i_1 \dots i_{k+1}}) \in \Sigma_{k+1}} \delta_{k+1}(p\overline{i_1 \dots i_{k+1}}) \\ &= \sum_{(p\overline{i_1 \dots i_{k+1}}) \in \Sigma_{k+1}} \left[\sum_{q<p:(qp\overline{i_1 \dots i_{k+1}}) \in \Sigma_{k+2}} (qp\overline{i_1 \dots i_{k+1}}) + \sum_{q>p:(qp\overline{i_1 \dots i_{k+1}}) \in \Sigma_{k+2}} (qp\overline{i_1 \dots i_{k+1}}) \right] \\ &= \sum_{\{q<p\}:(qp\overline{i_1 \dots i_{k+1}}) \in \Sigma_{k+2}} [(qp\overline{i_1 \dots i_{k+1}}) - (qp\overline{i_1 \dots i_{k+1}})] = 0. \end{aligned}$$

For the second claim, we compute, for an arbitrary $\omega \in <\Sigma_{k-1}>, \eta \in <\Sigma_{k+1}>$,

$$<\omega, \delta_{k-1}^*(\delta_k^*(\eta))> \quad = \quad <\delta_{k-1}(\omega), \delta_k^*(\eta)> = <\delta_k(\delta_{k-1}(\omega)), \eta> = 0.$$

Hence $\delta_{k-1}^*(\delta_k^*(\eta)) = 0$. ■

Now, in each $<\Sigma_k>$, define S_k for $0 \leq k < K$ and G_k for $0 < k \leq K$ as follows.

$$\begin{aligned} S_k &= \text{image}(\delta_k^*) = \{\omega \in <\Sigma_k> : \exists \eta \in <\Sigma_{k+1}> \text{ with } \delta_k^*(\eta) = \omega\} \\ G_k &= \text{image}(\delta_{k-1}) = \{\omega \in <\Sigma_k> : \exists \eta \in <\Sigma_{k-1}> \text{ with } \delta_{k-1}(\eta) = \omega\}. \end{aligned} \tag{10}$$

Lemma 2 *G_k and S_k are orthogonal subspaces of $<\Sigma_k>$.*

Proof. Let $\omega \in G_k, \eta \in S_k$. We have

$$\begin{aligned} \omega &= \delta_{k-1}(\omega'), \text{ for some } \omega' \in <\Sigma_{k-1}> \\ \eta &= \delta_k^*(\eta'), \text{ for some } \eta' \in <\Sigma_{k+1}> \\ <\omega, \eta> &= <\delta_{k-1}(\omega'), \delta_k^*(\eta')> = <\delta_k(\delta_{k-1}(\omega')), \eta'> = 0. \end{aligned}$$ ■

From Lemma (2), $G_k \cup S_k = G_k \oplus S_k$. Defining $H_k = (G_k \oplus S_k)^\perp$, we obtain $<\Sigma_k> = G_k \oplus H_k \oplus S_k$.

Lemma 3 *$G_k^\perp = kernel(\delta_{k-1}^*)$ and $S_k^\perp = kernel(\delta_k)$.*

Proof. Let $\omega \in G_k^\perp$ and suppose η is an arbitrary vector in $<\Sigma_{k-1}>$. Then

$$<\eta, \delta_{k-1}^*(\omega)> \quad = \quad <\delta_{k-1}(\eta), \omega> = 0 \quad \text{by Lemma (2) since } \delta_{k-1}(\eta) \in G_k.$$

We conclude $\delta_{k-1}^*(\omega) = 0$, $\omega \in \text{kernel}(\delta_{k-1}^*)$, and $G_k^\perp \subseteq \text{kernel}(\delta_{k-1}^*)$. Conversely, if $\omega \in \text{kernel}(\delta_{k-1}^*)$ and $\eta \in G_k = \text{image}(\delta_{k-1})$, we have $\eta = \delta_{k-1}(\eta')$ for some $\eta' \in <\Sigma_{k-1}>$ and

$$<\eta, \omega> \quad = \quad <\delta_{k-1}(\eta'), \omega> = <\eta', \delta_{k-1}^*(\omega)> = 0.$$

This forces $\omega \in G_k^\perp$ and consequently $\text{kernel}(\delta_{k-1}^*) \subseteq G_k^\perp$.

The second claim follows in a similar fashion.

$$\begin{aligned} \omega \in \text{kernel}(\delta_k), \eta \in S_k &\Rightarrow <\omega, \eta> = <\omega, \delta_k^*(\eta')> = <\delta_k(\omega), \eta'> = 0 \Rightarrow \omega \in S_k^\perp. \\ \omega \in S_k^\perp, \eta \in <\Sigma_{k+1}> &\Rightarrow \delta_k^*(\eta) \in \text{image}(\delta_k^*) = S_k \Rightarrow <\delta_k(\omega), \eta> = <\omega, \delta_k^*(\eta)> = 0 \\ &\Rightarrow \delta_k(\omega) = 0 \Rightarrow \omega \in \text{kernel}(\delta_k). \end{aligned}$$ ■

Lemma 4 *For $0 \leq k < K$, $S_k = \delta_k^*(G_{k+1})$. For $0 < k \leq K$, $G_k = \delta_{k-1}(S_{k-1})$. Also, $\delta_{k-1}^*|_{G_k}$ is a bijection from G_k to S_{k-1}, and $\delta_k|_{S_k}$ is a bijection from S_k to G_{k+1}.*

Proof. Using Lemma (3) in conjunction with the definitions (10), we have

$$\begin{aligned} S_k &= \delta_k^*(<\Sigma_{k+1}>) = \delta_k^*(G_{k+1} \oplus G_{k+1}^\perp) = \delta_k^*(G_{k+1} \oplus \text{kernel}(\delta_k^*)) = \delta_k^*(G_{k+1}) \\ G_k &= \delta_{k-1}(<\Sigma_{k-1}>) = \delta_{k-1}(S_{k-1} \oplus S_{k-1}^\perp) = \delta_{k-1}(S_{k-1} \oplus \text{kernel}(\delta_{k-1})) = \delta_{k-1}(S_{k-1}). \end{aligned}$$

Shifting indices, we obtain $\delta_{k-1}^*(G_k) = S_{k-1}$. That is, δ_{k-1}^* maps G_k onto S_{k-1}. If $\omega, \omega' \in G_k$ and $\delta_{k-1}^*(\omega) = \delta_{k-1}^*(\omega')$, then

$$\begin{aligned} (\omega - \omega') &\in \text{kernel}(\delta_{k-1}^*) = G_k^\perp \\ (\omega - \omega') &\in G_k \cap G_k^\perp \\ (\omega - \omega') &= 0, \end{aligned}$$

which establishes δ_{k-1}^* as a one-to-one mapping. A similar proof shows that $\delta_k|_{S_k}$ is also bijective. ■

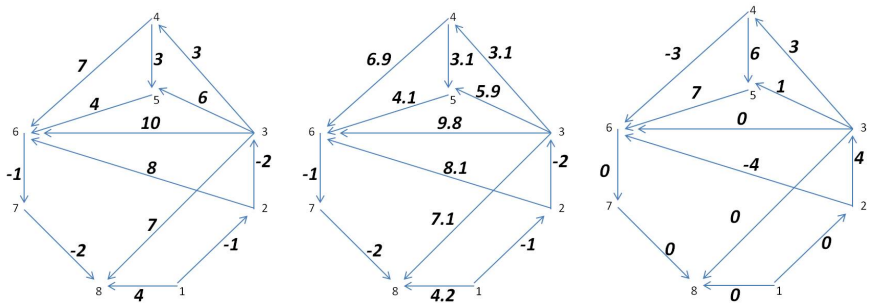


Figure 2: Widget flow among cities: (a) gradient, (b) nongradient, (c) solenoidal

Theorem 5 (Hodge Decomposition) *For* $0 < k < K$, $<\Sigma_k> = \delta_{k-1}(<\Sigma_{k-1}>) \oplus [kernel(\delta^*_{k-1}) \cap kernel(\delta_k)] \oplus \delta^*_k(<\Sigma_{k+1}>)$. *Every* $\omega \in <\Sigma_k>$ *admits a unique decomposition* $\omega = \omega_g + \omega_h + \omega_s$ *with*

$$\begin{aligned} \omega_g &\in G_k = \delta_{k-1}(\Sigma_{k-1}) \\ \omega_h &\in H_k = kernel(\delta^*_{k-1}) \cap kernel(\delta_k) \\ \omega_s &\in S_k = \delta^*_k(\Sigma_{k+1}). \end{aligned}$$

Proof. Our definitions established

$$<\Sigma_k> \;=\; G_k \oplus H_k \oplus S_k = \delta_{k-1}(<\Sigma_{k-1}>) \oplus [(G_k \oplus S_k]^\perp \oplus \delta^*_k(<\Sigma_{k+1}>).$$

It remains to show that $[(G_k \oplus S_k]^\perp = \text{kernel}(\delta^*_{k-1}) \cap \text{kernel}(\delta_k)$. Using Lemma (3), we argue

$$[G_k \oplus S_k]^\perp \;=\; G_k^\perp \cap S_k^\perp = \text{kernel}(\delta^*_{k-1}) \cap \text{kernel}(\delta_k). \quad .$$

The unique decomposition of $\omega \in <\Sigma_k>$ follows from the orthogonal subspace decomposition of $<\Sigma_k>$ ■

We now have the relationships implied by Figure (1). In particular, each G_k is isomorphic to the preceding S_{k-1}. It follows that $\dim(G_k) = \dim(S_{k-1})$.

4 Applications in finding best estimates

In this section, we will emphasize graphs as elements of $<\Sigma_1>$. That is, a given set of edge weights directly establishes an element of $<\Sigma_1>$. Figure (2a) shows a directed graph that depicts widget flow among cities. Traffic flows in both directions, and the assigned edge weight reflects the net flow in the direction of the arrow. Negative weights then specify a net flow against the arrow. This graph is the $<\Sigma_1>$-element ω, for which we compute $\delta^*_0(\omega)$ via the methods of the previous section:

$$\begin{aligned} \omega &= -(12) + 4(18) - 2(23) + 8(26) + 3(34) + 6(35) + 10(36) \qquad (11) \\ &\quad +7(38) + 3(45) + 7(46) + 4(56) - (67) - 2(78) \\ \delta^*_0(\omega) &= -[(2) - (1)] + 4[(8) - (1)] - 2[(3) - (2)] + 8[(6) - (2)] + 3[(4) - (3)] + 6[(5) - (3)] + 10[(6) - (3)] \\ &\quad +7[(8) - (3)] + 3[(5) - (4)] + 7[(6) - (4)] + 4[(6) - (5)] - [(7) - (6)] - 2[(8) - (7)] \\ &= -3(1) - 7(2) - 28(3) - 7(4) + 5(5) + 30(6) + (7) + 9(8). \text{ negative divergence in each point} \qquad (12) \end{aligned}$$

In vector calculus, the *divergence* of a vector field is the net outgoing flux at a point. We compute this value by surrounding the point with a small ball and integrating the outgoing normal field component over the ball's surface. The divergence is the limit of this integral divided by the ball volume, as the ball radius shrinks to zero; it gives a measure of the diverging or converging field strength at a point. By analogy, we consider our directed graph as a discrete vector field, defined at each vertex and having components directed to or from its neighbors with magnitudes given by the edge weights. At each vertex, we define the *divergence* as the excess of outgoing over incoming weights. That is, if $w_{\overline{i_1 i_2}}$ is the weight associated with edge $(\overline{i_1 i_2})$,

$$\text{divergence}(i_1) \;=\; \sum_{j>i_1:(\overline{i_1 j})\in<\Sigma_1>} w_{\overline{i_1 j}} \;-\; \sum_{j<i_1:(\overline{j i_1})\in<\Sigma_1>} w_{\overline{j i_1}}. \qquad (13)$$

net incoming vectors - net outcoming vectors

Returning to our example, we note that $\delta_0^*(\omega)$ assigns to each vertex the negative of its divergence, and the short proof below extends this property in the general case.

Lemma 6 *Let $\omega \in <\Sigma_1>$ represent a weighted directed graph. Then, for each vertex (i_1),*

$$[\delta_0^*(\omega)](i_1) = -\mathit{divergence}(i_1).$$

Proof. Suppose the graph has n vertices and $\omega = \sum_{(\overline{i_1 i_2})\in\Sigma_1} w_{\overline{i_1 i_2}}(\overline{i_1 i_2})$. From (9),

$$[\delta_0^*(\omega)]((i_1)) = \sum_{j:(ji_1)\in\Sigma_1} \omega(ji_1) = \sum_{j<i_1:(ji_1)\in\Sigma_1} \omega(ji_1) - \sum_{j>i_1:(i_1 j)\in\Sigma_1} \omega(i_1 j) = -\text{divergence}(i_1). \quad \blacksquare$$

For ω given by (11) and representing the graph of Figure (2a), we note from (12) that vertex (3) has the largest divergence, a net outflow of 28, and is therefore a net widget producer. Similarly, vertex (6) has the smallest divergence, a net inflow of 30, and is a net widget consumer. Divergence then provides one measure of vertex importance. In this example, we can rank the vertices from (6) with a divergence of -30 up through (3) with a divergence of 28, affording higher ranks to vertices with greater net production.

Continuing with the same ω, consider the $\eta \in \Sigma_0$ below, together with the edge assignments of $\delta_0(\eta)$:

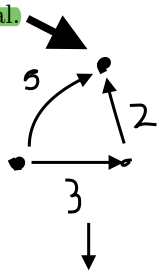
$$\begin{aligned} \eta &= -2(1) - 3(2) - 5(3) - 2(4) + (5) + 5(6) + 4(7) + 2(8) \qquad (14)\\ [\delta_0(\eta)](12) &= \eta(2) - \eta(1) = -1 = \omega(12) \\ &\vdots \qquad\qquad\qquad \vdots \\ [\delta_0(\eta)](78) &= \eta(8) - \eta(7) = -2 = \omega(78). \end{aligned}$$

delta(2) —> (12)—> + w(2)
delta(1) —>(21) —> -(12) —> -w(1)

We have $\delta_0(\eta) = \omega$. That is, $\omega \in \text{image}(\delta_0)$, and therefore its Hodge decomposition is $\omega_g = \omega$, $\omega_h = \omega_s = 0$. Also, $\eta \in \Sigma_0$ is a function that assigns values to vertices, and $\delta_0(\eta)$ assigns to edge $(\overline{i_1 i_2})$ the incremental change in that function in passing from vertex (i_1) to (i_2). By analogy with a similar interpretation in the vector calculus, we refer to δ_0 as the *gradient* operator. By extension, the graph $\delta_0(\eta)$ is a *gradient graph*.

In a gradient graph, the edge-weight sums along any paths connecting two given vertices are identical. That is, if $pi_1 i_2 \ldots i_s q$ and $pj_1 j_2 \ldots j_t q$ are two such paths in $\omega = \sum_{(i_1 i_2)\in\Sigma_1} a_{i_1 i_2}(i_1 i_2) = \delta_0(\eta)$, we have

$$\begin{aligned} a_{pi_1} + \left(\sum_{u=2}^{s} a_{i_{u-1} i_u}\right) + a_{i_s q} &= \eta(i_1) - \eta(p) + \sum_{k=2}^{s}[\eta(i_k) - \eta(i_{k-1})] + \eta(q) - \eta(i_s) = \eta(q) - \eta(p) \\ a_{pj_1} + \left(\sum_{u=2}^{t} a_{j_{u-1} j_u}\right) + a_{j_t q} &= \eta(j_1) - \eta(p) + \sum_{k=2}^{t}[\eta(j_k) - \eta(j_{k-1})] + \eta(q) - \eta(j_t) = \eta(q) - \eta(p). \end{aligned}$$



sum of edge weights is the same

An *undirected cycle* in a directed graph is a closed path that may move with or against the edge directions. When moving against the edge direction, we negate the edge weight. In this sense, the edge weights around any undirected cycle sum to zero for a gradient graph, as the reader can verify for Figure (2a).

Lemma 7 *G is a gradient graph if and only if the edge weight sum along any undirected cycle is 0.*

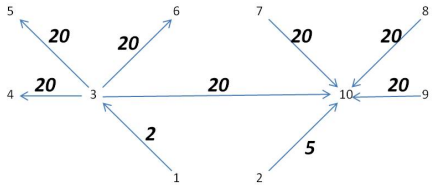
this means that there are no sinks or "fonts"

suda

Proof. The discussion above establishes that a gradient graph produces zero sums around any undirected cycle. For the converse, we construct an $\eta \in \Sigma_0$ such that $\omega = \delta_0(\eta)$. For each connected component of the undirected graph, choose an arbitrary starting vertex, say (i_1), and let $\eta(i_1) = 0$. Proceeding with or against edges, follow paths to other component vertices, summing edge weights, negated if necessary. On reaching a new vertex, say i_j, let $\eta(i_j)$ be the value of the sum at that point. Because the sum around any cycle is zero, if we encounter a vertex more than once, we will assign it the same value each time. An edge weight in ω is then the increment in η from its lower to higher vertex. That is, $\omega = \delta_0(\eta)$ is a gradient graph. $\blacksquare$

si agafem el ccle la sua total es 0 (3+2-5 o bé 5-3-2)

Actually, the gradient graph ω of Figure (2a) was constructed as $\delta_0(\eta)$, for the η in (14) above. However, as the construction of Lemma (7) involves arbitrary choices, a given gradient graph possesses many generators in Σ_0. In general, for gradient graph $\omega \in \Sigma_1$, let $\delta_0^{-1}(\omega) = \{\eta \in \Sigma_0 : \delta_0(\eta) = \omega\}$. On a given component of the underlying undirected graph, the difference between any two elements of $\delta_0^{-1}(\omega)$ will be a constant assignment to the vertex components. Consequently, on such a graph, all elements of $\delta_0^{-1}(\omega)$ provide the same vertex ranking within each component. In particular, any connected gradient graph ω admits a unique vertex ranking, obtained by arranging the vertex coefficients of any $\eta \in \delta_0^{-1}(\omega)$ in descending order.



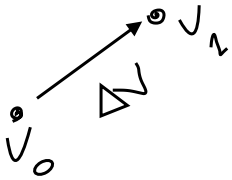
vertex	1	2	3	4	5	6	7	8	9	10
divergence	2	5	78	-20	-20	-20	20	20	20	-85
potential	22	5	20	0	0	0	20	20	20	0
divergence rank	5	6	10	2	3	4	7	8	9	1
potential rank	10	5	6	2	3	4	7	8	9	1

0 of potential. all flows converge to 10

Figure 3: Traffic flow designed to emphasize potential rank performance over divergence rank

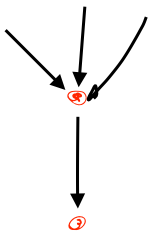
For any $\eta \in \delta_0^{-1}(\omega)$, we can view the vertex assignments of $-\eta$ as providing a pressure, or incentive, to produce the edge flows of ω. If the graph represents an electrical network with unit resistance on each edge, then voltages maintained at the values given by $-\eta$ create the currents given by ω. Similar observations apply when the $-\eta$ assignments are interpreted as hydrostatic pressures or constant temperature heat sources/sinks. Among physicists, voltages, pressures, temperatures, and the like are known as *potential* functions, implying that such functions induce certain voltage, pressure, or temperature gradients. By analogy, we refer to an element $\eta \in \delta_0^{-1}(\omega)$ as a potential function over the vertices that induces gradient ω.

The gradient graph of Figure (2a) admits the following comparisons, where the potential $\eta \in \delta_0^{-1}(\omega)$ is that of Equation (14).



vertex	1	2	3	4	5	6	7	8
divergence	3	7	28	7	-5	-30	-1	-9
potential	2	3	5	2	-1	-5	-4	-2
divergence rank	5	6	8	7	3	1	4	2
potential rank	5	7	8	6	4	1	2	3

higher potential
high divergence (<0)



Lower potential
low divergence (<0)

Divergence ranks vertex 7 higher than vertex 8, while potential rank reverses this assessment. Divergence deals only with vertex flow, while potential rank takes the strength of neighboring nodes into consideration.

In Figure (3) we contrive a graph in which the potential rank flips the divergence rank of nodes 1 and 2 by specifying the lower divergence node as exporting widgets into a prodigious widget producer. The larger potential rank acknowledges the node's ability to export into a hostile market. Referring to this graph as ω, we compute an $\eta \in \delta_0^{-1}(\omega)$ via an initial assignment of zero to vertex 10, obtaining the comparisons shown in the figure. This graph is clearly a gradient graph because there are no undirected cycles.

A general graph is not necessarily a gradient graph, but it will have a gradient component via its Hodge decomposition. The gradient component represents the best least squares match of a gradient graph to the given graph. That is, if $\omega = \omega_g + \omega_h + \omega_s$ is the Hodge decomposition of $\omega \in <\Sigma_1>$, then

$$||\omega - \omega_g|| = \min_{\omega' \in G_1} ||\omega - \omega'||, \tag{15}$$

where $||x|| = <x, x>^{1/2}$ is magnitude of vector x. This observation follows from an expansion of $||\omega - \omega'||$. That is, for $\omega' \in G_1$,

$$\begin{aligned} ||\omega - \omega'||^2 &= <\omega - \omega', \omega - \omega'> = <(\omega_g - \omega') + \omega_h + \omega_s, (\omega_g - \omega') + \omega_h + \omega_s> \\ &= <\omega_g - \omega', \omega_g - \omega'> + 2\ <\omega_g - \omega', \omega_h + \omega_s> + \ <\omega_h + \omega_s, \omega_h + \omega_s> \\ &= ||\omega_g - \omega'||^2 + ||\omega_h + \omega_s||^2, \end{aligned}$$

because $\omega_g - \omega' \in G_1$ and $\omega_h + \omega_s \in G_1^\perp$. The last expression clearly assumes its minimum value $||\omega_h + \omega_s||$ when $\omega' = \omega_g$. So, ω_g is the best least squares approximation to ω among the gradient graphs, and the magnitude of the residual $\omega_h + \omega_s$ measures how well ω_g approximates ω.

We now analyze non-gradient graphs with nontrivial Hodge decompositions. Figure (2b) is a modified version of Figure (2a). Edge weights around undirected cycles no longer sum to zero, and therefore it is not a pure gradient graph. Referring to this graph as $\omega \in \Sigma_1$, let $\omega = \omega_g + \omega_h + \omega_s$ be its Hodge decomposition. To compute ω_g, it suffices to know an $\alpha \in \delta_0^{-1}(\omega_g)$. Noting that $\omega_h + \omega_s \in G_1^\perp = \text{kernel}(\delta_0^*)$, we have

w_g = delta(alpha)

$$\delta_0^*(\omega) = \delta_0^*(\omega_g) + \delta_0^*(\omega_h + \omega_s) = \delta_0^*(\omega_g) = \delta_0^*(\delta_0(\alpha)). \tag{16}$$

$<\Sigma_0>$ is an n-dimensional real vector space, whose basis vectors are the 0-simplices: $(1), (2), \ldots, (n)$. Over

edge	ω	ω_g	ω_s	ω_h
(12)	-1.0000	-0.9759	0.0000	-0.0241
(18)	4.2000	4.1759	0.0000	0.0241
(23)	-2.0000	-1.9167	-0.0800	-0.0033
(26)	8.1000	8.0407	0.0800	-0.0207
(34)	3.1000	3.0037	0.1050	-0.0087
(35)	5.9000	5.9537	-0.0450	-0.0087
(36)	9.8000	9.9574	-0.1400	-0.0174
(38)	7.1000	7.0685	0.0000	0.0315
(45)	3.1000	2.9500	0.1500	0.0000
(46)	6.9000	6.9537	-0.0450	-0.0087
(56)	4.1000	4.0037	0.1050	-0.0087
(67)	-1.0000	-0.9444	0.0000	-0.0556
(78)	-2.0000	-1.9444	0.0000	-0.0556

(a)

vertex	α	div(ω)	div(ω_g)
(1)	0.0000	3.2	3.2
(2)	-0.9759	7.1	7.1
(3)	-2.8926	27.9	27.9
(4)	0.1111	6.9	6.9
(5)	3.0611	-4.9	-4.9
(6)	7.0648	-29.9	-29.9
(7)	6.1204	-1.0	-1.0
(8)	4.1759	-9.3	-9.3

triangle	γ	curl(ω)	curl(ω_s)
(236)	-0.0800	-0.3000	-0.3000
(345)	-1.1309	0.3000	0.3000
(346)	1.2359	0.2000	0.2000
(356)	-1.1759	0.2000	0.2000
(456)	1.2809	0.3000	0.3000

(b)

Table 2: (a) Hodge decomposition of Figure (2b)
(b) Solutions α and γ of Equations (18) and (20) respectively

this basis, the matrix form of $\delta_0^*\delta_0 : <\Sigma_0> \rightarrow <\Sigma_0>$ is particularly simple. The i^{th} column describes

$$\delta_0^*(\delta_0(i)) \quad = \quad \delta_0^* \left(\sum_{j:(ji)\in <\Sigma_1>} (ji) \right) = \sum_{j:(ji)\in <\Sigma_1>} [(i)-(j)] = n_i(i) - \sum_{j:(ji)\in <\Sigma_1>} (j),$$

where n_i = number of neighbors of (i). Column i of the matrix then has n_i in the diagonal position and a minus one in each row j such that an edge connects i and j. Being the (negated) divergence of a gradient, the map $\delta_0^*\delta_0$ is called the graph *Laplacian* and is analogous to the Laplacian of vector calculus. Here, we denote the Laplacian by $\triangle_0$. Substituting in (16), we obtain a suitable $\alpha \in \delta_0^{-1}(\omega_g)$ as a solution of

$$\triangle_0(\alpha) \quad = \quad \delta_0^*(\omega) = -\text{divergence}(\omega). \tag{17}$$

Assuming $\alpha = \sum_{i=1}^{n} a_i\,(i)$, we have the following linear system for ω of Figure (2b).

$$\begin{bmatrix} 2 & -1 & & & & & & -1 \\ -1 & 3 & -1 & & & -1 & & \\ & -1 & 5 & -1 & -1 & -1 & & -1 \\ & & -1 & 3 & -1 & -1 & & \\ & & -1 & -1 & 3 & -1 & & \\ & -1 & -1 & -1 & -1 & 5 & -1 & \\ & & & & & -1 & 2 & -1 \\ -1 & & -1 & & & & -1 & 3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix} = \begin{bmatrix} -3.2 \\ -7.1 \\ -27.9 \\ -6.9 \\ 4.9 \\ 29.9 \\ 1.0 \\ 9.3 \end{bmatrix} \tag{18}$$

unknown

-div(w)

From the solution α, we obtain $\omega_g = \delta_0(\alpha)$, calculated as $\omega_g(\overline{i_1 i_2}) = \alpha(i_2) - \alpha(i_1)$. Table (2a) displays the Hodge decomposition, $\omega = \omega_g + (\omega_h + \omega_s)$, the solution α, and the divergence of ω_g. At this point, we have shown only how to extract the gradient component ω_g. Subsequent discussion will take up a similar extraction of component ω_s, which will then enable the full decomposition shown in the table. Since $\omega_h, \omega_s \in G_1^{\perp} = \text{kernel}(\delta_0^*)$, we expect $\delta_0^*(\omega) = \delta_0^*(\omega_g) + \delta_0^*(\omega_h + \omega_s) = \delta_0^*(\omega_g)$, as confirmed by Table (2b).

We observe small vertex residuals: $\omega_h + \omega_s = \omega - \omega_g$. For widget flows, we might dismiss these residuals as traffic flow measurement errors. This same graph could arise from an electrical network, with the edge flows representing currents through unit resistances. In this case, $-\alpha$ gives a best least-squares estimate, up to an additive constant, of the node voltages. The actual currents forced by these voltages are given by the ω_g edge weights, and the small residuals $\omega_h + \omega_s$ give a measure of confidence in the approximation.

A similar analysis extracts the component $\omega_s \in S_1$. We know that $\omega_s = \delta_1^*(\gamma)$ for some $\gamma \in <\Sigma_2>$. Consequently, since both ω_g and ω_h reside in $S_1^{\perp} = \text{kernel}(\delta_1)$, we have

$$\delta_1(\omega) \quad = \quad \delta_1(\omega_g + \omega_h) + \delta_1(\delta_1^*(\gamma)) = \delta_1(\delta_1^*(\gamma)). \tag{19}$$

The term $\delta_1(\omega)$ is known as the graph *curl*. This term also derives from vector calculus, where the curl measures a field's rotational tendency at a given point. Unlike the scalar divergence, the curl is a vector quantity. Computationally, we orient a small disk perpendicular to the direction in which we want to measure the curl. We then integrate the tangential field component around the disk periphery, divide by the disk area, and take the limiting value as the disk radius shrinks to zero. We can verify that δ_1 performs an analogous measurement, although the graph's discrete nature leaves only triangles to play the role of "small disks." For a triangle $(\overline{i_1 i_2 i_3}) \in \Sigma_2$, we have

$$[\delta_1(\omega)](\overline{i_1 i_2 i_3}) \quad = \quad \omega(\overline{i_1 i_2}) + \omega(\overline{i_2 i_3}) - \omega(\overline{i_1 i_3}),$$

which is the edge sum around the triangle, starting at i_1 and proceeding toward i_2. That is, curl(ω) assigns a rotational value to each triangle, equal to the edge-flow sum on its boundary. Of course, if ω is a pure gradient graph, this cyclical sum will be zero, conforming to the vector calculus identity: curl $\circ$ gradient $= 0$.

Equation (19) now reads: $\delta_1 \delta_1^*(\gamma) = \text{curl}(\omega)$. Using this equation to solve for γ, we obtain $\omega_s = \delta_1^*(\gamma)$. We envision a matrix form for $\delta_1 \delta_1^* : <\Sigma_2> \rightarrow <\Sigma_2>$, in which the elements $(\overline{i_1 i_2 i_3})$ of Σ_2 label the rows and columns in increasing lexicographic order. In the graph of Figure (2b), for example, the row/column labels are $(236), (345), (346), (356), (456)$. Column $(\overline{i_1 i_2 i_3})$ then represents

$$\begin{aligned}
\delta_1(\delta_1^*(\overline{i_1 i_2 i_3})) \quad &= \quad \delta_1((\overline{i_2 i_3}) - (\overline{i_1 i_3}) + (\overline{i_1 i_2})) = \sum_{(j\overline{i_2 i_3}) \in \Sigma_2} (j\overline{i_2 i_3}) - \sum_{(j\overline{i_1 i_3}) \in \Sigma_2} (j\overline{i_1 i_3}) + \sum_{(j\overline{i_1 i_2}) \in \Sigma_2} (j\overline{i_1 i_2}) \\
&= \quad 3(\overline{i_1 i_2 i_3}) + \sum_{j \in J_1} (j\overline{i_2 i_3}) - \sum_{j \in J_2} (j\overline{i_1 i_3}) + \sum_{j \in J_3} (j\overline{i_1 i_2}),
\end{aligned}$$

where

$$\begin{aligned}
J_1 \quad &= \quad \{j : j \notin \{i_1, i_2, i_3\} \text{ and } (j\overline{i_2 i_3}) \in \Sigma_2\} \\
J_2 \quad &= \quad \{j : j \notin \{i_1, i_2, i_3\} \text{ and } (j\overline{i_1 i_3}) \in \Sigma_2\} \\
J_3 \quad &= \quad \{j : j \notin \{i_1, i_2, i_3\} \text{ and } (j\overline{i_1 i_2}) \in \Sigma_2\}.
\end{aligned}$$

We conclude that column $(\overline{i_1 i_2 i_3})$ contains a three on the diagonal. Moreover, for $(\overline{j_1 j_2 j_3}) \neq (\overline{i_1 i_2 i_3})$,

$$\text{row } (\overline{j_1 j_2 j_3}) \quad \text{contains} \quad \begin{cases} \epsilon_{j\overline{i_2 i_3}}, & \exists j \text{ with } \overline{j_1 j_2 j_3} \sim j\overline{i_2 i_3} \\ -\epsilon_{j\overline{i_1 i_3}}, & \exists j \text{ with } \overline{j_1 j_2 j_3} \sim j\overline{i_1 i_3} \\ \epsilon_{j\overline{i_1 i_2}}, & \exists j \text{ with } \overline{j_1 j_2 j_3} \sim j\overline{i_1 i_2} \\ 0, & \text{otherwise.} \end{cases}$$

This formula is notationally cumbersome but simple in practice. An off-diagonal matrix entry is zero unless the row and column labels differ in exactly one index. If so, replace the nonmatching index in the row label by the nonmatching index in the column label, or vice versa, and permute the resulting label to canonical form. The matrix entry is ± 1 as the permutation is even or odd. For example, suppose we are computing the matrix entry in row (236), column (346). We compute either of the sequences:

$$\begin{array}{llllll}
(236) & \text{replace 2 with 4} & \Rightarrow & (436) & \text{permute} & \Rightarrow & (346) \\
(346) & \text{replace 4 with 2} & \Rightarrow & (326) & \text{permute} & \Rightarrow & (236),
\end{array}$$

each requiring an odd number of transposition in the last step. Accordingly, the matrix entry is (-1).

Assuming γ takes the form $\sum_{(\overline{i_1 i_2 i_3}) \in \Sigma_2} c_{\overline{i_1 i_2 i_3}} (\overline{i_1 i_2 i_3})$, Equation (19) assumes the following form for the graph of Figure (2b).

delta_1 delta_1* = omega

$$\begin{bmatrix} 3 & & -1 & -1 & \\ & 3 & 1 & -1 & 1 \\ -1 & 1 & 3 & 1 & -1 \\ -1 & -1 & 1 & 3 & 1 \\ & 1 & -1 & 1 & 3 \end{bmatrix} \begin{bmatrix} c_{236} \\ c_{345} \\ c_{346} \\ c_{356} \\ c_{456} \end{bmatrix} = \begin{bmatrix} -0.3 \\ 0.3 \\ 0.2 \\ 0.2 \\ 0.3 \end{bmatrix} \rightarrow \text{curl(w)} \qquad (20)$$

From the solution γ, we obtain $\omega_s = \delta_1^*(\gamma)$. Using ω_g from our earlier partial decomposition, we complete the full Hodge decomposition with final component $\omega_h = \omega - \omega_g - \omega_s$. Table (2a) shows the decomposition,

the solution γ, and the curls of ω and ω_s. Since $\omega_h, \omega_g \in S_1^{\perp} = \text{kernel}(\delta_1)$, we have $\text{curl}(\omega) = \delta_1(\omega) = \delta_1(\omega_g + \omega_h) + \delta_1(\omega_s) = \delta_1(\omega_s) = \text{curl}(\omega_s)$, as confirmed by Table (2b).

The Hodge gradient component captures the graph divergence at each node. At each vertex in our widget examples, positive divergence indicates excess production over consumption; negative divergence indicates excess consumption. However if we add traffic that simply moves widgets around closed paths, these additional flows do not change any node divergence and consequently have no net impact on production or consumption. The residuals $\omega_s + \omega_h$ represent such cyclical traffic. Small residuals might suggest traffic flow measurements errors, while large residuals could signify unnecessary network congestion, in the sense that widget demands do not justify the measured circulation.

The smallest closed paths are triangles, and we use the term *solenoidal* flow to designate circulatory flow around triangles. The Hodge component ω_s measures, on each edge, the traffic portion due to solenoidal circulation around triangles. As with the edges, we adopt an orientation for triangles. If the flow follows the vertices along an even permutation of the canonical order $\overline{i_1 i_2 i_3}$, then the flow is positive. If it follows an odd permutation of the vertices, we negate its value. That is, the terms $16.3(245) = -16.3(425)$ both specify a flow of magnitude 16.3 proceeding around the triangle (245).

The solenoidal component cannot account for all divergence-free circulation because some circulation can occur around nontriangular paths. Path (12678) of Figure (2b) provides an example. The final Hodge component ω_h provides edge flows associated with such circulations. After computing ω_g and ω_s, we can obtain ω_h by subtracting the two known components from the given graph edge weights. Since $\omega_h \in \text{kernel}(\delta_0^*) \cap \text{kernel}(\delta_1)$, we have

$$(\delta_0 \delta_0^* + \delta_1^* \delta_1)(\omega_h) \quad = \quad 0.$$

For $k > 0$, the map $\delta_{k-1}\delta_{k-1}^* + \delta_k^* \delta_k : <\Sigma_k> \rightarrow <\Sigma_k>$ is called the *generalized Laplacian* and is denoted by $\triangle_k$. Hence $\triangle_1(\omega_h) = 0$, and since such solutions are termed *harmonic* in the continuous theory, we call ω_h the harmonic component.

We can imitate the observations in Equation (15) to show that

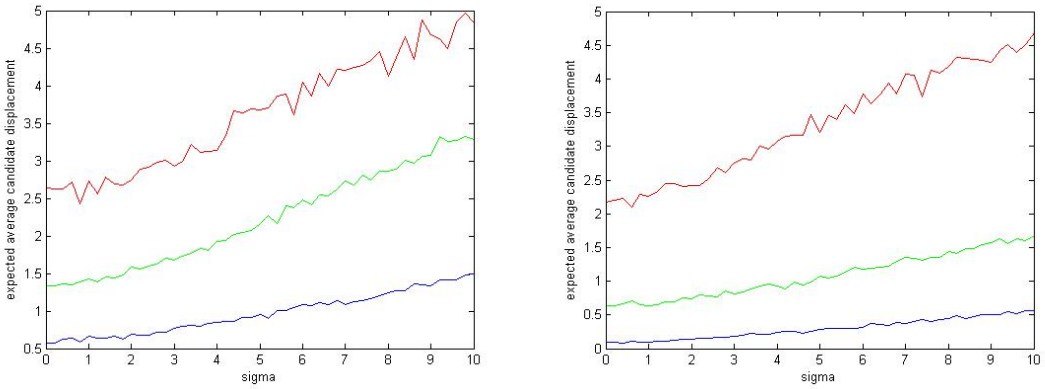
$$||\omega - \omega_s|| \quad = \quad \min_{\omega' \in S_1} ||\omega - \omega'||,$$

which means that ω_s gives the best least squares approximation to a given graph ω within the class of solenoidal graphs. For example, let ω represent the graph of Figure (2c), where a quick check reveals that the divergence is zero at each node. Consequently, this graph's decomposition should have no gradient component. Moreover, weights associated with edges that do not participate in triangles are zero. We then expect that the flow derives from triangular circulations alone, forcing the harmonic component to be zero. That is, the graph will be a pure solenoid, $\omega = \omega_s = \delta_1^*(\gamma)$ for some $\gamma \in <\Sigma_2>$. Indeed, using Equation (9), we find $\gamma = 4(236) + 2(345) + (346) + 3(356) + 4(456)$ yields $\delta_1^*(\gamma) = \omega$.

Recall that a given gradient graph can arise from several potentials in Σ_0, with the various contenders differing by constants on the graph's connected components. A similar multiplicity occurs with solutions γ of Equation (19): $\delta_1(\delta_1^*(\gamma)) = \text{curl}(\omega)$. In the example at hand, we can verify that for any constant c, $\gamma_c = c(345) - c(346) + c(356) - c(456)$ yields $\delta_1^*(\gamma_c) = 0$. Consequently, for any solution γ of $\delta_1(\delta_1^*(\gamma)) = \text{curl}(\omega)$, $\gamma + \gamma_c$ is also a solution. Multiple solutions present no difficulty, since all map to the same unique solenoidal component of the decomposition.

5 An application to global ranking

We return now to the lollipop problem that opened this tutorial. We generate a random triad selection by iterating through all three-vertex subsets and choosing the corresponding triad with some probability p. We reject any sampling in which the triad edges do not generate a connected graph on the candidate vertices. We used $p = 0.1$ to generate the lollipop data of Section(1). In that example, we computed triads judgments using overlapping Gaussians, centered at known ranking positions. In general, however, ground truth is not known, and we accept elementary triad judgments from a panel of experts. In either case, we construct a graph ω from the local judgments, in which a positive edge weight from vertex i to vertex j indicates a "preference" flow from weaker candidate i to stronger candidate j. In the absence of conflicts, this preference flow forms a gradient graph because the sum around any cycle is necessarily zero. In that case, the preference graph ω is equal to its Hodge gradient component ω_g. The gradient component in turn



(a) (b)

Figure 4: Algorithm performance: (a) few local judgments, (b) many judgments

derives from a potential α over the vertices, and $[\delta_0(\alpha)](ij) = \alpha(j) - \alpha(i)$ recovers the overall preference flow from i to j. The potential α then serves to globally rank the vertices.

Of course, we can hardly expect our conflicting local triad rankings to form a pure gradient graph. Such a miracle requires not only that each candidate pair be judged in the correct direction but also that the magnitude of the assigned preferences be consistent across the local judgments. Nevertheless, we conjecture that a common appreciation of merit among the rankers will produce a graph in which the gradient component approximates the group judgment. That is, if σ is small in the lollipop simulation, or if the panel is competent in the real-world counterpart, preference conflicts should be minor.

To this end, we convert our triad judgments into edge weights a_{ij}, such that $a_{ij} > 0$ when candidate j is perceived superior to candidate i. Specifically, since we are ranking triads, we record the assessment $\text{merit}(k) > \text{merit}(j) > \text{merit}(i)$ as $a_{ij} = 1, a_{jk} = 1, a_{ik} = 2$. Clearly this triangle in isolation is a pure gradient graph. At this point, however, a complication occurs. As we choose triads randomly, the same edge, say (ij), can occur in many triads and may therefore receive different weights. Since we are attempting to discover the group's judgment of candidate i versus candidate j, it is reasonable to accumulate the assigned weights. Opposing judgments then tend to reduce the preference flow, while reinforcing judgments increase it. Other methods might be appropriate, if, for example, we know that some rankers are better qualified to assess the relative merit of particular candidates. Using simple summation produces the result noted in the introduction. In brief, the algorithm proceeds as follows.

1. Establish a graph ω by adding the edge weights assigned to the local triangles.
2. Solve the Laplacian equation: $\Delta_0(\alpha) = \delta_0^*(\omega)$.
3. Rank the vertices via α. That is, $\alpha(j) > \alpha(i)$ implies j is ranked higher than i.

We could use edges or tetrhedra instead of triangles, and in our artificial data experiments we can vary the σ parameter to simulate more or fewer conflicts among the rankers. The curves in Figure (4) illustrate how algorithm performance varies with σ, measured as the average number of positions a candidate is displaced from ground truth. We expect this metric to rise monotonically with increasing σ. However, because the simulations choose random triads, it can happen that a triad group chosen with a larger σ can nevertheless outperform a group chosen with a smaller σ. To extract the trend, we smooth the data by averaging many simulations at each σ value. The graphs then report an expected average displacement. All exhibit greater displacements with increased σ, corresponding to greater error associated with less competent judges. It is notable that this displacement remains relatively small, even for large σ.

In both panels of Figure (4), the upper curve corresponds to local judgments pronounced independently on randomly chosen pairs. The middle curve corresponds to triads, and the lowest curve to quadruples. Performance improves markedly in passing from pairs to triads and also exhibits more stability with increasing σ. Further improvement appears when quadruples are chosen for the independent local judgments.

In the left panel, pairs were chosen with probability $p = 0.16$, while triads and quadruples were chosen with $p = 0.04$. Recall that the random selection is constrained to cover all candidates in the sense that any

two candidates are connected by a chain of local comparisons. For pairs, this constraint is difficult to realize with smaller acceptance probabilites. For triads or quadruples, it is easily satisfied with $p = 0.04$. In the right panel, pairs, triads, and quadruples were all chosen with $p = 0.2$. As expected, all curves improve with larger random selections, but this improvement is not without cost. For a given p, a random selection of triads is much larger than a corresponding selection of pairs. A quadruple selection is yet larger.

Hodge decomposition and ranking software is available from [5] as executable Java jar-files. The ranking software provides an extension of the triad-judgment reconciliation described in this paper. Specifically, it accommodates a heterogeneous collection of local judgments (pair, triads, quadruples, etc.) on the input file. The site also provides access to the Java source code for both decomposition and ranking. To compute solutions to the various matrix algebra equations that appear in our earlier discussion, the code adapts the minresQLP linear solver available from the Stanford Systems Optimization Laboratory [6].

References

[1] Arnold, Douglas N.; Falk, Richard S.; Winther, Ragnar. *Finite Element Exterior Calculus: from Hodge Theory to Numerical Stability*, Bulletin of the American Mathematical Society, Vol. 47, No. 2, April, 2010.

[2] Jiang, Xiaoye; Lim, Lek-Heng; Yao, Yuan; Ye, Yinyu. *Statistical Ranking and Combinatorial Hodge Theory*, Mathematical Programming, Vol. 127, No. 1, 2011.

[3] Jiang, Xiaoye; Lim, Lek-Heng; Yao, Yuan; Ye, Yinyu. *Learning to Rank with Combinatorial Hodge Theory*, `http://comptop.stanford.edu/u/preprints/hodge-preprint2.pdf, 2008`.

[4] Goldring, Tom. *Mining Directed Graphs with Discrete Hodge Theory*, (preliminary draft), 2011.

[5] Johnson, James L.: `http://www.cs.wwu.edu/johnson/hodge/hodgeSoftware.htm`, 2012.

[6] Stanford Systems Optimization Laboratory: `http://www.stanford.edu/group/SOL/download.html`.