Constructing the Irreducible Characters of J_4 with GAP

Thomas Breuer

Lehrstuhl D für Mathematik
RWTH, 52056 Aachen, Germany

July 13th, 1999

We assume that the "table head" of the ordinary character table of the sporadic simple Janko group J_4 is known. That is, the conjugacy classes and their centralizer orders are known as well as all p-th power maps, for primes p up to the maximal element order in J_4, which is 66. For the sake of simplicity, we take the table head of the character table contained in the GAP Character Table Library [Bre04].

```
gap> LoadPackage( "ctbllib" );
true
gap> tbl:= CharacterTable( "J4" );
CharacterTable( "J4" )
gap> NrConjugacyClasses( tbl );
62
```

Besides the table head with the trivial character, we assume that we know a nontrivial irreducible character of smallest degree –which is 1 333– and the nontrivial permutation character of smallest degree; the latter character corresponds to the action on the cosets of a maximal subgroup of type $2^{11}.M_{24}$ in J_4.

```
gap> irreducibles:= Irr( tbl ){ [ 1, 2 ] };
[ Character( CharacterTable( "J4" ), [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
      1 ] ), Character( CharacterTable( "J4" ),
    [ 1333, 53, -11, 10, -11, 5, -3, 3, -10, 2, -2, E(7)+E(7)^2+E(7)^4,
      E(7)^3+E(7)^5+E(7)^6, 1, -3, 1, 3, -1, 2, 2, -2, 2, 0,
      -E(7)-E(7)^2-E(7)^4, -E(7)^3-E(7)^5-E(7)^6, E(7)+E(7)^2+E(7)^4,
      E(7)^3+E(7)^5+E(7)^6, 0, -1, -1, -1, E(7)+E(7)^2+E(7)^4,
      E(7)^3+E(7)^5+E(7)^6, -2, 0, -1, 0, 0, -E(7)-E(7)^2-E(7)^4,
      -E(7)^3-E(7)^5-E(7)^6, -1, 0, 0, 0, 0, -1, -1, E(7)+E(7)^2+E(7)^4,
      E(7)^3+E(7)^5+E(7)^6, 1, 1, 1, 1, 1, -E(7)-E(7)^2-E(7)^4,
      -E(7)^3-E(7)^5-E(7)^6, 0, 0, 0, 0, 1, 1 ] ) ]
gap> max:= CharacterTable( Maxes( tbl )[1] );;
gap> pi:= TrivialCharacter( max ) ^ tbl;
Character( CharacterTable( "J4" ), [ 173067389, 52349, 8317, 737, 957, 253,
  141, 14, 77, 41, 37, 5, 5, 5, 13, 13, 14, 2, 0, 11, 9, 13, 3, 3, 3, 1, 1,
  2, 1, 2, 2, 2, 2, 0, 1, 2, 1, 1, 1, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ] )
```

The first new irreducible character is of course the complex conjugate of the degree 1 333 character shown above.

```
gap> AddSet( irreducibles, ComplexConjugate( irreducibles[2] ) );
```

Now our strategy is to create characters that are in general reducible, and to compute characters and virtual characters of smaller norm from them until eventually the list of irreducibles is complete.

We start with the characters obtained by induction from cyclic subgroups of J_4, and symmetrizations of the known irreducibles. Note that these two computations are possible only because we know the power maps of J_4.

```
gap> indcyc:= InducedCyclic( tbl, "all" );;
gap> sym2:= Symmetrizations( tbl, irreducibles, 2 );;
gap> sym3:= Symmetrizations( tbl, irreducibles, 3 );;
```

Before we start to work with characters, we raise the info level for computations of this kind. This will for example cause that a message is printed whenever a new irreducible character has been found.

```
gap> SetInfoLevel( InfoCharacterTable, 2 );
```

First we reduce the characters with the known irreducibles, that is, we project them into the orthogonal space of the vector space that is spanned by the known irreducibles.

```
gap> chars:= Concatenation( indcyc, [ pi ], sym2, sym3 );;
gap> Length( chars );
220
gap> chars:= ReducedCharacters( tbl, irreducibles, chars );;
#I  ReducedCharacters: irreducible character of degree 887778 found
#I  ReducedCharacters: irreducible character of degree 889111 found
#I  ReducedCharacters: irreducible character of degree 887778 found
#I  ReducedCharacters: irreducible character of degree 393877506 found
#I  ReducedCharacters: irreducible character of degree 789530568 found
gap> Length( chars.irreducibles );
5
gap> Length( chars.remainders );
206
```

We found five new irreducibles. For later use, we store those irreducibles for which no symmetrizations were used yet.

```
gap> newirr:= chars.irreducibles;;
```

In order to find out the dimension of the $\mathbb{Z}$-lattice spanned by the remaining reducible characters, we compute a lattice basis using the LLL algorithm.

```
gap> lll:= LLL( tbl, chars.remainders );;
#I  LLL: 4 irreducibles found
gap> List( lll.irreducibles, Degree );
[ 1981808640, 1981808640, 1981808640, 2267824128 ]
```

We were lucky, five new irreducibles were obtained as elements of the lattice basis; in order to work in their orthogonal space from now on, we reduce `chars` with them.

```
gap> Append( newirr, lll.irreducibles );
gap> chars:= ReducedCharacters( tbl, lll.irreducibles, chars.remainders );;
```

This yields no new irreducibles. Now let us look at the reducible vectors in the lattice basis computed by `LLL`.

```
gap> Length( lll.remainders );
50
gap> lll.norms;
[ 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 4, 2, 5, 4, 5, 4, 4, 5, 3,
  6, 6, 8, 4, 6, 6, 4, 8, 8, 7, 9, 7, 6, 7, 7, 8, 6, 9, 7, 7, 4, 6, 7, 8, 5 ]
```

Together with the 13 known irreducibles, the basis of length 49 spans the whole 62 dimensional space of irreducibles of J_4. The norms of the virtual characters in `lll.remainders` are listed in `lll.norms`, in our case they are very small. Since `LLL` does not reduce the virtual characters in the `remainders` list with the characters in the `irreducibles` list, we do this reduction now.

```
gap> lll:= ReducedClassFunctions( tbl, lll.irreducibles, lll.remainders );;
```

We use the new irreducibles to repeat the process of generating and reducing characters.

```
gap> Append( irreducibles, newirr );
gap> Length( irreducibles );
12
gap> sym2:= Symmetrizations( tbl, newirr, 2 );;
gap> sym3:= Symmetrizations( tbl, newirr, 3 );;
gap> newchars:= Concatenation( sym2, sym3 );;
gap> newchars:= ReducedCharacters( tbl, irreducibles, newchars );;
gap> chars:= Concatenation( chars.remainders, newchars.remainders );;
gap> lll:= LLL( tbl, chars );;
#I  LLL: 35 irreducibles found
gap> lll.norms;
[ 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2 ]
gap> Append( irreducibles, lll.irreducibles );
gap> Length( irreducibles );
47
```

Most of the virtual characters in `lll.remainders` have norm 2, and GAP has a function that inspects lattices geneated by norm 2 vectors for the occurrence of sublattices of types D_4 and D_5; in the latter case, the underlying irreducible characters can always be deduced, in the former case this is possible at least in some cases.

```
gap> dn:= DnLatticeIterative( tbl, lll );;
#I  ReducedClassFunctions: irreducible character of degree 786127419 found
#I  ReducedClassFunctions: irreducible character of degree 786127419 found
#I  ReducedClassFunctions: irreducible character of degree 1579061136 found
```

```
#I  ReducedClassFunctions: irreducible character of degree 2727495848 found
#I  ReducedClassFunctions: irreducible character of degree 3403149 found
#I  ReducedClassFunctions: irreducible character of degree 786127419 found
#I  ReducedClassFunctions: irreducible character of degree 230279749 found
#I  ReducedClassFunctions: irreducible character of degree 1842237992 found
gap> Length( dn.irreducibles );
9
gap> Append( irreducibles, dn.irreducibles );
gap> Length( irreducibles );
56
gap> dn.norms;
[ 2, 2, 2, 2, 2, 2 ]
```

Now 6 irreducible characters are missing, and we know a 6-dimensional lattice L of virtual characters inside the standard lattice spanned by these irreducibles. Let us compute the possible embeddings of L into the standard lattice, and try to deduce irreducible characters if possible.

```
gap> gram:= MatScalarProducts( tbl, dn.remainders, dn.remainders );
[ [ 2, 0, 0, 0, 0, 0 ], [ 0, 2, 0, 0, -1, 0 ], [ 0, 0, 2, 0, -1, 0 ],
  [ 0, 0, 0, 2, 0, 0 ], [ 0, -1, -1, 0, 2, 1 ], [ 0, 0, 0, 0, 1, 2 ] ]
gap> emb:= OrthogonalEmbeddingsSpecialDimension( tbl, dn.remainders, gram, 6 );;
#I  Decreased : computation of 2nd character failed
gap> Length( emb.irreducibles );
2
gap> Append( irreducibles, emb.irreducibles );
```

A four dimensional lattice is left, and the possible embeddings do not determine uniquely the irreducible characters. So we compute all possible embeddings, and inspect the different cases separately.

```
gap> chars:= emb.remainders;;
gap> gram:= MatScalarProducts( tbl, chars, chars );
[ [ 2, 0, -1, 0 ], [ 0, 2, -1, 0 ], [ -1, -1, 2, 1 ], [ 0, 0, 1, 2 ] ]
gap> emb:= OrthogonalEmbeddings( gram, 4 );
rec( vectors := [ [ -1, -1, 1, 0 ], [ -1, 1, 0, 0 ], [ -1, 0, 1, 1 ],
      [ -1, 0, 1, 0 ], [ 1, 0, 0, 1 ], [ 1, 0, 0, 0 ], [ 0, -1, 1, 1 ],
      [ 0, -1, 1, 0 ], [ 0, 1, 0, 1 ], [ 0, 1, 0, 0 ], [ 0, 0, 1, 1 ],
      [ 0, 0, 0, 1 ] ], norms := [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ],
  solutions := [ [ 1, 2, 11, 12 ], [ 3, 5, 8, 10 ], [ 4, 6, 7, 9 ] ] )
```

The `solution` component of the result tells us that there are three solution matrices, each given by a list of positions in the `vectors` component. The irreducible characters can now be extracted using `Decreased`.

```
gap> dec1:= Decreased( tbl, chars, emb.vectors{ emb.solutions[1] } );
#I  Decreased : computation of 1st character failed
fail
```

This means that the first solution does not correspond to irreducible characters.

```
gap> dec2:= Decreased( tbl, chars, emb.vectors{ emb.solutions[2] } );;
gap> Length( dec2.irreducibles );
4
gap> dec3:= Decreased( tbl, chars, emb.vectors{ emb.solutions[3] } );;
gap> Length( dec3.irreducibles );
4
gap> Intersection( dec2.irreducibles, dec3.irreducibles );
[  ]
```

This means that we are left with two possibilities to complete the irreducibles of J_4. The computation of orthogonal embeddings was independent of the table head of J_4, thus the power maps may provide information to exclude one of the possibilities. For example, we can check whether the symmetrizations of the candidates can be decomposed into irreducibles. This way the vectors in `dec2.irreducibles` can be proved not to be characters.

```
gap> sym2:= Symmetrizations( tbl, dec2.irreducibles, 2 );;
gap> ScalarProduct( dec2.irreducibles[1], sym2[1] );
7998193/2
```

Alternatively, one could also try to recompute the second power map from the candidates for irreducibles together with the element orders. For the possibility ruled out, we get a unique power map that is different from the one stored on the table.

```
gap> irr:= Concatenation( irreducibles, dec2.irreducibles );;
gap> pow:= PossiblePowerMaps( tbl, 2, rec( chars:= irr, subchars:= irr ) );
#I  PossiblePowerMaps: 2nd power map initialized; congruences, kernels and
#I    maps for smaller primes considered,
#I    the current indeterminateness is 839808.
#I  PossiblePowerMaps: no test of decomposability allowed
#I  PossiblePowerMaps: test scalar products of minus-characters
#I  PowerMapsAllowedBySymmetrizations: no character with indeterminateness
#I    between 1 and 100000 significant now
#I  PossiblePowerMaps: 1 solution(s)
[ [ 1, 1, 1, 4, 2, 2, 3, 8, 4, 4, 4, 12, 13, 5, 6, 6, 8, 8, 19, 20, 10, 10,
      11, 12, 13, 12, 13, 28, 16, 17, 17, 32, 33, 19, 20, 36, 22, 22, 26, 27,
      41, 28, 43, 44, 45, 46, 47, 48, 49, 51, 52, 50, 30, 31, 32, 33, 57, 58,
      59, 34, 46, 47 ] ]
gap> pow[1] = PowerMap( tbl, 2 );
false
```

Thus we have found the irreducible characters. Finally, let us check whether they coincide with the ones that are stored on the GAP library table.

```
gap> irr:= Concatenation( irreducibles, dec3.irreducibles );;
gap> Set( irr ) = Set( Irr( tbl ) );
true
```

For further computations, we reset the changed `Info` level to zero.

```
gap> SetInfoLevel( InfoCharacterTable, 0 );
```

(The whole computation took only a few minutes.)

References

[Bre04] Thomas Breuer, *Manual for the GAP Character Table Library, Version 1.1*, Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, 2004.