CODE FOR: SIGNATURES OF ALTERNATING GROUP ACTIONS WITH NON-ZERO QUOTIENT GENUS

JENNIFER PAULHUS AND AARON WOOTTON

Below we describe the code and output files used to prove the computational component in the paper named above. Details on notation and terminology used below are in that paper. We will write generating vectors for signatures $[1; n_1, \ldots, n_r]$ as $(a, b, c_1, \ldots, c_r)$.

1. Introduction

For $n < 40$ we use computer searches to prove that every potential signature for A_n with quotient genus 1 is an actual signature (except for one case each for $n = 5$ and 6). Of course, for each n, there are an infinite number of potential signatures so to successfully check small n values, we need to be able to reduce that list to a manageable finite number.

We first find generating vectors for $[1; k]$ for each k in $\mathcal{O}_{A_n}$. See Section 1 below for computational details on how we find these generating vectors.

Suppose we have a generating vector (a, b, c) corresponding to $[1; k]$. If k is odd, this generating vector guarantees a generating vector for the signature $[1; k, k]$, simply take (a, b, c^2, c^{-1}), since c^2 and c^{-1} will have the same order as c. Note we can repeat this idea to create a generating vector for any signature $[1; k, \ldots, k]$.

If k is even, the existence of a generating vector for $[1; k]$ guarantees a generating vector for $[1; k, k, k]$, here take (a, b, c, c^{-1}, c) as a generating vector (and similarly any odd number of k's). For k even, then, we must also search for generating vectors for $[1; k, k]$. Details about how we found these generating vectors are in Section 2 below. Once we have those generating vectors we can find a generating vector for $[1; k, k, k, k]$ (and indeed any even number of k's) using the same idea as we did for an odd number of k's.

At this point, we know that every signature $[1; k, \ldots, k]$ is an actual signature for each n (again, except for one case each for $n = 5$ and 6 which we discuss below).

Finally, for every pair $k_1, k_2 \in \mathcal{O}_{A_n}$ with $k_1 \neq k_2$ we demonstrate elements in A_n of order k_1 and k_2, respectively, which generate the group A_n. This guarantees that for any signature $[1; k_1, \ldots, k_r]$ where at least two distinct values appear among the k_i's we can find a generating vector. Simply use the computation above to pick elements

of order k_m and k_j ($k_m \neq k_j$) that generate the group, then choose any elements in the group of orders the remaining k_i for the rest of the generating vectors. Then since the elements of orders k_m and k_j already generate all of A_n, we simply use the work of Bertram (see Theorem 2.2 in the paper) to find ℓ-cycles a so that $aa^{-1} = c_1, \ldots, c_r$. Thus $[1; k_1, \ldots, k_r]$ is an actual signature. See Section 3 below for the details on how we computed all these examples.

2. $[1; k]$

We create generating vectors for signatures $[1; k]$ for every $k \in \mathcal{O}_{A_n}$. To do this, we construct an element in A_n of each order and with maximum support by searching through representatives of each conjugacy class in Magma (see `bertram.m`). Call that element c.

We then used code in Python which produces the elements described in Bertram's paper a so that $a \cdot a^{-1} = c$ (see the file `bertram.py`), and confirmed that the elements in Bertram's paper together with this element of maximum support generate the whole group (using code in `final_check.m`).

We do need to ensure a and a^{-1} are in the same conjugacy class. The code in `bertram.py` creates minimum length ℓ-cycles (where $\ell = \lfloor 3n/4 \rfloor$) and adds any values in $\{1, \ldots, n\}$ which are not in c (so as to get full support between a and c). For the examples we compute, those cycles have $\ell < n - 1$. This ensures a and a^{-1} are in the same conjugacy class (conjugacy classes of ℓ-cycles split in A_n if and only if ℓ is odd and equal to $n - 1$).

There were 8 cases where the maximum support element we found in Magma, together with the corresponding element from Bertram's paper did not generate the whole group. You can see generating vectors for those examples in the `README` file. All other examples can be found in the `OnePeriodOutputs` folder.

3. $[1; k, \ldots, k]$

Now we consider the cases $[1; k, k, ..., k]$. We treat the $k = 2$ case separately.

If we have $[1; 2]$ and $[1; 2, 2]$ (or $[1; 2, 2, 2]$ for $n = 5$ case) we get $[1; 2, 2, 2, ..., 2]$ by taking c_1 from the $[1; 2]$ case for odd r and adding two $(1\ 2)(3\ 4)$, or take c_1 and c_2 from the $[1; 2, 2]$ case and add two $(1\ 2)(3\ 4)$ for r even.

As mentioned in the introduction, for any k odd we already have $[1; k, \ldots, k]$ by the previous section since given a generating vector (a, b, c) for $[1; k]$, we get one for $[1; k, k]$ by using a, b, c^2, c^{-1}. This logic extends to create generating vectors for $[1; k, \ldots, k]$ with arbitrarily long periods.

For k even, if we know $[1; k]$, we also know $[1; k, k, k]$ since given a generating vector (a, b, c) for $[1; k]$, we can create the generating vector a, b, c, c^{-1}, c for $[1; k, k, k]$. However, we don't automatically get generating vectors for $[1; k, k]$ and so we need

to additionally search for $[1; k, k]$ for all k even. Once we have a generating vector for $[1; k, k]$ we can get a generating vector for $[1; k, k, k, k]$ by adding a c and c^{-1} like we did above. At this point, we can then find all generating vectors for $[1; k, \dots, k]$ with arbitrarily long periods.

To compute generating vectors for $[1; k, k]$ with k even we take $a = (1\ 2\ 3\ \dots\ n-1)$ or $a = (1\ 2\ 3\ \dots\ n)$ depending on whether n is even or odd. (We note that in the special case where $\mathrm{ord}(c) = n$ for n odd, we instead take $a = (1....n-2)$.) We then choose c to be any element of order k with maximum support. We then verify that $\langle a, c\rangle = A_n$. This gives us the generating vector $(a, \mathrm{id}, c, c^{-1})$ The code to run this is in the file `bertram-1kk.m`. And the generating vectors are stored in `bertram_pairs_logs`.

This method always created a generating vector except for a few cases in very small n (specifically when $n = 6$, 8, and 12). Instead in these cases we used Magma code based on work of Breuer to create a generating vector for $[1; k, k]$ directly. Those results are listed in the `README` file.

Because $[1; 3]$ for $n = 6$ (resp. $[1; 2]$ for $n = 5$) is a potential signature which we know is not an actual signatures, we do also have to find generating vectors for $[1; 3, 3]$ (resp. $[1; 2, 2, 2]$) in order to be able to guarantee the potential signatures $[1; 3, \dots, 3]$ (resp. $[1; 2, \dots, 2]$) are all actual signatures.

We computed these three examples individually and the generating vectors we found are listed in the `README` file.

4. $[1; k_1, k_2]$

Finally we find all signatures $[1; k_1, k_2]$ with $k_1 \neq k_2$. In order to do this, we iterate through all pairs of distinct non-trivial orders of elements of A_n, and for each such pair k_m and k_j, we find elements in A_n of order k_m and k_j respectively which generated all of A_n.

Our first attempt to find such generating vectors is to construct elements of orders k_m and k_j of maximal support as described in the proof of Lemma 3.1 and then use the construction in the proof of Lemma 3.2 to find elements of that same order and cycle types. We then check whether these two elements generate the group which will mean we can use Bertram's construction to find an a and b so that, with the two elements we found will create a generating vector. In the vast majority of cases, this construction worked. The code to compute these examples is at `mixed_pair_I.m`, the cases that failed are at `mixed_pairs_I_logs`, and the successful generating vectors in folders in `MixedPairsIResults`.

For a number of cases we instead had to find a different cycle type of maximal support of a given order and then repeat the process from the previous paragraph. The code to compute these examples is at `mixed_pair_II.m`, the cases that failed

are at `mixed_pairs_II_logs`, and the successful generating vectors in folders in `MixedPairsIIResults`.

In about a dozen cases, we again used Magma code based on work of Breuer to create generating vectors of the signature $[0; k_1, k_2, m]$ for various m. This will guarantee that the chosen c_1 and c_2 from the $[0; k_1, k_2, m]$ generating vector also generate the group, and thus $[1; k_1, k_2]$ will be a signature. These examples are on the `README` file.

In three cases (when $n = 6$, 7, and 8) A_n cannot be generated by an element of order 2 and an element of order 3. We can find examples of generating vectors which satisfy the signature $[1; 2, 3]$ but those examples also require the a and b in the generating vector to generate the whole group. So in order to be able to extend to longer periods with only 2's and 3's, we need to check that we can generate these groups with with two elements of order 2 and an element of order 3, or vice versa (see the last paragraph in the Introduction above for how we can extend to longer periods once we know we have these examples). These examples are in the last section of the `README` file.

Finally, for $n \equiv 3 \bmod 4$ so that $m = (n+3)/2$ is prime or $n \equiv 1 \bmod 4$ so that $m = (n+1)/2$ is prime, we had to independently find elements of order 2 and m which generate the group. In these cases we show that if $c_1 = (1\ 2)(3\ 4)\dots(n-2\ n-1)$ and $c_2 = (1\ 3\ 5\ 7\dots n-2\ n)$ if $n \equiv 1 \bmod 4$ or $c_1 = (1\ 2)(3\ 4)\dots(n-4\ n-3)$ and $c_2 = (1\ 3\ 5\ 7\dots n-2\ n-1\ n)$ if $n \equiv 3 \bmod 4$ then $\langle c_1, c_2\rangle = A_n$.

This work utilized resources from Unity, a collaborative, multi-institution high-performance computing cluster managed by UMass Amherst Research Computing and Data. We used Magma 2.28-14 and python3.