THE COHOMOLOGY OF FLAG VARIETIES AND SCHUBERT POLYNOMIALS

YUZE LUAN

Contents

1. Introduction

This senior thesis aims to introduce an interesting space called the *flag variety*, and its cohomology ring, called the *coinvariant algebra*. The flag variety is a space (algebraic variety), whose points correspond to complete flags, that is, sequences of vector subspaces

$$\{0\} = V_0 \subset V_1 \subset \cdots \subset V_n = \mathbb{C}^n,$$

where $\dim_{\mathbb{C}} V_i = i$. The coinvariant algebra is the quotient of the polynomial ring $\mathbb{Q}[x_1, \ldots, x_n]$ by the ideal generated by symmetric polynomials. Recall that a polynomial $p \in \mathbb{Q}[x_1, \ldots, x_n]$ is called *symmetric* if

$$p(x_1, \ldots, x_n) = p(x_{\sigma(1)}, \ldots, x_{\sigma(n)})$$

for any permutation σ of $\{1, \ldots, n\}$. The coinvariant algebra has a basis given by *Schubert polynomials*.

The original contribution is a computer program we wrote, which computes the coefficients of the product of two Schubert polynomials in the Schubert polynomial basis. This is motivated by an interesting open problem: there is no known combinatorial formula for these coefficients.

2. Topological preliminaries: homology and cohomology

In this section, we will discuss some preliminary topological concepts, namely homology and cohomology. All the definitions and propositions in this section are taken from Hatcher's book [1].

Homology is a very useful tool for the study of unknown spaces, and it can be considered to be a generalization of the well-known Euler characteristic. For any given polyhedron P, denote the number of vertices of this polyhedron by V, the number of edges by E, and the number of faces by F, then the Euler characteristic of the polyhedron is the number

$$\chi(P) = V - E + F.$$

We can compute, for instance, that the Euler characteristics of a tetrahedron, a cube and an octahedron are all 2. On the other hand, no matter how we triangulate a torus, the Euler characteristics of all the possible triangulations are always 0.

We can see that the Euler characteristic seems to be able to distinguish shapes that are similar to a sphere and shapes that are similar to a torus. Homology is a generalization of the Euler characteristic, in the sense that it can distinguish between more spaces than the Euler characteristic.

In order to define homology, we first need to define the n-simplex, and the Δ-complex structure, which are similar to "triangulating a space".

Definition 1. The *standard n-simplex* Δ^n is

$$\Delta^n = \left\{ (t_0, \ldots, t_n) \in \mathbb{R}^{n+1} : \sum_{i=0}^{n} t_i = 1 \text{ and } t_i \geq 0 \text{ for all } i \right\}.$$

The *interior* of Δ^n, denoted by $\Delta^n_\circ$, is

$$\Delta^n_\circ = \left\{ (t_0, \ldots, t_n) \in \Delta^n : t_i > 0 \text{ for all } i \right\}.$$

The *boundary* of Δ^n is

$$\partial\Delta^n = \Delta^n \backslash \Delta^n_\circ.$$

The boundary is the union of the *faces* of Δ^n.

Definition 2. A *Δ-complex structure* on a topological space X is a collection of maps $\sigma_\alpha : \Delta^n \to X$, with n depending on the index α, such that:

(1) The restriction $\sigma_\alpha|_{\Delta^n_\circ}$ is injective, and each point of X is in the image of exactly one such restriction $\sigma_\alpha|_{\Delta^n_\circ}$.

(2) Each restriction of σ_α to a face of Δ^n is one of the maps $\sigma_\beta : \Delta^{n-1} \to X$. Here we are identifying the face of Δ^n with Δ^{n-1} by the canonical linear homeomorphism between them that preserves the ordering of the vertices.
(3) A set $A \subset X$ is open iff $\sigma_\alpha^{-1}(A)$ is open in Δ^n for each σ_α.

Definition 3. The group of *n-chains* $\Delta_n(X)$ consists of (finite) formal sums

$$\sum_\alpha n_\alpha \sigma_\alpha,$$

where $\sigma_\alpha : \Delta^n \to X$ are the maps in the definition of a Δ-complex structure, and $n_\alpha \in \mathbb{Q}$.

Definition 4. The *boundary homomorphism* $\partial_n : \Delta_n(X) \to \Delta_{n-1}(X)$ is defined by requiring

$$\partial_n(\sigma_\alpha) = \sum_{i=0}^{n} (-1)^i \sigma_\alpha | [v_0, \ldots, \hat{v_i}, \ldots, v_n],$$

where $\hat{v_i}$ denotes deleting the element v_i, and $[v_0, \ldots, \hat{v_i}, \ldots, v_n]$ is the corresponding face of $\partial\Delta^n$, and then extending the definition to formal sums by linearity. The nth *simplicial homology group* is the quotient group

$$H_n(X, \mathbb{Q}) = \mathrm{Ker}(\partial_n)/\mathrm{Im}(\partial_{n+1}).$$

There are many ways to put Δ-complex structures on a space X, but they all give the same simplicial homology groups. This is a consequence of Theorem 2.27 in [1].

The homology of flag varieties is computed using cellular homology, a tool for computing homology groups. Cellular homology requires the structure of a *cell complex* or *CW complex* on the topological space of interest. A k-cell is an open ball of dimension k.

Example 1. An n-sphere can be constructed by attaching an n-cell to a 0-cell. A torus can be constructed first by attaching two 1-cells to a 0-cell to form a figure "8", and then by attaching a 2-cell to the figure 8. These two spaces are both CW complexes.

Here is a fun remark connecting back to the Euler characteristic.

Remark 1. For a finite CW complex X, the Euler characteristic of X is defined to be the alternating sum

$$\chi(X) = \sum_n (-1)^n c_n,$$

where c_n is the number of n-cells of X. This definition generalizes the $V - E + F$ formula for 2-dimensional complexes.

We now define cellular homology.

Definition 5. Let X be a CW complex. The *n-skeleton* of X, denoted by X^n, is the union of all cells of dimension less than or equal to n. The *cellular chain complex* is

$$\cdots \to H_{n+1}(X^{n+1}, X^n) \to H_n(X^n, X^{n-1}) \to H_{n-1}(X^{n-1}, X^{n-2}) \to \cdots .$$

The *cellular homology groups* of X are the homology groups of its cellular chain complex.

Theorem 2.35 in [1] proves that computing homology using cellular homology gives the same result as using the simplicial homology groups defined earlier.

Theorem 1. The cellular homology groups are isomorphic to the simplicial homology groups.

In fact, we need to combine the two theorems mentioned above (2.27 and 2.35) to prove Theorem 1.

Lemma 1. If X is a CW complex, then $H_n(X^n, X^{n-1})$ is a free abelian group and has a basis in one-to-one correspondence with the n-cells of X.

Lemma 1 is proved in [1]. We now state the key insight which will allow us to compute the homology of the flag variety from the filtration discussed in the next section.

Lemma 2. Let X be a CW complex with the following property: all cells of X have **even** dimension. Let c_n be the number of n-cells of X. Then

$$H_n(X, \mathbb{Q}) = \begin{cases} \mathbb{Q}^{c_n}, & \text{if } n \text{ is even,} \\ 0, & \text{if } n \text{ is odd.} \end{cases}$$

Proof. The cellular chain complex (1) is of the form

$$\ldots \to \mathbb{Q}^{c_{2k+2}} \to 0 \to \mathbb{Q}^{c_{2k}} \to 0 \to \mathbb{Q}^{c_{2k-2}} \to \ldots \tag{2}$$

It is trivial to compute the homology of this cellular chain complex, because all homomorphisms are necessarily identically 0. Indeed, we have

$$\mathrm{Ker}(0 \to \mathbb{Q}^{c_{2k}}) = 0 \quad \text{and} \quad \mathrm{Im}(0 \to \mathbb{Q}^{c_{2k}}) = 0,$$

and

$$\mathrm{Ker}(\mathbb{Q}^{c_{2k}} \to 0) = \mathbb{Q}^{c_{2k}} \quad \text{and} \quad \mathrm{Im}(\mathbb{Q}^{c_{2k}} \to 0) = 0,$$

for all k, from which the Lemma follows. □

Now we define cohomology, which gives us a ring structure rather than a group structure.

Definition 6. Given a space X, we form its chain complex

$$\ldots \to \Delta_{n+1} \to \Delta_n \xrightarrow{\partial_n} \Delta_{n-1} \to \ldots,$$

and dualize this chain complex by replacing each Δ_n with

$$\Delta_n^* = \mathrm{Hom}(\Delta_n, \mathbb{Q}),$$

the group of homomorphisms $\Delta_n \to \mathbb{Q}$. Also replace the boundary maps ∂_n by their dual maps ∂_n^* from Δ_{n-1}^* to Δ_n^*. We obtain

$$\cdots \to \Delta_{n-1}^* \xrightarrow{\partial_n^*} \Delta_n^* \to \Delta_{n+1}^* \to \cdots.$$

The *cohomology group* $H^n(X, \mathbb{Q})$ is $\mathrm{Ker}(\partial_{n+1}^*)/\mathrm{Im}(\partial_n^*)$.

The *cohomology ring of* X is

$$H^*(X, \mathbb{Q}) = \bigoplus_n H^n(X, \mathbb{Q}),$$

and its multiplication operation is given by the *cup product*, as explained in Chapter 3 of [1].

The cohomology groups are sometimes isomorphic to the homology groups.

Theorem 2. (Poincaré duality) If M is an n-dimensional oriented closed manifold, then there is an isomorphism

$$H^k(M, \mathbb{Q}) \cong H_{n-k}(M, \mathbb{Q})$$

between the kth cohomology group and the $(n-k)$th homology group.

3. (Co)homology of Flag Varieties

Recall from the introduction the definition of a complete flag.

Definition 7. A *complete flag* of an n-dimensional vector space V over $\mathbb{C}$ is an increasing sequence of subspaces

$$\{0\} = V_0 \subset V_1 \subset V_2 \subset \ldots \subset V_n = V,$$

where $\dim_{\mathbb{C}} V_i = i$.

Definition 8. The *(complete) flag variety* Fl_n is the set whose elements are the complete flags of $\mathbb{C}^n$. It has the structure of a manifold, and of a projective algebraic variety.

A natural questions is what Fl_n looks like. Obviously, the complete flag

$$\{0\} \subset \mathbb{C} \subset \mathbb{C}^2 \subset \ldots \subset \mathbb{C}^n \tag{3}$$

is a point in Fl_n. Here, by $\mathbb{C}^k$, we mean the subspace of $\mathbb{C}^n$ spanned by $e_1, \ldots, e_k$, where $e_1, \ldots, e_n$ is the standard basis of $\mathbb{C}^n$. If we "wiggle"

any subspace in this sequence without changing its dimension and its inclusion relation with the previous and next subspaces, the resulting sequence of subspaces will also be a point in the flag variety. So we can intuitively guess that the flag variety Fl_n is a manifold. It is stated in Fulton's book [2] that Fl_n is indeed a manifold of dimension $\binom{n}{2}$.

We can use the tools of homology and cohomology to study the flag variety.

Question 1. What is the (co)homology of the flag variety?

It is quite difficult to put a Δ-complex structure on the flag variety, for the purpose of computing its simplicial homology. However, Fl_n has a natural cellular decomposition into *Schubert cells*, which will therefore allow us to compute $H^*(Fl_n, \mathbb{Q})$ by means of cellular (co)homology.

Definition 9. Denote the complete flag in (3) by

$$F_\bullet = (\{0\} = F_0 \subset F_1 \subset \cdots \subset F_n = \mathbb{C}^n).$$

For any permutation $\omega \in S_n$, define the *Schubert cell*

$$X_\omega = \{E_\bullet \in Fl_n : \dim(E_p \cap F_q) = \#\{i \leq p | \omega(i) \leq q\}\}.$$

A more intuitive way to understand this definition is as follows: define an equivalence relation on Fl_n by

$$E_\bullet \sim E'_\bullet \quad \text{if} \quad \dim(E_p \cap F_q) = \dim(E'_p \cap F_q), \text{ for all } p, q.$$

Then the Schubert cells are the equivalence classes of the equivalence relation "$\sim$".

Notice that there are $n!$ Schubert cells because the number of permutations is $n!$.

The key to computing $H^*(Fl_n, \mathbb{Q})$ is the following. It can be proved that all Schubert cells are affine spaces $\mathbb{A}^k$ of some dimension k. Because $\mathbb{A}^k$ is a cell of dimension $2k$, we can apply Lemma 2 (and Poincaré duality) to conclude the following.

Theorem 3. *$H^*(Fl_n, \mathbb{Q})$ has a basis whose elements are in correspondence with the Schubert cells. These are called Schubert classes.*

An immediate corollary of Theorem 3 is that

$$\dim_{\mathbb{Q}} H^*(Fl_n, \mathbb{Q}) = n!. \tag{4}$$

Example 2. In this example, we compute the cohomology of the flag variety Fl_3. As above, denote the standard basis of $\mathbb{C}^3$ as $\{e_1, e_2, e_3\}$, and fix the flag $F_\bullet$

$$(F_0 = \{0\}) \subset (F_1 = \langle e_1 \rangle) \subset (F_2 = \langle e_1, e_2 \rangle) \subset (F_3 = \mathbb{C}^3).$$

Recall that for any $\omega \in S_3$, X_ω consists of all complete flags $E_\bullet$ such that

$$\dim(E_p \cap F_q) = \#\{i \leq p | \omega(i) \leq q\},$$

for all $1 \leq p, q \leq 3$. For simplicity of notation, we write

$$d_{i,j} = \dim(E_p \cap F_q)$$

in the table below.

ω	$d_{1,1}$	$d_{1,2}$	$d_{2,1}$	$d_{2,2}$
$[1,\ 2,\ 3]$	1	1	1	2
$[1,\ 3,\ 2]$	1	1	1	1
$[2,\ 1,\ 3]$	0	1	1	2
$[2,\ 3,\ 1]$	0	1	0	1
$[3,\ 1,\ 2]$	0	0	1	1
$[3,\ 2,\ 1]$	0	0	0	1

Because $\#\{i \leq p | \omega(i) \leq 3\} = p$, and $\#\{i \leq 3 |\ \omega(i) \leq q\} = q$, $d_{p,3} = p$ and $d_{3,q} = q$ for all p and q. This is not surprising because E_3 and F_3 are all equal to $\mathbb{C}^3$, and when they are intersected with a smaller subspace, i.e. E_p or F_q, the dimension of the intersection should certainly be p and q. These dimensions of intersections with $\mathbb{C}^3$ along with the dimension of intersections with F_0 are trivial and so are not listed in the table.

Now, we want to compute the Schubert cells from the table.

For $[1,\ 2,\ 3]$, because $\dim(E_1 \cap F_1) = 1$ and $\dim(E_2 \cap F_2) = 2$, the only flag that satisfies the condition is the fixed flag $F_\bullet$. So the Schubert cell corresponding to $[1,\ 2,\ 3]$ is a point,

$$X_{[1,\ 2,\ 3]} = \mathbb{A}^0.$$

For $[1,\ 3,\ 2]$, we also know immediately from $\dim(E_1 \cap F_1) = 1$ that E_1 is F_1, and we also know from $\dim(E_2 \cap F_2) = 1$ that E_2 is all the 2-dimensional subspaces (i.e. planes) in $\mathbb{C}^3$ that contain e_1 and do not contain e_2. So the Schubert cell corresponding to $[1,\ 3,\ 2]$ is $\mathbb{A}^1$,

$$X_{[1,\ 3,\ 2]} = \mathbb{A}^1.$$

For $[2,\ 1,\ 3]$, we know from $\dim(E_2 \cap F_2) = 2$ that E_2 is F_2, and E_1 is contained in E_2 but E_1 is not F_1, so E_1 can be all the lines in $\langle e_1, e_2 \rangle$ except for $\langle e_1 \rangle$. So the Schubert cell corresponding to $[2,\ 1,\ 3]$ is $\mathbb{A}^1$,

$$X_{[2,\ 1,\ 3]} = \mathbb{A}^1.$$

For $[2,\ 3,\ 1]$, because E_1 is in F_2 but not in F_1, E_1 is the set of all lines through the origin in F_2 except for the line $\langle e_1 \rangle$. E_2 does not contain F_1, but it contains E_1, and it's intersection with F_2 is exactly

E_1. So E_2 is the set of all planes that contains E_1 except for F_2. The corresponding Schubert cell is an $\mathbb{A}^1$-bundle over $\mathbb{A}^1$, which is $\mathbb{A}^2$,

$$X_{[2,\ 3,\ 1]} = \mathbb{A}^2.$$

For $[3,\ 1,\ 2]$, E_1 is not in F_2, so E_1 is the set of all lines in $\mathbb{C}^3$ except for the lines in F_2. F_1 is in E_2. So E_2 is completely determined by two lines F_1 and E_1. E_1 is isomorphic to $\mathbb{A}^2$, so the Schubert cell is $\mathbb{A}^2$,

$$X_{[3,\ 1,\ 2]} = \mathbb{A}^2.$$

For $[3,\ 2,\ 1]$, again E_1 is not in F_2 so E_1 is the set of all lines in $\mathbb{C}^3$ except for the lines in F_2. E_2 does not contain F_1 but the dimension of intersection with F_2 is 1. So E_2 is the set of all planes in $\mathbb{C}^3$ that contains E_1 and the intersection of E_2 with F_2 is a line that is not F_1 in F_2. E_1 is isomorphic to $\mathbb{A}^2$ and E_2 brings another degree of freedom. the Schubert cell is an $\mathbb{A}^2$ bundle over $\mathbb{A}^1$, which is $\mathbb{A}^3$,

$$X_{[3,\ 2,\ 1]} = \mathbb{A}^3.$$

We summarize the discussion so far in the table below.

permutation (ω)	$[1,2,3]$	$[1,3,2]$	$[2,1,3]$	$[2,3,1]$	$[3,1,2]$	$[3,2,1]$
Schubert cell (X_ω)	$\mathbb{A}^0$	$\mathbb{A}^1$	$\mathbb{A}^1$	$\mathbb{A}^2$	$\mathbb{A}^2$	$\mathbb{A}^3$
cohomology	H^6	H^4	H^4	H^2	H^2	H^0.

In conclusion, we have shown that

$$H^*(Fl_3, \mathbb{Q}) = \mathbb{Q} \oplus 0 \oplus \mathbb{Q}^2 \oplus 0 \oplus \mathbb{Q}^2 \oplus 0 \oplus \mathbb{Q}.$$

In particular, $\dim H^*(Fl_3, \mathbb{Q}) = 6$, which is consistent with formula (4).

4. The coinvariant algebra and Schubert polynomials

In this section, we define the coinvariant algebra and its basis, the Schubert polynomials.

Definition 10. A polynomial $p \in \mathbb{Q}[x_1, \ldots, x_n]$ is *symmetric* if interchanging any two variables doesn't change the polynomial. More formally, the symmetric group S_n acts on $\mathbb{Q}[x_1, \ldots, x_n]$ by

$$\sigma * p = p(x_{\sigma(1)}, \ldots, x_{\sigma(n)}).$$

Then p is symmetric if $\sigma * p = p$ for all $\sigma \in S_n$.

For instance, the polynomial $x_1x_2x_3$ is symmetric in $\mathbb{Q}[x_1, x_2, x_3]$ but not symmetric in $\mathbb{Q}[x_1, \ldots, x_4]$, because in $\mathbb{Q}[x_1, \ldots, x_4]$ we can have $(3\ 4) * x_1x_2x_3 = x_1x_2x_4 \neq x_1x_2x_3$. The polynomial $x_1x_2 + x_3x_4 \in \mathbb{Q}[x_1, \ldots, x_4]$ is not symmetric because we can interchange x_1 with x_3 and have $x_2x_3 + x_1x_4 \neq x_1x_2 + x_3x_4$.

We now define the coinvariant algebra.

Definition 11. Let I denote the ideal generated by all the symmetric polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$. Then the quotient ring

$$R_n = \mathbb{Q}[x_1, \ldots, x_n]/I$$

is called the *coinvariant algebra*.

The coinvariant algebra has the set of Schubert polynomials as its basis. The *Schubert polynomials* are a set of polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$ that are defined recursively. For each permutation $\omega \in S_n$, there is a unique Schubert polynomial σ_ω defined with respect to it.

We use the square bracket $[a_1 \ldots a_n]$ to denote the one-line notation of a permutation.

Definition 12. The *base case of the Schubert polynomials* is

$$\sigma_{[n\ n-1\ \ldots\ 3\ 2\ 1]} = x_1^{n-1} x_2^{n-2} \cdots x_{n-1}.$$

The *recursion relation* is

$$\delta_i \sigma_\omega = \sigma_{\omega s_i} \quad \text{if} \quad \omega(i) > \omega(i+1),$$

where s_i is the transposition $(i\ \ i{+}1)$ that permutes the ith element with the $(i+1)$st element, and it also acts on $\mathbb{Q}[x_1, \ldots, x_n]$ as in Definition 10, and δ_i is the divided difference operator that acts on polynomials by

$$\delta_i(p) = \frac{p - s_i * p}{x_i - x_{i+1}}.$$

Remark 2. The identity permutation corresponds to the polynomial 1.

Remark 3. Despite being a fraction of polynomials in the definition, the Schubert polynomials are always polynomials with integer coefficients.

Example 3. We compute some of the Schubert polynomials in 3 variables. The base case is $\sigma_{[3\ 2\ 1]} = x_1^2 x_2$. Now want to use the divided difference operator on the base case to compute the Schubert polynomials of other permutations in S_n.

Notice that the 1st entry of [3 2 1] is 3 and it is greater than the 2nd entry of [3 2 1] which is 2. So let $i = 1$ and $s_i = (1\ 2)$, and by the recursion relation formula

$$\sigma_{[3\ 2\ 1](2\ 1)} = \delta_1 \sigma_{[3\ 2\ 1]} = \frac{\sigma[3\ 2\ 1] - (1\ 2) * \sigma[3\ 2\ 1]}{x_1 - x_2}$$

$$= \frac{x_1^2 x_2 - x_2^2 x_1}{x_1 - x_2} = x_1 x_2.$$

Therefore,

$$\sigma_{[2\ 3\ 1]} = \sigma_{(3\ 2\ 1)(2\ 1)} = x_1 x_2.$$

Similarly, apply δ_2 on $\sigma_{[2\ 3\ 1]}$ and $\sigma_{[3\ 2\ 1]}$. We can compute that

$$\sigma_{[2\ 1\ 3]} = \sigma_{[2\ 3\ 1](2\ 3)} = \frac{\sigma[2\ 3\ 1] - (2\ 3)\sigma[2\ 3\ 1]}{x_2 - x_3}$$
$$= \frac{x_1 x_2 - x_1 x_3}{x_2 - x_3} = x_1,$$

and

$$\sigma_{[3\ 1\ 2]} = \sigma_{[3\ 2\ 1](2\ 3)} = \delta_2 \sigma_{[3\ 2\ 1]} = \frac{\sigma[3\ 2\ 1] - (2\ 3)\sigma[3\ 2\ 1]}{x_2 - x_3}$$
$$= \frac{x_1^2 x_2 - x_1^2 x_3}{x_2 - x_3} = x_1^2.$$

The Schubert polynomials can be computed in Maple by the function called `schubertpoly` in the program in section 6, given a permutation as input.

Theorem 4. The Schubert polynomials form a basis for the coinvarant ring.

An immediate corollary is that the coinvariant ring has dimension $n!$ because there are $n!$ Schubert polynomials.

Definition 13. A polynomial $p \in \mathbb{Q}[x_1, \ldots, x_n]$ is *homogeneneous* of degree d if all the terms in the polynomial have the same degree d.

For instance, the polynomial $x_1^2 x_3^5 - x_4^7 + 6x_2 x_3 x_4^5 \in \mathbb{Q}[x_1, \ldots, x_4]$ is homogeneous of degree 7. The polynomial $x_2^3 + x_1 x_2 \in \mathbb{Q}[x_1, x_2, x_3]$ is not homogeneous because it contains terms of degrees 2 and 3.

Definition 14. A ring is *graded* if it can be written as a direct sum

$$R = \bigoplus_{i \geq 0} R_i$$

such that $R_i R_j \subset R_{i+j}$.

Remark 4. The coinvariant algebra is graded.

We have already seen that the coinvariant algebra and the cohomology ring of the flag variety have the same dimension $n!$, by Remark 5 and (4). The connection between the coinvariant algebra and the flag variety is explained by the following important theorem, proved in [2].

Theorem 5. The cohomology ring of the flag variety is isomorphic to the coinvariant algebra,

$$R \cong H^*(Fl_n, \mathbb{Q}).$$

Under this correspondence, the Schubert polynomials correspond precisely to the Schubert classes defined in the previous section. This result in Theorem 5 is extremely remarkable, however, proving this theorem is beyond the purpose of this senior thesis.

5. The open question

Due to geometric interpretations, when the product of two schubert polynomials is expressed as a linear combination of the Schubert polynomials, the all the coefficients are polynomials with integer coefficients.

Open Question 1. Find a combinatorial formula for these integers.

6. An Approach with the Grobner Basis

We wrote a program in Maple that computes the coefficients of the product of two Schubert polynomials in the Schubert polynomial basis.

We can use Maple to express every Schubert polynomial in the monomial basis G generated from the Grobner basis of $R = \mathbb{Q}[x_1, \ldots, x_n]/I$, where I is the symmetric ideal. Then record the coefficients and compile them into a matrix, and we have the $n!$ by $n!$ change-of-basis matrix M from Schubert polynomials to G. Use the same algorithms again to express the product of two Schubert polynomials in G, and then multiply the coefficients by M^{-1}, and we can find the coefficient of the product of Schubert polynomials in the Schubert polynomial basis. Indeed, the coefficients are all positive integers.

As an example, at the end of the Maple program attached below, the product of Schubert polynomials corresponding to [2 1 4 3] and [3 2 1 4] is computed to be the sum of the Schubert polynomials corresponding to [4 2 3 1] and [4 3 1 2].

References

[1] A. Hatcher, *Algebraic Topology*, Cambridge University Press (2001)
[2] W. Fulton, *Schubert Varieties and Degeneracy Loci*, Springer (1998)

```
> with(Groebner)
```

[Basis, FGLM, HilbertDimension, HilbertPolynomial, HilbertSeries, Homogenize, InitialForm, InterReduce, IsBasis, IsProper, IsZeroDimensional, LeadingCoefficient, LeadingMonomial, LeadingTerm, MatrixOrder, MaximalIndependentSet, MonomialOrder, MultiplicationMatrix, MultivariateCyclicVector, NormalForm, NormalSet, RationalUnivariateRepresentation, Reduce, RememberBasis, SPolynomial, Solve, SuggestVariableOrder, Support, TestOrder, ToricIdealBasis, TrailingTerm, UnivariatePolynomial, Walk, WeightedDegree] (1)

```
> with(combinat)
```

[Chi, bell, binomial, cartprod, character, choose, composition, conjpart, decodepart, encodepart, eulerian1, eulerian2, fibonacci, firstcomb, firstpart, firstperm, graycode, inttovec, lastcomb, lastpart, lastperm, multinomial, nextcomb, nextpart, nextperm, numbcomb, numbcomp, numbpart, numbperm, partition, permute, powerset, prevcomb, prevpart, prevperm, randcomb, randpart, randperm, rankcomb, rankperm, setpartition, stirling1, stirling2, subsets, unrankcomb, unrankperm, vectoint] (2)

```
> with(GroupTheory)
```

[<|>, AbelianGroup, AbelianInvariants, AllPerfectGroups, AllSmallGroups, AllTransitiveGroups, Alt, AlternatingGroup, AreConjugate, AreIsomorphic, BabyMonster, CayleyGraph, CayleyTable, CayleyTableGroup, Center, Centraliser, Centralizer, Centre, Character, CharacterTable, ClassNumber, ComplexProduct, ConjugacyClass, ConjugacyClasses, Conjugator, ConwayGroup, Core, CustomGroup, CycleIndexPolynomial, CyclicGroup, Degree, DerivedLength, DerivedSeries, DerivedSubgroup, DicyclicGroup, DihedralGroup, DirectProduct, DrawCayleyTable, DrawSubgroupLattice, ElementOrder, ElementPower, ElementaryGroup, Elements, Embedding, ExceptionalGroup, Exponent, FPGroup, Factor, FischerGroup, FittingSubgroup, FrattiniSubgroup, FreeGroup, GL, GaloisGroup, GeneralLinearGroup, GeneralOrthogonalGroup, GeneralUnitaryGroup, Generators, Group, GroupOrder, HaradaNortonGroup, HeldGroup, HigmanSimsGroup, Hypercenter, Hypercentre, IdentifySmallGroup, Index, Intersection, IsAbelian, IsAlternating, IsCommutative, IsCyclic, IsElementary, IsFinite, IsNilpotent, IsNormal, IsPGroup, IsPerfect, IsPermutable, IsPrimitive, IsQuasinormal, IsRegular, IsSimple, IsSoluble, IsSolvable, IsSubgroup, IsSubnormal, IsSymmetric, IsTransitive, JankoGroup, Labels, LeftCoset, LeftCosets, LowerCentralSeries, LyonsGroup, MathieuGroup, McLaughlinGroup, MetacyclicGroup, MinPermRepDegree, MinimumPermutationRepresentationDegree, Monster, NilpotencyClass, NilpotentResidual, NonRedundantGenerators, (3)

```
    NormalClosure, NormalSubgroups, Normaliser, NormaliserSubgroup,
    NormalizerSubgroup, NumAbelianGroups, NumGroups, NumPerfectGroups,
    NumTransitiveGroups, ONanGroup, Operations, Orbit, Orbits,
    OrthogonalGroup, PCore, PGL, PGU, PGroupPrime, PSL, PSU, PSp,
    PerfectGroup, PermApply, PermCommutator, PermCompare, PermConjugate,
    PermCycleType, PermDegree, PermFixed, PermInverse, PermLeftQuotient,
    PermOrder, PermParity, PermPower, PermProduct, PermRightQuotient,
    PermSupport, PermutationGroup, PresentationComplexity,
    ProjectiveGeneralLinearGroup, ProjectiveGeneralUnitaryGroup,
    ProjectiveSpecialLinearGroup, ProjectiveSpecialUnitaryGroup,
    ProjectiveSymplecticGroup, QuaternionGroup, RandomElement,
    RandomSmallGroup, Relators, RightCoset, RightCosets, RubiksCubeGroup,
    RudvalisGroup, SL, SearchSmallGroups, SearchTransitiveGroups, Simplify,
    SmallGroup, SolubleResidual, SolvableResidual, SpecialLinearGroup,
    SpecialOrthogonalGroup, SpecialUnitaryGroup, Stabiliser, Stabilizer,
    Subgroup, SubgroupLattice, SubgroupMembership, Supergroup, SuzukiGroup,
    SylowSubgroup, Symm, SymmetricGroup, SymplecticGroup, ThompsonGroup,
    TitsGroup, TransitiveGroup, Transitivity, TrivialGroup,
    UpperCentralSeries]
> with(PolynomialIdeals)
[ <, >, Add, Contract, EliminationIdeal, EquidimensionalDecomposition,      (4)
    Generators, HilbertDimension, IdealContainment, IdealInfo,
    IdealMembership, Intersect, IsMaximal, IsPrimary, IsPrime, IsProper,
    IsRadical, IsZeroDimensional, MaximalIndependentSet, Multiply,
    NumberOfSolutions, Operators, PolynomialIdeal, PrimaryDecomposition,
    PrimeDecomposition, Quotient, Radical, RadicalMembership, Saturate,
    Simplify, UnivariatePolynomial, VanishingIdeal,
    ZeroDimensionalDecomposition, in, subset]
> with(SolveTools)
[AbstractRootOfSolution, Basis, CancelInverses, Combine, Complexity, Engine,      (5)
    GreaterComplexity, Identity, Inequality, Linear, Parametric, Polynomial,
    PolynomialSystem, RationalCoefficients, SemiAlgebraic, SortByComplexity]
> with(LinearAlgebra)
[&x, Add, Adjoint, BackwardSubstitute, BandMatrix, Basis, BezoutMatrix,      (6)
    BidiagonalForm, BilinearForm, CARE, CharacteristicMatrix,
    CharacteristicPolynomial, Column, ColumnDimension, ColumnOperation,
    ColumnSpace, CompanionMatrix, CompressedSparseForm, ConditionNumber,
    ConstantMatrix, ConstantVector, Copy, CreatePermutation, CrossProduct,
    DARE, DeleteColumn, DeleteRow, Determinant, Diagonal, DiagonalMatrix,
    Dimension, Dimensions, DotProduct, EigenConditionNumbers, Eigenvalues,
```

```
    Eigenvectors, Equal, ForwardSubstitute, FrobeniusForm,
    FromCompressedSparseForm, FromSplitForm, GaussianElimination,
    GenerateEquations, GenerateMatrix, Generic, GetResultDataType,
    GetResultShape, GivensRotationMatrix, GramSchmidt, HankelMatrix,
    HermiteForm, HermitianTranspose, HessenbergForm, HilbertMatrix,
    HouseholderMatrix, IdentityMatrix, IntersectionBasis, IsDefinite,
    IsOrthogonal, IsSimilar, IsUnitary, JordanBlockMatrix, JordanForm,
    KroneckerProduct, LA_Main, LUDecomposition, LeastSquares, LinearSolve,
    LyapunovSolve, Map, Map2, MatrixAdd, MatrixExponential, MatrixFunction,
    MatrixInverse, MatrixMatrixMultiply, MatrixNorm, MatrixPower,
    MatrixScalarMultiply, MatrixVectorMultiply, MinimalPolynomial, Minor,
    Modular, Multiply, NoUserValue, Norm, Normalize, NullSpace,
    OuterProductMatrix, Permanent, Pivot, PopovForm, ProjectionMatrix,
    QRDecomposition, RandomMatrix, RandomVector, Rank, RationalCanonicalForm,
    ReducedRowEchelonForm, Row, RowDimension, RowOperation, RowSpace,
    ScalarMatrix, ScalarMultiply, ScalarVector, SchurForm, SingularValues,
    SmithForm, SplitForm, StronglyConnectedBlocks, SubMatrix, SubVector,
    SumBasis, SylvesterMatrix, SylvesterSolve, ToeplitzMatrix, Trace,
    Transpose, TridiagonalForm, UnitVector, VandermondeMatrix, VectorAdd,
    VectorAngle, VectorMatrixMultiply, VectorNorm, VectorScalarMultiply,
    ZeroMatrix, ZeroVector, Zip]
```

```
> #First, we want to write a function that computes the base case of Schubert
    polynomials. This function takes the number of variables as input and
    returns the base case of the Schubert polynomials.
```

```
> basecase:=proc(n)
   local p, i;
   p:=1;
   for i from 1 to n do
          p := p*x[i]^(n-i)
           end do;
   return(p); end proc;
```

$$\mathit{basecase} := \mathbf{proc}(n) \quad \mathbf{(7)}$$

```
   local p, i;
   p := 1; for i to n do p := p*x[i]^(n - i) end do; return p
end proc
```

```
> #As an example, if we want to compute the base case of Schubert polynomials of
    4 variables (the polynomial that correspond to the permutation [4,3,2,1],)
    then we compute:
```

```
> basecase(4)
```

$$x_1^3\, x_2^2\, x_3 \quad \mathbf{(8)}$$

```
>
```

```
> #The secound tool we wrote is the part in the divided difference operator that
    takes a polynomial and interchanegs the ith and (i+1)th variable, then
    returns the new polynomial:
> divdiffoperator := proc (p, i)
        local k, s;
        k := eval(p, [x[i] = s, x[i+1] = x[i]]);
        k := eval(k, s = x[i+1]);
  return k; end proc;
```

$$divdiffoperator := \mathbf{proc}(p,\ i) \quad \textbf{(9)}$$

```
  local k, s;
  k := eval(p, [x[i] = s, x[i+1] = x[i]]); k := eval(k, s = x[i+1]); return
  k
end proc
```

```
> #example:
> divdiffoperator(x[1]^3*x[2]+x[3]*x[4]^2, 3)
```

$$x_2\ x_1^3 + x_3^2\ x_4 \quad \textbf{(10)}$$

```
> #Finally, we start to compute the Schubert polynomials recursively
    . This function schubertpoly takes a permutation t as input, and
    if the permutation t is the base case =[seq(n .. 1, -1)],
    then the function returns the base case polynomial
    . If the permutation t is not the base case,
    then compute the Schubert polynomials recursively.
>
> #We first observe that, for example, if we want to compute the Schubert
    polynomial of t=[2, 4, 1, 3], then according to the definition,
     we should step by step move the larger number to the front in every pair
    ·(2, 4), (4, 1)(1, 3) in this permutation t. So the procdure is :
     schubertpoly([2, 4, 1, 3]) =calculated
    by the relation with schubertpoly([4, 2, 1, 3]) =calculated
    by the relation with schubertpoly([4, 2, 3, 1]) =calculated
    by the relation with schubertpoly([4, 3, 2, 1]). Essentially,
     the list h is tracking the moves of every larger number in the pairs
    in the permutation to the front,
    and the Schubert polynomial of t can be computed by tracing back
    to each of the different moves in h, and eventually to the base case.
> schubertpoly := proc (t)
        local p, s, h, q, i, r, n;
        n := numelems(t);
        p := basecase(n);
        s := [seq(n .. 1, -1)]; h := t; q := 0;


        if h = s then return p
        else
              for i to n-1 do
```

```
                        if h[i] < h[i+1]
                        then
                                q := h[i];
                                h[i] := h[i+1]; h[i+1] := q;
                                r := (schubertpoly(h, n)
    -divdiffoperator(schubertpoly(h, n), i))/(x[i]-x[i+1]);
                                r := simplify(r);
          return r;
          end if; end do; end if;
  end proc;
schubertpoly := proc(t)                                          (11)
   local p, s, h, q, i, r, n;
   n := numelems(t);
   p := basecase(n);
   s := [seq(n..1, -1)];
   h := t;
   q := 0;
   if h = s then
      return p
   else
      for i to n - 1 do
         if h[i] < h[i+1] then
            q := h[i];
            h[i] := h[i+1];
            h[i+1] := q;
            r := (schubertpoly(h, n) - divdiffoperator(schubertpoly(h,
            n), i))/(x[i] - x[i+1]);
            r := simplify(r);
            return r
         end if
      end do
   end if
end proc
>
> #Here is an example of the computation of all the Schubert polynomials of
    permutations of 4 variables. The permutation is printed first and then its
    corresponding Schubert polynomial is printed:
> for t in permute(4) do
   print(t);
   expand(schubertpoly(t));
   end do
                                [1, 2, 3, 4]
```

$$\mathbf{1}$$

$$[1,\ 2,\ 4,\ 3]$$

$$x_1 + x_2 + x_3$$

$$[1,\ 3,\ 2,\ 4]$$

$$x_1 + x_2$$

$$[1,\ 3,\ 4,\ 2]$$

$$x_1\ \ x_2 + x_3\ \ x_1 + x_2\ \ x_3$$

$$[1,\ 4,\ 2,\ 3]$$

$$x_1^2 + x_1\ \ x_2 + x_2^2$$

$$[1,\ 4,\ 3,\ 2]$$

$$x_1^2\ \ x_2 + x_1^2\ \ x_3 + x_2^2\ \ x_1 + x_1\ \ x_2\ \ x_3 + x_2^2\ \ x_3$$

$$[2,\ 1,\ 3,\ 4]$$

$$\boldsymbol{x_1}$$

$$[2,\ 1,\ 4,\ 3]$$

$$x_1^2 + x_1\ \ x_2 + x_3\ \ x_1$$

$$[2,\ 3,\ 1,\ 4]$$

$$x_1\ \ x_2$$

$$[2,\ 3,\ 4,\ 1]$$

$$x_1\ \ x_2\ \ x_3$$

$$[2,\ 4,\ 1,\ 3]$$

$$x_1^2\ \ x_2 + x_2^2\ \ x_1$$

$$[2,\ 4,\ 3,\ 1]$$

$$x_1^2\ \ x_2\ \ x_3 + x_2^2\ \ x_1\ \ x_3$$

$$[3,\ 1,\ 2,\ 4]$$

$$\boldsymbol{x_1^2}$$

$$[3,\ 1,\ 4,\ 2]$$

$$x_1^2\ \ x_2 + x_1^2\ \ x_3$$

$$[3,\ 2,\ 1,\ 4]$$

$$x_1^2\ \ x_2$$

$$[3,\ 2,\ 4,\ 1]$$

$$x_1^2\ \ x_2\ \ x_3$$

$$[3,\ 4,\ 1,\ 2]$$

$$x_1^2\ \ x_2^2$$

$$[3,\ 4,\ 2,\ 1]$$

$$x_1^2\ \ x_2^2\ \ x_3$$

$$[4,\ 1,\ 2,\ 3]$$

$$x_1^3$$

$$[4,\ 1,\ 3,\ 2]$$

$$x_2\ x_1^3 + x_3\ x_1^3$$

$$[4,\ 2,\ 1,\ 3]$$

$$x_2\ x_1^3$$

$$[4,\ 2,\ 3,\ 1]$$

$$x_1^3\ x_2\ x_3$$

$$[4,\ 3,\ 1,\ 2]$$

$$x_1^3\ x_2^2$$

$$[4,\ 3,\ 2,\ 1]$$

$$x_1^3\ x_2^2\ x_3 \qquad (12)$$

```
>
```

```
> #Next, we want to generate the Grobner basis for the symmetric ideal. The
    generators of the symmetric ideal are [x[1]+..x[n], x[1]^2+...+x[n]^2,...,
    x[1]^n+...+x[n]^n]. The generators are computed in the list t
    . p is the list [x[1],..., x[n]], and we use the Basis tool
    in the Grobner basis package in Maple
    to compute the Grobner basis of the symmetric ideal.
```

```
>
```

```
> generategrobner:=proc(n)
   local t, i, h, j, p, F;
   t:=[ ]; p:=[ ];
   for i from 1 to n do
   h:=0;
   for j from 1 to n do
  h:= h+x[j]^i; end do;
   t:=[op(t), h];end do;

   for i from 1 to n do
   p:=[op(p), x[i]];end do;

  F:=Basis(t, plex(op(p)));
  return(F); end proc;
```

```
generategrobner := proc(n)                                    (13)
   local t, i, h, j, p, F;
   t := [ ];
   p := [ ];
   for i to n do
      h := 0; for j to n do h := h+x[j]^i end do; t := [op(t), h]
   end do;
   for i to n do p := [op(p), x[i]] end do;
```

```
    F := Groebner:-Basis(t, plex(op(p)));
    return F
end proc
```

```
> #As an example, we compute the Grobner basis of the symmetric ideal of 3,4 and
    5 variables.
```

```
> generategrobner(3)
```

$$\left[x_3^3,\ x_2^2+x_2\ x_3+x_3^2,\ x_1+x_2+x_3\right] \tag{14}$$

```
> generategrobner(4)
```

$$\left[x_4^4,\ x_3^3+x_3^2\ x_4+x_3\ x_4^2+x_4^3,\ x_2^2+x_2\ x_3+x_2\ x_4+x_3^2+x_3\ x_4+x_4^2,\ x_1+x_2+x_3+x_4\right] \tag{15}$$

```
> generategrobner(5)
```

$$\left[x_5^5,\ x_4^4+x_4^3\ x_5+x_4^2\ x_5^2+x_4\ x_5^3+x_5^4,\ x_3^3+x_3^2\ x_4+x_3^2\ x_5+x_3\ x_4^2+x_3\ x_4\ x_5+x_3\ x_5^2+x_4^3\right. \tag{16}$$
$$+x_4^2\ x_5+x_4\ x_5^2+x_5^3,\ x_2^2+x_2\ x_3+x_2\ x_4+x_2\ x_5+x_3^2+x_3\ x_4+x_3\ x_5+x_4^2+x_4\ x_5+$$
$$\left.x_5^2,\ x_1+x_2+x_3+x_4+x_5\right]$$

```
> #From the examples, we can see the pattern that there exists a basis for the
    quotient ring Q[x[1],...x[n]] mod the symmetric ideal of n variables,
    and the basis is the set of all monomials in the form x[1]^a1·...·x[n]^an,
     where ai < i for all i. We call this basis G
    and generate it with the following function.
```

```
>
```

```
> generatebasisG :=proc(n)
    local l, h, i, m, j, t;
   l := [1];
    if n > 1 then

    for i from 2 to n do
    m := [ ];
    for j from 2 to i do
    for t in l do
    m := [op(m), t·x[i]^(j-1)]; end do;
    end do; l := [op(l), op(m)];end do; end if;
    return(l); end proc;
```

```
generatebasisG := proc(n)                                                    (17)
    local l, h, i, m, j, t;
    l := [1];
    if 1 < n then
        for i from 2 to n do
            m := [ ];
            for j from 2 to i do
                for t in l do
                    m := [op(m), t*x[i]^(j - 1)]
```

```
            end do
        end do;
        l := [op(l), op(m)]
    end do
  end if;
  return l
end proc
```

```
> #Examples:
```

```
> generatebasisG(3)
```

$$\left[1,\ x_2,\ x_3,\ x_2\ x_3,\ x_3^2,\ x_2\ x_3^2\right] \tag{18}$$

```
> numelems(generatebasisG(3))
```

$$6 \tag{19}$$

```
> generatebasisG(4)
```

$$\left[1,\ x_2,\ x_3,\ x_2\ x_3,\ x_3^2,\ x_2\ x_3^2,\ x_4,\ x_2\ x_4,\ x_3\ x_4,\ x_2\ x_3\ x_4,\ x_3^2\ x_4,\ x_2\ x_3^2\ x_4,\ x_4^2,\ x_2\ x_4^2,\ x_3\ x_4^2,\ x_2\ x_3\ x_4^2,\ x_3^2\ x_4^2,\ x_2\ x_3^2\ x_4^2,\ x_4^3,\ x_2\ x_4^3,\ x_3\ x_4^3,\ x_2\ x_3\ x_4^3,\ x_3^2\ x_4^3,\ x_2\ x_3^2\ x_4^3\right] \tag{20}$$

```
> numelems(generatebasisG(4))
```

$$24 \tag{21}$$

```
> #numelems computes the number of elements in a list. We can see indeed that
    there are n! number of basis elements in the list, which matchs with the
    dimension of the quotient ring.
>
> #Now, we want a function that takes any polynomial along with the number of
    variables of the polynomial ring as input, and returns the list of
    coefficients of the polynomial written as a linear combination of the basis
    G. To do this, we first use the Maple tool NormalForm to reduce the part in
    the symmetric ideal of the input polynomial, and then we use the Maple tool
    RationalCoefficients to write the reduced polynomial as a linear
    combination of the basis G.
> converttobasisG :=proc(a, n)
  local p, i, A;

  p := [ ];

 for i from 1 to n do
  p := [op(p), x[i]];end do;


  A := NormalForm(expand(a), generategrobner(n), plex(op(p)));


  return(RationalCoefficients(A, generatebasisG(n)));

  end proc;
converttobasisG := proc(a, n)
```

(22)

```
    local p, i, A;
    p := [ ];
    for i to n do p := [op(p), x[i]] end do;
    A := Groebner:-NormalForm(expand(a), generategrobner(n), plex(op(p)));
    return SolveTools:-RationalCoefficients(A, generatebasisG(n))
end proc
```

```
> #Example:
> converttobasisG(x[1]^2*x[2], 3)
```

$$[0,\ 0,\ 0,\ 0,\ 0,\ -1]\ -x_2\ x_3^2 \tag{23}$$

```
> #x[1]^2*x[2] is normalized to be -x[2]x[3]^2, which is -1 times the last element
      in the basis
>
> #Now we want a change of basis matrix M. So when we write the product of two
      Schubert polynomials as a linear combination of the basis G, then we
      multiply M with the vector of coefficients, then we will have the
      coefficient vector of the product in the basis of Schubert polynomials.
> #To compute the matrix M, we first use the convertobasisG function to convert
      every Schubert polynomial as a linear combination of the basis G. Then we
      record the coefficient of the linear combincation of the first Schubert
      polynomial in the first row, the second Schubert polynomial in the second
      row and so on. By basic calculation in linear algebra, the transpose and
      inverse of this matrix is the matrix M we want.
>
> changeofbasis :=proc(n)
   local L, t, M;
   L := [ ];

   for t in permute(n) do
  L := [op(L), converttobasisG(expand(schubertpoly(t)), n)];
  end do;
   M := Matrix(L);
   M := Transpose(M);
   M := MatrixInverse(M);
  return (M);
  end proc;
```

```
changeofbasis := proc(n)
    local L, t, M;
    L := [ ];
    for t in combinat:-permute(n) do
        L := [op(L), converttobasisG(expand(schubertpoly(t)), n)]
    end do;
    M := Matrix(L);
```
(24)

```
    M := LinearAlgebra:-Transpose(M);
    M := MatrixInverse(M);
    return M
end proc
```

```
> #Example: We will always get a n! by n! change of basis matrix
> changeofbasis(3)
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{25}$$

```
> #Now, here is an example of computing the linear combination of the product of
    two Schubert polynomials in the basis of Schubert polynomials
> #We first pick two random Schubert polynomials, schubertpoly([2, 1, 4, 3])
    and schubertpoly([3, 2, 1, 4]). Multiply them
    and convert the resulting polynomial into the linear combination of basis G
    . We will get a list of 24 entries and we convert the list into a vector K.
>
> K := Vector[column](converttobasisG((schubertpoly([2, 1, 4, 3])·schubertpoly([3,
    2, 1, 4])), 4))
```

$$K := \begin{bmatrix} 1 \;..\; 24 \; Vector_{column} \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix} \tag{26}$$

```
> #Now multiply the vector K with the change of basis matrix
> P := MatrixVectorMultiply(changeofbasis(4), K)
```

$$P := \begin{bmatrix} 1 \;..\; 24 \; Vector_{column} \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran\_order \end{bmatrix} \tag{27}$$

```
> P[1..10]
```

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (28)$$

```
> P[11..20]
```

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (29)$$

```
> P[21..24]
```

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (30)$$

```
> #From the result, we see that the product of schubertpoly([2, 1, 4, 3])
    and schubertpoly([3, 2, 1, 4]) should be the sum of the third to last
    and the second to last Schubert polynomials.
>
> expand(schubertpoly([2, 1, 4, 3])·schubertpoly([3, 2, 1, 4]))
```

$$x_1^4\ x_2 + x_1^3\ x_2^2 + x_1^3\ x_2\ x_3 \quad (31)$$

```
> permute(4)
```

$$[[1, 2, 3, 4], [1, 2, 4, 3], [1, 3, 2, 4], [1, 3, 4, 2], [1, 4, 2, 3], [1, 4, 3, 2], \quad (32)$$

```
[2, 1, 3, 4], [2, 1, 4, 3], [2, 3, 1, 4], [2, 3, 4, 1], [2, 4, 1, 3], [2, 4, 3,
1], [3, 1, 2, 4], [3, 1, 4, 2], [3, 2, 1, 4], [3, 2, 4, 1], [3, 4, 1, 2], [3, 4,
2, 1], [4, 1, 2, 3], [4, 1, 3, 2], [4, 2, 1, 3], [4, 2, 3, 1], [4, 3, 1, 2], [4,
3, 2, 1]]
```

```
> schubertpoly([4, 2, 3, 1], 4) + schubertpoly([4, 3, 1, 2], 4)
```

$$x_1^3\ x_2^2 + x_1^3\ x_2\ x_3 \tag{33}$$

```
> NormalForm(x[2]x[1]^4, generategrobner(4), plex(x[1], x[2], x[3], x[4]))
```

$$0 \tag{34}$$

```
> #We see that the product of schubertpoly([2, 1, 4, 3]) and schubertpoly([3, 2,
    1, 4]) is indeed the sum of schubertpoly([4, 2, 3, 1], 4)
    and schubertpoly([4, 3, 1, 2], 4) except for the term x[1]^4x[2],
     but the NormalForm computation shows that x[1]^4x[2] is
    in the symmetric ideal, so the calculation is indeed correct.
>
>
```