Explanations of the calculations for the paper "Heptagon relations from a simplicial 3-cocycle, and their cohomology"

Igor G. Korepanov

1 Preliminaries

s:4f

1.1 Technical: package `PL`

The calculations have been done in GAP computer algebra system, with our additional package `PL` that should be downloaded from https://sourceforge.net/projects/plgap/ and installed.

Remark. `PL` package is still at the "pre-alpha" state. The part of it used in the present work is, however, fully functional and well tested.

1.2 A basis in permitted colorings of one 4-face, and coordinates of coboundary-induced colorings in that basis

Let ω be a simplicial 3-cocycle given on some Δ^4, satisfying

$$.\omega_{ijkl} \quad \text{for any 3-face } ijkl. \tag{1}$$ onv

The colors of $\Delta^4 \subset \Delta^m$ belong, by definition, to the *three*-dimensional F-linear space F^3, identified below with the space V_{Δ^4} consisting of 3-cocycles ν on Δ^4 taken *to within adding a multiple of* ω (restricted to Δ^4). This means that we have a chosen basis in each V_{Δ^4}, that is, for any $\vec{p}_4 = \omega|_{\Delta^4}$. Note that any specific choice of these bases does not affect our theoretical constructions; convenient bases must be, though, specified for calculations.

There is a remarkable symmetric bilinear form on V_{Δ^4}. Let ν, η be two 3-cocycles; we make first from them 3-cochain μ as follows:

$$\mu_{ijkl} \stackrel{\text{def}}{=} \frac{\nu_{ijkl}\eta_{ijkl}}{\omega_{ijkl}}, \tag{2}$$ mu

and then define the bilinear form as the value on $\Delta^4 = ijklm$ of its simplicial *coboundary*:

$$Q_{\Delta^4}(\nu, \eta) \stackrel{\text{def}}{=} (\delta\mu)(\Delta^4) = \frac{\nu_{jklm}\,\eta_{jklm}}{\omega_{jklm}} - \cdots + \frac{\nu_{ijkl}\,\eta_{ijkl}}{\omega_{ijkl}} \tag{3}$$ bf4

p:Q **Proposition 1.** *$Q_{\Delta^4}(\nu,\eta)$ depends actually only on the equivalence classes of ν and η modulo ω, both belonging to V_{Δ^4}.*

Proof. This follows from the fact that $Q_{\Delta^4}(\nu,\eta)$ clearly vanishes if either ν or η is proportional to ω. □

We will take the liberty of denoting as $\delta(ijk)$ the coboundary of the simplicial 2-cochain taking value $1 \in F$ on triangle ijk and zero on other triangles. For example, within 4-simplex $\Delta^4 = 12345$, it means that $\delta(234)$ takes value $1 \in F$ on tetrahedron 1234, value $-1 \in F$ on tetrahedron 2345, and zero on other tetrahedra. We call such colorings corresponding to cocycles $\nu = \delta(ijk)$ *triangle vectors*. Surely, *any* simplicial 3-cocycle ν on Δ^4 can be represented (and not uniquely) as a linear combination of triangle vectors.

Bilinear form Q_{Δ^4} provides an elegant way of introducing *coordinates* in our linear space V_{Δ^4} of colors of Δ^4. For instance, here are the coordinates of triangle vectors that we used in our actual calculations, on the example of pentachoron $\Delta^4 = 12345$. We chose them to be proportional to $Q_{\Delta^4}\big(\nu, \delta(s)\big)$, taking three triangles for s, namely $s = 345$, $s = 125$ and $s = 123$ (we write simply Q instead of Q_{12345} in (4) below):

$$
\begin{array}{cc}
\text{Cocycle } \nu & \begin{array}{c}\text{Three coordinates}\\ \text{of the corresponding coloring of } \Delta^4 = 12345\end{array} \\
 & \begin{array}{ccc} -Q\big(\nu,\delta(345)\big) & -Q\big(\nu,\delta(125)\big) & -Q\big(\nu,\delta(123)\big) \\ \cdot\,\omega_{1345}\,\omega_{2345} & \cdot\,\omega_{1235}\,\omega_{1245} & \cdot\,\omega_{1234}\,\omega_{1235} \end{array} \\
\begin{array}{c} \delta(123)\\ \delta(124)\\ \delta(125)\\ \delta(134)\\ \delta(135)\\ \delta(145)\\ \delta(234)\\ \delta(235)\\ \delta(245)\\ \delta(345) \end{array} &
\begin{pmatrix}
0 & \omega_{1245} & \omega_{1234}-\omega_{1235} \\
0 & -\omega_{1235} & \omega_{1235} \\
0 & \omega_{1235}-\omega_{1245} & -\omega_{1234} \\
\omega_{2345} & 0 & -\omega_{1235} \\
-\omega_{2345} & \omega_{1245} & \omega_{1234} \\
\omega_{2345} & -\omega_{1235} & 0 \\
-\omega_{1345} & 0 & \omega_{1235} \\
\omega_{1345} & -\omega_{1245} & -\omega_{1234} \\
-\omega_{1345} & \omega_{1235} & 0 \\
-\omega_{2345}+\omega_{1345} & 0 & 0
\end{pmatrix}
\end{array}
\qquad (4)
$$
v12345

These coordinates work for a generic ω. To be exact, it can be checked that matrix (4) has rank < 3 in the only case where

$$\omega_{1345} = \omega_{2345}, \qquad (5)$$
13

keeping (1) in mind, of course. If we add, however, the "missing" seven columns to (4), so that they will correspond to *all* ten triangles—2-faces of our Δ^4, the rank will, remarkably, be three for *any* ω satisfying (1), as the following proposition states.

p:Qr **Proposition 2.** *Q_{Δ^4}, considered as a bilinear form on V_{Δ^4}, has the full rank—that is, three.*

Proof. If the rank of Q_{Δ^4} were less than 3, then not only (5), but also other similar relations would hold, with other subscripts. This would lead to a contradiction between (1) and the cocycle condition $\omega_{1234} - \omega_{1235} + \omega_{1245} - \omega_{1345} + \omega_{2345} = 0$. □

Due to Propositions 1 and 2, there is the canonical isomorphism between V_{Δ^4} and its conjugate space $V^*_{\Delta^4}$, this latter consisting of course of linear forms on 3-cocycles vanishing on ω.

1.3 Recalling the explicit expressions for 5-cocycles in characteristics two and three

The value of the 5-cocycle in characteristic two on a 5-simplex w can be expressed as

$$c(\nu,\eta) = \sum_{\substack{v,v'\subset w\\ v<v'}} Q_v(\nu,\eta) Q_{v'}(\nu,\eta) + \sum_{v\subset w} \tilde\epsilon_v^{(w)} \bigl(Q_v(\nu,\eta)\bigr)^2, \qquad (6)$$ h2

where v and v' are faces of w. We assume in (6) that these faces are *numbered*, and understand their numbers when writing "$v < v'$".

In characteristic three:

$$\sum_{\substack{v_1,v_2,v_3\subset w\\ v_1<v_2<v_3}} \epsilon_{v_1}^{(w)} \epsilon_{v_2}^{(w)} \epsilon_{v_3}^{(w)}\, Q_{v_1}(\nu,\eta)\, Q_{v_2}(\nu,\eta)\, Q_{v_3}(\nu,\eta), \qquad (7)$$ h3

where v_1, v_2 and v_3 are faces of w.

2 Manifold: cells, coboundaries, cohomology

s:M

2.1 Function `ei( M, m, mp1_cell, m_cell )`

For manifold `M`, returns the incidence coefficient $= 0$ or ± 1 between the $(\mathtt{m}+1)$-simplex with number `mp1_cell` and `m`-simplex with number `m_cell`.

2.2 Function `coboundary( M, addr )`

`M` is a manifold, and `addr` must be a two-component list

$$\mathtt{addr} = [\, m-1,\ \text{the number of } (m-1)\text{-simplex} \,]$$

The function returns the coboundary of an $(m-1)$-simplex in manifold `M` in the form of the list of the same length as `M.faces[`m`]`—that is, the number of m-faces—where at each place the incidence coefficient stays between the corresponding m-face and the $(m-1)$-face determined according to `addr`.

2.3 Function `basis_cohomology( M, field, n )`

Returns a basis in the space of `n`-cohomologies of manifold `M` with coefficients in `field`, in the form of a list of `n`-cocycles representing the basis cohomology classes. Each `n`-cocycle is represented as a list of `field`-valued coefficients at `n`-cells.

2.4 Function `PolTriangulatedOrient( M )`

For a triangulated manifold `M`, returns consistent orientations ± 1 of its cells of the maximal dimension. Orientations are taken with respect to the orientations determined by the vertices of each cell taken in their increasing order.

3 Heptagon colorings of a manifold

s:mc

3.1 Function `evt( M, triangle, pentachoron, cocycle )`

For a given `triangle` (= 2-face) we take simplicial 3-cocycle equal to the coboundary of the delta-function of that `triangle`. That is, the cochain taking value 1 on that `triangle` and 0 on the others. We fix a basis in each 3-dimensional space of permitted colorings for a given `pentachoron` (= 4-face), determined by the `cocycle` ω, according to (4). Function `evt` substitutes right vertex numbers instead of 12345 in (4), and returns the 3-row-vector of components of the (permitted) coloring of the `pentachoron` appearing due to that 3-`cocycle`. Zero of course if `triangle` is not a face of `pentachoron`.

`M` is a triangulated manifold.

3.2 Function `e( M, g_simplex, cocycle )`

Here n=3, hence 2*n-2=4. Function e returns the list of 3-row-vectors (made by evt) for all 4-faces.

3.3 Function `all_e( M, cocycle )`

Given are manifold M and cocycle omega. Function `all_e` returns matrix whose rows correspond to triangles, while each row is as returned by `e` with the only difference that all the inner square brackets are removed. That is, each triple of columns corresponds to a 4-face.

3.4 Function `colorings( M, cocycle )`

Here $k = 3$. And within it:

Inner function `r_d_s_g( d_simplex )` What we do: the rows of `on_d_simplex` give a basis of permitted colorings of a given `d_simplex`, that is, 5-simplex. Then we make matrix r of orthogonal *rows*. Then we make matrix m which is matrix r with zeroes added at the places corresponding to other 5-simplices (not our given `d_simplex`). `r_all_result` is the matrix made of such `m = r_d_s_g(t)` for all 5-simplices as `d_simplex`, by placing them one under another. The rows of `r_all_result` give this way all restrictions on permitted colorings of M. No more problems: `colorings.g` gives a basis of linear space V_r, while `colorings.p` gives a basis of linear space V_p.

4 Heptagon cocycles

s:hc

4.1 Function `gran( M, cocycle, pentachoron, x, y )`

`gran` means, in principle, "face".

Here, specifically, the value of bilinear form Q (3) is returned on chosen a 4-face, namely, `pentachoron`. Arguments `x` and `y` are two permitted colorings of manifold `M` (ν and η in (3) are their restrictions on the `pentachoron` Δ^4) given in the same format as the *rows* of the output of function `all_e`: each triple of columns corresponds to a 4-face.

4.2 Function `on_d( M, cocycle, d_simplex, x, y )`

This function is not actually used in calculations; it is just to check that the previous function `gran` gave indeed a 4-cocycle. Function `on_d` takes a 5-simplex `d_simplex` and sums the values of Q along its 4-faces, with proper signs. Some permitted colorings `x` and `y` must be supplied as arguments (in the same format as in function `gran`).

4.3 Function `on_d_2( M, cocycle, d_simplex, x, y )`

This function calculates the value (6) of 5-cochain in characteristic two on `d_simplex`

4.4 Function `on_d_3( M, cocycle, d_simplex, x, y )`

This function calculates the value (7) of 5-cochain in characteristic three on `d_simplex`

5 Calculations

s:calc

The calculations are done for 3-cocycle ω being an $\mathbb{F}_2$-cocycle (this is important!) plus a coboundary whose components belong to a big enough finite field F of characteristic two. Vector spaces V_p and V_g are taken over F.

Concerning characteristic three, no nontrivial results could be obtained as yet, so our calculations go just in characteristic two.

5.1 Function `itogi_function( )`: returned values

This function returns the list of lists `[ vektor, subset_V_p_V_g, d_V_p, d_V_g, sum_sumsum, sum, rank_A ]` of the function's local variables. Of these,

- `vektor` runs, essentially, over all third cohomology classes of M with coefficients in $\mathbb{F}_2$, starting from the zero class. To be exact, `vektor` shows the coordinates of the class in some basis,
- `subset_V_p_V_g` is a technical Boolean variable serving just to check that the inclusion of vector spaces $V_g \subset V_p$ holds indeed (so it looks like the calculations go right),
- `d_V_p` is $\dim V_p$ —the dimension of the space of permitted colorings of M,
- `d_V_g` is $\dim V_g$ —the dimension of the space of g-colorings of M,
- `sum_sumsum` is one more technical Boolean variable showing whether the `sum` calculated in two ways (and called `sumsum` in one of its versions) fives the same result, where
- `sum` is the polynomial of two permitted colorings from which our invariant comes out. It turns out to be, in all our examples, a symmetric bilinear form of the *squares* of coordinates of the two permitted colorings ν and η of M,
- `rank_A` is the rank of that symmetric bilinear form.

5.2 Function `itogi_function( )`: inner variables

- `basis_p` consists of permitted colorings whose equivalence classes modulo g-colorings form a basis in V_p/V_g,
- `u` and `v` are lists of `field`-valued coefficients with which we make linear combinations of the vectors from `basis_p`, see next item,
- `x` and `y` are these linear combinations for `u` and `v`, respectively. They are what is denoted ν and η in the paper,

- `sum` is $I(M,\omega)$ from the paper. Remember that ω is determined by the above (vektor),

- `U` and `V` are *not* actually needed for calculations, and stay here just for control. They are lists of coefficients that can be placed at the basis vectors of V_g, see next item,

- `xx` and `yy` are the vectors `x` and `y` to which linear combinations of three basis vectors of V_g are added, whose coefficients are indeterminates from lists `U` and `V`, respectively,

- `sumsum` is like `sum` but calculated with `xx` and `yy`. These sums must be equal, and this is controled by the Boolean variable `sum_sumsum`. Remark: in simple cases, like $M = S^2 \times S^3$, `sum_sumsum` may return `false`, because both sums are zero, but GAP thinks that they belong to different fields (`GF(2)` and `<field in characteristic 2>`, respectively). This brings about no difficulties,

- `A` is the matrix of the symmetric bilinear form `sum` of the *squares* of variables in `x` and `y`. Because, looking at `sum`, we see that it has exactly such form in all our examples.

5.3 Integer cohomology

Sometimes it may make sense to obtain a description of cocycles ω, like we did for $M = \mathbb{R}P^4 \times S^1$. We did that by calculating the cohomology with coefficients in $\mathbb{Z}$, reducing the obtained cocycle $\bmod 2$, and comparing with each of the four cocycles ω. Here is the relevant GAP function:

```
basis_cohomology_integers := function( M, field, n )
 local coboundaries, mat, cocycles, f, bas;
 coboundaries := FreeLeftModule( Integers,
  List([1..Length(M.faces[n-1])],
  nm1_cell -> coboundary(M,[n-1,nm1_cell]) ) );
 mat := List([1..Length(M.faces[n])], n_cell ->
     coboundary(M,[n,n_cell]) );
 cocycles := FreeLeftModule( Integers, NullspaceIntMat( mat ) );
 f:= NaturalHomomorphismBySubspace( cocycles, coboundaries );
 bas := Basis(cocycles/coboundaries);
 return List( bas, x -> PreImagesRepresentative( f, x )
   # + Random( coboundaries )      # !!!
    );
end;
```