

Implementing the
Burnside-Dixon Algorithm
in Gap

Richard Thomas Bayley
Queen Mary University of London

May 20, 2006

Contents

1		2
1.1	Introduction	2
1.2	Burnside	2
1.2.1	Burnside's Algorithm with S_4	4
1.3	Dixon	6
1.3.1	Burnside-Dixon Algorithm	6
1.3.2	Dixon's algorithm with A_4	7
1.3.3	Recovering Complex Characters	9
1.3.4	Dixon's Algorithm for D_4	10
A	Gap Code	13

Chapter 1

1.1 Introduction

In this chapter we look at how to compute the character tables of a finite group G using an algorithm of Burnside's [1] with some refinements given by Dixon [3] and Schneider [6]. We look at how these algorithms can be implemented in Gap [5] by concentrating on some given gap commands and by writing some of our own. These algorithms have been implemented in Gap already and the character table of a group may be found by using commands such `Display(CharacterTable(G))` where G is a group. We do not use this command though or similar ones as our aim is to gain a deeper understanding of the algorithm. We assume that G has conjugacy classes K_1, K_2, \dots, K_k with orders h_1, h_2, \dots, h_k respectively and that the irreducible characters of G are $\text{Irr}(G) = \{\chi_1, \dots, \chi_k\}$ with degrees d_1, \dots, d_k . It is quite reasonable to also assume that $K_1 = \{1\}$ and χ_1 is the trivial representation 1_G . Also Let $K_{j'} = K_j^{-1}$ for each conjugacy class. K_j^{-1} is just the conjugacy class obtained by taking the inverse of all the elements in K_j . These two classes are in fact equal for the Symmetric group.

1.2 Burnside

Burnside [1] gives us a systematic way of calculating the irreducible characters of a finite group. We start though with a few definitions.

Definition 1. Class Coefficients

Let $c_{rst}(r, s, t = 1, 2, \dots, k)$ be the number of solutions (x, y) to $xy = z$ with $x \in K_r$, $y \in K_s$, for a given $y \in K_t$. This number is independent of which z we choose from the conjugacy class. We take these class coefficients and form $k \times k$ matrices M_r with the (s, t) -th entry being c_{rst} .

Example 1. Here we consider the Symmetric group on four letters S_4 . This group has five conjugacy classes which can be found using the command `HConjClass(G, G)` where `G:=SymmetricGroup(4)`. Alternatively you can use the gap command `K:=ConjugacyClasses(G)` and then find the elements using `Elements(K[i])` for $1 \leq i \leq k$.

$$\begin{aligned}
 K_1 &= \{()\}, & K_2 &= \{(3, 4), (2, 4), (2, 3), (1, 4), (1, 3), (1, 2)\}, \\
 K_3 &= \{(2, 3, 4), (2, 4, 3), (1, 3, 4), (1, 4, 3), (1, 2, 4), (1, 4, 2), (1, 2, 3), (1, 3, 2)\}, \\
 K_4 &= \{(1, 2)(3, 4), (1, 3)(2, 4), (1, 4)(2, 3)\}, \\
 K_5 &= \{(1, 2, 3, 4), (1, 2, 4, 3), (1, 3, 2, 4), (1, 3, 4, 2), (1, 4, 2, 3), (1, 4, 3, 2)\}
 \end{aligned} \tag{1.1}$$

By using the command `ClassCoeffs(K)` given in Appendix A we can compute the class coefficient matrices straight off, where K is the set of conjugacy classes.

$$M_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad M_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 6 & 0 & 3 & 2 & 0 \\ 0 & 4 & 0 & 0 & 4 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 3 & 4 & 0 \end{pmatrix} \quad M_3 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 4 \\ 8 & 0 & 4 & 8 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 4 & 0 & 0 & 4 \end{pmatrix}$$

$$M_4 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 3 & 0 & 0 & 2 & 0 \\ 0 & 2 & 0 & 0 & 1 \end{pmatrix} \quad M_5 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 3 & 4 & 0 \\ 0 & 4 & 0 & 0 & 4 \\ 0 & 2 & 0 & 0 & 1 \\ 6 & 0 & 3 & 2 & 0 \end{pmatrix}$$

Note that the first matrix M_1 is always the identity.

Definition 2.

$$\omega_{ij} = \frac{h_j \chi_i(K_j)}{\chi_i(K_1)}$$

Lemma 1. *If $1 \leq i \leq k$ then*

$$\sum_r \frac{\omega_{ir} \omega_{ir'}}{h_r} = \frac{|G|}{\chi^2(K_1)}$$

Lemma 2.

$$\chi_i(K_j) = \frac{\omega_{ij} \chi_i(K_1)}{h_j}$$

We won't actually need to use all of these matrices in Burnside's algorithm. What we are going to do is find a set of linearly independent eigenvectors with the first entry normalised to one for all of these matrices which will give us each ω_{ij} . The ij th entry of the eigenvector v_i is ω_{ij} and then we use these to simply calculate the character table. This is done by using Lemma 1 to calculate the degrees of the representations and then we can use Lemma 2 to calculate the other characters. The major difficulty here lies in calculating the k linearly independent eigenvectors. If one of the matrices M_r has a set of k distinct eigenvalues then the eigenvectors for this matrix will be eigenvectors for all the matrices. Otherwise we have to do some more work to find these eigenvectors which involves finding a set of one dimensional eigenspaces. We will describe this situation with the Dihedral group D_4 in Dixon's algorithm later as the method is similar.

1.2.1 Burnside's Algorithm with S_4

Consider the matrix M_2 given in Example 1. This matrix has distinct eigenvalues $\{6, 2, 0, -2, -6\}$. These can be obtained by using the command `Eigenvalues(Rationals,M[2])`. Note that we use the command `Eigenvectors(Rationals,TransposedMat(M[2]))` to find the eigenvectors. We use `TransposedMat` to ensure that we compute right eigenvectors and not left ones as Gap is set up to

do. The eigenvectors that we find are

$$\begin{aligned}
v_1 &= (\omega_{11}, \omega_{12}, \omega_{13}, \omega_{14}, \omega_{15}) = (1, 6, 8, 3, 6) & \lambda = 6 \\
v_2 &= (\omega_{21}, \omega_{22}, \omega_{23}, \omega_{24}, \omega_{25}) = (1, 2, 0, -1, -2) & \lambda = 2 \\
v_3 &= (\omega_{31}, \omega_{32}, \omega_{33}, \omega_{34}, \omega_{35}) = (1, 0, -4, 3, 0) & \lambda = 0 \\
v_4 &= (\omega_{41}, \omega_{42}, \omega_{43}, \omega_{44}, \omega_{45}) = (1, -2, 0, -1, 2) & \lambda = -2 \\
v_5 &= (\omega_{51}, \omega_{52}, \omega_{53}, \omega_{54}, \omega_{55}) = (1, -6, 8, 3, -6) & \lambda = -6
\end{aligned}$$

Now we use the ω_{ij} to calculate the degrees of the irreducible representations of the group using Lemma 2.

$$\begin{aligned}
\sum_r \frac{\omega_{1r}\omega_{1r'}}{h_r} &= \frac{1 \cdot 1}{1} + \frac{6 \cdot 6}{6} + \frac{8 \cdot 8}{8} + \frac{3 \cdot 3}{3} + \frac{6 \cdot 6}{6} = 24 \Rightarrow \chi_1^2(K_1) = \frac{24}{24} \Rightarrow \chi_1(K_1) = d_1 = 1 \\
\sum_r \frac{\omega_{2r}\omega_{2r'}}{h_r} &= \frac{1 \cdot 1}{1} + \frac{2 \cdot 2}{6} + \frac{0 \cdot 0}{8} + \frac{-1 \cdot -1}{3} + \frac{-2 \cdot -2}{6} = \frac{8}{3} \Rightarrow \chi_2^2(K_1) = \frac{24 \cdot 3}{8} \Rightarrow \chi_2(K_1) = d_2 = 3 \\
\sum_r \frac{\omega_{3r}\omega_{3r'}}{h_r} &= \frac{1 \cdot 1}{1} + \frac{0 \cdot 0}{6} + \frac{-4 \cdot -4}{8} + \frac{3 \cdot 3}{3} + \frac{0 \cdot 0}{6} = 6 \Rightarrow \chi_3^2(K_1) = \frac{24}{6} \Rightarrow \chi_3(K_1) = d_3 = 2 \\
\sum_r \frac{\omega_{4r}\omega_{4r'}}{h_r} &= \frac{1 \cdot 1}{1} + \frac{-2 \cdot -2}{6} + \frac{0 \cdot 0}{8} + \frac{-1 \cdot -1}{3} + \frac{2 \cdot 2}{6} = \frac{8}{3} \Rightarrow \chi_4^2(K_1) = \frac{24 \cdot 3}{8} \Rightarrow \chi_4(K_1) = d_4 = 3 \\
\sum_r \frac{\omega_{5r}\omega_{5r'}}{h_r} &= \frac{1 \cdot 1}{1} + \frac{-6 \cdot -6}{6} + \frac{8 \cdot 8}{8} + \frac{3 \cdot 3}{3} + \frac{-6 \cdot -6}{6} = 24 \Rightarrow \chi_5^2(K_1) = \frac{24}{24} \Rightarrow \chi_5(K_1) = d_5 = 1
\end{aligned}$$

Now that we have all of the degrees we can use Lemma 2 to find all of the other characters.

$$\begin{aligned}
\chi_1(K_2) &= \frac{\omega_{12}d_1}{h_2} = \frac{6 \cdot 1}{6} = 1 & \chi_1(K_3) &= \frac{\omega_{13}d_1}{h_3} = \frac{8 \cdot 1}{8} = 1 \\
\chi_1(K_4) &= \frac{\omega_{14}d_1}{h_4} = \frac{3 \cdot 1}{3} = 1 & \chi_1(K_5) &= \frac{\omega_{15}d_1}{h_5} = \frac{6 \cdot 1}{6} = 1 \\
\chi_2(K_2) &= \frac{\omega_{22}d_2}{h_2} = \frac{2 \cdot 3}{6} = 1 & \chi_2(K_3) &= \frac{\omega_{23}d_2}{h_3} = \frac{0 \cdot 3}{8} = 0 \\
\chi_2(K_4) &= \frac{\omega_{24}d_2}{h_4} = \frac{-1 \cdot 3}{3} = -1 & \chi_2(K_5) &= \frac{\omega_{25}d_2}{h_5} = \frac{-2 \cdot 3}{6} = -1 \\
\chi_3(K_2) &= \frac{\omega_{32}d_3}{h_2} = \frac{0 \cdot 2}{6} = 0 & \chi_3(K_3) &= \frac{\omega_{33}d_3}{h_3} = \frac{-4 \cdot 2}{8} = -1 \\
\chi_3(K_4) &= \frac{\omega_{34}d_3}{h_4} = \frac{3 \cdot 2}{3} = 2 & \chi_3(K_5) &= \frac{\omega_{35}d_3}{h_5} = \frac{0 \cdot 2}{6} = 0 \\
\chi_4(K_2) &= \frac{\omega_{42}d_4}{h_2} = \frac{-2 \cdot 3}{6} = -1 & \chi_4(K_3) &= \frac{\omega_{43}d_4}{h_3} = \frac{0 \cdot 3}{8} = 0 \\
\chi_4(K_4) &= \frac{\omega_{44}d_4}{h_4} = \frac{-1 \cdot 3}{3} = -1 & \chi_4(K_5) &= \frac{\omega_{45}d_4}{h_5} = \frac{2 \cdot 3}{6} = 1 \\
\chi_5(K_2) &= \frac{\omega_{52}d_5}{h_2} = \frac{-6 \cdot 1}{6} = -1 & \chi_5(K_3) &= \frac{\omega_{53}d_5}{h_3} = \frac{8 \cdot 1}{8} = 1 \\
\chi_5(K_4) &= \frac{\omega_{54}d_4}{h_4} = \frac{3 \cdot 1}{3} = 1 & \chi_5(K_5) &= \frac{\omega_{55}d_5}{h_5} = \frac{-6 \cdot 1}{6} = -1
\end{aligned}$$

To do all of this calculation we can use the function `ComputeDegs(W,G,K)` given in Appendix A where W is the matrix of eigenvectors, G is the group and K is the set of conjugacy classes found

using `HConjClass(G,G)`. The character table of S_4 can now be given below.

S_4	K_1	K_2	K_3	K_4	K_5
χ_1	1	1	1	1	1
χ_2	3	1	0	-1	-1
χ_3	2	0	-1	2	0
χ_4	3	-1	0	-1	1
χ_5	1	-1	1	1	-1

1.3 Dixon

Computing eigenvectors for matrices can be done relatively easily by many packages such as Maple, Matlab and Mathematica which have inbuilt routines for computing them over \mathbb{C} . There are often slight rounding errors though as computers are not able to compute in infinite precision. In practical situations in applied mathematics and engineering there are methods to eliminate this by partial pivoting etc but not to totally eradicate them. Gap doesn't compute eigenvectors over \mathbb{C} as it will not approximate a number by rounding. To compute characters we often need the calculations to be done symbolically rather than numerically and as stated by Dixon in [1] most of the time that we use character tables we need the entries to be in exact form as algebraic integers. The essence behind Dixon's algorithm is that of taking the problem from the field of complex numbers to that of the field \mathbb{Z}_p , the field of integers modulo p for some suitable prime p . This removes any rounding issues and often allows the calculations to be done much quicker.

1.3.1 Burnside-Dixon Algorithm

Let m be the exponent of G , the least common multiple of all the elements of G . This can be found using the Gap command `Exponent(G)`; . If $x \in G$ and $\chi_i \in \text{Irr}(G)$ then $\chi_i(x)$ is a sum of complex m th roots of unity. Therefore if ζ is a complex m th root of unity, found by command `E(m)`; , then the characters of G all lie in $\mathbb{Z}[\zeta] = \{f(\zeta) | f(x) \in \mathbb{Z}[x]\}$, the ring of polynomials in ζ with integer

coefficients. Now we need to choose the prime p over which we will do our computations. Due to a theorem by Dirichlet on primes in an arithmetic progression there is a prime p such that m divides $p-1$, and then we can find an integer z such that $z^m \equiv 1 \pmod{p}$ and $z^f \not\equiv 1$ for all $f, 0 < f < m$. We will look at the rest of Dixon's theory for this later when it is used more extensively in converting the characters found in \mathbb{Z}_p to characters in \mathbb{C} .

1.3.2 Dixon's algorithm with A_4

Consider the alternating group of degree four. Its conjugacy classes and class coefficient matrices are given below .

$$\begin{array}{ll}
 K_1 = \{()\} & K_2 = \{(2, 3, 4), (1, 4, 3), (1, 2, 4), (1, 3, 2)\} \\
 K_3 = \{(2, 4, 3), (1, 3, 4), (1, 4, 2), (1, 2, 3)\} & K_4 = \{(1, 2)(3, 4), (1, 3)(2, 4), (1, 4)(2, 3)\} \\
 M_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} & M_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 4 & 0 & 0 & 4 \\ 0 & 3 & 0 & 0 \end{pmatrix} \\
 M_3 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 4 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 3 & 0 \end{pmatrix} & M_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 3 & 0 & 0 & 2 \end{pmatrix}
 \end{array}$$

We now choose a suitable prime, here it turns out that $p = 7$ and thus we will do all of our work in \mathbb{Z}_7 from now on. It is quite easy to work in \mathbb{Z}_p in gap though the notation takes a little time to get used to. To distinguish between integers and elements of \mathbb{Z}_p Gap uses Zech logarithms. It is best shown by an example so we will convert all the elements of \mathbb{Z}_7 into the way gap sees them by using Zech logarithms. To do this we just multiply the integers by the identity element $Z(p)^0$; and so $\{0,1,2,3,4,5,6\} = \{0*Z(7), Z(7)^0, Z(7)^2, Z(7), Z(7)^4, Z(7)^5, Z(7)^3\}$. Notice here that 3 is the generator in Z_7 . We can use the gap command `Inverse` to find the inverse of an element and then the command `Int` to convert from the Zech logarithms to integers. For example

$\text{Inverse}(Z(7)^4) = Z(7)^2$. $\text{Int}(Z(7)^2) = 2$ and $4 \cdot 2 = 8 = 1 \pmod{7}$. To change all the entries in a matrix from integers to Zech logarithms we can use $M := M * \text{One}(\text{GF}(p))$ where GF stands for Galois Field. To change them back we can use the function $Z\text{ToMat}(M, p)$. Once our matrices are in the right form we need to calculate the eigenvalues over \mathbb{Z}_p , this can be done by using the gap command $\text{Eigenvalues}(\text{GF}(p), M)$ and we find that M_2 has distinct eigenvalues $\{0, 4, 1, 2\}$. Note here that Gap doesn't solve the characteristic polynomial for this matrix M_2 but tries a few values of $\{1, 2, \dots, p\}$ and finds which are solutions by just seeing if they are roots. As you can see it is a lot easier to find roots \mathbb{Z}_p than in \mathbb{C} . As M_2 has distinct eigenvalues its eigenvectors will be linearly independent eigenvectors for all of our matrices. We are fortunate that in A_4 there is a matrix that has distinct eigenvalues. This is not always the case and we shall use D_4 again in the next section to illustrate what happens if there are not distinct eigenvalues for one of the class coefficient matrices. The eigenvector computation is done by using Gaussian elimination in \mathbb{Z}_p to find the null space of the associated matrix. To compute the eigenvectors we do much the same but remembering to transpose the matrix. By using $\text{eig} := \text{Eigenvectors}(\text{GF}(p), \text{TransposedMat}(M))$ and we find that the eigenvectors are

$$\begin{aligned}
 v_1 &= [1, 0, 0, 6] & v_2 &= [1, 4, 4, 3] \\
 v_3 &= [1, 1, 2, 3] & v_4 &= [1, 2, 1, 3]
 \end{aligned}$$

As before with Burnside's algorithm we are able to compute the degrees of the irreducible representations by Lemma 1 but this time making sure that everything is done in \mathbb{Z}_p . It is worth remembering though that $K_{j'} \neq K_j$ in this group. To find the inverse conjugacy classes we can run the function $\text{InvConjdash}(K)$ given in Appendix A. Given a full set of conjugacy classes this function will return a list of integers which are the j' . For example with A_4 the set of integers is $[1, 3, 2, 4]$. Lemma 1 will give use the degrees d_i^2 and we can use the Gap function $\text{RootsMod}(d_i^2, p)$ to tell us the two roots. We choose the one less than $\frac{p}{2}$ as $p > 2 \cdot |G|^{\frac{1}{2}}$. This can all be done with the function $\text{ComputeDegDixon}(\text{eig}, p, G, K)$ where eig is the set of linearly independent eigenvectors in Zech logarithm form. This function returns a list of degrees for the representations and for A_4 we get that the degrees are $[3, 1, 1, 1]$. An easy check to see if these

degrees are right is to see if the sum of the squares adds to the order of our group, rest assured $3^2 + 1^2 + 1^2 + 1^2 = 12 = |A_4|$. To find the characters of A_4 in \mathbb{Z}_p we simply use Lemma 2 like we did in Burnside, the difference being that this time all calculations are done in \mathbb{Z}_p . This can be done by using the function `ComputeCharDixZp(deg,eig, p, G, K)` given in Appendix A where `deg` is the list of degrees found above. Computing these characters in \mathbb{Z}_p is relatively stress-free; the answer for A_4 is given below

A_4	K_1	K_2	K_3	K_4
χ_1	3	0	0	6
χ_2	1	1	1	1
χ_3	1	2	4	1
χ_4	1	4	2	1

With a little judicious reordering we can swap χ_1 and say χ_2 so that we get the trivial character in the first place but apart from that we have a complete character table in \mathbb{Z}_7 for A_4 .

1.3.3 Recovering Complex Characters

The tricky part with Dixon's algorithm comes with finding the complex characters. The proofs behind much of the following can be found in [3] and [4] what we shall describe here is how to get all the little pieces of information which are required to actually compute the characters. First of all we must find a $z \in G$ such that the order of z is equal to e the exponent of G . Before we defined what K'_j meant when we look at conjugacy classes, we shall now define what $K_{j(n)}$ means.

Definition 3. The conjugacy class $K_{j(n)}$ is the class obtained by taking all the elements of $K_j = \{g_1, g_2, \dots, g_{h_j}\}$ and putting them to the power of n so that $K_{j(n)} = \{g_1^n, g_2^n, \dots, g_{h_j}^n\}$. We ensure that we ignore any repeats.

To find these $K_{j(n)}$ we can use the function `PowerConjClassDash(K,n)` given in Appendix A. For example for A_4 we get that `PowerConjClassDash(K,2)` returns the list `[1,3,2,1]` and these are the $j(n)$ for $n = 2$. Next we need to calculate integers μ_{ijs} such that $0 \leq \mu_{ijs} < p$ which is easily done

with the following sum.

$$\mu_{ijs} \equiv \frac{1}{m} \sum_{n=0}^{m-1} \chi_i(K_{j(n)}) z^{-sn} \pmod{p} \quad (1.2)$$

where $i, j = 1, 2, \dots, k; s = 0, 1, 2, \dots, m-1$

and the χ_i are the characters in Z_p . Next we compute the following sum;

$$\chi_i(K_j) = \sum_{s=0}^{m-1} \mu_{ijs} z^s \quad (i, j = 1, 2, \dots, k) \quad (1.3)$$

Here the χ_i are the complex characters that we are looking for. These equations 1.2 and 1.3 are both incorporated into the function `ComputeCharComplex(char, deg, p, G, K, m)` given in Appendix A where `char` is a matrix of the characters found using `ComplexCharDixZp`, therefore `char[i][j] : = $\chi_i(K_j)$` in \mathbb{Z}_p . We find that for A_4 we get the character table given below where the trivial character has been swapped with χ_2 to ensure it is in the first position.

A_4	K_1	K_2	K_3	K_4
χ_1	1	1	1	1
χ_2	3	0	0	-1
χ_3	1	$E(3)$	$E(3)^2$	1
χ_4	1	$E(3)^2$	$E(3)$	1

Where $E(3) = e^{\frac{2\pi i}{3}}$ a primitive third root of unity. The whole process from beginning to end is done by using the command `DixonAlg(G)` and this will compute the character table for any finite group so long as it is not too large. The reason that it was comparatively easy to compute the characters for A_4 is that there is a class coefficient matrix which has distinct eigenvalues but this is not always the case. Here we consider D_4 which doesn't have any class coefficient matrices with distinct eigenvalues.

1.3.4 Dixon's Algorithm for D_4

D_4 may be summarized by the following presentation

$$D_4 = \langle f_1, f_2 : f_1^2 = f_2^2 = 1, f_2 * f_1 * f_2 = f_1 \rangle$$

This group has conjugacy classes

$$K_1 = \{1\} \quad K_2 = \{f_1\} \quad K_3 = \{f_2\} \quad K_4 = \{f_1 * f_2\}$$

The class coefficient matrices are

$$M_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad M_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad M_3 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad M_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad (1.4)$$

Each of these matrices except for M_1 has two eigenvalues $\{1, 4\}$. We can find the eigenvectors associated to each eigenvalue by using the command `NullspaceMat(M)`. Where M is the matrix obtained by subtracting an eigenvalue off all the diagonal elements. For example we chose M_2 and find that it has eigenvectors $\{[4, 1, 0, 0], [0, 0, 4, 1]\}$ associated to $\lambda = 4$ but these need to be normalised to have first entry 1 so we get $\{[1, 4, 0, 0], [0, 0, 1, 4]\}$ associated to the eigenvalue 4 and eigenvectors $\{[1, 1, 0, 0], [0, 0, 1, 1]\}$ associated to the eigenvalue 1. Let V_1 and V_2 stand for each of these eigenspaces respectively. We are going to act on these eigenspaces with the other coefficient matrices M_i and use this action to find our eigenvectors. Let $V_1 = \{v_{11}, v_{12}\} = \{[4, 1, 0, 0], [0, 0, 4, 1]\}$. Then $M_3 \times v_{11} = [0, 0, 1, 4] = 1 \times v_{12}$ and $M_3 \times v_{12} = [4, 1, 0, 0] = 1 \times v_{11}$. We then use these coefficients to form a smaller split matrix S which here is

$$S = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (1.5)$$

This matrix has distinct eigenvalues and so we can use it to find our eigenvectors. The eigenvectors of S are $[1, 4]$ and $[1, 1]$. We can now use the individual values of these eigenvectors to act as coefficients of the vectors in V_1 as follows. Let our first vector be e_1 and we compute like this;

$e_1 = 1 \times v_{11} + 4 \times v_{12} = [1,4,4,16] = [1,4,4,1] \pmod{5}$. Similarly we get $e_2 = [1, 4, 1, 4]$. If we let M_3 act on the eigenspace V_2 we can find some more vectors and we get that $e_3 = [1, 1, 4, 4]$ and $e_4 = [1, 1, 1, 1]$ and so we have found a set of linearly independent vectors which are eigenvectors for all of our matrices M_i . It is worth noting here that the split matrix S might not always have distinct eigenvalues. If this is the case then we ignore it and form other split matrices. In our case we acted on the eigenspaces of M_2 with M_3 and found that this was sufficient to find all our eigenvectors, this does not always happen and we may have to act on the eigenspaces of M_2 with some other M_i . Even when this is done we may not get all of our eigenvectors. We may have to act on another few M_i with other M_i to finally get all of our eigenvectors. We should note that M_1 doesn't help in the splitting and it is worthless to act on the eigenspaces of M_i with M_i . With our full set of eigenvectors we can calculate the character table just as before. The command to do all this splitting is `EigenSplit(M, p)` where p is the prime from section 1.3.1 and M is the list of class coefficient matrices with entries as Zech Logaritms . For D_4 we find that $p = 5$ and we find that for D_4 we get the character table given below which just as before can be found using the command `DixonAlg(G)`

D_4	K_1	K_2	K_3	K_4
χ_1	1	1	1	1
χ_2	1	1	-1	-1
χ_3	1	-1	1	-1
χ_4	1	-1	-1	1

Appendix A

Gap Code

www.maths.qmul.ac.uk/~rtb

Bibliography

- [1] Burnside, *Theory of Groups*, 2nd ed. (1911), Dover, New York, 1955.
- [2] S. Donkin, *Some Relative Character Theory*, Journal of Pure and Applied Algebra 97. (1994), 281 - 301.
- [3] J. D. Dixon, *High Speed Computation of Group Characters*, Num Math. 10 (1967), 446-450.
- [4] L. C. Grove, *Groups and Characters*, Wiley-Interscience, New York, 1997.
- [5] The GAP Group, *GAP - Groups, Algorithms, and Programming, Version 4.4*; 2004, (<http://www.gap-system.org>).
- [6] G. J. A. Schneider, *Dixon's Character Table Algorithm Revisited*, J. Symb. Comp. 9 (1990), 601-606.