End of Second Year Statement

Elliot Costi

September 2007

1 Reading

1.1 Books

Second Year

1. Certain chapters of Don Taylor's The Geometry of Classical Groups

First Year

1. Larry C. Grove - Classical Groups and Geometric Algebra;
2. Certain chapters of Derek Holt's Handbook of Computation Group Theory;
3. Certain chapters of James and Liebeck's book;
4. Humphries;
5. Humphries.

1.2 Notes

Second Year

1. Alexei Skorobogatov's Lie Algebras course notes.

First Year

1. Robert Wilson's notes on the web regarding alternating and classical groups
2. Generators of the Symplectic Group by Yaim Cooper (notes on a postgrad course taken from the web)
3. Notes on the classical groups taken from a course delivered by Peter Cameron
4. The Magma manual
5. The SL2 recognition code

1.3 Papers

Second Year

1. Constructive Recognition of Classical Groups in Odd Characteristic by Charles Leedham-Green and Eamonn O'Brien

First Year

1. The Computational Matrix Group Project by Charles Leedham-Green
2. Computational Group Theory, Chares C Sims
3. An Introduction to Computational Group Theory, Akos Seress

4. Tensor Products are Projective Geometries, Charles Leedham-Green and Eamonn O'Brien
5. Towards Effective Algorithms for Linear Groups, Eamonn O'Brien
6. Recognising Tensor-Induced Matrix Groups, Charles Leedham-Green and Eamonn O'Brien
7. Constructive Recognition of PSL(2, q), Conder, Charles Leedham-Green and Eamonn O'Brien
8. Constructing Representations of Finite Simple Groups and Covers, Vahid Dabbaghian-Abdoly

2 Seminars

Second Year

1. Gave a presentation on my research at the Postgraduate conference held in Cambridge.
2. Attended Rob Curtis' 60th birthday conference
3. Attended Peter Cameron's 60th birthday conference
4. Gave a talk to the Representation Theory Study Group on constructive recognition of the symplectic groups in a non-natural representation
5. Gave a QuIPS seminar on constructive recognition of the special linear group in a non-natural representation
6. Attended group theory two day seminar session in Auckland
7. Attended the Algebra Colloquium seminars
8. Attended the Pure Maths Seminars
9. Attended the QuIPS seminars
10. Attended the Representation Theory Seminars
11. Organised all QuIPS talks for the 2006 year.

First Year

1. Wrote and gave a two part talk on the Symplectic Groups for the Representation Theory seminars
2. Gave a QuIPS talk on my research at the time - finding an element of SL(d, q) in its natural representation as a word in the generators.

3. Gave a presentation on my research at the time at the Postgraduate conference held in Southampton - finding an element of SL(d, q) in a non-natural representation as a word in the generators.

4. Attended a conference in Birmingham

5. Attended a conference held at Imperial

6. Attended a conference held for Charles Leedham-Green's retirement

7. Attended a conference held in Southampton for postgraduates

3 Prize

In 2006, I won the Eileen Colyer Prize

I was given a grant of 1,000 plus a further 500 from the mathematics department to work in Auckland with Eamonn O'Brien on constructive recognition of the classical groups in their non-natural representation. I found the experience very useful as Eamonn is an expert in computing and was able to teach me about the principals of writing concise code and how to analyse code so that you can get it to run faster.

I am grateful to Alan Camina for giving me the opportunity of having this experience.

4 Research

4.1 A word in the generators for $\mathrm{SL}(d, q)$ - the natural representation

In the Spring term of 2006, my task was to write a piece of code to find an element of $\mathrm{SL}(d, q)$ in its natural representation as a word in the generators.

The generators of $\mathrm{SL}(d, q)$ are given as they appear in the paper Constructive Recognition of Classical Groups in Odd Characteristic by Charles Leedham-Green and Eamonn O'Brien.

Let the element of $\mathrm{SL}(d, q)$ that you wish to find in terms of the generators be A. The first step of the algorithm is to add a multiple of one row to the top to get 1 in the (1, 1) entry. Now, the generators u and v generate a permutation group that is homomorphic to S_d. With these you can manipulate the matrix A in question to move row i to row 1 and column j to column 1 and then use various combinations of conjugates of t and δ to add a multiple of the first row/column to every other entry in first row/column until they are all zero. Working through the matrix A in this way, you will eventually be left with the identity matrix. You will then have $x_1 \dots x_m A x_{m+1} \dots x_n = I$, where the x_i are elements of the generating set. Then you can rearrange the equation to get A in terms of the generators.

The algorithm was done so that it returns the element A as an SLP in the generators of $\mathrm{SL}(d, q)$.

Having completed this, I modified the code to solve the problem for a different set of generators to be used by Eamonn O'Brien's team in Auckland. This code is now in use by the Auckland team.

4.2 A word in the generators for $\mathrm{SL}(d, q)$ - non-natural representations

In the summer term of 2006, my task was to write a piece of code to find an element of $\mathrm{SL}(d, q)$ in a non-natural representation as a word in the image of the standard generators. The first non-natural representation that was looked at was that of the exterior square of the natural representation conjugated by a random matrix. The method of the algorithm, however, will work for any irreducible non-standard representation in the natural characteristic that you wish to deal with.

Consider the subgroup H of $\mathrm{SL}(d, q)$ that fixes the space spanned by the first basis element and then map H to the non-natural representation $\mathrm{SL}(n, q')$ by a map ϕ, where q and q' are both powers of the same prime. We will call the image H^ϕ in order to know that we are working in the higher dimension. Now, H^ϕ acts reducibly on the underlying vector space $(F'_q)^n$ since it has a normal p-subgroup (a theorem from representation theory). So H^ϕ affords a non-trivial submodule U of $(F'_q)^n$. Now if g is the element of $\mathrm{SL}(n, q')$ that we wish to find in terms of the generators, let $W = U^g$. We want to kill the first row of the matrix g in $\mathrm{SL}(d, q')$.

As H is maximal in $\mathrm{SL}(d, q)$ and $H \leq N(U) < \mathrm{SL}(d, q)$, then $H = N(U)$.

Consider the elementary abelian group K, a subgroup of $\mathrm{SL}(d, q)$, generated by those transvections with some power of the primitive element of the ground field in the top row. Now consider K^ϕ. We want to find the element x of K^ϕ that maps W back to U. We will then have $U^{gx} = U$. Hence $gx \in N(U) = H$ and so we have killed the top row of the pre-image of g. We then dualise this process to kill the first column of g.

There already exists an algorithm to provide this x. Written by Ruth Schwingel, it has been dubbed Ruth2, which will be explained later.

Having done this, you then consider how the p-group K^ϕ acts on the reduced g. By forming $K_i^{\phi g}$, where $\{K_i^\phi\}$ are the generators of K^ϕ, and then mapping back to the natural representation, you can discover what each row of the pre-image of g is in the natural representation. I wrote an algorithm to write elements of K^ϕ as words in their generating set in order to map them back to the natural representation and this will be discussed later.

So we can construct a candidate for the pre-image of g. This candidate will in actual fact be a multiple of the pre-image of g as usually, the non-natural representation is isomorphic to $\mathrm{PGL}(d, q)$ or $\mathrm{PSL}(d, q)$. We can find out what the multiple should be by writing the element of g as a word in its generators using the natural representation algorithm, re-evaluating this word in the non-natural representation and dividing the resulting matrix by g. This will give a scalar matrix from which we can read off the first entry to discover the correct multiple.

The word in the generators for the pre-image of g is then recalculated and returned by the algorithm.

The input to this algorithm will be the generators of the image of the standard generators in the non-natural representation. Code is in production to map an arbitrary generating set for the non-natural representation to the image of the standard set.

4.3 Modifying Ruth Schwingel's second algorithm: Ruth2

During the Summer holidays of 2006 and 2007, my task was to modify Ruth2 so that it worked over a prime power field and returned an SLP in the generators of the input p-group. This algorithm will work for any unipotent matrix group.

The idea behind the modification was thus. The algorithm works by canonising an input subspace U in relation to an input p-group K. For the constructive recognition problem, U is the submodule afforded by H and the input p-group K is K^{ϕ}. The canonisation works by considering an ordering on U and the canonical form of U is the minimal element with respect to this ordering under the orbit of K. The ordering is defined by considering the depth of the basis vectors of U; the depth of a basis vector being the position of its first non-zero element reading from left to right.

We now redefine depth to cope with non prime fields. The Depth Plus of a vector consists of a pair. The first entry is the vector's depth in the old sense. If you consider an element of the prime field as a polynomial with the primitive element ω as the indeterminant, then the field can be considered as a vector space over the prime field; the coefficients of the powers of ω being the entries in the vector. For example, in the field $\mathrm{GF}(5^3)$, the element $2 + 4\omega + 3\omega^2$ would map to the vector $(2\ 4\ 3)$. If the depth of the vector v is i, then the second entry in Depths Plus is the depth of the i-th entry of v considered as a vector in this way. For example, the vector over the field of 125 elements $(0\ 0\ \omega\ 1)$ has Depth Plus of [3, 2] as 3 is the depth in the old sense and ω corresponds to the vector $(0\ 1\ 0)$, which has depth 2.

However, it was found that there was still a problem with the original code. The p-group that you are using to canonise the basis vectors of U must generate successive terms a specific chief series throughout the algorithm. This chief series must also retain the property that no two elements of the p-group are of the same "matrix weight".

The weight of an upper unitriangular matrix A is defined as follows. The matrix weight is an ordered triple containing positive integers. The first element of the triple is the number of diagonals above the leading diagonal that you need to move before you get a non-zero entry. The second entry in the triple is how far you need to move down said diagonal from top to bottom before you hit a non-zero entry, say x. Now consider x as a polynomial in $\mathbb{F}_p$ over the primitive element; the third entry in the triple is how far you need to move down the coefficients of x from the constant term upwards before you get a non-zero entry.

Hence, the matrix weight of:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 2 & w & 0 \\ 0 & 1 & 0 & 0 & 3 & 0 & 6 \\ 0 & 0 & 1 & 0 & w & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

is [2, 3, 2].

After each time a generator, say g, of the p-group K is used to kill an entry in a basis vector, say v of U, it is removed from the p-group. The generators k of K that have the property that the depth of $v - vg =$ depth of $v - vk$ are then replaced with kg^α, for some integer α so that each $v - vkg^\alpha$ has greater depth and hence can be used to kill entries further down in v.

Once this process has been done, all entries in the p-group are checked to make sure that no two are of the same matrix weight before the algorithm continues. This ensures that the maximum possible entries in each basis vector of U are always affected and hence U is canonised to the true minimum element under the orbit of K.

4.4 An algorithm to write an element of any unipotent matrix group as a word in its generating set

This algorithm is based on the principals of Ruth2. The input unipotent group K is made upper unitriangular and made into the required chief series. The element Y that you wish to find in terms of the generating set is also made upper unitriangular. You then use the generating set for K to kill entries in the matrix Y by considering elements of K of least matrix weight (as defined above). Once a generator g of K has been used to kill an entry in Y, you use g to perform the same process on the other generators of K as you do with Ruth2, defined above with respect to the matrix weight.

4.5 A word in the generators for other classical groups in their natural representations

My next task was to write a piece of code to find an element of the other classical groups in their natural representations as a word in the generating sets as outlined in the paper Constructive Recognition of Classical Groups in Odd Characteristic by Charles Leedham-Green and Eamonn O'Brien. The generators spoken about below will be as defined in this paper. The cases to consider are: $\mathrm{Sp}(d,q)$, $\mathrm{SU}(2n,q)$, $\mathrm{SU}(2n+1,q)$, $\Omega^{+}(2d,q)$, $\Omega^{-}(2d,q)$ and $\Omega(2d+1,q)$.

The algorithms all work in a similar way to the SL case in the sense that row and column operations are used in order to kill each entry of an arbitrary element of each classical group. Here we outline the differences in each case.

4.5.1 Symplectic Groups

The Weyl group for the symplectic group acts imprimitively on the hyperbolic basis vectors of the underlying space as blocks $\{e_i, f_i\}$. Let A be the matrix that you wish to reduce to the identity.

To kill a row, you add a suitable multiple of one of the rows to the first row as before in order to get a 1 in the upper left hand slot. You then swap the first 2 columns using s. This negates the 1 to a -1. You then add a suitable multiple of the second column to the third using x conjugated by suitable powers of δ. The effects of applying x to kill the third entry on the top row to the first column are, for the time being, ignored.

You then apply usu to A to swap the third and fourth columns and reapply x to kill what is now the third entry on the top row. u and v generate the group $S_{d/2}$ and permute the columns of A without destroying the block structure $\{e_i, f_i\}$. So use u and v to cycle the second to $d/2$-th blocks and now work on the next block. You continue in this way until the top row looks like (* -1 0 ... 0).

Then use s to swap the first two columns to get (1 * 0 ... 0) and use t to kill the remaining place. Once this last place has been killed, you find that you have also killed the second column as the following lemma shows.

Lemma 4.1 *Let the symplectic form of a matrix group be given by the matrix:*

$$J = \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ -1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & -1 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & & \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & 0 & 0 & \dots & -1 & 0 \end{pmatrix}$$

If the top row of a symplectic matrix with respect to this form is $(\,1 \quad 0 \quad 0 \quad \dots \quad 0\,)$, *then the second column of the matrix is* $(\,0 \quad 1 \quad 0 \quad \dots \quad 0\,)$.

Proof: Let a matrix $A \in \mathrm{SL}(2n,q)$ have top row $(\,1 \quad 0 \quad 0 \quad \dots \quad 0\,)$. Then A^{-1} has the same top row as A because A is in the group of matrices that fix the first basis element of the vector space on which it acts and so A^{-1} is of the same form. Hence $(A^{-1})^T$, the transpose of the inverse of A, has $(\,1 \quad 0 \quad 0 \quad \dots \quad 0\,)$ as its first column.

If you multiply A by J on the right, it has the effect of swapping each column in pairs whilst negating the the second column in each pair, i.e. the second column becomes negated and is swapped with the first, the fourth column is negated and is swapped with the third, etc.

$$J^{-1} = \begin{pmatrix} 0 & -1 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & -1 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & & \\ 0 & 0 & 0 & 0 & \dots & 0 & -1 \\ 0 & 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix}$$

Multiplying A by J^{-1} on the left has the same effect on the rows of A. So, the second row becomes negated and is swapped with the first, the fourth row is negated and is swapped with the third, etc. Hence, if you perform A^J, the second row of the resulting matrix will be $(\,0 \quad 1 \quad 0 \quad \dots \quad 0\,)$. Now, if A is a symplectic matrix, $A^J = (A^{-1})^T$. This means that the first column of A^J is $(\,1 \quad 0 \quad 0 \quad \dots \quad 0\,)$, meaning that the second column of A must be $(\,0 \quad 1 \quad 0 \quad \dots \quad 0\,)$.

□

4.5.2 Unitary Groups in even dimension

This works in a similar way to the symplectic case due to the Weyl group preserving the hyperbolic basis structure. One difference, however, is how you kill the [1, 2] entry of the arbitrary element A once you have killed the rest of the top row. We do not have an element that enables you to kill every field element as t has $\alpha = w^{(q+1)/2}$ in its [1, 2] position.

However, the subfield of GF(q^2) generated by α, is still GF(q^2). Hence, you can find the [1, 2] entry of A as a polynomial in α and use conjugates of t by powers of y to kill said entry.

Conjecture: At this point of the algorithm, the [1, 2] entry of A is a sum of odd powers of α. Conjugates of t by powers of y only enable you to kill odd powers of α.

4.5.3 Unitary Groups in odd dimension

With respect to a hyperbolic basis, $\mathrm{SU}(2d, q)$ sits as a subgroup of $\mathrm{SU}(2d+1, q)$ in the top left hand corner of the matrices, leaving one basis vector at the bottom. See Don Taylor's The Geometry of Classical Groups for details.

The algorithm needs to find the generators of $\mathrm{SU}(2d, q)$ as words in the generators of $\mathrm{SU}(2d+1, q)$. You then use the generators of $\mathrm{SU}(2d, q)$ to kill the top $d \times d$ left block of A in $\mathrm{SU}(2d+1, q)$ and then use conjugates of x with powers of y to kill the remaining basis vector.

4.5.4 Omega Plus

From now on, we assume that we are working over a field of odd characteristic.

This works almost exactly the same as the symplectic group case. The only difference is that once you've killed the third to d-th entries on the top row, the second entry is automatically killed (conjecture).

4.5.5 Omega Minus

This case is markedly different from all the other cases. With respect to a hyperbolic basis, $\Omega^{+}(2d-2,q)$ sits as a subgroup of $\Omega^{-}(2d,q)$ in the top left hand corner of the matrices, leaving two basis vectors at the bottom. See Don Taylor's The Geometry of Classical Groups for details. So, in a similar way to the Unitary case in odd dimension, we find the generators of $\Omega^{+}(2d-2,q)$ in terms of the generators of $\Omega^{-}(2d,q)$ and kill the upper $(2d-4)\times(2d-4)$ block.

At present, the algorithm then uses an application of Ruth2 to kill the remaining two entries on the top row. It considers the remaining portion as $\mathrm{SL}(2,q^2)\otimes\mathrm{SL}(2,q^2)$ in a non-natural representation (isomorphic to $\Omega^{-}(4,q)$). At present, am I looking at ways to do this using first principals as Ruth2 can be potentially expensive.

You are then left with a 4×4 block sandwiched in the bottom right hand corner of the matrix A. This is an element of $\Omega^{-}(4,q)$ and hence is isomorphic to an element $\mathrm{SL}(2,q^2)\otimes SL(2,q^2)$. You find said element by first principals. Kill the matrix in $\mathrm{SL}(2,q^2)$ and then map the information back to $\Omega^{-}(4,q)$ to finish the job off.

4.5.6 Omega Circle

This has not yet been done. I expect that this will work in a similar way to Unitary in the odd dimension.

4.5.7 Summary

So far, I have produced algorithms to solve this problem for $\mathrm{Sp}(d,q)$, $\mathrm{SU}(2n,q)$, $\mathrm{SU}(2n+1,q)$, $\Omega^{+}(2d,q)$ and $\Omega^{-}(2d,q)$ and am yet to produce one for $\Omega(2d+1,q)$.

4.6 A word in the generators for other classical groups in their non-natural representations

4.6.1 Symplectic Groups

This works in pretty much the same way as the SL case. Differences include H being the subgroup that stabilises the space spanned by the first two basis vectors of the underlying vector space and the p-group being made up of t plus conjugates of x by elements of the Weyl group.

Towards the end of the algorithm, it is necessary to check that your candidate for the pre-image of your arbitary element A in the natural representation is an element of $\mathrm{Sp}(d,q)$. You do this by dividing the pre-image by a suitable field element to get the determinant 1 and then dividing again to make sure the equation $A^TJA = J$ holds, where J is the matrix representing the symplectic form. If the equation doesn't hold, then A^TJA will be a multiple of J and so you divide appropriately to get the equation to hold.

4.6.2 Omega Plus

This is almost exactly the same as the Symplectic case. H is a subgroup that stabilises the space spanned by the first two basis vectors. K is a p-group generated by certain conjugates of t with δ and the Weyl group.

When it comes to checking whether the candidate for the pre-image of the element A is in Omega Plus, you need to perform an additional check to the symplectic case. We need to make sure that the element we have is in $\Omega^+(2d, q)$ and not in the other coset of $\mathrm{SO}(2d, q)$. We do this by dividing by -1 if necessary.

4.6.3 Unitary in even dimension

This is very similar to the other cases but with a few twists.

4.6.4 Others

$\mathrm{SU}(2n+1, q)$, $\Omega^-(2d, q)$ and $\Omega^+(2d+1, q)$ are yet to be done. All the algorithms outlined here also need to be tidied up considerably and made to run as fast as possible.

4.7 Future Questions

I have found that if you perform on of the algorithms for the natural representation on a matrix that doesn't preserve the required form, then the input matrix is not reduced to the identity. Could this be used as a membership test?

What is the complexity of these algorithms?

What is the length of the straight line programmes? It is thought to be $d^2 \log q$.

Can similar algorithms be produced for other groups? For example the exceptional groups of Lie type.

Can an algorithm be found to write an element of $\mathrm{PSX}(d, q)$ as an element of its generators by considering how the generators act on the projective points?