Journal of the Faculty of Environmental Science and Technology, Okayama University
Vol. 1, No. 1, pp.1-13, March 1996

The Gap Condition for S_5 and GAP Programs

Masaharu MORIMOTO* and Mamoru YANAGIHARA†

(Received November 17, 1995)

Abstract

In transformation groups on manifolds, it has been an interesting problem to ask whether for a given finite group G, there exists a real G-module V such that $\dim V^P > 2\dim V^{>P}$ for all subgroups P of prime power order and such that $V^H = 0$ for certain large subgroups H of G. This paper provides GAP programs to show that S_5 does not admit such a real S_5-module V.

KEYWORDS: GAP, fixed point, gap condition.

1991 MATHEMATICS SUBJECT CLASSIFICATION: 57R91, 20C15, 68Q40.

1. Introduction

Let G be a finite group. A real G-module V is said to satisfy the *gap condition* if $\dim V^P > 2\dim V^{>P}$ for all subgroups P of prime power order and such that $V^H = 0$ for certain large subgroups H of G (precisely to say, for all $H \in \mathcal{L}(G)$ defined below). The existence problem of such modules is closely related to equivariant surgery theory (cf. [PR], [M1]) and construction of exotic actions on closed, smooth manifolds. Our purpose in the present paper is to show that S_5 the symmetric group of degree 5 does not admit a real S_5-module satisfying the gap condition, employing the computer software GAP (Groups, Algorithms, and Programming) [S]. This result was announced in [MY] (1994) and the present paper includes the details.

Let G be a finite group. Let $\mathcal{S}(G)$ denote the set of all subgroups of G and $\mathcal{P}(G)$ the set of all subgroups of G of prime power order. (Particularly, the trivial group $\{1\}$ belongs to $\mathcal{P}(G)$.) For each prime p we define a characteristic subgroup G^p by

$$G^p = \bigcap\{H \lhd G \mid |G/H| \text{ is a power of } p\}.$$

Then the set $\mathcal{L}(G)$ mentioned above is defined by

$$\mathcal{L}(G) = \{H \leq G \mid H \supset G^p \text{ for some prime } p\}.$$

Let $\mathcal{M}(G)$ denote the complementary set $\mathcal{S}(G) \setminus \mathcal{L}(G)$. If p and q are primes or 1 and n is a positive integer, let $\mathcal{G}_p^q[n]$ denote the family of all finite groups K having a series $P \lhd H \lhd K$ such that P is of p-power order, H/P is a cyclic group of order n, and K/H is of q-power order. Set

$$\mathcal{G}_p^q = \bigcup_n \mathcal{G}_p^q[n],\quad \mathcal{G}_p = \bigcup_q \mathcal{G}_p^q,\quad \mathcal{G}^q = \bigcup_p \mathcal{G}_p^q,\quad \mathcal{G} = \bigcup_q \mathcal{G}^q,\quad \text{and}\quad \mathcal{G}_{\text{odd}}^{\text{odd}}[2] = \bigcup_{p,\, q \text{ odd}} \mathcal{G}_p^q[2].$$

If a finite group K does not belong to $\mathcal{G}$ then K is called an *Oliver group*. By [O, Theorem 7], a finite group K is an Oliver group if and only if K has a smooth fixed-point free action on a disk. These sets $\mathcal{L}(G)$ and

*Department of Environmental and Mathematical Sciences, Faculty of Environmental Science and Technology, Okayama University, Okayama, 700 Japan.
†Department of Industrial and Information Studies, Komatsu College, Komatsu, Ishikawa, 923 Japan.

$\mathcal{M}(G)$ of subgroups of G play very important roles if G is an Oliver group (cf. [LM], [M2]). We proved the next theorem in [MY].

Theorem 1.1. *Let G be a finite group not of prime power order. If $G^2 = G$ or*

(OC) *$G^p \neq G$ for some odd prime p and $G \notin \mathcal{G}^{\rm odd}_{\rm odd}[2]$,*

then there exist real G-modules satisfying the gap condition.

The remainder of the current paper is devoted to giving GAP programs to confirm the next theorem.

Theorem 1.2. *The symmetric group S_5 of degree 5 does not admit a real S_5-module satisfying the gap condition.*

Set $\mathcal{G}^q_p(G) = \mathcal{S}(G) \cap \mathcal{G}^q_p$, $\mathcal{G}_p(G) = \mathcal{S}(G) \cap \mathcal{G}_p$, $\mathcal{G}^q(G) = \mathcal{S}(G) \cap \mathcal{G}^q$, and $\mathcal{G}(G) = \mathcal{S}(G) \cap \mathcal{G}$,

$$\begin{aligned}
\mathcal{PU}(G) &= \{(P,H) \in \mathcal{P}(G) \times \mathcal{S}(G) \mid P < H\}, \\
\mathcal{P}2\mathcal{S}(G) &= \{(P,H) \in \mathcal{PU}(G) \mid P < H,\ [H:P] = 2, \\
&\qquad [HG^2 : PG^2] = 2, \text{ and } PG^q = G\ (\forall q \text{ odd prime})\}, \text{ and} \\
\mathcal{P}2\mathcal{S}(G)_{\rm odd} &= \{(P,H) \in \mathcal{P}2\mathcal{S}(G) \mid P \text{ is of odd order}\}.
\end{aligned}$$

The organization of the paper is as follows. In Section 2, we give programs to determine the sets $\mathcal{P}(G)$, $\mathcal{L}(G)$ and $\mathcal{P}2\mathcal{S}(G)_{\rm odd}$. In Section 3, we present programs to compute the fixed point dimensions of real S_5-modules, related to the gap condition. In Section 4, we explain how to use the obtained results in Section 3 in order to prove Theorem 1.2.

2. Structure of subgroups of G

We perform computation using the computer software GAP. Let us begin with giving GAP the definition of the group G for which we perform computation. As a usual method in GAP, definition of a group is described by generators being permutations. This is done with the built-in function **Group(-)**. For example, since the symmetric group S_5 of degree 5 is generated by the cyclic permutations $(1,2,3,4,5)$ and $(1,2)$, we can make GAP realize the definition of S_5 in the form:

```
G := Group((1,2,3,4,5), (1,2));;
```

The set of all conjugacy classes $CCS(G)$ (= `CCSG`) of subgroups of G is obtained by the built-in function **ConjugacyClassesSubgroups(-)**:

```
CCSG  := ConjugacyClassesSubgroups(G);
```

and the complete set $RCCS(G)$ (= `RCCSG`) of representatives of $CCS(G)$ is obtained by the built-in function **List(-,-)**:

```
RCCSG := List(CCSG, h -> Representative(h));
```

For example, we obtain the following result in the case $G = S_5$.

Result 2.1. *There are* 19 *conjugacy classes of subgroups of S_5. They have the following representatives.*

```
RCCSG[1]  = Subgroup( G, [  ] ),
RCCSG[2]  = Subgroup( G, [ (4,5) ] ),
RCCSG[3]  = Subgroup( G, [ (2,3)(4,5) ] ),
RCCSG[4]  = Subgroup( G, [ (3,4,5) ] ),
RCCSG[5]  = Subgroup( G, [ (2,3)(4,5), (2,4)(3,5) ] ),
RCCSG[6]  = Subgroup( G, [ (2,3)(4,5), (2,4,3,5) ] ),
RCCSG[7]  = Subgroup( G, [ (4,5), (2,3) ] ),
RCCSG[8]  = Subgroup( G, [ (1,2,3,4,5) ] ),
RCCSG[9]  = Subgroup( G, [ (3,4,5), (4,5) ] ),
RCCSG[10] = Subgroup( G, [ (3,4,5), (1,2)(4,5) ] ),
RCCSG[11] = Subgroup( G, [ (4,5), (1,3,2) ] ),
RCCSG[12] = Subgroup( G, [ (4,5), (2,3), (2,4)(3,5) ] ),
RCCSG[13] = Subgroup( G, [ (1,2,3,4,5), (2,5)(3,4) ] ),
RCCSG[14] = Subgroup( G, [ (2,3)(4,5), (2,4)(3,5), (3,4,5) ] ),
RCCSG[15] = Subgroup( G, [ (4,5), (1,3,2), (2,3) ] ),
RCCSG[16] = Subgroup( G, [ (1,2,3,4,5), (2,5)(3,4), (2,3,5,4) ] ),
RCCSG[17] = Subgroup( G, [ (2,3)(4,5), (2,4)(3,5), (3,4,5), (4,5) ] ),
RCCSG[18] = Subgroup( G, [ (1,3,2), (2,4,3), (2,3)(4,5) ] ),
RCCSG[19] = Subgroup( G, [ (1,2,3,4,5), (1,2) ] ) = G.
```

In our computation of $\mathcal{L}(G)$, we use

$$\mathcal{L}(G)_{\mathrm{normal}} = \{H \in \mathcal{L}(G) \mid H \triangleleft G\} \quad (= \texttt{LGnormal}),$$

and the next function **makeLGnormal(-)** computes the set $\mathcal{L}(G)_{\mathrm{normal}}$.

```
makeLGnormal := function()
        local S, H, i, ns, ni;
        S := [];
        ns := Length(RCCSG);
        for i in [1..ns] do
                H := RCCSG[i];
                ni := Index(G, H) ;
                if IsPrimePowerInt(ni) and IsNormal(G,H) then
                        Add(S, i);
                elif ni = 1 then
                        Add(S, i);
        fi;
        od;
        return S;
end;
```

Program 2.2.

After making GAP read Program 2.2, we can obtain $\mathcal{L}(G)_{\mathrm{normal}}$ by typing

```
LGnormal := makeLGnormal();
```

in GAP.

Next we give a function **testLG**(-) which checks whether a subgroup H lies in $\mathcal{L}(G)$ or not. If $H \in \mathcal{L}(G)$ then **testLG**(-) returns true and else false. This **testLG**(-) is given by a program including a function **isSubConjugate**(-,-) that assigns to subgroups `RCCSG[h]` and `RCCSG[k]`, true if `RCCSG[h]` is conjugate to a subgroup of `RCCSG[k]` and false otherwise.

```
isSubConjugate := function(k, h)
        local size_k, size_h, conj, hh;
        size_k := Size(RCCSG[k]);
        size_h := Size(RCCSG[h]);
        if (size_k = Size(G)) or (k = h) then
                return true;
        fi;
        if not (IsInt(size_k/size_h)) then
                return false;
        fi;
        if size_k = size_h then
                return false;
        fi;
        for hh in Elements(CCSG[h]) do
                if IsSubgroup(RCCSG[k], hh) then
                        return true;
                fi;
        od;
        return false;
end;
```

Program 2.3.

The function **testLG**(-) is given by the program:

```
testLG := function(h)
        local h1;
        for h1 in LGnormal do
                if isSubConjugate(h, h1) then
                        return true;
                fi;
        od;
        return false;
end;
```

Program 2.4.

The set $\mathcal{L}(G)$ (= `LG`) is computed by the function **makeLG**():

```
makeLG := function()
        local S, n, i;
        S := [];
        n := Length(RCCSG);
        for i in [1..n] do
                if testLG(i) then
```

```
                Add(S, i);
            fi;
        od;
        return S;
end;
LG := makeLG();
```

Program 2.5.

Result 2.6. *If* $G = S_5$ *then* `LG = [18, 19]`, *i.e.* $\mathcal{L}(G) = [RCCSG[18], RCCSG[19]]$.

Let $Prime(G)$ (= `PrimeG`) be the set of primes dividing $|G|$ (the order of G). $Prime(G)$ is computed by

```
PrimeG := Set(Factors(Size(G)));
```

The next function **coSylow(-)** assigns to a prime $p \in Prime(G)$ the normal subgroup G^p (called the *coSylow p-subgroup* of G):

```
coSylow := function(p)
        local ind, max_ind, Gp, h;
        max_ind := 1;
        Gp := Length(RCCSG);
        for h in LG do
                ind := Index(G, RCCSG[h]);
                if IsInt(ind / p) and (max_ind < ind) then
                        max_ind := ind;
                        Gp := h;
                fi;
        od;
        return Gp;
end;
```

Program 2.7.

The set $CoSylow(G) = \{(p, G^p) \mid p \in Prime(G)\}$ (= `CoSylowG`) is obtained by the function **makeCoSylow()**:

```
makeCoSylow := function()
        local  S, n, i;
        S := [];
        n := Length(PrimeG);
        for i in [1..n] do
                S[i] := [PrimeG[i], coSylow(PrimeG[i])];
        od;
        return S;
end;
CoSylowG := makeCoSylow();
```

Program 2.8.

Result 2.9. *If* $G = S_5$ *then* $CoSylow(G) = \{(2, RCCSG[18]), (3, RCCSG[19]), (5, RCCSG[19])\}$.

We compute $\mathcal{P}2\mathcal{S}(G)_{\mathrm{odd}}$ (= `P2SGodd`) as follows. The function **subgProduct(-,-)** defined below assigns a subgroup HN to a subgroup H and a normal subgroup N of G.

```
subgProduct := function(H, N)
        local gen;
        gen := Union(H.generators, N.generators);
        return Subgroup(G, gen);
end;
```

Program 2.10.

We also use $\mathcal{P}(G)$ (= PG) in our computation of $\mathcal{P}2\mathcal{S}(G)_{\text{odd}}$, and the function **makePG()** computes $\mathcal{P}(G)$.

```
makePG := function()
        local pg, i, ns, size;
        pg := [];
        ns := Length(RCCSG);
        for i in [1..ns] do
                size := Size(RCCSG[i]);
                if IsPrimePowerInt(size) or (size = 1) then
                        Add(pg, i);
                fi;
        od;
        return pg;
end;
PG := makePG();
```

Program 2.11.

If RCCSG[i] is in PG, we check whether (RCCSG[i], RCCSG[j]) is in $\mathcal{P}2\mathcal{S}(G)$ (= P2SG) or not, with the function **testP2SG**(-, -):

```
testP2SG := function(i, j)
        local P, H, gsize, pair, p, Gp, K1, K2;
        P := RCCSG[i];
        H := RCCSG[j];
        if not (Size(H) / Size(P) = 2) then
                return false;
        fi;
        if isSubConjugate(j, i) = false then
                return false;
        fi;
        gsize := Size(G);
        for pair in CoSylowG do
                p := pair[1];
                Gp := RCCSG[pair[2]];
                K1 := subgProduct(P, Gp);
                if p = 2 then
                        K2 := subgProduct(H, Gp);
                        if not (Index(K2, K1) = 2) then
                                return false;
                        fi;
```

```
                else
                        if not (Size(K1) = gsize) then
                                return false;
                        fi;
                fi;
        od;
        return true;
end;
```

Program 2.12.

We can obtain the list $\mathcal{P}2\mathcal{S}(G)$ using the function **makeP2SG()**:

```
makeP2SG := function()
        local S, np, ns, i, j;
        S := [];
        np := Length(PG);
        ns := Length(RCCSG);
        for i in [1..np] do
                for j in [1..ns] do
                        if testP2SG(PG[i], j) then
                                Add(S, [PG[i], j]);
                        fi;
                od;
        od;
        return S;
end;
P2SG := makeP2SG();
```

Program 2.13.

Result 2.14. *If $G = S_5$ then one obtains the result:*

$$\begin{aligned}\mathcal{P}2\mathcal{S}(G) = \{ & (RCCSG[1], RCCSG[2]), \quad (RCCSG[3], RCCSG[6]), \\ & (RCCSG[3], RCCSG[7]), \quad (RCCSG[4], RCCSG[9]), \\ & (RCCSG[4], RCCSG[11]), \quad (RCCSG[5], RCCSG[12]) \}.\end{aligned}$$

The set $\mathcal{P}2\mathcal{S}(G)_{\mathrm{odd}}$ is computed by the function **makeP2SGodd()**:

```
makeP2SGodd := function()
        local S, n, i;
        S := [];
        n := Length(P2SG);
        for i in [1..n] do
                if not IsInt(P2SG[i][1] / 2) then
                        Add(S, P2SG[i]);
                fi;
        od;
        return S;
```

```
end;
P2SGodd := makeP2SGodd();
```

Program 2.15.

Result 2.16. *If $G = S_5$ then one obtains the result:*

$$\mathcal{P}2\mathcal{S}(G)_{\text{odd}} = \{ (RCCSG[1], RCCSG[2]), (RCCSG[3], RCCSG[6]), (RCCSG[3], RCCSG[7]), (RCCSG[5], RCCSG[12]) \}.$$

3. H-Fixed point dimensions of irreducible G-representations

The built-in function **CharTable(-)** gives the character table of irreducible representations. Before using this function, we must set **G.conjugacyClasses**.

The character table will be obtained in the order of **G.conjugacyClasses**. Set

```
G.conjugacyClasses := ConjugacyClasses(G);;
```

Result 3.1. *If $G = S_5$ then one obtains the result:*

```
c1 = G.conjugacyClasses[1] = ConjugacyClasses( G, () ),
c2 = G.conjugacyClasses[2] = ConjugacyClasses( G, (4,5) ),
c3 = G.conjugacyClasses[3] = ConjugacyClasses( G, (3,4,5) ),
c4 = G.conjugacyClasses[4] = ConjugacyClasses( G, (2,3)(4,5) ),
c5 = G.conjugacyClasses[5] = ConjugacyClasses( G, (2,3,4,5) ),
c6 = G.conjugacyClasses[6] = ConjugacyClasses( G, (1,2)(3,4,5) ),
c7 = G.conjugacyClasses[7] = ConjugacyClasses( G, (1,2,3,4,5) ).
```

Next apply the function:

```
CTG := CharTable(G);;
```

The irreducible character table is tabulated by **CTG.irreducibles** from the data CTG, and the value of the i-th irreducible character on the j-th conjugacy class is given by **CTG.irreducibles[i][j]**.

Result 3.2. *If $G = S_5$ then one obtains the result:*

	conjugacy classes						
	c_1	c_2	c_3	c_4	c_5	c_6	c_7
χ_1 = CTG.irreducibles[1]	1	1	1	1	1	1	1
χ_2 = CTG.irreducibles[2]	1	-1	1	1	-1	-1	1
χ_3 = CTG.irreducibles[3]	4	-2	1	0	0	1	-1
χ_4 = CTG.irreducibles[4]	4	2	1	0	0	-1	-1
χ_5 = CTG.irreducibles[5]	5	1	-1	1	-1	1	0
χ_6 = CTG.irreducibles[6]	5	-1	-1	1	1	-1	0
χ_7 = CTG.irreducibles[7]	6	0	0	-2	0	0	1

Table 3.2 : Irreducible Characters of S_5

In order to regard the data in CTG of a irreducible character as a function from G to the complex number field, we prepare the function **irrCharacter(-, -)**. This function assigns $\chi_j(x)$ to the j-th irreducible character χ_j and $x \in G$.

```
irrCharacter := function(j, x)
        local k, n;
        n := Length(CTG.irreducibles);
        for k in [1..n] do
                if x in G.conjugacyClasses[k] then
                        return CTG.irreducibles[j][k];
                fi;
        od;
end;
```

Program 3.3.

Let V be a complex G-representation. The dimension $\dim_{\mathbf{C}} V^H$ of H-fixed point set V^H is calculated with the formula

$$\dim_{\mathbf{C}} V^H = \frac{1}{|H|} \sum_{h \in H} \chi_V(h),$$

where χ_V is the character of G, canonically identified with V. We give the function **fixedDim(-, -)** assigning $\dim_{\mathbf{C}} V^H$ to the i-th subgroup $H =$ `RCCSG[i]` in `RCCSG` and the j-th irreducible character $V =$ `CTG.irreducibles[j]` of G by

```
fixedDim := function(i, j)
        local h, x, s, d;
        if (i = Length(RCCSG)) then
                if (j = 1) then
                        return 1;
                else
                        return 0;
                fi;
        fi;
        h := RCCSG[i];
        s := Size(h);
        d := Sum(Elements(h), x -> irrCharacter(j, x)) / s;
        return d;
end;
```

Program 3.4.

Now we make the table `FDT` of the fixed dimensions. For a subgroup `RCCSG[i]`, `FDT[i]` is a list of the fixed dimension of the j-th irreducible representation by the i-th subgroup.

```
FDT[i] := List([1..n], j -> fixedDim(i, j));
```

Result 3.5. *If $G = S_5$ then one obtains the result:*

	irreducible modules						
	V_1	V_2	V_3	V_4	V_5	V_6	V_7
RCCSG[1]	1	1	4	4	5	5	6
RCCSG[2]	1	0	1	3	3	2	3
RCCSG[3]	1	1	2	2	3	3	2
RCCSG[4]	1	1	2	2	1	1	2
RCCSG[5]	1	1	1	1	2	2	0
RCCSG[6]	1	0	1	1	1	2	1
RCCSG[7]	1	0	0	2	2	1	1
RCCSG[8]	1	1	0	0	1	1	2
RCCSG[9]	1	0	0	2	1	0	1
RCCSG[10]	1	1	1	1	1	1	0
RCCSG[11]	1	0	1	1	1	0	1
RCCSG[12]	1	0	0	1	1	1	0
RCCSG[13]	1	1	0	0	1	1	0
RCCSG[14]	1	1	1	1	0	0	0
RCCSG[15]	1	0	0	1	1	0	0
RCCSG[16]	1	0	0	0	0	1	0
RCCSG[17]	1	0	0	1	0	0	0
RCCSG[18]	1	1	0	0	0	0	0
RCCSG[19]	1	0	0	0	0	0	0

Table 3.6 : S_5-Fixed Dimensions

In order to obtain such FDT, we give the function **makeFixedDimTable()**:

```
makeFixedDimTable := function()
        local S, nr, ni, i, j;
        S := [];
        nr := Length(RCCSG);
        ni := Length(CTG.irreducibles);
        for i in [1..nr] do
                S[i] := List([1..ni], j -> fixedDim(i, j))
        od;
        return S;
end;
FDT := makeFixedDimTable();
```

Program 3.7.

Let $Irr(G)$ denote the set of all isomorphism classes of irreducible complex G-representations. A complete set of representatives of $Irr(G)$ is denoted by $RIrr(G)$. The set $RIrr(G)$ is identified with the set of all irreducible characters of G. Let $Irr(G, \mathcal{M}(G))$ be the set of all isomorphism classes of irreducible complex G-representations V such that $V^H = 0$ for all $H \in \mathcal{L}(G)$. Let $RIrr(G, \mathcal{M}(G))$ be a complete set of representatives of $Irr(G, \mathcal{M}(G))$. The next function **testIrrMG(-)** tells whether an irreducible G-representation belongs to $Irr(G, \mathcal{M}(G))$ or not.

```
testIrrMG := function(i)
        local j;
        for j in LG do
                if not (FDT[j][i] = 0) then
```

```
                        return false;
                fi;
        od;
        return true;
end;
```

Program 3.8.

A set $RIrr(G, \mathcal{M}(G))$ is obtained by the function:

```
makeRIrrMG := function()
        local S, i, n;
        S := [];
        n := Length(CTG.irreducibles);
        for i in [1..n] do
                if testIrrMG(i) then
                        Add(S, i);
                fi;
        od;
```

Program 3.9.

Result 3.10. *If $G = S_5$ then one obtains the result: $RIrr(G, \mathcal{M}(G)) = \{ V_3, V_4, V_5, V_6, V_7 \}$.*

Two irreducible complex G-representations V and W are said to be *Galois conjugate* if $\dim_{\mathbf{C}} V^H = \dim_{\mathbf{C}} W^H$ for all subgroups H of G. Let $GCCIrr(G, \mathcal{M}(G))$ be the set of all Galois conjugate classes of representations in $Irr(G, \mathcal{M}(G))$, and let $RGCCIrr(G, \mathcal{M}(G))$ be a complete set of representatives of $GCCIrr(G, \mathcal{M}(G))$. The next function **testGaloisConjugate(-, -)** checks whether, given a set S of irreducible representations, a irreducible representation is Galois conjugate to an element in S or not.

```
testGaloisConjugate := function(Irrs, i)
        local n, j, k, s;
        n := Length(RCCSG);
        for j in Irrs do
                s := Sum([1..n], k -> AbsInt(FDT[k][i] - FDT[k][j]));
                if (s = 0) then
                        return true;
                fi;
        od;
        return false;
end;
```

Program 3.11.

We can find a set $RGCCIrr(G, \mathcal{M}(G))$ by the next function:

```
makeRGaloisCCIrrMG := function()
        local a, i, j, k, gcc;
        gcc := [RIrrMG[1]];
        for i in RIrrMG do
```

```
            if not testGaloisConjugate(gcc, i) then
                    Add(gcc, i);
            fi;
        od;
        return gcc;
end;
RGCCIrrMG := makeRGaloisCCIrrMG();
```

Program 3.12.

Result 3.13. *If* $G = S_5$ *then one obtains* $RGCCIrr(G, \mathcal{M}(G)) = \{ V_3, V_4, V_5, V_6, V_7 \}$.

The function **fixedDimDiff(-, -)** below assigns to `Pairs` (a set consisting of pairs (H, K) of subgroups of G) and a set `Irrs` of irreducible representations, the list of $\dim_{\mathbf{C}} V^H - 2\dim_{\mathbf{C}} V^K$, where (H, K) runs over `Pairs` and V does over `Irrs`.

```
fixedDimDiff := function(Pairs, Irrs)
        local S, pair, h, k, i, b;
        S:= [];
        for pair in Pairs do
                h := pair[1];
                k := pair[2];
                b := List(Irrs, i -> FDT[h][i] - 2 * FDT[k][i]);
                Add(S, b);
        od;
        return S;
end;
```

Program 3.14.

Result 3.15. *If* $G = S_5$ *then by* **fixedDimDiff(P2SG, RGCCIrrMG)**, *one obtains the result:*

	irreducible modules				
	V_3	V_4	V_5	V_6	V_7
(RCCSG[1], RCCSG[2])	2	-2	-1	1	0
(RCCSG[3], RCCSG[6])	0	0	1	-1	0
(RCCSG[3], RCCSG[7])	2	-2	1	1	0
(RCCSG[4], RCCSG[9])	2	-2	-1	1	0
(RCCSG[4], RCCSG[11])	0	0	-1	1	0
(RCCSG[5], RCCSG[12])	1	-1	0	0	0

Table 3.16 : Differences of S_5-Fixed Dimensions

4. Proof of Theorem 1.2

Let $G = S_5$. If V is a real G-module satisfying the gap condition then the complex module $\mathbf{C} \otimes_{\mathbf{R}} V$ satisfies the gap condition with respect to complex dimension. That is, the function

$$f_V(P, H) = \dim_{\mathbf{C}} V^P - 2\dim_{\mathbf{C}} V^H$$

is positive for all $P \in \mathcal{P}(G)$ and $H > P$, and in addition, $\dim_{\mathbf{C}} V^K = 0$ for all $K \in \mathcal{L}(G)$. Suppose that there exists a complex G-module satisfying the gap condition. Replacing each irreducible summand by a Galois conjugate module in $RGCCIrr(G, \mathcal{M}(G))$, we obtain a complex G-module

$$V = a_3V_3 \oplus a_4V_4 \oplus a_5V_5 \oplus a_6V_6 \oplus a_7V_7,$$

where a_i are nonnegative integers, satisfying the gap condition. Since

$$f_V(P, H) > 0 \quad \text{for} \quad (P, H) = (RCCSG[3], RCCSG[6]) \quad \text{and} \quad (RCCSG[4], RCCSG[11]),$$

it follows from Table 3.16 that $a_5 - a_6 > 0$ and $-a_5 + a_6 > 0$. This is a contradiction. Thus there never exists a real G-module satisfying the gap condition if $G = S_5$.

References

[LM] Laitinen, E. and Morimoto, M.; Finite groups with smooth one fixed point actions on spheres, Reports Dept. Math. Univ. Helsinki no. 25, Helsinki, 1993.

[M1] Morimoto, M.; Bak groups and equivariant surgery, K-Theory **2** (1989), 465–483.

[M2] ______ ; Equivariant surgery theory: Construction of equivariant normal maps, Publ. RIMS Kyoto Univ. **31** (1995), 145–167.

[MY] Morimoto, M. and Yanagihara, M.; On gap conditions of finite group actions, Preprint 1994.

[O] Oliver, R.; Fixed point sets of group actions on finite acyclic complexes, Comment. Math. Helvetici **50** (1975), 155–177.

[PR] Petrie, T. and Randall, J. D.; *Transformation Groups on Manifolds*, Dekker, New York, 1984.

[S] Schönert, M. et.al.; *GAP-Groups, Algorithms, and Programming* (fourth ed.), Lehrstuhl D für Mathematik, RWTH, Aachen, 1994.