RWTH AACHEN UNIVERSITY

Master's thesis

Bruhat Decomposition in Orthogonal Groups over Finite Fields

Daniel Rademacher

September 25, 2020

supervised by Prof. Dr. Alice Niemeyer
and Dominik Bernhardt M. Sc.

The present work was submitted to

Chair of Algebra and Representation Theory

RWTH Aachen University

Contents

1 Introduction

Thematically related to my Bachelor's thesis *Bruhat Decomposition in Unitary and Symplectic Groups over Finite Fields* [DR] we deal with the orthogonal group in this Master's thesis and pursue the same two goals. Firstly, we introduce the Bruhat decomposition for the orthogonal group over finite fields and prove the correctness of an algorithm for computing this decomposition. Secondly, we describe a procedure to write every element of the orthogonal group as a word in specific generators, namely the Leedham-Green-O'Brien standard generators (LGO) [LGO]. The basic procedure for the special linear group is described in the paper *Straight-line programs with memory and matrix Bruhat decomposition* by A. C. Niemeyer, T. Popiel and C. E. Praeger [NPP].

Before we embark on describing how to achieve our goals, we give basic definitions and introduce notation in the second chapter. In the third chapter we summarise the beforenamed paper and discuss the general approach to design an algorithm for our goals.
Depending on the quadratic form on the underlying vector space, one distinguishes three types of orthogonal groups, namely SO^+ (orthogonal group of plus type), SO^- (orthogonal group of minus type) and SO° (orthogonal group of circle type), where the Witt index plays a central role and is discussed in more depth in Chapter 2. We have to consider each of these cases separately since the dimension of the anisotropic space leads to different transformations and thus to different algorithms. The fourth chapter deals with the orthogonal group of plus type whereas the fifth chapter examines the circle type. In this thesis, the results for the orthogonal group of minus type are not presented, but will be considered independently later. The fourth and fifth chapter are structured identically as follows.
In the first subsection we start by getting to know the group of respective type better. An important element is a standardized bilinear form on the underlying finite dimensional vector space over a finite field on which we want to analyze the group for the rest of the chapter. In addition, we examine a homomorphism, in particular a monomorphism, from the special linear group to the special orthogonal group in order to gain an initial intuition about the format of the matrices. In the second subsection we deepen this understanding and construct additional matrices that are sufficient for the desired method. Therefore, we can specify an algorithm to compute the Bruhat decomposition in the third subsection and prove its correctness. The order is chosen such that the difficulties of the algorithms are increasing, i.e., we first deal with the plus and then with the circle type. In the final part of each chapter, we express every element of the orthogonal group of corresponding type as a word in specified generators, namely the LGO standard generators. In order to use these generators, we apply a base change to transform the matrices into a form that matches our previous definition of the bilinear form. Then we start with the monomial matrices from the algorithm for the Bruhat decomposition as we can assume that we can find two (unitriangular) matrices b_1 and b_2 such that for a matrix a in the orthogonal group we have

$$b_1 \cdot a \cdot b_2 = m$$

where m is a monomial matrix in the same group as a. Thus, if we can write the matrices b_1, b_2 and m in terms of standard generators, we know exactly what a looks like since

$$a = b_1^{-1} \cdot m \cdot b_2^{-1}.$$

As b_1 and b_2 are the product of matrices computed by the Bruhat decomposition algorithm, we consider the matrices used there, write them as words of the LGO standard generators and can, therefore, directly represent b_1 and b_2.
We know that it is also necessary to be able to describe monomial matrices in the orthogonal group as words in specified generators. To achieve this, we proceed as, again, in the aforementioned paper *Straight-line programs with memory and matrix Bruhat decomposition* [NPP] and write a monomial matrix m' in terms of the LGO generators such that

$$m' \cdot m = h,$$

where h is a diagonal matrix. The algorithm to achieve this is simpler for the circle type than for the plus type. Finally, we write the diagonal matrix h as a word of LGO generators and obtain

$$a = b_1^{-1} \cdot (m')^{-1} \cdot h \cdot b_2^{-1}.$$

In the first three parts of Chapter 4 and 5 we see that the Bruhat decomposition of the special linear group consisting of monomial matrices and lower unitriangular matrices also exists for the orthogonal group of plus and circle type, which is why we also want to realize this decomposition there. In the orthogonal group of minus type, the anisotropic space enforces that the restriction to lower triangular matrices is too large. As already mentioned above, this is not further discussed in this thesis.

The results achieved in this thesis are relevant to a larger project in the area of computational group theory, namely the matrix group recognition problem [BHGO]. To show the connection to the matrix group recognition problem, we consider the following situation. Let $a_1, \ldots, a_k \in \mathrm{GL}(n, q)$ and $G = \langle a_1, \ldots, a_k \rangle$. The aim of the matrix group recognition project is to acquire information about a matrix group like G, such as the order. We also want to solve the "membership problem" that is to decide if a matrix $a \in \mathrm{GL}(n, q)$ is in the matrix group G and, if so, to solve the "word problem" that is to write a as a word in the generators $a_1, \ldots, a_k$.
In order to solve this problem, Aschbacher's Theorem [MA] is employed to seek a homomorphism from a given group to a group lying in one of the Aschbacher families. The problem of determining information about the original group G is then reduced to determining information about two new groups, namely the image of G under the homomorphism and its kernel. This process can be repeated with the two new groups, forming a tree structure. The process ends when only certain groups remain as leaves in which the classical groups play an

important role. To decide whether any matrix is in the special orthogonal group and, if so, how to write the element with particular generators, can be solved with the results of this thesis.

All algorithms presented in this thesis are implemented by the author in GAP [GAP] for the first time and are available in the package `BruhatDecomposition`. We use the concept of a straight-line program with memory (MSLP) to store words in a generator set of a group efficiently. We analyze the maximum length and number of slots needed for the MSLP constructed in our implementation. The main theorems about the existence of an MSLP for the Bruhat decomposition are Theorem 4.27 for the orthogonal group of plus type and Theorem 5.29 for the orthogonal group of circle type.

The membership problem in classical groups has also been solved and implemented by Csaba Schneider in Magma [Magma] with the results of Elliot Costi. However, they used a completely different method which did not involve the Bruhat decomposition. The algorithms presented in this thesis are new and have not previously been published.

It should also be mentioned that some aspects this thesis builds upon my Bachelor's thesis *Bruhat Decomposition in Unitary and Symplectic Groups over Finite Fields* [DR] and, therefore, uses results from it.

Index of Symbols

Symbol	Meaning
G, H, U	Groups
g	Element of a group
$\varphi, \phi, \tau, \kappa$	Maps
V	Vector space
$\mathcal{B}$	Basis of a vector space
Φ	Bilinear form
Q	Quadratic form
$\mathrm{Bifo}(V)/\,\mathrm{Qfo}(V)$	The set of all bilinear forms/quadratic forms on a vector space V
GL	General linear group
$\mathrm{End}(V)$	Group of homomorphisms from V to V
SO	Special orthogonal group
O	Orthogonal group
a, b, c	Matrices
I_n	$\mathrm{diag}(1, \ldots, 1)$
$E_{i,j}$	A matrix with a 0 in every position except for a 1 in position (i, j)
n	Dimension of a vector space
m	Mostly $n = 2m$, $n = 2m + 1$ or $n = 2m + 2$
$e_1, \ldots, e_n$	Standard basis of $\mathbb{F}_q^n$
h	Diagonal matrix
q	Prime power
p	Prime number
f	Natural number (Mostly $q = p^f$)
ν, w, z	Elements of a vector space
$\mathbb{F}$	A finite field
$\imath, \jmath$	Elements of a field
ω	Primitive element of a field
ℓ	Commonly used as an exponent
$a_{i,-}$	ith row of a
$a_{-,j}$	jth column of a
$.^L$	Multiplication from left
$.^R$	Multiplication from right
$I_{i,j}(\imath)$	Transvection in the special linear group
$T_{i,j}(\imath)$	Transformation in the special orthogonal group
$s, s', t, t', \delta, \delta', u, v, \sigma$	LGO standard generators (see Chapter 4 and 5)
δ_*	$\mathrm{diag}(\omega, 1, \ldots, 1, \omega^{-1})$
$J_n, J^\circ_{n,q}$	Gram-matrix in special orthogonal group
$M - N$	$M \setminus N$ for sets M and N
v^φ	$\varphi(v)$
V^*	Dual space of a vector space V
i, j, k	Indices

2 Background

This chapter introduces basic definitions and notations that are important in all subsequent chapters. First, we discuss the mathematical aspects while the second section of this chapter deals with the programming aspects of this thesis. The material in this chapter is well known and can, for example, be found in [Taylor].

2.1 Preliminaries

This subsection introduces the basic mathematical definitions and expressions of this thesis.

First, we give some definitions that are needed later to describe the classical groups. For the remainder of this chapter, $\mathbb{F}$ denotes a finite field of order q.

Definition 2.1: Let V be an $\mathbb{F}$-vector space.
A map $\Phi\colon V \times V \to \mathbb{F}, (\nu, w) \mapsto \Phi(\nu, w)$ is called a *bilinear form* on V, if Φ is linear in each component, which is to say

(i) $\Phi(\imath\nu_1 + \jmath\nu_2, w) = \imath \cdot \Phi(\nu_1, w) + \jmath \cdot \Phi(\nu_2, w)$ for all $\imath, \jmath \in \mathbb{F}$ and $\nu_1, \nu_2, w \in V$,

(ii) $\Phi(\nu, \imath w_1 + \jmath w_2) = \imath \cdot \Phi(\nu, w_1) + \jmath \cdot \Phi(\nu, w_2)$ for all $\imath, \jmath \in \mathbb{F}$ and $\nu, w_1, w_2 \in V$.

We define $\mathrm{Bifo}(V)$ as the set of all bilinear forms on V.

Remark 2.2: From now on, we will refer to a bilinear form only as a *form*.

In contrast to the unitary and symplectic group, the orthogonal group cannot be defined by bilinear forms for general characteristics. However, for finite fields whose characteristic is different from 2, we can consider the corresponding bilinear form.

Definition 2.3: Let V be an $\mathbb{F}$-vector space.

1.) A bilinear form $\Phi \in \mathrm{Bifo}(V)$ is called *non-singular* or *non-degenerate* if for each $\nu \in V - \{0\}$ there is a $w \in V$ with $\Phi(\nu, w) \neq 0$.

2.) A bilinear form $\Phi \in \mathrm{Bifo}(V)$ is called *symmetric* if $\Phi(\nu, w) = \Phi(w, \nu)$ for all $\nu, w \in V$. The set of symmetric bilinear forms on V is denoted by $\mathrm{Bifo}_+(V)$.

3.) If $\mathrm{char}(\mathbb{F}) \neq 2$, then a non-singular and symmetric bilinear form $\Phi \in \mathrm{Bifo}(V)$ is called *orthogonal*.

In order to analyze orthogonal groups in characteristic 2 and to give the general definition of orthogonal groups, we still need the concept of quadratic forms.

Definition 2.4: A map $Q\colon V \to \mathbb{F}$ is a *quadratic form* if there is an associated bilinear form Φ_Q, called the *polar form* of Q, such that

1.) $Q(\lambda\nu) = \lambda^2 Q(\nu)$ for all $\lambda \in \mathbb{F}, \nu \in V$ and

2.) $\Phi_Q(\nu, w) = Q(\nu + w) - Q(\nu) - Q(w)$ for all $\nu, w \in V$.

The set of all quadratic forms on V is denoted by $\mathrm{Qfo}(V)$. A quadratic form $Q \in \mathrm{Qfo}(V)$ is called *non-degenerate* if the polar form of Q is non-singular.

Now, we consider some connections between quadratic forms and the corresponding polar form.

Remark 2.5: Let Q be a quadratic form on an $\mathbb{F}$-vector space V and Φ_Q the polar form of Q.

1.) Φ_Q is uniquely determined by condition 2.) of Definition 2.4. Moreover, Φ_Q is symmetric since

$$\Phi_Q(\nu, w) = Q(\nu + w) - Q(\nu) - Q(w) = Q(w + \nu) - Q(w) - Q(\nu) = \Phi_Q(w, \nu).$$

2.) Let $\mathrm{char}(\mathbb{F}) \neq 2$ and Φ be a symmetric form on V. Define $Q_\Phi\colon V \to \mathbb{F}, \nu \mapsto 2^{-1}\Phi(\nu, \nu)$. We check whether Q_Φ is a quadratic form with polar form Φ.

1.) $Q_\Phi(\lambda\nu) = 2^{-1}\Phi(\lambda\nu, \lambda\nu) = \lambda^2 2^{-1}\Phi(\nu, \nu) = \lambda^2 Q_\Phi(\nu)$ for all $\lambda \in \mathbb{F}, \nu \in V$.

2.) For all $\nu, w \in V$ we have

$$\begin{aligned} & Q_\Phi(\nu + w) - Q_\Phi(\nu) - Q_\Phi(w) \\ &= 2^{-1}(\Phi(\nu + w, \nu + w) - \Phi(\nu, \nu) - \Phi(w, w)) \\ &= 2^{-1}(\Phi(\nu, \nu) + \Phi(w, w) + 2\Phi(\nu, w) - \Phi(\nu, \nu) - \Phi(w, w)) \\ &= 2^{-1}2\Phi(\nu, w) = \Phi(\nu, w). \end{aligned}$$

Hence, Q_Φ defines a quadratic form.

There are special endomorphisms on an $\mathbb{F}$-vector space equipped with a bilinear or quadratic form, namely isometries, which are used to define the classical groups. To define these, we need another definition beforehand.

Definition 2.6: Let V be an n-dimensional $\mathbb{F}$-vector space. Define

1.) $\mathrm{End}(V) \coloneqq \{\varphi \mid \varphi\colon V \to V,\ \varphi \text{ is a vector space homormorphism}\}$,

2.) $\mathrm{GL}(V) \coloneqq \{\varphi \mid \varphi\colon V \to V,\ \varphi \text{ is an endomorphism and bijective}\}$,

3.) $\mathrm{GL}(n,q) \coloneqq \{a \in \mathbb{F}_q^{n\times n} \mid a \text{ is invertible}\}$,

4.) $\mathrm{SL}(n,q) \coloneqq \{a \in \mathbb{F}_q^{n\times n} \mid a \text{ is invertible and } \det(a) = 1\}$.

Remark 2.7:

1.) Notice that $\mathrm{GL}(n,q)$ is well-defined since every finite field of order $q = p^f$ is unique up to isomorphism.

2.) $\mathrm{End}(V)$ and $\mathrm{GL}(V)$ are groups with composition. $\mathrm{GL}(n,q)$ and $\mathrm{SL}(n,q)$ are groups with matrix multiplication.

3.) Note that in this thesis groups act on the right.

There is an important connection between these groups.

Theorem 2.8: Let V be an n-dimensional $\mathbb{F}$-vector space. Let $\mathcal{B} \in V^n$ be a basis of V. Then the map

$$\mathrm{End}(V) \to \mathbb{F}^{n\times n}, \varphi \mapsto {}^{\mathcal{B}}\varphi^{\mathcal{B}}$$

is an isomorphism.

Corollary 2.9: Let V be an n-dimensional $\mathbb{F}$-vector space. Let $\mathcal{B} \in V^n$ be a basis of V. Then

$$\mathrm{GL}(V) \to \mathrm{GL}(n,q), \varphi \mapsto {}^{\mathcal{B}}\varphi^{\mathcal{B}}$$

is an isomorphism.

We are now in a position to define the special endomorphisms that yield the definition of classical groups.

Definition 2.10: Let (V, Φ) be an $\mathbb{F}$-vector space equipped with a form Φ or equipped with a quadratic form Q. A map $\varphi \in \mathrm{End}(V)$ is called an *isometry* from V to V if $\Phi(\nu^\varphi, w^\varphi) = \Phi(\nu, w)$ for all $\nu, w \in V$ or $Q(\nu^\varphi) = Q(\nu)$ for all $\nu \in V$. The group of all bijective isometries of (V, Φ) or (V, Q) is denoted by $I(V, \mathbb{F}, \Phi)$ or $I(V, \mathbb{F}, Q)$.
The subgroup $S(V, \mathbb{F}, \Phi) := I(V, \mathbb{F}, \Phi) \cap \mathrm{SL}(V)$ or $S(V, \mathbb{F}, Q) := I(V, \mathbb{F}, Q) \cap \mathrm{SL}(V)$ is called the group of *special isometries* of V.

Remark 2.11: On one hand let φ be a bijective isometry on an $\mathbb{F}$-vector space V equipped with a quadratic form Q and Φ_Q be the polar form of Q. Then $Q(\nu^\varphi) = Q(\nu)$ holds for all $\nu \in V$. For the polar form Φ_Q we have

$$\begin{aligned}\Phi_Q(\nu^\varphi, w^\varphi) &= Q(\nu^\varphi + w^\varphi) - Q(\nu^\varphi) - Q(w^\varphi)\\ &= Q((\nu + w)^\varphi) - Q(\nu^\varphi) - Q(w^\varphi)\\ &= Q(\nu + w) - Q(\nu) - Q(w) = \Phi_Q(\nu, w)\end{aligned}$$

for all $\nu, w \in V$. Thus, φ is also a bijective isometry on Φ_Q. On the other hand let φ be a bijective isometry on Φ_Q. Then we have

$$\begin{aligned}&&\Phi(\nu, \nu) &= \Phi(\nu^\varphi, \nu^\varphi)\\ \iff &&4Q(\nu) - Q(\nu) - Q(\nu) &= 4Q(\nu^\varphi) - Q(\nu^\varphi) - Q(\nu^\varphi)\\ \iff &&2Q(\nu) &= 2Q(\nu^\varphi)\end{aligned}$$

for all $\nu \in V$. So we can only conclude that φ is a bijective isometry on Q if $\mathrm{char}(\mathbb{F}) \neq 2$. In this case we also have that $I(V, \mathbb{F}, \Phi) = I(V, \mathbb{F}, Q)$.

At this point it is possible to define the classical group SO.

Definition 2.12: Let V be an n-dimensional $\mathbb{F}$-vector space with a non-singular quadratic form Q. The group $I(V, \mathbb{F}, Q)$ of bijective isometries is called the *orthogonal group* and denoted $\mathrm{O}(V)$. The subgroup of special isometries is called the *special orthogonal group*, denoted $\mathrm{SO}(V)$.

Remark 2.13:

1.) From now on, we only consider non-singular bilinear or quadratic forms and finite fields with characteristic different from 2.

2.) Note that if $\mathrm{char}(\mathbb{F}) \neq 2$, then the orthogonal group is the group of isometries of the non-singular symmetric bilinear form Φ_Q. Therefore, we can also consider an orthogonal form (see Remark 2.5 and Remark 2.11).

3.) Note that the orthogonal group is independent of the chosen quadratic form. [Taylor, page 138/139]

We consider three different cases for the orthogonal group with the Witt index being an important distinguishing feature.

Definition 2.14: Let V be an $\mathbb{F}$-vector space with a non-singular orthogonal form Φ or a quadratic form Q with polar form Φ.

1.) Let $W \leq V$. We call W *non-singular*, if $\Phi|_W$ is non-singular and *totally isotropic*, if $\Phi|_W = 0$. It is called *totally-singular* if $Q(w) = 0$ for all $w \in W$.

2.) We call a subspace U of V a *maximally totally isotropic* (totally singular) subspace of V, if U is totally isotropic (totally singular) and there is no proper totally isotropic (totally singular) subspace W of V containing U. We say a subspace V_0 of V is *anisotropic*, if $\Phi(\nu, \nu) \neq 0$ for all $\nu \in V_0 - \{0\}$.

Definition 2.15: Let (V, Φ) be an $\mathbb{F}$-vector space with a non-singular orthogonal form Φ or a quadratic form Q, in which case Φ is the polar form of Q. The dimension of a maximally totally isotropic subspace of V, or in case of a quadratic form of a maximally totally singular subspace of V, is called the *Witt index* of (V, Φ).

Theorem 2.16: Let Φ be an orthogonal form on an $\mathbb{F}$-vector space V. Then there exist vectors $\nu_1, \ldots, \nu_m, w_1, \ldots, w_m \in V$ such that

$$V = \langle \nu_1, w_1 \rangle \perp \ldots \perp \langle \nu_m, w_m \rangle \perp V_0$$

where $\langle \nu_i, w_i \rangle$ is a hyperbolic plane, i.e. $\Phi(\imath\nu_i + \jmath w_i, \imath\nu_i + \jmath w_i) = \imath\jmath$ for all $\imath, \jmath \in \mathbb{F}$, and V_0 is anisotropic, such that V has Witt index m and one of the following holds:

1.) $\dim(V_0) = 0$ and $\dim(V) = 2m$.

2.) $\dim(V_0) = 1$ and $\dim(V) = 2m + 1$.

3.) $\dim(V_0) = 2$ and $\dim(V) = 2m + 2$.

Proof. [Taylor, page 138/139]. □

We introduce names for the different cases.

Definition 2.17: Let Φ be an orthogonal form on an $\mathbb{F}$-vector space V. Let

$$V = \langle u_1, v_1 \rangle \perp \ldots \perp \langle u_m, v_m \rangle \perp V_0$$

as in Theorem 2.16. We define the following types of orthogonal groups:

1.) If $\dim(V_0) = 0$, we denote $I(V, \mathbb{F}_q, \Phi)$ by $\mathrm{O}^+(2m, q)$ and say that the group is an *orthogonal group of plus type* and the form Φ is called *hyperbolic*. Moreover, we denote $S(V, \mathbb{F}_q, \Phi)$ by $\mathrm{SO}^+(2m, q)$ and say that the group is a *special orthogonal group of plus type*.

2.) If $\dim(V_0) = 1$, we denote $I(V, \mathbb{F}_q, \Phi)$ by $\mathrm{O}^\circ(2m+1, q)$ and say that the group is an *orthogonal group of circle type* and the form Φ is called *parabolic*. Moreover, we denote $S(V, \mathbb{F}_q, \Phi)$ by $\mathrm{SO}^\circ(2m+1, q)$ and say that the group is a *special orthogonal group of circle type*.

3.) If $\dim(V_0) = 2$, we denote $I(V, \mathbb{F}_q, \Phi)$ by $\mathrm{O}^-(2m+2, q)$ and say that the group is an *orthogonal group of minus type* and the form Φ is called *elliptic*. Moreover, we denote $S(V, \mathbb{F}_q, \Phi)$ by $\mathrm{SO}^-(2m+2, q)$ and say that the group is a *special orthogonal group of minus type*.

To understand the Bruhat decomposition, it is also important to define (B, N) pairs.

Definition 2.18: Let G be a group. A *(B, N) pair* is a pair of subgroups B and N of G such that all of the following conditions hold:

1.) $G = \langle B, N \rangle$,

2.) $H = B \cap N$ is a normal subgroup of N,

3.) The group $W \coloneqq N/H$ is generated by elements $\{w_i \mid i \in I\}$ where $I \neq \emptyset$ is an index set such that $w_i^2 = 1$ for all $i \in I$,

4.) If $w_i = n_i H$ and $n \in N$, then

   a.) $n_i B n \subseteq (B n_i n B) \cup (B n B)$ and

   b.) $n_i B n_i \neq B$.

This definition looks complicated, so we give a brief example in order to understand (B, N) pairs better. We need two more definitions for this example which are also needed later.

Definition 2.19: Let $n \in \mathbb{N}$ and $a \in \mathbb{F}^{n \times n}$. Then a is called a *monomial matrix* if there is exactly one non-zero element in each row and column of a.

Definition 2.20: Let $n \in \mathbb{N}$ and $a \in \mathbb{F}^{n \times n}$. Then a is called a *lower-unitriangular matrix* if both of the following conditions hold:

(i) $a_{i,i} = 1$ for all $i \in \{1, 2, \ldots, n\}$,

(ii) $a_{i,j} = 0$ for all $i, j \in \{1, 2, \ldots, n\}$ and $j > i$.

Example 2.21: Let $n \in \mathbb{N}$, V an n-dimensional $\mathbb{F}$-vector space, $G = \mathrm{GL}(n, \mathbb{F})$ the general linear group and $a \in G$.
As $\det(a) \neq 0$ each row and column of a contains a non-zero entry. Multiplying a matrix a by a lower-unitriangular matrix from the left side corresponds to elementary row operations and multiplication from the right side corresponds to elementary column operations. Using the Gaussian elimination procedure, every matrix in G can be transformed into reduced row echelon form using only elementary row and column operations. Such a matrix has only one non-zero entry in each row and column, so it is a monomial matrix. In particular, there are unitriangular matrices $b_1, b_2 \in G$ such that $b_1 a b_2 = w$ where w is a monomial matrix.
Let B be the group of lower-triangular matrices and N the group of monomial matrices. It is easy to show that (B, N) is a (B, N) pair for G.

Remark 2.22: The previous example is important because the presented (B, N) pair is one for all classical groups (see [Taylor]). We use the (B, N) pair also for the special orthogonal group in this thesis.

Finally, we can define the Bruhat decomposition.

Definition 2.23: Let G be a group with a (B, N) pair, $H = B \cap N$ and $W = N/H$. The *Bruhat decomposition* of G is the decomposition of G into

$$G = BWB = \prod_{w \in W} BwB.$$

We now define transvections in a group G. Transvections take the role of elementary row and column operations in the Gaussian elimination. The proof of Lemma 3.2 is a good example for that.

Definition 2.24: Let V be an n-dimensional $\mathbb{F}$-vector space. Let $H \leq V$ such that $\dim(H) = n - 1$. An element $\tau \in \mathrm{GL}(V)$ is called *transvection* with respect to the *hyperplane* H if

1.) $w^\tau = w$ for all $w \in H$,

2.) $\nu^\tau - \nu \in H$ for all $\nu \in V$.

Transvections play an important role as they generate the $\mathrm{SL}(n,q), \mathrm{SU}(n,q)$ and $\mathrm{Sp}(n,q)$. Since the matrices for elementary row and column operations of the Gaussian algorithm are not contained in $\mathrm{SU}(n,q)$ and $\mathrm{Sp}(n,q)$, transvections in these groups can be seen as a substitute. In the author's Bachelor's thesis [DR], therefore, transvections were used for the Bruhat decomposition in $\mathrm{SU}(n,q)$ and $\mathrm{Sp}(n,q)$. However, this is not the case for the orthogonal group in general as shown in Corollary 2.29. In order to express elements of a group in terms of particular generators, we need to work with transvections which lie in the corresponding group.

Definition 2.25: Let V be an n-dimensional $\mathbb{F}$-vector space and $\tau \in \mathrm{GL}(V)$ be a transvection. τ is called an *orthogonal transvection* if $\tau \in \mathrm{SO}(V)$.

Transvections can be completely characterized by the following lemma.

Lemma 2.26: Let V be an n-dimensional $\mathbb{F}$-vector space. Let $\tau \in \mathrm{GL}(V)$ be a transvection with respect to the hyperplane $H \leq V$. Let $\mu \colon V \to \mathbb{F} \in V^*$ with $H = \ker(\mu)$. Then the following holds:

1.) There is a vector $u \in H - \{0\}$ such that $\nu^\tau = \nu - \nu^\mu \cdot u$ for all $\nu \in V$.

2.) If $z \in V - \{0\}$ with $z \in \ker(\mu)$, then $\tau_{z,\mu} \colon V \to V, \nu \mapsto \nu - \nu^\mu \cdot z$ is a transvection.

Proof. [Taylor, page 20]. □

When working with transvections, the following properties are commonly used.

Lemma 2.27: Let V be an n-dimensional $\mathbb{F}$-vector space. Let $H \leq V$ be a hyperplane and $\mu_1, \mu_2 \in V^*$ with $\ker(\mu_1) = \ker(\mu_2) =: U$, $\jmath \in \mathbb{F}^*$ and $z_1, z_2 \in H - \{0\}$. Then all of the following hold:

1.) $\tau_{z_1,\mu_1}\tau_{z_2,\mu_1} = \tau_{z_1+z_2,\mu_1}$,

2.) $\tau_{z_1,\mu_1}\tau_{z_1,\mu_2} = \tau_{z_1,\mu_1+\mu_2}$,

3.) $\tau_{\jmath z_1,\mu_1} = \tau_{z_1,\jmath\mu_1}$.

Proof. [Taylor, page 21 Theorem 4.2]. □

The following result shows that we cannot work with orthogonal transvections.

Theorem 2.28: There are no orthogonal transvections over a finite field $\mathbb{F}$ whose characteristic is different from 2. If $\mathbb{F}$ has characteristic 2, then an orthogonal transvection for a quadratic form Q has the form

$$x \mapsto x - Q(a)^{-1}\Phi_Q(x, a)a$$

where $Q(a) \neq 0$ and Φ_Q is the polar form of Q.

Proof. [Taylor, page 145 Theorem 11.11]. □

Corollary 2.29: An orthogonal group over a finite field $\mathbb{F}$ whose characteristic is different from 2 is not generated by orthogonal transvections.

Clearly, we have to consider some other type of transformations, namely the Siegel transformations, to take the role of the elementary row and column operations. Since Siegel transformations are more complex than transvections, many proofs and algorithms in the orthogonal groups are more difficult.

Definition 2.30: Let V be an n-dimensional $\mathbb{F}$-vector space with quadratic form Q and polar form Φ_Q. Let $\nu \in V$ be singular, i.e. $Q(\nu) = 0$, and $w \in \langle\nu\rangle^\perp$. We define

$$\rho_{\nu,w}(x) := x + \Phi_Q(x, w)\nu - \Phi_Q(x, \nu)w - Q(w)\Phi_Q(x, \nu)\nu$$

for $x \in V$. Then we have that $\rho_{\nu,w}$ is a well-defined element of $\mathrm{SO}(V)$ for fields of all characteristics. We call $\rho_{\nu,w}$ a *Siegel transformation*. [Taylor, page 148]

Example 2.31: Let $f \in \mathbb{N}$ and $q = p^f$ a prime power. For example, the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix} \in \mathbb{F}_q^{4\times 4}$$

is a Siegel Transformation. We elaborate on this in Remark 4.15.

Analogous to the properties of the transvections the following holds.

Lemma 2.32: Let V be an n-dimensional $\mathbb{F}$-vector space with quadratic form Q and polar form Φ_Q. Let $\nu \in V$ be singular, $w, w_1, w_2 \in \langle\nu\rangle^\perp$, $\varphi \in O(V)$ and $\jmath \in \mathbb{F}^*$. Then all of the following hold:

1.) $\rho_{\jmath\nu,w} = \rho_{\nu,\jmath w}$,

2.) $\rho_{\nu,w_1}\rho_{\nu,w_2} = \rho_{\nu,w_1+w_2}$,

3.) $\varphi\rho_{\nu,w}\varphi^{-1} = \rho_{\nu^\varphi,w^\varphi}$.

Proof. [Taylor, Theorem 11.19]. □

The following result is similar to [DR, Theorem 4.1] for $\mathrm{SU}(n,q)$ and [DR, Theorem 5.1] for $\mathrm{Sp}(n,q)$ except that we consider Siegel transformations instead of transvections here.

Theorem 2.33: Let V be an n-dimensional $\mathbb{F}$-vector space with quadratic form Q and polar form Φ_Q. If $\dim(V) \geq 3$, the Witt index of V at least 1, and $S(V,\mathbb{F},\Phi) \neq \mathrm{SO}^+(4,2)$, then $S(V,\mathbb{F},\Phi)$ is generated by the Siegel transformations of V.

Proof. [Taylor, Theorem 11.46]. □

2.2 Straight-Line Programs

In this subsection, we present an efficient way to express a group element in specified generators using basic group operations. Therefore, it is possible to encode group elements as words over a given generating set. Rather than storing the word as a string, we write it as a word in other subwords. This allows us to evaluate the encoded word efficiently under a group homomorphism.
We start by defining which operations can be used. There are no loops, no conditional statements, no comparisons or other special techniques from programming languages. We restrict ourselves to multiplying and inverting (see [NPP]).

Remark 2.34: MSLPs and the Bruhat decomposition of the classical groups play an important role in the matrix group recognition project. In the problem of this project some matrices $a_1, \ldots, a_k \in \mathrm{GL}(n, q)$ over a finite field $\mathbb{F}_q$ are given. We consider the group $G = \langle a_1, \ldots, a_k \rangle$ and want to compute certain properties of G, for example its order.
Moreover, we could ask whether a matrix $a \in \mathrm{GL}(n, q)$ is an element of G and if this is the case, how a can be expressed in terms of the generators $a_1, \ldots, a_k$. For this, the results of this thesis can be used. More information about this project can be found in [BHGO].

Definition 2.35: Let G be a group, $b \in \mathbb{N}_0$ and $M = [m_1, \ldots, m_b]$ an ordered list of b elements of G. A modification $\mathcal{I}$ of M is called an *instruction* if it has one of the following forms:

(*i*) $m_k \leftarrow m_i$ with $i, k \in \{1, \ldots, b\}$. This instruction stores m_i in the list M in slot k.

(*ii*) $m_k \leftarrow m_i \cdot m_j$ with $i, j, k \in \{1, \ldots, b\}$. This instruction stores $m_i \cdot m_j$ in the list M in slot k.

(*iii*) $m_k \leftarrow m_i^{-1}$ with $i, k \in \{1, \ldots, b\}$. This instruction stores m_i^{-1} in the list M in slot k.

(*iv*) Show(A) where $A \subseteq \{1, \ldots, b\}$. Here no action is required. Instead the specified slots are displayed.

Remark 2.36: The MSLPs of this thesis are written in GAP [GAP] syntax. Therefore, we need an additional type of instruction to initialize the start memory. We use the notation

$$m_i \leftarrow g$$

in this thesis. This means that the element g is stored in the memory slot m_i.

A single instruction does not seem interesting, but we can define a sequence of instructions.

Definition 2.37: Let G be a group, $b \in \mathbb{N}_0$ and $M = [m_1, \ldots, m_b]$ an ordered list of b elements of G. A *straight-line program with memory* (MSLP) is a sequence $S = [\mathcal{I}_1, \ldots, \mathcal{I}_n]$ of instructions $\mathcal{I}_r$ with $1 \leq r \leq n$.
The number $b \in \mathbb{N}_0$ is called the *memory quota* of S and S is said to be a b-MSLP. The number $n \in \mathbb{N}_0$ is the length of S. The empty sequence is permitted with length 0.

In the evaluation process, M is used as memory for the elements that are needed. The instruction (i) of Definition 2.35 can be used to overwrite a slot and minimize the memory quota. The idea of an MSLP is to reduce the memory required for evaluating a particular word in a group by repeatedly overwriting memory slots which are no longer used. The length n of an MSLP describes the number of operations during the entire evaluation.

Let $\mathcal{X}$ be a set and $F_\mathcal{X}$ the free group on $\mathcal{X}$. An obvious way to record a word in $F_\mathcal{X}$ is to simply store the sequence of elements of $\mathcal{X}$ in the word. This, however, is often not very efficient if we want to evaluate the image of the word under a group homomorphism $\varphi\colon F_\mathcal{X} \to G$. For example, if G is a matrix group this would entail $n-1$ matrix multiplications if the word has length n. To improve this, the aim of an SLP is to give an algorithm how to evaluate homomorphisms on this word efficiently, for example by computing and storing subwords only once that occur repeatedly.

Remark 2.38: The instructions of Definition 2.35 use elements of $\{1, \ldots, b\}$. The memory M is secondary in the description. This implies that an MSLP is independent of the group and the chosen elements for the memory. Hence, it is possible to encode an element as a word in one group and to execute exactly the same constructed steps in another group. We use this later to transform a monomial matrix into a diagonal matrix.

The following example helps to understand the previous remark and MSLPs.

Example 2.39: In most computations with groups, elements need to be raised to some powers. To shorten this, fast exponentiation can be used.
Let G be a group, $g \in G$ and $n \in \mathbb{N}$. We can express n in binary form as $n = \sum_{i=0}^{m} a_i \cdot 2^i$ with $a_i \in \{0, 1\}$ for $0 \leq i \leq m$ and $m \in \mathbb{N}$. Then

$$g^n = g^{\left(\sum_{i=0}^{m} a_i \cdot 2^i\right)} = \prod_{i=0}^{m} (g^{2^i})^{a_i}.$$

A way to construct an MSLP for fast exponentiation with memory quota 2 is given in Algorithm 1 in pseudo code.

For example, the algorithm produces the following MSLP for $n = 8$:

$$L = [[m_2 \leftarrow g], [m_1 \leftarrow 1_G], [m_2 \leftarrow m_2 \cdot m_2], [m_2 \leftarrow m_2 \cdot m_2],$$
$$[m_2 \leftarrow m_2 \cdot m_2], [m_1 \leftarrow m_1 \cdot m_2], [m_2 \leftarrow m_2 \cdot m_2], [m_1 \leftarrow m_1]].$$

In this case, we need 8 instructions which is as much as n. However, for a growing n, the length of the SLPs converges to $\log_2(n)$.
Now, let $G := \mathbb{F}_5$ and $g := 2 \in G$. If we evaluate the MSLP L with g and 1_G, we get $g^8 = 1_G$ in slot 1. But we can also choose $G := C_9$ and $g := (1, \ldots, 9) \in G$. If we evaluate the MSLP L with g and (), we obtain $g^8 = g^{-1}$ in slot 1. So we can use a given MSLP on every group and every selection of elements.

Algorithm 1: Fast exponentiation

Input: $k \in \mathbb{N}_0$ a natural number
Output: An MSLP which computes g^k using fast exponentiation for a group element of G

```
function Power(k)
    I_1 := (m_2 ← g);
    I_2 := (m_1 ← 1_G);          // First two instructions to store g and 1_G of a group G.
    i := 3;
    while k > 0 do
        if k odd then
            I_i := (m_1 ← m_1 · m_2);      // The factor in the binary representation is not 0.
            i := i + 1;
        I_i := (m_2 ← m_2 · m_2);                    // Square the element in slot 2.
        k := Floor(k/2);
        i := i + 1;
    I_i := (m_1 ← m_1);
    return [I_1, ..., I_i];
```

Remark 2.40: Let $n, f \in \mathbb{N}$, $q = p^f$ a prime power, $\langle a_1, \ldots, a_k \rangle =: G \cong \mathrm{SO}(n, q)$ and $a \in G$. Since $a \in G$, we can write a as a word in the generators $a_1, \ldots, a_k$ by multiplying these elements and their inverse elements. Therefore, we can compute an MSLP S that, if we evaluate S with the generators $a_1, \ldots, a_k$, computes a. From now on, the construction of such an MSLP will be referred to as the word problem.

3 Bruhat Decomposition in the Special Linear Group

The paper *Straight-line programs with memory and matrix Bruhat decomposition* [NPP] describes and analyzes an algorithm to compute an MSLP for the Bruhat decomposition in the special linear group SL based on the algorithm in [Taylor].
In the following section, we want to describe the method they used, because we use a similar strategy in the special orthogonal group SO. First, we show that SL is generated by transvections. In this chapter, $\mathbb{F}$ is a finite field of order $q = p^f$ for p prime and $f \in \mathbb{N}$.

Definition 3.1: Let $n \in \mathbb{N}$ and $\mathbb{F}$ a field. We call

$$I_n = \mathrm{diag}(1, \ldots, 1) \in \mathbb{F}^{n \times n}$$

the *identity matrix* and define $E_{i,j} \in \mathbb{F}^{n \times n}$ for $i, j \in \{1, \ldots, n\}$ as follows:

$$(E_{i,j})_{a,b} = \begin{cases} 1, & \text{if } i = a \text{ and } j = b, \\ 0, & \text{else.} \end{cases}$$

Lemma 3.2: Let $n, f \in \mathbb{F}$ and $q = p^f$ a prime power. Then $\mathrm{SL}(n, q)$ is generated by transvections.

Proof. First, we look at matrices of the form $I_{i,j}(\imath) \coloneqq I_n + \imath \cdot E_{i,j}$ for $i, j \in \{1, \ldots, n\}$ with $i \neq j$ and $\imath \in \mathbb{F}^*$ and show that these are transvections as described in Definition 2.24. Let $e_1, \ldots, e_n$ be the standard basis of $V = \mathbb{F}_q^n$ and let $H = \langle e_1, e_2, \ldots, e_{i-1}, e_{i+1}, \ldots, e_n \rangle$. We show that $I_{i,j}(\imath)$ is a transvection with respect to the hyperplane H.

1.) Let $w \in H$. Then it follows that

$$w \cdot (I_n + \imath \cdot E_{i,j}) = w \cdot I_n + w \cdot \imath \cdot E_{i,j} = w + 0 = w.$$

2.) Let $\nu \in V$. We can assume $\nu = w + \jmath \cdot e_i$ for $\jmath \in \mathbb{F} - \{0\}$ and $w \in H$. Then we have

$$\nu(I_n + \imath \cdot E_{i,j}) - \nu = \nu I_n + \nu \imath E_{i,j} - \nu = \nu - \nu + w\imath E_{i,j} + \imath\jmath e_i E_{i,j} = \imath\jmath e_i E_{i,j} \in H.$$

Thus, $I_{i,j}(\imath) = I_n + \imath \cdot I_{i,j}$ is a transvection. Moreover, $\det(I_{i,j}(\imath)) = 1$ for $i \neq j$ which implies $I_{i,j}(\imath) \in \mathrm{SL}(n, q)$.
Now, we see that every matrix in $\mathrm{SL}(n, q)$ can be expressed as a product of these transvections. The proof is done by induction on n.
For $n = 1$ there is nothing to show. Assume that $n > 1$ and let $(a_{i,j})_{i,j=1,\ldots,n} \in \mathrm{SL}(n, q)$.

1.) If $a_{1,2} = 0$, then choose some $j \in \{1, \ldots, n\}$ with $a_{1,j} \neq 0$ and add column j to column 2 by multiplying $(a_{i,j})_{n \times n}$ from the right side with $I_n + E_{j,2}$ to get $a_{1,2} \neq 0$.

2.) Multiply $I_n + \frac{(1-a_{1,1})}{a_{1,2}} \cdot E_{2,1}$ from the right side to add the second column $\frac{(1-a_{1,1})}{a_{1,2}}$ times to the first column to get $a_{1,1} = 1$.

3.) For $j > 1$ use transvections as row and column operations to get $a_{1,j} = 0$ and $a_{j,1} = 0$.

Now, every entry of the first row and column is zero except $a_{1,1}$.
The claim follows by induction. □

Remark 3.3: In SL, let B denote the subgroup of lower-triangular matrices and N the subgroup of monomial matrices. Then (B, N) forms a (B, N) pair for SL [Taylor, page 28/29].

We can use the following algorithm to compute such a Bruhat decomposition in SL. Since we use only lower-unitriangular matrices, we cannot guarantee that there is a 1 in the desired places as in the Lemma 3.2. The algorithm is taken from the paper *Straight-line programs with memory and matrix Bruhat decomposition* [NPP].

Algorithm 2: UnitriangularDecompositionSL

Input: $a \in \mathrm{SL}(n, q)$.
Output: A monomial matrix $h \in \mathrm{SL}(n, q)$ and two lower unitriangular matrices $u_1, u_2 \in \mathrm{SL}(n, q)$ such that $h = u_1 \cdot a \cdot u_2$.

function UnitriangularDecompositionSL(a)
- $u_1 \coloneqq I_n$, $u_2 \coloneqq I_n$;
- // We iterate over the columns of a from the right side to the left side.
- **for** $c \in [n, \ldots, 2]$ **do**
  - // Search in column c the first non-zero entry from the top.
  - Search the first entry $i \in [1, \ldots, n]$ with $a_{i,c} \neq 0$;
  - Set $r \coloneqq i$ and piv $\coloneqq a_{r,c}$;
  - **for** $i \in [r+1, \ldots, n]$ **do**
    - $\alpha \coloneqq -\frac{a_{i,c}}{\mathrm{piv}}$;
    - // Multiply transvection $E_{i,c}(\alpha)$ from the left side.
    - $a_{i,-} \coloneqq a_{i,-} + \alpha a_{r,-}$ and $u_{1i,-} \coloneqq u_{1i,-} + \alpha u_{1r,-}$;
  - **for** $j \in [c-1, \ldots, 1]$ **do**
    - $\alpha \coloneqq -\frac{a_{r,j}}{\mathrm{piv}}$;
    - // Multiply transvection $E_{r,j}(\alpha)$ from the right side.
    - $a_{-,j} \coloneqq a_{-,j} + \alpha a_{-,c}$ and $u_{2-,j} \coloneqq u_{2-,j} + \alpha u_{2-,r}$;
- **return** $[a, u_1, u_2]$;

The word problem of SL can be subsequently solved as follows.
Let $a \in \mathrm{SL}$. The description of the matrix a with special generators consists of three steps. First, we calculate the Bruhat decomposition of the matrix a. It is important that all operations performed during the algorithm can be described with the special generators. With Algorithm 2, this step can be performed in SL. However, the description of the used matrices with the special generators has been omitted in Algorithm 2.
Now, if the monomial matrix of the Bruhat decomposition can be described with the special generators, a description of a with the special generators is found. For this purpose, the monomial matrix of the Bruhat decomposition should be represented as a product of another monomial matrix and a diagonal matrix. The second step, therefore, is to find a monomial matrix in SL, such that the product of the monomial matrices forms a diagonal matrix. In the final step, only the description of the diagonal matrix with the special generators remains.

Remark 3.4: Since these three steps are important for the rest of this thesis, we compose a brief summary:

1.) Construct two matrices b_1, b_2 such that $m = b_1 a b_2$ is a monomial matrix and b_1, b_2 can be described as words in the specified generators.

2.) Write a monomial matrix m' as a word of the specified generators such that $m'm = h$ is a diagonal matrix.

3.) Write h in terms of the specified generators.

We want to perform the same procedure consisting of these three steps in the special orthogonal group now. For each type, the first step is performed in the 2nd and 3rd subsection, while the 4th subsection deals with the second and third step.

4 Bruhat Decomposition in Special Orthogonal Groups of Plus Type

This chapter deals with the special orthogonal group of plus type.

First, we look at how such groups can be described using a Gram-matrix to gain an understanding of the elements in this group. In the second subsection, we focus on the Siegel transformations in the special orthogonal group of plus type as we aim for a similar procedure as in the special linear group (see Chapter 3). In the third subsection, we describe an algorithm to obtain the Bruhat decomposition in the special orthogonal group of plus type and prove its correctness. Finally, we discuss the implementation to realize this procedure as an MSLP and also how to solve the word problem for the plus type continuing as described in Remark 3.4.

4.1 Mathematical Background for the Bruhat Decomposition in SO^+

We start with some mathematical background for this group, which we need in the rest of the chapter.

The matrix $J_n \in \mathbb{F}^{n\times n}$ as defined below plays an important role for various proofs.

Definition 4.1: Let $n \in \mathbb{N}$ and define $J_n \coloneqq (J_n)_{i,j\in\{1,\dots,n\}}$ as follows:

$$(J_n)_{i,j} = \begin{cases} 1, & \text{if } i+j = n+1, \\ 0, & \text{else.} \end{cases}$$

We call J_n the *anti-diagonal matrix*.

Example 4.2: For $n = 4$ we have

$$J_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

Remark 4.3: Observe that $J_n^2 = I_n$.

The next result shows that we may assume that $\mathrm{O}^+(n,q)$ preserves a form with Gram-matrix J_n.

Theorem 4.4: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m$ and $\mathbb{F}_q$ a field. Then

$$\mathrm{O}^+(n,q) = \{a \in \mathrm{GL}(n,q) \mid aJ_n a^{\mathrm{T}} = J_n\}.$$

Proof. We can find a basis $\{e_1, f_1, \ldots, e_m, f_m\}$ such that (e_i, f_i) is a hyperbolic pair for $i \in \{1, \ldots, m\}$ [LGO, page 840, item 5]. We can order this basis into $(e_1, e_2, \ldots, f_2, f_1)$ such that the corresponding Gram-matrix has the form J_n. Therefore, $a \in \mathrm{O}^+(n,q)$ if and only if $aJ_n a^{\mathrm{T}} = J_n$. □

Corollary 4.5: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m$ and $\mathbb{F}_q$ a field. Then

$$\mathrm{SO}^+(n,q) = \{a \in \mathrm{SL}(n,q) \mid aJ_n a^{\mathrm{T}} = J_n\}.$$

Proof. We have $\mathrm{SO}^+(n,q) = \mathrm{O}^+(n,q) \cap \mathrm{SL}(n,q)$. □

The next result describes what some elements of $\mathrm{SO}^+(n,q)$ look like. We use this to gain an intuition for elements of the orthogonal group and then to construct matrices, which lie in $\mathrm{SO}^+(n,q)$ and have entries in positions other than the upper left and lower right block.

Definition 4.6: Let $n, m, f \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m$ and $a, b \in \mathrm{GL}(m,q)$. We set

$$\mathrm{diag}(a,b) \coloneqq \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \in \mathrm{GL}(n,q).$$

Lemma 4.7: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m$ and $\mathbb{F}_q$ be a field. We set $a^{-\mathrm{T}} \coloneqq (a^{\mathrm{T}})^{-1} = (a^{-1})^{\mathrm{T}}$. Then we have that

$$\kappa\colon \mathrm{GL}(m,q) \hookrightarrow \mathrm{O}^+(n,q), a \mapsto \mathrm{diag}(a, J_m a^{-\mathrm{T}} J_m)$$

is a monomorphism.

Proof. Let $a, b \in \mathrm{GL}(m,q)$. Then we have

$$(ab)^{-\mathrm{T}} = (b^{\mathrm{T}} a^{\mathrm{T}})^{-1} = a^{-\mathrm{T}} b^{-\mathrm{T}}.$$

Therefore,

$$\begin{aligned}\kappa(ab) &= \mathrm{diag}(ab, J_m(ab)^{-\mathrm{T}}J_m)\\ &= \mathrm{diag}(ab, J_m a^{-\mathrm{T}}b^{-\mathrm{T}}J_m)\\ &= \mathrm{diag}(ab, J_m a^{-\mathrm{T}}J_m J_m b^{-\mathrm{T}}J_m)\\ &= \mathrm{diag}(a, J_m a^{-\mathrm{T}}J_m)\cdot\mathrm{diag}(b, J_m b^{-\mathrm{T}}J_m) = \kappa(a)\kappa(b).\end{aligned}$$

Hence, κ is a group homomorphism. Moreover, we see that the homomorphism κ is injective by inspecting the top left block of $\mathrm{diag}(a, J_m a^{-\mathrm{T}}J_m)$. It remains to show that $\kappa(a) \in \mathrm{O}^+(n,q)$ for $a \in \mathrm{GL}(m,q)$. We verify this by using Theorem 4.4.

$$\begin{aligned}\kappa(a)J_n\kappa(a)^{\mathrm{T}} &= \begin{pmatrix} a & 0\\ 0 & J_m a^{-\mathrm{T}}J_m\end{pmatrix}\begin{pmatrix} 0 & J_m\\ J_m & 0\end{pmatrix}\begin{pmatrix} a^{\mathrm{T}} & 0\\ 0 & J_m a^{-1}J_m\end{pmatrix}\\ &= \begin{pmatrix} 0 & aJ_m\\ J_m a^{-\mathrm{T}} & 0\end{pmatrix}\begin{pmatrix} a^{\mathrm{T}} & 0\\ 0 & J_m a^{-1}J_m\end{pmatrix}\\ &= \begin{pmatrix} 0 & aa^{-1}J_m\\ J_m a^{-\mathrm{T}}a^{\mathrm{T}} & 0\end{pmatrix}\\ &= J_n.\end{aligned}$$

Hence, $\kappa(a) \in \mathrm{O}^+(n,q)$. □

Corollary 4.8: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m$ and $\mathbb{F}$ be a field. Then

$$\kappa\colon \mathrm{SL}(m,q) \hookrightarrow \mathrm{SO}^+(n,q), a \mapsto \mathrm{diag}(a, J_m a^{-\mathrm{T}}J_m)$$

is a monomorphism.

Proof. For a matrix $a \in \mathrm{SL}(m,q)$ we have $\det(a) = 1$, $\det(a^{-\mathrm{T}}) = 1$, $\det(J_m) = \pm 1$ and hence

$$\begin{aligned}\det(\mathrm{diag}(a, J_m a^{-\mathrm{T}}J_m)) &= \det(a)\cdot\det(J_m a^{-\mathrm{T}}J_m)\\ &= \det(a)\cdot\det(a^{-\mathrm{T}})\cdot\det(J_m)^2\\ &= 1.\end{aligned}$$

Hence, the result follows directly by the previous lemma. □

4.2 Siegel Transformations in SO^+

In this subsection, we focus on Siegel transformations in the orthogonal group of plus type. We describe Siegel transformations and prove that they lie in the special orthogonal group of plus type. We use κ as defined in the following remark.

Remark 4.9: For the remainder of this chapter, we fix some notations of the previous subsection. First, we set

$$\mathrm{SO}^+(n,q) \coloneqq \{a \in \mathrm{SL}(n,q) \mid aJ_n a^{\mathrm{T}} = J_n\}.$$

We also want to investigate κ of Corollary 4.8 in more detail. Therefore, we set

$$\kappa\colon \mathrm{SL}(m,q) \hookrightarrow \mathrm{SO}^+(n,q), a \mapsto \mathrm{diag}(a, J_m a^{-\mathrm{T}} J_m).$$

We start by studying the image of monomial matrices under κ.

Lemma 4.10: Let $a \in \mathrm{SL}(m,q)$ be a monomial matrix. Then $\kappa(a)$ is a monomial matrix.

Proof. The monomial matrices form a group under matrix multiplication. Therefore, a^{-1} is a monomial matrix and so is $(a^{-1})^{\mathrm{T}} = a^{-\mathrm{T}}$. Multiplying a monomial matrix by J_m from the left side, induces a permutation of the rows and, hence, $J_m a^{-\mathrm{T}}$ is again a monomial matrix. Multiplying a monomial matrix by J_m from the right side, induces a permutation of the columns and, hence, $(J_m a^{-\mathrm{T}})J_m$ is again a monomial matrix. Clearly, $\mathrm{diag}(a, J_m a^{-\mathrm{T}} J_m)$ is a monomial matrix. □

We can also investigate the image of transvections under κ. Here, $E_{i,j}$ is as defined in Definition 3.1.

Lemma 4.11: Let $I_m + \imath \cdot E_{i,j} = I_{i,j}(\imath) \in \mathrm{SL}(m,q)$ for $\imath \in \mathbb{F}_q$, $i,j \in \{1,\dots,m\}$ and $j < i$ be a lower-unitriangular matrix. Then $\kappa(I_{i,j}(\imath))$ is a lower-unitriangular matrix.

Proof. It suffices to check that $J_m \cdot (I_{i,j}(\imath))^{-\mathrm{T}} \cdot J_m$ is a lower-unitriangular matrix. We see that $(I_{i,j}(\imath))^{-\mathrm{T}} = I_{j,i}(-\imath)$ is an upper-unitriangular matrix. Moreover, we have that $J_m \cdot I_{j,i}(-\imath) \cdot J_m = I_{n+1-j,n+1-i}(-\imath)$. Now, $j < i$ if and only if $n+1-j > n+1-i$. Hence, $J_m \cdot (I_{i,j}(\imath))^{-\mathrm{T}} \cdot J_m$ is a lower-unitriangular matrix. □

Unfortunately, we cannot use these matrices to eliminate entries in the lower left or upper right block of a matrix. But now we have an idea what elements of $\mathrm{SO}^+(n,q)$ look like in general.

Example 4.12: Let $f \in \mathbb{N}$ and $q = p^f$ a prime power. We look at matrices with a field element in the lower left block. For example, we could define

$$a = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ a_{3,1} & 0 & 1 & 0 \\ 0 & a_{4,2} & 0 & 1 \end{pmatrix} \in \mathbb{F}_q^{4\times 4}.$$

We notice that

$$a \cdot J_4 \cdot a^{\mathrm{T}} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & a_{3,1} + a_{4,2} \\ 1 & 0 & a_{4,2} + a_{3,1} & 0 \end{pmatrix}.$$

This shows that $a \in \mathrm{SO}^+(4, q)$ if and only if $a_{3,1} = -a_{4,2}$.

We prove this result in general.

Theorem 4.13: Let $n, f \in \mathbb{N}$, n even and $q = p^f$ a prime power. Define

$$T_{i,j}(\imath) \coloneqq I_n + E_{i,j}(\imath) + E_{n-j+1,n-i+1}(-\imath)$$

for $i, j \in \{1, \ldots, n\}$, $j < i$, $i + j \neq n + 1$ and $\imath \in \mathbb{F}_q$. Then $T_{i,j}(\imath) \in \mathrm{SO}^+(n, q)$.

Proof. We use Corollary 4.5 and show that $T_{i,j}(\imath) \cdot J_n \cdot T_{i,j}(\imath)^{\mathrm{T}} = J_n$. We have

$$\begin{aligned} T_{i,j}(\imath) \cdot J_n \cdot T_{i,j}(\imath)^{\mathrm{T}} &= (I_n + E_{i,j}(\imath) + E_{n-j+1,n-i+1}(-\imath)) \cdot J_n \cdot T_{i,j}(\imath)^{\mathrm{T}} \\ &= (J_n + E_{i,n-j+1}(\imath) + E_{n-j+1,i}(-\imath)) \cdot T_{i,j}(\imath)^{\mathrm{T}} \\ &= (J_n + E_{i,n-j+1}(\imath) - E_{n-j+1,i}(\imath)) \cdot (I_n + E_{j,i}(\imath) - E_{n-i+1,n-j+1}(\imath)) \\ &= J_n + E_{i,n-j+1}(\imath) - E_{n-j+1,i}(\imath) + E_{n-j+1,i}(\imath) - E_{i,n-j+1}(\imath) \\ &= J_n. \end{aligned}$$

Hence, $T_{i,j}(\imath) \in \mathrm{SO}^+(n, q)$. □

Note that for the previous theorem to hold we assumed certain conditions on i and j, e.g. $i+j \neq n+1$. From now on, we use the following notation to work with Siegel transformations in the special orthogonal group of plus type.

Definition 4.14: Let $n, f \in \mathbb{N}$, n even and $q = p^f$ a prime power. For $\imath \in \mathbb{F}_q^*$, $i, j \in \{1, \dots, n\}$, $j < i$ and $i + j \neq n + 1$ we set

$$T_{i,j}(\imath) \coloneqq I_n + E_{i,j}(\imath) + E_{n-j+1,n-i+1}(-\imath).$$

Finally, we want to check whether these matrices are indeed Siegel transformations.

Lemma 4.15: Let $n, f \in \mathbb{N}$, n even and $q = p^f$ a prime power. $T_{i,j}(\imath)$ is a Siegel transformation for $\imath \in \mathbb{F}_q^*$, $i, j \in \{1, \dots, n\}$, $j < i$ and $i + j \neq n + 1$.

Proof. We choose $\nu, w \in \mathbb{F}_q^n$ where $\nu = \imath e_i$ and $w = e_{n-j+1} \in \langle \nu \rangle^\perp$. Notice that $\Phi(w, w) = 0$ and, hence, we have

$$\begin{aligned}
\rho_{\nu,w}(x) &= x + (x^T J_n e_{n-j+1})\imath e_i - (x^T J_n \imath e_i) e_{n-j+1} \\
&= x + \imath x_j e_i - \imath x_{n-i+1} e_{n-j+1} \\
&= (x_1, \dots, x_{i-1}, x_i + \imath x_j, x_{i+1}, \dots, x_{n-j+2}, x_{n-j+1} - \imath x_{n-i+1}, x_{n-j}, \dots, x_n)^T.
\end{aligned}$$

Moreover,

$$T_{i,j}(\imath)x = (x_1, \dots, x_{i-1}, x_i + \imath x_j, x_{i+1}, \dots, x_{n-j+2}, x_{n-j+1} - \imath x_{n-i+1}, x_{n-j}, \dots, x_n)^T.$$

□

4.3 Algorithm for the Bruhat Decomposition in SO^+

We now describe an algorithm to compute the Bruhat decomposition for elements of $\mathrm{SO}^+(n,q)$. First, we give a brief outline of the algorithm explaining its main ideas, then we give pseudo code for the algorithm and prove its correctness. There are slight differences between the plus and circle type of SO, but the algorithms are mostly identical.

Suppose $a \in \mathrm{SO}^+(n,q)$ and we wish to compute the Bruhat decomposition of a.

The algorithm iterates over the following basic step, which replaces a by a new matrix which we again call a:
Suppose a is an element of $\mathrm{SO}^+(n,q)$ with n even and $k \in \{\frac{n}{2}, \dots, n\}$ is the least element such that for all $j \in \{k+1, \dots, n\}$ the jth column has only one non-zero entry, say $a_{i,j}$, and $a_{i,j}$ is the only non-zero entry of the ith row. This means that we can find a kth column of a such that every column on its right has only one non-zero entry and this entry is also the only non-zero entry in its row. Therefore, every column on the right of column k has the desired monomial form.
In the basic step, we search for the least $i \in \{1, \dots, n\}$ such that $a_{i,k} \neq 0$ and multiply lower unitriangular matrices, in particular the Siegel transformations from Definition 4.14, from the left and the right such that column k also has exactly one non-zero element $a_{i,k}$ and $a_{i,k}$ is the only non-zero entry in row i. Since we cannot use a Siegel transformation $T_{i,j}(\imath)$ with $i + j = n + 1$, we have to ensure that the operations used are still sufficient to clear the column or row respectively. Lemma 4.18 and Lemma 4.19 deal with this. Moreover, we can ensure that the columns $k+1, \dots, n$ and the corresponding rows stay in the desired form.

Having described the basic step, we now describe the main algorithm. Our first aim is to transform a into a matrix in which all entries in the nth column below a pivot element $a_{i,n}$ are zero and all entries in the ith row to the left of $a_{i,n}$ are also zero. We achieve this by multiplying a by matrices $T_{i,j}(\imath)$ as in Definition 4.14.
We iterate over all columns from right to left starting with the nth column. Now, we go through this column from top to bottom and look for the first non-zero entry $a_{i,n}$. We use this entry to clear all entries to the left and below. First, we iterate over the entries $a_{k,n}$ for $k \in \{i+1, \dots, n\}$. If $a_{k,n} \neq 0$, we multiply a by the matrix $T_{i,k}(-\frac{a_{k,n}}{a_{i,n}})$ from the left side such that this entry becomes zero. Hence, we can transform a into a matrix such that the only non-zero entry in the nth column is $a_{i,n}$.
Now, we consider all the entries on the left of $a_{i,n}$, that is, $a_{i,k}$ for $k \in \{1, \dots, n-1\}$. Similarly, we multiply the matrices $T_{k,n}(-\frac{a_{i,k}}{a_{i,n}})$ from the right side to get rid of these entries, leaving only one non-zero entry in the row. This is the first use of the basic step.
We continue this procedure with the basic step until the matrix gets the desired monomial form.

It remains to understand why lower unitriangular matrices are enough in the course of the algorithm.

Assume that a has a form where for $k \in \{\frac{n}{2}, \dots, n\}$ every column to the right of the kth column has been modified by the basic step and, therefore, has the described monomial form. Next, we apply the basic step on the kth column.
First, we want to find our pivot element in the kth column. For this we iterate over the kth column from top to bottom and select the first non-zero element, say $a_{i,k}$. Since $a_{i,k}$ is the first non-zero entry from the top, we do not have to clear entries above $a_{i,k}$. Therefore, we only have to clear everything below $a_{i,k}$ using lower unitriangular matrices.
Assume that there is a non-zero entry on the right of $a_{i,k}$ say $a_{i,j}$ for $j \in \{k+1, \dots, n\}$. Then $a_{i,k}$ would be zero since we used the basic step on $a_{i,j}$. Therefore, we only have to clear everything to the left of $a_{i,k}$ using lower unitriangular matrices.

We give the algorithm UnitriangularDecompositionSOPlus in pseudo code and Example 4.16 demonstrates a computation.

Algorithm 3: UnitriangularDecompositionSOPlus

Input: $a \in \mathrm{SO}^+(n,q)$.
Output: A monomial matrix $w \in \mathrm{SO}^+(n,q)$ and two lower unitriangular matrices $u_1, u_2 \in \mathrm{SO}^+(n,q)$ such that $w = u_1 \cdot a \cdot u_2$.

function UnitriangularDecompositionSOPlus(a)
- $u_1 \coloneqq I_n$, $u_2 \coloneqq I_n$;
- **for** $c \in [n, \dots, \frac{n}{2}]$ **do**
  - Find first entry $i \in [1, \dots, n]$ with $a_{i,c} \neq 0$;
  - Set $r \coloneqq i$ and piv $\coloneqq a_{r,c}$;
  - **for** $i \in [r+1, \dots, n]$ **do**
    - $\alpha \coloneqq -\frac{a_{i,c}}{\text{piv}}$;
    - **if** $r + i \neq n+1$ **then**
      - $a_{i,-} \coloneqq a_{i,-} + \alpha a_{r,-}$; $\quad u_{1i,-} \coloneqq u_{1i,-} + \alpha u_{1r,-}$;
      - $a_{n-r+1,-} \coloneqq a_{n-r+1,-} - \alpha a_{n-i+1,-}$; $\quad u_{1n-r+1,-} \coloneqq u_{1n-r+1,-} - \alpha u_{1n-i+1,-}$;
  - **for** $j \in [c-1, \dots, 1]$ **do**
    - $\alpha \coloneqq -\frac{a_{r,j}}{\text{piv}}$;
    - **if** $c + j \neq n+1$ **then**
      - $a_{-,j} \coloneqq a_{-,j} + \alpha a_{-,c}$; $\quad u_{2-,j} \coloneqq u_{2-,j} + \alpha u_{2-,c}$;
      - $a_{-,n-c+1} \coloneqq a_{-,n-c+1} - \alpha a_{-,n-j+1}$; $\quad u_{2-,n-c+1} \coloneqq u_{2-,n-c+1} - \alpha u_{2-,n-j+1}$;
- **return** $[a, u_1, u_2]$;

For a better understanding how and why this algorithm works, we give an example.

Example 4.16: We demonstrate how Algorithm 3 works on the example of

$$a := \begin{pmatrix} 6 & 6 & 2 & 5 \\ 3 & 6 & 1 & 5 \\ 4 & 4 & 4 & 3 \\ 5 & 3 & 5 & 4 \end{pmatrix} \in \mathrm{SO}^+(4,7).$$

The algorithm first considers the fourth column. We start by picking entry $a_{1,4} = 5$ and clear everything below this element in the fourth column.

$$\begin{pmatrix} 6 & 6 & 2 & 5 \\ 3 & 6 & 1 & 5 \\ 4 & 4 & 4 & 3 \\ 5 & 3 & 5 & 4 \end{pmatrix} \xrightarrow{\cdot^L T_{2,1}(6)} \begin{pmatrix} 6 & 6 & 2 & 5 \\ 4 & 0 & 6 & 0 \\ 4 & 4 & 4 & 3 \\ 2 & 0 & 2 & 0 \end{pmatrix} \xrightarrow{\cdot^L T_{3,1}(5)} \begin{pmatrix} 6 & 6 & 2 & 5 \\ 4 & 0 & 6 & 0 \\ 6 & 6 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{pmatrix}$$

We observe that as soon as we have ensured that every entry in the fourth column, except the entry in position $(4,4)$, is zero the entry in position $(4,4)$ automatically is also zero and every entry in the fourth row is zero except in position $(4,1)$. We continue the algorithm and clear the first row with $a_{1,4} = 5$.

$$\begin{pmatrix} 6 & 6 & 2 & 5 \\ 4 & 0 & 6 & 0 \\ 6 & 6 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^R T_{4,3}(1)} \begin{pmatrix} 0 & 6 & 0 & 5 \\ 4 & 0 & 6 & 0 \\ 0 & 6 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^R T_{4,2}(3)} \begin{pmatrix} 0 & 0 & 0 & 5 \\ 0 & 0 & 6 & 0 \\ 0 & 6 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{pmatrix} =: h$$

We notice the same effect for the first row as in the fourth column. Since h is a monomial matrix, we are done and

$$h = T_{3,1}(5)T_{2,1}(6)aT_{4,3}(1)T_{4,2}(3).$$

Hence, we have the Bruhat decomposition of a.

$\mathrm{SO}^+(4,q)$ is a special case since we are always finished after applying the algorithm on the last column. We investigate a second example where we look at a matrix of dimension 6 to gain an even better understanding of the procedure.

Example 4.17: For our second demonstration of Algorithm 3 we choose the matrix

$$a := \begin{pmatrix} 5 & 6 & 5 & 1 & 2 & 5 \\ 1 & 1 & 1 & 1 & 6 & 0 \\ 4 & 2 & 0 & 5 & 1 & 3 \\ 5 & 2 & 5 & 1 & 2 & 1 \\ 0 & 0 & 3 & 0 & 0 & 5 \\ 3 & 4 & 5 & 2 & 6 & 5 \end{pmatrix} \in \mathrm{SO}^+(6,7).$$

The algorithm first considers the sixth column. The entry $a_{1,6} = 5$ becomes our pivot element and we clear everything below this element in the sixth column.

$$\begin{pmatrix} 5 & 6 & 5 & 1 & 2 & 5 \\ 1 & 1 & 1 & 1 & 6 & 0 \\ 4 & 2 & 0 & 5 & 1 & 3 \\ 5 & 2 & 5 & 1 & 2 & 1 \\ 0 & 0 & 3 & 0 & 0 & 5 \\ 3 & 4 & 5 & 2 & 6 & 5 \end{pmatrix} \xrightarrow{\cdot^L T_{3,1}(5)} \begin{pmatrix} 5 & 6 & 5 & 1 & 2 & 5 \\ 1 & 1 & 1 & 1 & 6 & 0 \\ 1 & 4 & 4 & 3 & 4 & 0 \\ 5 & 2 & 5 & 1 & 2 & 1 \\ 0 & 0 & 3 & 0 & 0 & 5 \\ 6 & 1 & 1 & 4 & 3 & 0 \end{pmatrix} \xrightarrow{\cdot^L T_{4,1}(4)} \begin{pmatrix} 5 & 6 & 5 & 1 & 2 & 5 \\ 1 & 1 & 1 & 1 & 6 & 0 \\ 1 & 4 & 4 & 3 & 4 & 0 \\ 4 & 5 & 4 & 5 & 3 & 0 \\ 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 6 & 6 & 6 & 1 & 0 \end{pmatrix}$$

Now, we clear the entry $a_{5,6} = 5$ and the sixth column is dealt with. However, the same effect as in the previous example takes place. We investigate this further in Lemma 4.18 and Lemma 4.19.

$$\begin{pmatrix} 5 & 6 & 5 & 1 & 2 & 5 \\ 1 & 1 & 1 & 1 & 6 & 0 \\ 1 & 4 & 4 & 3 & 4 & 0 \\ 4 & 5 & 4 & 5 & 3 & 0 \\ 0 & 0 & 3 & 0 & 0 & 5 \\ 2 & 6 & 6 & 6 & 1 & 0 \end{pmatrix} \xrightarrow{\cdot^L T_{5,1}(6)} \begin{pmatrix} 5 & 6 & 5 & 1 & 2 & 5 \\ 1 & 1 & 1 & 1 & 6 & 0 \\ 1 & 4 & 4 & 3 & 4 & 0 \\ 4 & 5 & 4 & 5 & 3 & 0 \\ 2 & 1 & 5 & 6 & 5 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Note that this effect is not only advantageous for the runtime but also necessary for the success of the algorithm since in the further course of the algorithm, we no longer need to add something to the last row which preserves the cleared last column.

We continue the algorithm and clear the first row with $a_{1,6} = 5$.

$$\begin{pmatrix} 5 & 6 & 5 & 1 & 2 & 5 \\ 1 & 1 & 1 & 1 & 6 & 0 \\ 1 & 4 & 4 & 3 & 4 & 0 \\ 4 & 5 & 4 & 5 & 3 & 0 \\ 2 & 1 & 5 & 6 & 5 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^R T_{6,5}(1)} \begin{pmatrix} 6 & 6 & 5 & 1 & 0 & 5 \\ 0 & 1 & 1 & 1 & 6 & 0 \\ 4 & 4 & 4 & 3 & 4 & 0 \\ 6 & 5 & 4 & 5 & 3 & 0 \\ 1 & 1 & 5 & 6 & 5 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^R T_{6,4}(4)} \begin{pmatrix} 0 & 6 & 5 & 0 & 0 & 5 \\ 3 & 1 & 1 & 1 & 6 & 0 \\ 2 & 4 & 4 & 3 & 4 & 0 \\ 4 & 5 & 4 & 5 & 3 & 0 \\ 2 & 1 & 5 & 6 & 5 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The same effect as with the columns occurs here.

$$\begin{pmatrix} 0 & 6 & 5 & 0 & 0 & 5 \\ 3 & 1 & 1 & 1 & 6 & 0 \\ 2 & 4 & 4 & 3 & 4 & 0 \\ 4 & 5 & 4 & 5 & 3 & 0 \\ 2 & 1 & 5 & 6 & 5 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^R T_{6,3}(6)} \begin{pmatrix} 0 & 6 & 0 & 0 & 0 & 5 \\ 4 & 1 & 1 & 1 & 6 & 0 \\ 5 & 4 & 4 & 3 & 4 & 0 \\ 2 & 5 & 4 & 5 & 3 & 0 \\ 1 & 1 & 5 & 6 & 5 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^R T_{6,2}(3)} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 1 & 1 & 1 & 6 & 0 \\ 0 & 4 & 4 & 3 & 4 & 0 \\ 0 & 5 & 4 & 5 & 3 & 0 \\ 0 & 1 & 5 & 6 & 5 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

At this stage, the first and last row as well as the first and last column each contain exactly one non-zero entry. We can look at the inner matrix and proceed by induction. We select $a_{2,5} = 6$ and clear column 5 and row 2.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 1 & 1 & 1 & 6 & 0 \\ 0 & 4 & 4 & 3 & 4 & 0 \\ 0 & 5 & 4 & 5 & 3 & 0 \\ 0 & 1 & 5 & 6 & 5 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^L T_{3,2}(4)} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 1 & 1 & 1 & 6 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 5 & 4 & 5 & 3 & 0 \\ 0 & 2 & 3 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^L T_{4,2}(3)} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 1 & 1 & 1 & 6 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 1 & 1 & 1 & 6 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^R T_{5,4}(1)} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 1 & 0 & 6 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^R T_{5,3}(1)} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} =: h$$

Therefore,

$$h = T_{4,2}(3)T_{3,2}(4)T_{5,1}(6)T_{4,1}(4)T_{3,1}(5)aT_{6,5}(1)T_{6,4}(4)T_{6,3}(6)T_{6,2}(3)T_{5,4}(1)T_{5,3}(1),$$

where h is a monomial matrix. Hence, we have computed the Bruhat decomposition of a.

The effect we have seen in the previous example on both rows as well as columns is paramount for the correctness of the algorithm as it ensures that the rows and columns that have already been cleared are preserved. Furthermore, we can stop the algorithm after we have used the basic step on half of the columns of the matrix since we already know that the remaining matrix has monomial form. Therefore, we prove that the observed effect appears for all matrices in the special orthogonal group of plus type. We start with a cleared row.

Lemma 4.18: Let $n, f \in \mathbb{N}$, n even, $q = p^f$ a prime power, $j \in \{1, \dots, n\}$ and $a \in \mathrm{SO}^+(n, q)$ such that there exists $k \in \{1, \dots, n\}$ with $a_{k,j} \neq 0$ and $a_{k,i} = 0$ for all $i \in \{1, \dots, n\}$ with $i \notin \{j, n-j+1\}$. Then

$$a_{i,n-j+1} = \begin{cases} a_{k,j}^{-1}, & \text{if } i = n-k+1, \\ 0, & \text{else.} \end{cases}$$

Proof. We set $t \coloneqq n-j+1$. Since $a \in \mathrm{SO}^+(n, q)$, we know by Corollary 4.5 that $a \cdot J_n \cdot a^{\mathrm{T}} = J_n$. Hence,

$$\begin{aligned}
& a \cdot J_n \cdot a^{\mathrm{T}} \\
&= \begin{pmatrix}
* & \dots & * & a_{1,t} & * & \dots & * & * & * & \dots & * \\
* & \dots & * & a_{2,t} & * & \dots & * & * & * & \dots & * \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
* & \dots & * & a_{k-1,t} & * & \dots & * & * & * & \dots & * \\
0 & \dots & 0 & a_{k,t} & 0 & \dots & 0 & a_{k,j} & 0 & \dots & 0 \\
* & \dots & * & a_{k+1,t} & * & \dots & * & * & * & \dots & * \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
* & \dots & * & a_{n,t} & * & \dots & * & * & * & \dots & *
\end{pmatrix} \cdot J_n \cdot a^{\mathrm{T}} \\
&= \begin{pmatrix}
* & \dots & * & * & * & \dots & * & a_{1,t} & * & \dots & * \\
* & \dots & * & * & * & \dots & * & a_{2,t} & * & \dots & * \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
* & \dots & * & * & * & \dots & * & a_{k-1,t} & * & \dots & * \\
0 & \dots & 0 & a_{k,j} & 0 & \dots & 0 & a_{k,t} & 0 & \dots & 0 \\
* & \dots & * & * & * & \dots & * & a_{k+1,t} & * & \dots & * \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
* & \dots & * & * & * & \dots & * & a_{n,t} & * & \dots & *
\end{pmatrix} \cdot \begin{pmatrix}
* & \dots & * & 0 & * & \dots & * \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
* & \dots & * & 0 & * & \dots & * \\
a_{1,t} & \dots & a_{k-1,t} & a_{k,t} & a_{k+1,t} & \dots & a_{n,t} \\
* & \dots & * & 0 & * & \dots & * \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
* & \dots & * & 0 & * & \dots & * \\
* & \dots & * & a_{k,j} & * & \dots & * \\
* & \dots & * & 0 & * & \dots & * \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
* & \dots & * & 0 & * & \dots & *
\end{pmatrix} \\
&= \begin{pmatrix}
0 & & 0 & \dots & 0 & & 1 \\
0 & & 0 & \dots & 1 & & 0 \\
\vdots & & \vdots & \ddots & \vdots & & \vdots \\
0 & & 0 & \dots & 0 & & 0 \\
a_{k,j}a_{1,t} + a_{k,t}a_{1,j} & & a_{k,j}a_{2,t} + a_{k,t}a_{2,j} & \dots & a_{k,j}a_{n-1,t} + a_{k,t}a_{n-1,j} & & a_{k,j}a_{n,t} + a_{k,t}a_{n,j} \\
0 & & 0 & \dots & 0 & & 0 \\
\vdots & & \vdots & \ddots & \vdots & & \vdots \\
0 & & 1 & \dots & 0 & & 0 \\
1 & & 0 & \dots & 0 & & 0
\end{pmatrix} \\
&= J_n.
\end{aligned}$$

Now, we have $(a \cdot J_n \cdot a^{\mathrm{T}})_{k,k} = a_{k,j}a_{k,t} + a_{k,t}a_{k,j} = 2a_{k,j}a_{k,t} = 0$. Since $2 \neq 0$ and $a_{k,j} \neq 0$, it follows that $a_{k,t} = 0$. Therefore, the claim follows. □

We show the same result for a column with one non-zero entry.

Lemma 4.19: Let $n, f \in \mathbb{N}$, n even, $q = p^f$ a prime power, $j \in \{1, \ldots, n\}$ and $a \in \mathrm{SO}^+(n, q)$ such that there exists $k \in \{1, \ldots, n\}$ with $a_{k,j} \neq 0$ and $a_{i,j} = 0$ for all $i \in \{1, \ldots, n\}$ with $i \notin \{k, n-k+1\}$. Then

$$a_{n-k+1,i} = \begin{cases} a_{k,j}^{-1}, & \text{if } i = n-j+1, \\ 0, & \text{else.} \end{cases}$$

Proof. We know by Corollary 4.5 that $aJ_na^{\mathrm{T}} = J_n$. Since $J_n^{-1} = J_n$, we see that

$$(aJ_na^{\mathrm{T}})^{-1} = (J_n)^{-1} \iff a^{-\mathrm{T}}J_na^{-1} = J_n.$$

Hence, $a^{-\mathrm{T}} \in \mathrm{SO}^+(n, q)$ and so $a^{\mathrm{T}} \in \mathrm{SO}^+(n, q)$. Therefore, the claim follows by Lemma 4.18. □

Now, we have everything to prove the correctness of the algorithm UnitriangularDecompositionSOPlus.

Theorem 4.20: The algorithm UnitriangularDecompositionSOPlus (Algorithm 3) is correct and terminates.

Proof. Let $a \in \mathrm{SO}^+(n, q)$. We start with column n and pick the least i such that the entry $a_{i,n} \neq 0$. Such an entry must exist since $\det(a) = 1$. By Theorem 4.13 we can use the matrices $T_{i,j}$ to clear all entries $a_{k,n}$ for $k \in \{i+1, \ldots, n\}$ and $k + i \neq n + 1$. Therefore, everything above and below the entry $a_{i,n}$ in column n is zero except $a_{n-i+1,n}$ and we used only lower-unitriangular matrices. Again, we use Theorem 4.13 and clear every entry $a_{i,k}$ left from $a_{i,n}$ for $k \in \{1, \ldots, n-1\}$ and $k+i \neq n+1$. Therefore, the row i also contains only one non-zero entry and $a_{i,1}$. By Lemma 4.18 and Lemma 4.19 it follows that $a_{n-i+1,n} = a_{i,1} = 0$ and that row $n - i + 1$ as well as column $n - n + 1 = 1$ contain only one non-zero entry. Hence, we can ensure that over the course of the algorithm one non-zero entry remains in the cleared row and column. Therefore, the claim follows iteratively. □

4.4 Implementation of the Bruhat Decomposition in SO^+

In this subsection, we discuss the implementation of the algorithm for the Bruhat decomposition in SO^+ (Algorithm 3 UnitriangularDecompositionSOPlus) and how it can be described with an MSLP. In addition, we discuss how a monomial matrix can be described as the product of particular generators.

First, we introduce the Leedham-Green-O'Brien standard generators [LGO] of $\mathrm{SO}^+(n,q)$ since we want to use them to solve the word problem described in Remark 2.40.

Definition 4.21: Let q be a prime power, say $q = p^f$, $n \in \mathbb{N}$ even, $n \geq 4$, $\omega \in \mathbb{F}_q$ a primitive element and $x, y \in \mathbb{N}_0$ such that $q - 1 = 2^y \cdot x$ where x is odd. The *Leedham-Green-O'Brien standard generators* [LGO] of $\mathrm{SO}^+(n,q)$ are the following:

$$\widetilde{s} = \begin{pmatrix} s_0 & 0 \\ 0 & I_{n-4} \end{pmatrix} \text{ for } s_0 = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

$$\widetilde{s'} = \begin{pmatrix} s'_0 & 0 \\ 0 & I_{n-4} \end{pmatrix} \text{ for } s'_0 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix},$$

$$\widetilde{t} = \begin{pmatrix} t_0 & 0 \\ 0 & I_{n-4} \end{pmatrix} \text{ for } t_0 = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\widetilde{t'} = \begin{pmatrix} t'_0 & 0 \\ 0 & I_{n-4} \end{pmatrix} \text{ for } t'_0 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix},$$

$$\widetilde{\delta} = \mathrm{diag}(\omega, \omega^{-1}, \omega, \omega^{-1}, 1, \ldots, 1),$$

$$\widetilde{\delta'} = \mathrm{diag}(\omega, \omega^{-1}, \omega^{-1}, \omega, 1, \ldots, 1),$$

$$\widetilde{u} = I_n,$$

$$\widetilde{v} = \begin{cases} \begin{pmatrix} 0 & I_{n-2} \\ I_2 & 0 \end{pmatrix}, & \frac{n}{2} \text{ odd}, \\ \begin{pmatrix} 0 & I_{n-2} \\ -I_2 & 0 \end{pmatrix}, & \frac{n}{2} \text{ even and} \end{cases}$$

$$\widetilde{\sigma} = \mathrm{diag}(\omega^x, \omega^{-x}, 1, \ldots, 1).$$

Leedham-Green and O'Brien use a different basis in their paper. Therefore, we have to perform a basis transformation. Let $\{e_1, \ldots, e_n\}$ be the standard basis of $\mathbb{F}_q^n$ and n even. We use the following matrix

$$T_n^{\mathrm{SO}^+} = \begin{pmatrix} e_1 & e_n & e_2 & e_{n-1} & \ldots & e_{\frac{n}{2}} & e_{\frac{n}{2}+1} \end{pmatrix} \in \mathrm{GL}(n, q).$$

Example 4.22: For $n = 6$ we have

$$T_6^{\mathrm{SO}^+} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

When we use the matrix $T_n^{\mathrm{SO}^+}$ to change the basis, we obtain the following generating set.

Definition 4.23: Let q be a prime power, say $q = p^f$, $4 \leq n$ a positive even integer, $\omega \in \mathbb{F}_q$ a primitive element and $x, y \in \mathbb{N}_0$ such that $q - 1 = 2^y \cdot x$ where x is odd. The Leedham-Green-O'Brien standard generators [LGO] of $\mathrm{SO}^+(n, q)$ transformed with $T_n^{\mathrm{SO}^+}$ are the following:

A double transposition: $$s = \begin{pmatrix} 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & I_{n-4} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{pmatrix},$$

A double transposition: $$s' = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_{n-4} & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

A Siegel transformation: $t = I_n + E_{1,n-1} - E_{2,n}$,

A Siegel transformation: $t' = I_n + E_{1,2} - E_{n-1,n}$,

A diagonal matrix: $\delta = \mathrm{diag}(\omega, \omega, 1, \ldots, 1, \omega^{-1}, \omega^{-1})$,

A diagonal matrix: $\delta' = \mathrm{diag}(\omega, \omega^{-1}, 1, \ldots, 1, \omega, \omega^{-1})$,

The identity matrix: $u = I_n$,

Product of two cycles of length $\frac{n}{2}$: $$v = \begin{cases} \begin{pmatrix} 0 & I_{\frac{n}{2}-1} & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & I_{\frac{n}{2}-1} & 0 \end{pmatrix}, & \frac{n}{2} \text{ odd}, \\ \begin{pmatrix} 0 & I_{\frac{n}{2}-1} & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & I_{\frac{n}{2}-1} & 0 \end{pmatrix}, & \frac{n}{2} \text{ even}, \end{cases}$$

A diagonal matrix: $\sigma = \operatorname{diag}(\omega^x, 1, \ldots, 1, \omega^{-x})$.

Example 4.24: We get the following generating set for $n = 6$ and $q = 7$:

$$s = \begin{pmatrix} 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad s' = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad t = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 6 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$t' = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \delta = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix}, \quad \delta' = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix},$$

$$u = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad v = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad \sigma = \begin{pmatrix} 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 \end{pmatrix}.$$

Now, we can continue as shown in the paper *Straight-line programs with memory and matrix Bruhat decomposition* from A. C. Niemeyer, T. Popiel and C. E. Praeger [NPP].

First, we write the matrices from Algorithm 3 UnitriangularDecompositionSOPlus as words in terms of the LGO generators. For this, we show how we can generate the special matrix $\mathrm{diag}(\omega, 1, \ldots, 1, \omega^{-1}) \in \mathrm{SO}^+(n, q)$.

Lemma 4.25: Let q be a prime power, say $q = p^f$, $4 \leq n = 2m$ a positive even integer, $\omega \in \mathbb{F}_q$ a primitive element and $x, y \in \mathbb{N}_0$ such that $q - 1 = 2^y \cdot x$, where x is odd. Then we have that

$$\delta_* \coloneqq \mathrm{diag}(\omega, 1, \ldots, 1, \omega^{-1}) \in \mathrm{SO}^+$$

and δ_* can be written in the LGO standard generators as

$$\delta_* = \begin{cases} \sigma, & \text{if } x = 1, \\ (\delta')^{\frac{1-x}{2}} \delta^{\frac{1-x}{2}} \sigma, & \text{else.} \end{cases}$$

Proof. If $x = 1$, we clearly have

$$\sigma = \mathrm{diag}(\omega^x, 1, \ldots, 1, \omega^{-x}) = \mathrm{diag}(\omega, 1, \ldots, 1, \omega^{-1}) \in \mathrm{SO}^+.$$

If $x \neq 1$, we set $i \coloneqq \frac{1-x}{2}$ and get

$$\begin{aligned} &(\delta')^{\frac{1-x}{2}} \delta^{\frac{1-x}{2}} \sigma \\ &= \mathrm{diag}(\omega^i, \omega^{-i}, 1, \ldots, 1, \omega^i, \omega^{-i}) \cdot \mathrm{diag}(\omega^i, \omega^i, 1, \ldots, 1, \omega^{-i}, \omega^{-i}) \cdot \mathrm{diag}(\omega^x, 1, \ldots, 1, \omega^{-x}) \\ &= \mathrm{diag}(\omega^{2i+x}, 1, \ldots, 1, \omega^{-(2i+x)}) \in \mathrm{SO}^+ . \end{aligned}$$

Moreover, $2i + x = 2\frac{1-x}{2} + x = 1 - x + x = 1$. Therefore, the claim follows. □

Now, we have all we need to describe the Siegel transformations of $\mathrm{SO}^+(n, q)$. Most of the statements needed to prove the following theorem go back to [DR, Theorem 5.23] where we described the transvections of the symplectic group. This is also an indication for the similarity of the symplectic group and the orthogonal group of plus type. This is discussed further at the end of this subsection in Remark 4.48.

Theorem 4.26: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $4 \leq n = 2m$ and $\omega \in \mathbb{F}_q$ a primitive element of $\mathbb{F}_q$. Let $(s, s', t, t', \delta, \delta', u, v, \sigma)$ be the LGO standard generators as in Definition 4.23 and δ_* as in Lemma 4.25. We set

$$\begin{aligned} u_1 &\coloneqq s', \\ u_i &\coloneqq v^{-1} u_{i-1} v. \end{aligned}$$

Then all matrices of the Algorithm UnitriangularDecompositionSOPlus (Algorithm 3) are a product of the LGO standard generators as shown in the following tables:

Entry in the bottom left block.

Matrix	**Word in terms of the LGO generators**
$T_{n,2}(\omega^\ell)$	$(\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2) s t s^{-1} (\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2)^{-1}$
$T_{n,i}(\omega^\ell)$	$u v^{-1} T_{n,i-1}(\omega^\ell) v u^{-1}$
$T_{j,i}(\omega^\ell)$	$u_{n-j+1}^{-1} T_{j-1,i}(\omega^\ell) u_{n-j+1}$

Entry in the top left or bottom right block.

Matrix	**Word in terms of the LGO generators**
$T_{2,1}(\omega^\ell)$	$(\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2) s' (t')^{-1} (s')^{-1} (\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2)^{-1}$
$T_{i+1,i}(\omega^\ell)$	$v^{-1} T_{i,i-1}(\omega^\ell) v$
$T_{i,j}(\omega^\ell)$	$u_j T_{i,j+1}(\omega^\ell) u_j^{-1}$

Proof. Observe that

$$\begin{aligned} sts^{-1} &= \begin{pmatrix} 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & I_{n-4} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{pmatrix} (I_n + E_{1,n-1} - E_{2,n}) \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & I_{n-4} & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \\ &= I_n - E_{n-1,1} + E_{n,2}, \\ s'(t')^{-1}(s')^{-1} &= \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_{n-4} & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} (I_n - E_{1,2} + E_{n-1,n}) \begin{pmatrix} 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_{n-4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 \end{pmatrix} \\ &= I_n + E_{2,1} - E_{n,n-1}. \end{aligned}$$

The rest follows by [DR, Theorem 5.23]. Notice that the matrices are the same except for the sign of some entries. Because of the order of the multiplication, the sign of all elements of the matrix in the middle stays the same. □

Finally, we can estimate the length and the amount of memory slots for an MSLP, which computes the Bruhat decomposition for an element in the orthogonal group of plus type.

Theorem 4.27: Let $q = p^f$ with p prime, $f \geq 1$, $n = 2m$ an even integer with $n \geq 4$ and $a \in \mathrm{SO}^+(n, q)$. Set $b = 25 + 2f + m$ and

$$\lambda = q + 20f + n - 5 + (n^2 - n)(f + 2\log_2(q) + 2\log_2(m-2) + n).$$

There exists a b-MSLP S of length at most λ such that if S is evaluated with memory containing the list $\{s, s', t, t', \delta, \delta', u, v, \sigma\}$, then S outputs memory containing (u_1, u_2) where $a = u_1 w u_2$ is the Bruhat decomposition of a.

Proof. In order to avoid the logarithm oracle, we use the following method to calculate the Siegel transformations.
Since $\mathbb{F}_q$ is a $\mathbb{F}_p$ vector space with basis $\{\omega^0, \omega^1, \ldots, \omega^{f-1}\}$ where ω is a primitive element of $\mathbb{F}_q$, we only need to save the Siegel transformations $T_{2,1}(\omega^0), T_{2,1}(\omega^1), \ldots, T_{2,1}(\omega^{f-1})$ and obtain $T_{2,1}(\beta)$ for $\beta \in \mathbb{F}_q$ through

$$T_{2,1}(\beta) = T_{2,1}(\sum_{i=0}^{f-1} k_i \omega^i) = \prod_{i=0}^{f-1} T_{2,1}(\omega^i)^{k_i}.$$

Hence, we just calculate the Siegel transformations $T_{2,1}(\omega^0), T_{2,1}(\omega^1), \ldots, T_{2,1}(\omega^{f-1})$ and store them. We use the same method for the Siegel transformations with entries in the lower left block.
Therefore, we start by computing these Siegel transformations. First, we look at the Siegel transformations $T_{2,1}(\omega^i)$ for $i \in \{0, \ldots, f-1\}$ and use the formula defined in Theorem 4.26. For this, we need δ_* which we can compute via Lemma 4.25 with at most $2 + 2\frac{q-1}{2} = 2 + q - 1 = q + 1$ operations. We need 6 operations for $\delta_*^{-\ell} v^{-2} \delta_*^{\ell} v^2$ (notice that we compute $\delta_*^{\ell+1}$ by one multiplication of δ_*^{ℓ} and δ_*) and thus 10 operations in total for

$$(\delta_*^{-\ell} v^{-2} \delta_*^{\ell} v^2) s' (t')^{-1} (s')^{-1} (\delta_*^{-\ell} v^{-2} \delta_*^{\ell} v^2)^{-1}.$$

Since we do this f times, we need $10f$ operations. Analogously, it follows that we need $10f$ operations to calculate

$$T_{n,2}(\omega^i) = (\delta_*^{-\ell} v^{-2} \delta_*^{\ell} v^2) s t s^{-1} (\delta_*^{-\ell} v^{-2} \delta_*^{\ell} v^2)^{-1}$$

where $i \in \{0, \ldots, f-1\}$. Moreover, we store u_i for $i \in \{1, \ldots, m-2\}$ from Theorem 4.26 and need $2(m-3) = n - 6$ operations to calculate them. In total, we need $20f + n - 6$ operations for the initialization.
Now, Algorithm 3 begins. We consider a matrix with a non-zero entry in every position. We need at most $f - 1 + f2\log_2(p) = f - 1 + 2\log_2(q)$ operations to calculate a Siegel transformation $T_{2,1}(\beta)$ or $T_{n,2}(\beta)$. According to the formula from Theorem 4.26 we need at

most $3 + 2\log_2(m-2)$ operations to shift $T_{n,2}(\beta)$ by an index j and to shift by an index i we need at most $2(m-1)$ operations. Maximal we need to do this $\frac{n^2-n}{2}$ times. If we sum up these results, we get

$$\tfrac{n^2-n}{2}(f + 2\log_2(q) + 2\log_2(m-2) + n)$$

operations. Analogously, it follows that we need at most

$$\begin{aligned}&\frac{n^2-n}{2}(f - 1 + 2\log_2(q) + 2 + 2\log_2(m-2) + (m-1)2)\\ \leq\ &\frac{n^2-n}{2}(f + 2\log_2(q) + 2\log_2(m-2) + n)\end{aligned}$$

operations for all the Siegel transformations in the top left and lower right block, yielding the claimed maximum total length of the MSLP:

$$q + 20f + n - 5 + (n^2-n)(f + 2\log_2(q) + 2\log_2(m-2) + n).$$

We need 18 memory slots for the LGO standard generators and their inverse, 2 memory slots for repeated squaring, 2 memory slots for the results u_1 and u_2, 1 memory slot for the current Siegel transformation, 1 additional memory slot to save some calculations and 1 memory slot for δ_*.
Moreover, we need f memory slots for the matrices $T_{2,1}(\omega^i)$ for $i \in \{0, \ldots, f-1\}$, f memory slots for the matrices $T_{n,2}(\omega^i)$ for $i \in \{0, \ldots, f-1\}$ and $m-2$ memory slots for the remaining u_i for $i \in \{1, \ldots, m-2\}$. Thus, we need

$$18 + 2 + 2 + 1 + 1 + 1 + f + f + m - 2 = 25 + 2f + m$$

memory slots in total. □

At this point, we can decompose a matrix $a \in \mathrm{SO}^+(n,q)$ into $a = b_1 w b_2$, where $w \in \mathrm{SO}^+(n,q)$ is a monomial matrix and b_1, b_2 can be written as words in the LGO standard generators. The next goal is to transform a monomial matrix in $\mathrm{SO}^+(n,q)$ into a diagonal matrix in $\mathrm{SO}^+(n,q)$ and to construct an MSLP for this.

Let $M_{\mathrm{SO}^+} \leq \mathrm{SO}^+(n,q)$ be the subgroup of monomial matrices in $\mathrm{SO}^+(n,q)$. Let $\{e_1, \ldots, e_n\}$ be the standard basis of $\mathbb{F}_q^n$. We define

$$\varphi_{\mathrm{SO}^+} \colon M_{\mathrm{SO}^+} \to S_n$$

such that $g \in M_{\mathrm{SO}^+}$ permutes the subspaces $\langle e_1\rangle, \ldots, \langle e_n\rangle$ in the same way that $\varphi_{\mathrm{SO}^+}(g)$ permutes $1, \ldots, n$.
Clearly, $\varphi_{\mathrm{SO}^+}(M_{\mathrm{SO}^+})$ is generated by the images of the standard generators s, s' and v from Definition 4.23. We set $m = \frac{n}{2}$ and obtain

$$\varphi_{\mathrm{SO}^+}(M_{\mathrm{SO}^+}) = \langle (1,n-1)(2,n), (1,2)(n-1,n), v_n^+ \rangle =: G_{\mathrm{SO}^+}$$

where $v_n^+ = (1,2,\ldots,m-1)(m+1,n,n-1,\ldots,m+2)$. Notice that

$$(1,2)(n-1,n)\cdot(1,n-1)(2,n) = (1,n)(2,n-1)$$

and, therefore, we have that

$$G_{\mathrm{SO}^+} = \langle (1,n)(2,n-1), (1,2)(n-1,n), v_n^+ \rangle.$$

Now, let $\pi \in G_{\mathrm{SO}^+}$. We would like to write π as a word in $\{(1,n)(2,n-1), (1,2)(n-1,n), v_n^+\}$. The basic idea for this algorithm is the same as for the monomial matrices in SO°. However, because of the generator $(1,n)(2,n-1)$ we have to rethink our approach. Since this algorithm is much more difficult and complex than the one for the circle type, the procedure in SO° is assumed to be known below (see page 82).

The problem with the generator $(1,n)(2,n-1)$ is that conjugation with this element can destroy a cycle which we already consider to be done. For example, we deal with the permutation $\pi = (1,3)(2,9)(4,11)(10,12) \in S_{12}$. The cycle $(10,12)$ has the desired form since $10, 12 > 6$. Therefore, we consider the cycle $(4,11)$. We want to get rid of the element 11 such that the cycle is also completed. Hence, we conjugate with $(2,11)(3,10)$ yielding

$$(1,3)(2,9)(4,11)(10,12)^{(2,11)(3,10)} = (1,10)(11,9)(4,2)(1,12).$$

Unfortunately, the cycle $(10,12)$ is no longer finished since we got the cycle $(1,12)$. If we now try to remove the element 12 of the cycle $(1,12)$, we start again with π and end up in an infinite loop.

We start by considering some cases that are easy to solve. Then we combine the results to get an overall algorithm. Analogous to the circle type we set $\Omega_1 = \{1,\dots,\frac{n}{2}\}$ and $\Omega_2 = \{\frac{n}{2}+1,\dots,n\}$ for $n \in \mathbb{N}$ even. We start by introducing some new definitions.

Definition 4.28: Let $n \in \mathbb{N}$ and n even. Let $k \in \{1,\dots,n\}$. Then we call $n-k+1$ the *mirror element* of k.

Remark 4.29: We need elements to work with in G_{SO^+}. Let $n, m \in \mathbb{N}$, $n = 2m$ and $\imath, \jmath \in \{1,\dots,m\}$ with $\imath < \jmath$. Consider $\pi = (\imath, n-\imath+1)(\jmath, n-\jmath+1) \in G_{\mathrm{SO}^+}$. We can write π as a word in terms of the generators $\{(1,n)(2,n-1), (1,2)(n-1,n), v_n^+\}$ as follows:

a.) First, notice that

$$(i,i+1)(n-i,n-i+1)^{(1,2,\dots,m-1)(m+1,n,n-1,\dots,m+2)} = (i+1,i+2)(n-i-1,n-i)$$

for $i \in \{1,\dots,m-2\}$. Since we have $(1,2)(n-1,n)$ as a generator, we can, therefore, construct the elements

$$(i,i+1)(n-i,n-i+1)$$

for $i \in \{1, \dots, m-1\}$.

b.) Second, we consider $(1,n)(2,n-1)$. We have that

$$(1,n)(2,n-1)^{(2,3)(n-2,n-1)} = (1,n)(3,n-2).$$

Hence, we can use $(i,i+1)(n-i,n-i+1)$ for $i \in \{1,\dots,m-1\}$ from a.) iteratively to construct

$$(1,n)(\jmath, n-\jmath+1).$$

After that, we can apply the same idea to the transposition $(1,n)$. This means

$$(1,n)(\jmath,n-\jmath+1)^{(1,2)(n-1,n)} = (2,n-1)(\jmath,n-\jmath+1).$$

Iteratively, we obtain the element

$$(\imath, n-\imath+1)(\jmath,n-\jmath+1).$$

Definition 4.30: Let $n, m \in \mathbb{N}$ and $n = 2m$. Let $\imath, \jmath \in \{1,\dots,m\}$ with $\imath < \jmath$. Then we call

$$(\imath, n-\imath+1)(\jmath,n-\jmath+1)$$

a *double transposition* in G_{SO^+}.

We start with the simplest case, where we can refer to a slightly modified version of Algorithm 7 MonomialSOCircle.

Definition 4.31: Let $n \in \mathbb{N}$, n even and $z_1 \dots z_k = \pi \in G_{\mathrm{SO}^+}$ where $z_i \in S_n$ is a cycle. If there is an $\imath \in \{1,\dots,n\}$ and $\jmath \in \{1,\dots,k\}$ such that $z_\jmath = (\imath, n-\imath+1)$ or $\imath^\pi = \imath$ and $(n-\imath+1)^\pi = n-\imath+1$, then we call $(\imath, n-\imath+1)$ a *phantom transposition of* π.

Lemma 4.32: Let $n, m \in \mathbb{N}$, $n = 2m$ and $z_1 \dots z_k = \pi \in G_{\mathrm{SO}^+}$ where $z_i \in S_n$ is a cycle. If π contains a phantom transposition, say $(\imath, n-\imath+1)$, then we can write π as a word of the generators $\{(1,n)(2,n-1), (1,2)(n-1,n), v_n^+\}$ as shown in the proof below.

Proof. As described in Remark 4.29 we can always construct the double transposition

$$(\imath, n-\imath+1)(\jmath,n-\jmath+1)$$

for $\jmath \in \{\imath+1,\dots,m\}$ and

$$(\jmath, n-\jmath+1)(\imath, n-\imath+1)$$

for $\jmath \in \{1, \ldots, \imath-1\}$. Therefore, we can apply Algorithm 7 MonomialSOCircle. If we are in the second case, we conjugate with a double transposition containing $(\imath, n-\imath+1)$. If π contains $(\imath, n-\imath+1)$, the conjugation with the double transposition does not change that. If π does not contain $(\imath, n-\imath+1)$, the conjugation with the double transposition does not change π either. Hence,

$$\pi^{(\imath,n-\imath+1)(\jmath,n-\jmath+1)} = \pi^{(\jmath,n-\jmath+1)},$$
$$\pi^{(\jmath,n-\jmath+1)(\imath,n-\imath+1)} = \pi^{(\jmath,n-\jmath+1)}.$$

If we are in the third case, we multiply with $(\imath, n-\imath+1)(\jmath, n-\jmath+1)$ or $(\jmath, n-\jmath+1)(\imath, n-\imath+1)$ from the left side. If π contains $(\imath, n-\imath+1)$, the multiplication removes $(\imath, n-\imath+1)$. If π does not contain $(\imath, n-\imath+1)$, π contains $(\imath, n-\imath+1)$ after the multiplication. Notice that this does not change the effect of $(\jmath, n-\jmath+1)$ on π. □

Remark 4.33: Observe that it is not possible that the phantom transposition still occurs in π after applying the algorithm as described in the proof of Lemma 4.32.

In the next lemma, we describe a technique to solve the problem at hand for a cycle z_i of a certain form.

Remark 4.34: Notice that all generators

$$\{(1,n)(2,n-1), (1,2)(n-1,n), (1,2,\ldots,m-1)(m+1,n,n-1,\ldots,m+2)\}$$

are mirrored. Therefore, every product of the generators is also mirrored.

Lemma 4.35: Let $n, m \in \mathbb{N}$, $n = 2m$ and $z_1 \ldots z_k = \pi \in G_{\mathrm{SO}^+}$ where $z_i \in S_n$ is a cycle. Let z_i be a cycle of odd length or of even length with $|\operatorname{supp}(z_i) \cap \{1, \ldots, m\}|$ even. Let z_j be the mirrored cycle of z_i. Then we can transform π into a permutation

$$z_1 \ldots z_{i-1} z_{i'} z_{i+1} \ldots z_{j-1} z_{j'} z_{j+1} \ldots z_k = \pi' \in G_{\mathrm{SO}^+}$$

where $z_{i'} \in S_{\Omega_1}$ and $z_{j'} \in S_{\Omega_2}$ or vice versa.

Proof. First, we assume that z_i is a cycle of odd length. There are two cases to consider. Either $|\operatorname{supp}(z_i) \cap \{1, \ldots, m\}|$ or $|\operatorname{supp}(z_i) \cap \{m+1, \ldots, n\}|$ is even. We choose the set of even size. Then we apply double transpositions by conjugation of two elements of the set on π. Since the set size is even, we can do this for all elements. Clearly, z_i has the required form afterwards. We can use the same strategy for z_i with even length as for z_i with odd length, if z_i fulfills the requirements of the lemma. □

Now, there are two cases to consider. A cycle can either be broken down into smaller cycles or can be transformed into the form of the previous lemma.

Lemma 4.36: Let $n, m \in \mathbb{N}$, $n = 2m$ and $z_1 \dots z_k = \pi \in G_{\mathrm{SO}^+}$ where $z_i \in S_n$ is a cycle. Let z_i be a cycle of even length such that for all $k \in \operatorname{supp}(z_i)$ we have $(n-k+1)^{z_i} = n-k+1$. That is to say that z_i does not contain an element and its mirrored element in its support. Let z_j be the mirrored cycle of z_i. Then we can transform π into a permutation

$$z_1 \dots z_{i-1} z_{i'} z_{i+1} \dots z_{j-1} z_{j'} z_{j+1} \dots z_k = \pi' \in G_{\mathrm{SO}^+}$$

where $|\operatorname{supp}(z_i) \cap \{1, \dots, m\}|$ is even.

Proof. If $|\operatorname{supp}(z_i) \cap \{1, \dots, m\}|$ is even, we are done. If z_i has length 2, there are two cases.

1.) If $z_i = (\imath, n-\imath+1)$, then Lemma 4.32 is applicable.

2.) If $z_i = (\imath, \jmath)$ with $\imath \in \{1, \dots, m\}$ and $\jmath \in \{m+1, \dots, n\}$, then $z_j = (n-\imath+1, n-\jmath+1)$ and we have

$$(\imath, n-\imath+1)(\jmath, n-\jmath+1) \cdot (\imath, \jmath)(n-\imath+1, n-\jmath+1) = (\imath, n-\jmath+1)(\jmath, n-\imath+1).$$

Otherwise, z_i contains a sequence $k_1, \dots, k_l$ of odd length with either $k_1, \dots, k_l \in \{1, \dots, m\}$ or $k_1, \dots, k_l \in \{m+1, \dots, n\}$, $l \geq 3$ and the elements before and behind the sequence are contained in the other set (If all sequences would have even length, this would be a contradiction to $|\operatorname{supp}(z_i) \cap \{1, \dots, m\}|$ being odd since z_i has even length). Then the sequence $n-k_1+1, \dots, n-k_l+1$ is contained in z_j. Now, we multiply the double transposition $(k_2, n-k_2+1)(k_l, n-k_l+1)$ from the left side. This yields

$$\begin{aligned}
&(k_2, n-k_2+1)(k_l', n-k_l'+1) \cdot (\dots, k_1, \dots, k_l, \dots)(\dots, n-k_1+1, \dots, n-k_l+1, \dots) \\
&= (\dots, k_1, k_2, n-k_3+1, \dots, n-k_l+1, \dots)(\dots, n-k_1+1, n-k_2+1, k_3, \dots, k_l, \dots) \\
&= z_{i'} z_{j'}
\end{aligned}$$

Therefore, we switched an odd number of elements of the sets $\{1, \dots, m\}$ and $\{m+1, \dots, n\}$. Hence, the claim follows. □

Lemma 4.37: Let $n, m \in \mathbb{N}$, $n = 2m$ and $z_1 \dots z_k = \pi \in G_{\mathrm{SO}^+}$ where $z_i \in S_n$ is a cycle. Let z_i be a cycle of even length at most m such that there exists a $k \in \{1, \dots, m\}$ with $k, n-k+1 \in \operatorname{supp}(z_i)$ and z_j its mirror cycle. Then we can transform π into a permutation

$$z_1 \dots z_{i-1} z_{i_1} z_{i_2} z_{i+1} \dots z_{j-1} z_{j_1} z_{j_2} z_{j+1} \dots z_k = \pi' \in G_{\mathrm{SO}^+}$$

where $z_{i_1}, z_{i_2}, z_{j_1}, z_{j_2}$ have a length smaller than z_i.

Proof. Notice that z_i and z_j have the same length. Since $k, n-k+1 \in \mathrm{supp}(z_i)$, there has to be an element $k' \in \{1, \ldots, m\}$ with $k', n-k'+1 \in \mathrm{supp}(z_j)$. Then we can multiply with $(k, n-k+1)(k', n-k'+1)$ and the claim follows by the third case of Algorithm 7. □

Remark 4.38: Notice that if π consists of only one cycle of length n, we can multiply with a power of v_n^+ to divide π into cycles of smaller length.

Now, we have everything needed to prove the correctness of the algorithm.

Theorem 4.39: Let $n \in \mathbb{N}$, n even and $z_1 \ldots z_k = \pi \in G_{\mathrm{SO}^+}$ where $z_i \in S_n$ is a cycle. Then π can be written as a word in the generators $\{(1,n)(2,n-1), (1,2)(n-1,n), v_n^+\}$ by following the steps of Algorithm 4.

Proof. Proceed as in Algorithm 4 MonomialSOPlus. The claim follows with the previous lemmas. □

We give the algorithm MonomialSOPlus in pseudo code and Example 4.40 shows some computations.

Algorithm 4: MonomialSOPlus

Input: $a \in M_{\mathrm{SO}^+}(n, q)$.
Output: MSLP S which outputs the permutation $\varphi_{\mathrm{SO}^+}(a)$ with the generators of G_{SO^+}

```
function MonomialSOPlus(a)
    π := φ_{SO+}(a);
    u_1 := (), u_2 := ();                          // Neutral element of S_n.
    Add instructions to MSLP S;
    while π contains no phantom transposition or divides into π_1π_2 do
        c_p := Cycles of permutation π in a list;
        for c ∈ c_p do
            if c ∉ S_{Ω_1} or c ∉ S_{Ω_2} then
                if c fulfills the requirements of Lemma 4.35 then
                    Continue as described there;  Add instructions to S;
                if c fulfills the requirements of Lemma 4.36 then
                    Continue as described there;  Add instructions to S;
                    Continue as described in Lemma 4.35;  Add instructions to S;
                else
                    Continue as described in Lemma 4.37;  Add instructions to S;
```

if π does not divide into $\pi_1\pi_2$ **then**
  Apply the algorithm of Lemma 4.32;
  Add instructions to S;
Initialize as in [NPP, Chapter 3.3];
for $i \in [1, \ldots, m-1]$ **do**
  $e := i^{\pi_1} - i$;
  $\pi_1 := \pi_1 v_i^e$;
  $v_{i+1} := s_i v_i$;
  $s_{i+1} := s_i^{v_1^{-1}}$;
  Add instructions to S;
return S;

Since the algorithm is quite complicated, we give a couple of examples.

Example 4.40: For the remainder of this example all elements are in the group

$$G_{\mathrm{SO}^+} = \langle (1,12)(2,11), (1,2)(11,12), (1,2,3,4,5,6)(7,12,11,10,9,8) \rangle.$$

This is the corresponding permutation group for the orthogonal group of plus type for $n = 12 = 2 \cdot 6 = 2m$.

1.) We start with a phantom transposition. Consider

$$\pi := (1,9,7,10,12,4,6,3)(2,11)$$

with phantom transposition $(2,11)$. Hence, we can continue as described in Lemma 4.32 and apply a slightly different version of Algorithm MonomialSOCircle 7 on

$$\pi' := (1,9,7,10,12,4,6,3).$$

The largest moved point of π' is $k = 12$ and $1^{\pi'} \neq 1$. Therefore, we are in the third case and have to multiply with $(1,12)$ from the left side. Since we do not have this element in G_{SO^+}, we use the phantom transposition and multiply with $(1,12)(2,11)$. This yields

$$\pi_1' := (1,12)(2,11) \cdot (1,9,7,10,12,4,6,3)(2,11) = (1,4,6,3)(7,10,12,9).$$

Clearly, we are done since $(1,4,6,3) \in S_{\Omega_1}$ and $(7,10,12,9) \in S_{\Omega_2}$.

2.) For a second more complicated example, we consider

$$\pi \coloneqq (1,10,6,4,5,12,3,7,9,8)(2,11)$$

with phantom transposition $(2,11)$ and $\pi' = (1,10,6,4,5,12,3,7,9,8)$. As in 1.) we are in the third case and multiply with $(1,12)(2,11)$ from the left side. Now, we have

$$\pi_1' \coloneqq (1,12)(2,11)\cdot(1,10,6,4,5,12,3,7,9,8)(2,11) = (1,3,7,9,8)(4,5,12,10,6).$$

The next cycle is $(1,3,7,9,8)(4,5,12,10,6)$. The largest moved point is $k=9$ and $12-9+1=4$. Since $4^{(1,3,7,9,8)}=4$, we are in the second case. Hence, we conjugate with $(4,9)(2,11)$:

$$\pi_2' \coloneqq (1,3,7,9,8)(4,5,12,10,6)^{(2,11)(4,9)} = (1,3,7,4,8)(5,12,10,6,9).$$

Again, we consider $(1,3,7,4,8)(5,12,10,6,9)$ with largest moved point $k=8$ and $12-8+1=5$. Again, we are in the second case yielding

$$\pi_3' \coloneqq (1,3,7,4,8)(5,12,10,6,9)^{(2,11)(5,8)} = (1,3,7,4,5)(6,9,8,12,10).$$

If we do this again for $k=7$, we finish with

$$\pi_4' \coloneqq (1,3,7,4,5)(6,9,8,12,10)^{(2,11)(6,7)} = (1,3,6,4,5)(7,9,8,12,10).$$

3.) Now, we consider an example without phantom transposition. Our element of choice is

$$\pi \coloneqq (1,4,7,3,5,2)(6,10,8,11,12,9).$$

Notice that no cycle contains both k and $n-k+1$ for $k \in \{1,\dots,6\}$. Moreover, $|\operatorname{supp}((1,4,7,3,5,2)) \cap \{1,\dots,6\}| = 5$ is odd. Hence, Lemma 4.36 is applicable. We find the sequence $3,5,2,1,4$ in $(1,4,7,3,5,2)$ and the sequence $10,8,11,12,9$ in $(6,10,8,11,12,9)$. We have to multiply with $(4,9)(5,8)$:

$$\pi_1 \coloneqq (4,9)(5,8)\cdot(1,4,7,3,5,2)(6,10,8,11,12,9) = (1,4,6,10,8,2)(3,5,11,12,9,7)$$

Afterwards, we can use Lemma 4.35 and apply the double transposition $(3,10)(5,8)$ via conjugation:

$$\pi_2 \coloneqq (1,4,6,10,8,2)(3,5,11,12,9,7)^{(3,10)(5,8)} = (1,4,6,3,5,2)(7,10,8,11,12,9)$$

4.) Let us take a look at one last example with the permutation

$$\pi := (1, 8, 2, 12, 5, 11)(3, 9, 6, 10, 4, 7).$$

We consider the cycle $(1, 8, 2, 12, 5, 11)(3, 9, 6, 10, 4, 7)$. It contains the elements 1 and $12 - 1 + 1 = 12$. Therefore, we can use Lemma 4.37. Now, consider the second cycle $(3, 9, 6, 10, 4, 7)$. Hence, we have to multiply with $(1, 12)(3, 10)$ from the left side which yields

$$\begin{aligned}\pi_1 &:= (1, 12)(3, 10) \cdot (1, 8, 2, 12, 5, 11)(3, 9, 6, 10, 4, 7)\\ &= (1, 5, 11)(2, 12, 8)(3, 4, 7)(6, 10, 9).\end{aligned}$$

All of the cycles in this decomposition have length 3, therefore, we can use the double conjugation strategy. $(1, 5, 11)$ contains an even number of elements smaller than $m + 1 = 7$. Hence, we conjugate with $(1, 12)(5, 8)$:

$$\pi_2 := (1, 5, 11)(2, 12, 8)(3, 4, 7)(6, 10, 9)^{(1,12)(5,8)} = (1, 5, 2)(3, 4, 7)(6, 10, 9)(8, 11, 12)$$

The cycle $(6, 10, 9)$ contains an even number of elements greater than $m = 6$. Hence, we conjugate with $(3, 10)(4, 9)$:

$$\pi_3 := (1, 5, 2)(3, 4, 7)(6, 10, 9)(8, 11, 12)^{(3,10)(4,9)} = (1, 5, 2)(3, 4, 6)(7, 10, 9)(8, 11, 12)$$

We also want to know the length and memory quota of an MSLP for this algorithm.

Theorem 4.41: Let $n, m \in \mathbb{N}$, $n = 2m$, $\pi \in G_{\mathrm{SO}^+}$, $b = 12$ and

$$\lambda = (2^{\lceil \log_2(n) \rceil} - 1)(8n - 33) + 2n \log_2(n) + 4n.$$

There exists a b-MSLP S of length at most λ such that if S is evaluated with memory containing $X = \{s, u, v, v^{-1}\}$, then S outputs memory containing a monomial matrix $g \in \mathrm{SO}^+$ such that $\varphi_{\mathrm{SO}^+}(g) = \pi$.

Proof. In the worst case, we start off with a phantom transposition. Then we can only apply one operation per step in contrast to the double transpositions where we always exert two steps simultaneously. It is most expensive to construct the double transposition $(m - 1, m + 2)(m, m + 1)$. For this, we need $(\imath, \imath + 1)(n - \imath + 1, n - \imath)$ which we can calculate recursively as described in Remark 4.29. Each conjugation with v_n^+ requires two multiplications and we need two more operations for the conjugation with $(\imath, \imath + 1)(n - \imath + 1, n - \imath)$. Therefore, we need $4(m - 2)$ operations to shift $(1, n)(2, n - 1)$ into $(1, n)(m, m + 1)$. With the same argument, we need at most $4(m - 3) + 2 + 4(m - 2) = 4(2m - 5) + 2$ operations in total to construct $(m - 1, m + 2)(m, m + 1)$. Knowing that, we can continue as in the proof

of Theorem 5.31. Notice that the remaining permutation has length at most $n-2$.
In the worst case, we have to split the permutation into a product of 2-cycles. Therefore, we are in the third case at most $2^{\lceil \log_2(n-2) \rceil} - 1$ times. The multiplication from the left with a double transposition needs one multiplication. Therefore, we need at most

$$(2^{\lceil \log_2(n-2) \rceil} - 1)(4(2m-5)+3)$$

MSLP instructions. In the worst case, all 2-cycles are in the second case. Therefore, we have to perform the second case at most $2^{\lceil \log_2(n-2) \rceil} - 1$ times. This time we conjugate our permutation which needs 2 more instructions. Hence, we need at most

$$(2^{\lceil \log_2(n) \rceil} - 1)(4(2m-5)+4)$$

MSLP instructions. Finally, we use the algorithm from [NPP, Section 3.3]. This needs at most $2n\log_2(n) + 4n$ MSLP instructions, yielding the claimed maximum total length of the MSLP:

$$(2^{\lceil \log_2(n) \rceil} - 1)(8(2m-5)+7) + 2n\log_2(n) + 4n$$

In addition to storing the input X, the memory quota for the MSLP is as follows. Two memory slots are needed to store the permutations we multiply from the left and right side. Additionally, we need three memory slots for the algorithm from [NPP, Section 3.3]. Three memory slots are also needed for the calculation of $(\imath, n-\imath+1)(\jmath, n-\jmath+1)$. Thus, we need

$$4+2+3+3 = 12$$

memory slots in total. □

Finally, we have to write a diagonal matrix in SO^+ in terms of the LGO standard generators. We observe the following property of diagonal matrices in SO^+ which helps us solve the problem at hand.

Lemma 4.42: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $a = \mathrm{diag}(a_1, \ldots, a_n) \in \mathrm{SO}^+(n,q)$ a diagonal matrix and $n = 2m$. Then $a_i = a_{n-i+1}^{-1}$ for $i \in \{1, \ldots, n\}$.

Proof. We use Corollary 4.5 and recall that $a \cdot J_n \cdot a^{\mathrm{T}} = J_n$. We have

$$\begin{aligned} a \cdot J_n \cdot a^{\mathrm{T}} &= a \cdot J_n \cdot a \\ &= \mathrm{diag}(a_1, \ldots, a_n) \cdot J_n \cdot \mathrm{diag}(a_1, \ldots, a_n) \\ &= \begin{pmatrix} 0 & 0 & \ldots & 0 & a_1 \\ 0 & 0 & \ldots & a_2 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n-1} & \ldots & 0 & 0 \\ a_n & 0 & \ldots & 0 & 0 \end{pmatrix} \cdot \mathrm{diag}(a_1, \ldots, a_n) \end{aligned}$$

$$= \begin{pmatrix} 0 & 0 & \dots & 0 & a_1a_n \\ 0 & 0 & \dots & a_2a_{n-1} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n-1}a_2 & \dots & 0 & 0 \\ a_na_1 & 0 & \dots & 0 & 0 \end{pmatrix}$$
$$= J_n.$$

Hence, the claim follows. □

We now want to use certain matrices to write diagonal matrices. We define these matrices and show that we can express them in terms of the LGO standard generators.

Definition 4.43: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m$ and $\omega \in \mathbb{F}_q$ a primitive element. We set

$$h_j^{\mathrm{SO}^+} := \mathrm{diag}(\underbrace{1, \dots, 1}_{j-1}, \omega, 1, \dots, 1, \omega^{-1}, \underbrace{1, \dots, 1}_{j-1})$$

for $j \in \{1, \dots, m\}$.

The next lemma shows how we can write these matrices in terms of LGO standard generators which also confirms that they lie in $\mathrm{SO}^+(n, q)$.

Lemma 4.44: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m$ and $\omega \in \mathbb{F}_q$ a primitive element. We have

$$h_1^{\mathrm{SO}^+} = (\delta')^{(q-x)/2}(\delta)^{(q-x)/2}\sigma \text{ and}$$
$$h_j^{\mathrm{SO}^+} = v^{-1}h_{j-1}^{\mathrm{SO}^+}v.$$

Proof. We have

$$\begin{aligned} &(\delta')^a(\delta)^a\sigma \\ &= \mathrm{diag}(\omega^a, \omega^{-a}, 1, \dots, 1, \omega^a, \omega^{-a}) \cdot \mathrm{diag}(\omega^a, \omega^a, 1, \dots, 1, \omega^{-a}, \omega^{-a}) \cdot \mathrm{diag}(\omega^x, 1, \dots, 1, \omega^{-x}) \\ &= \mathrm{diag}(\omega^{2a+x}, 1, \dots, 1, \omega^{-(2a+x)}). \end{aligned}$$

We wish that $2a + x \equiv_{q-1} 1$. We choose $a = \frac{q-x}{2}$ and get

$$2 \cdot \tfrac{q-x}{2} + x = q - x + x = q \equiv_{q-1} 1.$$

Clearly, we obtain $h_1^{\mathrm{SO}^+}$. Now, we use φ_{SO^+} and notice that

$$\varphi_{\mathrm{SO}^+}(v) = (1, 2, \dots, m)(m+1, n, n-1, \dots, m+2).$$

Hence, we showed that $v^{-1}h_{j-1}^{\mathrm{SO}^+}v$ induces the desired permutation on the diagonal entries. □

We can now write every diagonal matrix in $\mathrm{SO}^+(n,q)$ in terms of $h_j^{\mathrm{SO}^+}$.

Theorem 4.45: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m$ and $\omega \in \mathbb{F}_q$ a primitive element. Let $a = \mathrm{diag}(a_1, \ldots, a_n) \in \mathrm{SO}^+(n,q)$ be a diagonal matrix. Then a is a product of $h_j^{\mathrm{SO}^+}$ for $j \in \{1, \ldots, m\}$.

Proof. We know by Lemma 4.42 that the matrix a has the form

$$a = \mathrm{diag}(a_1, \ldots, a_n) = \mathrm{diag}(a_1, \ldots, a_m, a_m^{-1}, \ldots, a_1^{-1}).$$

For $j \in \{1, \ldots, m\}$ we choose $k_j \in \{1, \ldots, q\}$ such that $\omega^{k_j} = a_j$. We show that

$$a = \prod_{j=1}^{m} (h_j^{\mathrm{SO}^+})^{k_j}.$$

We have

$$\begin{aligned}
\prod_{j=1}^{m} (h_j^{\mathrm{SO}^+})^{k_j} &= (h_1^{\mathrm{SO}^+})^{k_1} \ldots (h_m^{\mathrm{SO}^+})^{k_m} \\
&= \mathrm{diag}(\omega, 1, \ldots, 1, \omega^{-1})^{k_1} \ldots \mathrm{diag}(1, \ldots, 1, \omega, \omega^{-1}, 1, \ldots, 1)^{k_m} \\
&= \mathrm{diag}(\omega^{k_1}, 1, \ldots, 1, (\omega^{-1})^{k_1}) \ldots \mathrm{diag}(1, \ldots, 1, \omega^{k_m}, (\omega^{-1})^{k_m}, 1, \ldots, 1) \\
&= \mathrm{diag}(a_1, 1, \ldots, 1, a_1^{-1}) \ldots \mathrm{diag}(1, \ldots, 1, a_m, (a_m)^{-1}, 1, \ldots, 1) \\
&= \mathrm{diag}(a_1, \ldots, a_m, a_m^{-1}, \ldots, a_1^{-1}) \\
&= a.
\end{aligned}$$

Hence, the claim follows. □

We give the algorithm DiagonalSOPlus in pseudo code and Example 4.46 demonstrates a computation.

Algorithm 5: DiagonalSOPlus

Input: $a \in \mathrm{SO}^+(n,q)$ a diagonal matrix.
Output: An MSLP S such that a is the output of the evaluation of S with the LGO standard generators.

```
function DiagonalSOPlus(a)
    Write q = 2^y x with x odd;  s := (q-x)/2;
    Add instructions to MSLP S to generate (δ')^s (δ)^s σ;        // h_1^{SO+}.
    for i ∈ [1, ..., n/2] do
        a_i := DiscreteLogarithm(a_{i,i}, ω);
        Add instructions to S for (h_i^{SO+})^{a_i} and h_{i+1}^{SO+} = v^{-1} h_i^{SO+} v;
    return S;
```

We give a brief example for the Algorithm 5 DiagonalSOPlus.

Example 4.46: We consider the matrix

$$\begin{pmatrix} \omega^{10} & 0 & 0 & 0 & 0 & 0 \\ 0 & \omega^{37} & 0 & 0 & 0 & 0 \\ 0 & 0 & \omega^{30} & 0 & 0 & 0 \\ 0 & 0 & 0 & \omega^{18} & 0 & 0 \\ 0 & 0 & 0 & 0 & \omega^{11} & 0 \\ 0 & 0 & 0 & 0 & 0 & \omega^{38} \end{pmatrix} \in \mathrm{SO}^+(6, 7^2).$$

We have that $(\delta')^{(q-x)/2}(\delta)^{(q-x)/2}\sigma = (\delta')^{(49-3)/2}(\delta)^{(49-3)/2}\sigma = \mathrm{diag}(\omega, 1, 1, 1, 1, \omega^{-1})$ where ω is a primitive element of $\mathbb{F}_{49}$. Now, the algorithm starts

i=1: We calculate DiscreteLogarithm$(a_{1,1} = \omega^{10}, \omega) = 10$.

i=2: We calculate DiscreteLogarithm$(a_{2,2} = \omega^{37}, \omega) = 37$.

i=3: We calculate DiscreteLogarithm$(a_{3,3} = \omega^{30}, \omega) = 30$.

Therefore,

$$\begin{aligned} a &= \mathrm{diag}(\omega, 1, 1, 1, 1, \omega^{-1})^{10}\,\mathrm{diag}(1, \omega, 1, 1, \omega^{-1}, 1)^{37}\,\mathrm{diag}(1, 1, \omega, \omega^{-1}, 1, 1)^{30} \\ &= (h_1^{\mathrm{SO}^+})^{10}(h_2^{\mathrm{SO}^+})^{37}(h_3^{\mathrm{SO}^+})^{30} \\ &= (h_1^{\mathrm{SO}^+})^{10}(v^{-1}h_1^{\mathrm{SO}^+}v)^{37}(v^{-2}h_1^{\mathrm{SO}^+}v^2)^{30}. \end{aligned}$$

For those who are interested, there is an MSLP for this case in the Appendix.

It should be noted that matrices do not always have this representation with entries as powers of a primitive element. In [GAP], the limit for calculations of this form is 2^{16}. Above that the computation of the discrete logarithm is much more expensive.

Finally, we calculate the maximum length and the number of memory slots of an MSLP, which computes a diagonal matrix from the orthogonal group of plus type.

Theorem 4.47: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m$ and $\omega \in \mathbb{F}_q$ a primitive element. Let $\lambda = m + 4\log_2(q) + n - 1 + n\log_2(q)$ and $b = 22$. Let $h \in \mathrm{SO}^+(n, q)$ be a diagonal matrix given in the form of $h = \mathrm{diag}(\omega^{k_1}, \ldots, \omega^{k_n})$. There exists a b-MSLP S of length at most λ such that if S is evaluated with memory containing the set $X = \{s, s^{-1}, s', (s')^{-1}, t, t^{-1}, t', (t')^{-1}, \delta, \delta^{-1}, \delta', (\delta')^{-1}, v, v^{-1}, u, u^{-1}, \sigma, \sigma^{-1}\}$, then S outputs final memory containing h.

Proof. The MSLP computes h as described in Theorem 4.45. The $h_j^{\mathrm{SO}^+}$ are computed in terms of the LGO standard generators as shown in Lemma 4.44. The construction of the $h_j^{\mathrm{SO}^+}$ using Lemma 4.44 costs $4\log_2(q) + (\frac{n}{2} - 1)2 + 1 = 4\log_2(q) + n - 1$ MSLP instructions:

- $h_1^{\mathrm{SO}^+}$ takes at most $4\log_2(\frac{q-x}{2}) + 1 \leq 4\log_2(q) + 1$ multiplications,
- $h_{j+1}^{\mathrm{SO}^+}$ arises from $h_j^{\mathrm{SO}^+}$ by conjugating with v which needs 2 multiplications.

Raising each $h_j^{\mathrm{SO}^+}$ up to its k_jth power can be done by repeated squaring [NPP, Section 2.2(ii)] which costs at most

$$2\log_2(k_j) \leq 2\log_2(q)$$

MSLP instructions. Since we have to do this m times, the computation of all of the $(h_j^{\mathrm{SO}^+})^{k_j}$ needs at most

$$\tfrac{n}{2}(2\log_2(q)) = n\log_2(q)$$

instructions. Moreover, we have to multiply the $(h_j^{\mathrm{SO}^+})^{k_j}$ together which requires

$$m - 1 < m = \tfrac{n}{2}$$

instructions. Overall, this results in a maximum total length of the MSLP of

$$m + 4\log_2(q) + n - 1 + n\log_2(q).$$

The memory quota for this MSLP is as follows:

- 18 memory slots are needed to store the input X,
- 1 memory slot is needed to multiply the $(h_j^{\mathrm{SO}^+})^{k_j}$ together,
- 1 memory slot is needed to store the $h_j^{\mathrm{SO}^+}$ because the recursion for the $h_j^{\mathrm{SO}^+}$ refers back to $h_{j-1}^{\mathrm{SO}^+}$ and v, hence, each $h_j^{\mathrm{SO}^+}$ can be written into the slot from which it is computed and
- 2 memory slots are needed for computing the $(h_j^{\mathrm{SO}^+})^{k_j}$ with repeated squaring [NPP, Section 2.2(ii)].

Thus, we need

$$18 + 2 + 1 + 1 = 22$$

memory slots in total. □

Remark 4.48: It should be noted that the orthogonal group of plus type can be treated in a similar way to the symplectic group at many points. A slight difference is that there are also transformations $T_{i,j}$ with $i + j = n + 1$ in the symplectic group and these are also used in the algorithm for the Bruhat decomposition. The most essential difference, however, lies in the permutation group that is generated by the images of the monomial matrices. Here the orthogonal group of plus type takes a special role in comparison with all other classical groups. For further details, the reader is refered to my Bachelor's thesis *Bruhat Decomposition in Unitary and Symplectic Groups over Finite Fields* [DR].

5 Bruhat Decomposition in Special Orthogonal Groups of Circle Type

This chapter deals with the special orthogonal group of circle type and is structured identically to the previous chapter about the plus type.

In the first subsection, we start off again by gathering initial knowledge about the group. The Gram-matrix has a form similar to the one for the plus type except for the entry in the middle. Then we construct a homomorphism from the special linear group into the orthogonal group of circle type. This is the same as for the plus type, except that we leave the entry in the middle invariant which is a good indicator that this entry needs special consideration. In the second subsection, we construct matrices for the Bruhat decomposition algorithm where not all produced matrices are Siegel transformations. In the third subsection, we present the algorithm for this type and prove its correctness. In the last subsection, we deal again with an MSLP for this algorithm as well as with monomial and diagonal matrices of the group.

5.1 Mathematical Background for the Bruhat Decomposition in SO°

In this subsection, we prove some first general properties of the orthogonal group of the circle type. The structure is analogous to the one for the plus type.

The matrix $J^\circ_{n,q} \in \mathbb{F}^{n\times n}$ defined below takes the role of J_n in the plus type.

Definition 5.1: Let $n \in \mathbb{N}$ with n odd and define $J^\circ_{n,q} := (J^\circ_{n,q})_{i,j\in\{1,\dots,n\}}$ via:

$$(J^\circ_{n,q})_{i,j} := \begin{cases} 1, & \text{if } i+j = n+1 \text{ and } i \neq \frac{n+1}{2}, \\ -2^{-1}, & \text{if } i = \frac{n+1}{2} \text{ and } j = \frac{n+1}{2}, \\ 0, & \text{else.} \end{cases}$$

We call $J^\circ_{n,q}$ the *anti-diagonal matrix of circle type*.

Example 5.2: For $n = 5$ and $\mathbb{F}_7$ we have

$$J^\circ_{5,7} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The next result shows that exactly the elements which leave $J^\circ_{n,q}$ invariant are in $\mathrm{O}^\circ(n,q)$. As we have seen for a similar result for the plus type, we use this for later proofs.

Theorem 5.3: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m+1$ and $\mathbb{F}_q$ a field. Then

$$\mathrm{O}^\circ(n,q) = \{a \in \mathrm{GL}(n,q) \mid aJ^\circ_{n,q}a^{\mathrm{T}} = J^\circ_{n,q}\}.$$

Proof. We can find a basis $\{e_1, f_1, \ldots, e_m, f_m, w\}$ such that (e_i, f_i) is a hyperbolic pair for $i \in \{1, \ldots, m\}$, $we_i = wf_i = 0$ for $i \in \{1, \ldots, m\}$ and $ww = -2^{-1}$ [LGO, page 840, item 7]. We can order this basis into $(e_1, e_2, \ldots, e_m, w, f_m, \ldots, f_2, f_1)$ such that the corresponding Gram-matrix has the form $J^\circ_{n,q}$. Therefore, $a \in \mathrm{O}^\circ(n,q)$ if and only if $aJ^\circ_{n,q}a^{\mathrm{T}} = J^\circ_{n,q}$. □

Corollary 5.4: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m+1$ and $\mathbb{F}_q$ a field. Then

$$\mathrm{SO}^\circ(n,q) = \{a \in \mathrm{SL}(n,q) \mid aJ^\circ_{n,q}a^{\mathrm{T}} = J^\circ_{n,q}\}.$$

Proof. We have $\mathrm{SO}^\circ(n,q) = \mathrm{O}^\circ(n,q) \cap \mathrm{SL}(n,q)$. □

The homomorphism into the orthogonal group of circle type looks similar to the one of plus type, except that we fix the entry in the middle of the matrix with 1. The subsequent proof can thus be performed analogously.

Definition 5.5: Let $n, m, f \in \mathbb{N}$, $q = p^f$ a prime power, n odd with $n = 2m+1$ and $a, b \in \mathrm{GL}(m,q)$. We set

$$\mathrm{diag}(a,1,b) \coloneqq \begin{pmatrix} a & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & b \end{pmatrix}.$$

Lemma 5.6: Let $n, m, f \in \mathbb{N}$, $q = p^f$ a prime power, n odd with $n = 2m+1$ and $\mathbb{F}$ be a field. We set $a^{-\mathrm{T}} \coloneqq (a^{\mathrm{T}})^{-1} = (a^{-1})^{\mathrm{T}}$. Then

$$\kappa^\circ \colon \mathrm{GL}(m,q) \hookrightarrow \mathrm{O}^\circ(n,q), a \mapsto \mathrm{diag}(a, 1, J_m a^{-\mathrm{T}} J_m)$$

is a monomorphism.

Proof. Let $a, b \in \mathrm{GL}(m,q)$. Then we have

$$(ab)^{-\mathrm{T}} = (b^{\mathrm{T}}a^{\mathrm{T}})^{-1} = a^{-\mathrm{T}}b^{-\mathrm{T}}.$$

Therefore,

$$\begin{aligned}\kappa^\circ(ab) &= \mathrm{diag}(ab, 1, J_m(ab)^{-\mathrm{T}}J_m)\\ &= \mathrm{diag}(ab, 1, J_m a^{-\mathrm{T}}b^{-\mathrm{T}}J_m)\\ &= \mathrm{diag}(ab, 1, J_m a^{-\mathrm{T}}J_m J_m b^{-\mathrm{T}}J_m)\\ &= \mathrm{diag}(a, 1, J_m a^{-\mathrm{T}}J_m)\cdot\mathrm{diag}(b, 1, J_m b^{-\mathrm{T}}J_m) = \kappa^\circ(a)\kappa^\circ(b).\end{aligned}$$

Hence, κ° is a group homomorphism. Moreover, we see that the homomorphism κ° is injective by inspecting the top left block of $\mathrm{diag}(a, 1, J_m a^{-\mathrm{T}}J_m)$. It remains to show that $\kappa^\circ(a) \in \mathrm{O}^\circ(n, q)$ for $a \in \mathrm{GL}(m, q)$. We verify this by using Theorem 5.3.

$$\begin{aligned}\kappa^\circ(a)J^\circ_{n,q}\kappa^\circ(a)^{\mathrm{T}} &= \begin{pmatrix} a & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & J_m a^{-\mathrm{T}}J_m\end{pmatrix}\begin{pmatrix} 0 & 0 & J_m\\ 0 & -\frac{1}{2} & 0\\ J_m & 0 & 0\end{pmatrix}\begin{pmatrix} a^{\mathrm{T}} & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & J_m a^{-1}J_m\end{pmatrix}\\ &= \begin{pmatrix} 0 & 0 & aJ_m\\ 0 & -\frac{1}{2} & 0\\ J_m a^{-\mathrm{T}} & 0 & 0\end{pmatrix}\begin{pmatrix} a^{\mathrm{T}} & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & J_m a^{-1}J_m\end{pmatrix}\\ &= \begin{pmatrix} 0 & 0 & aa^{-1}J_m\\ 0 & -\frac{1}{2} & 0\\ J_m a^{-\mathrm{T}}a^{\mathrm{T}} & 0 & 0\end{pmatrix}\\ &= J^\circ_{n,q}.\end{aligned}$$

Hence, $\kappa^\circ(a) \in \mathrm{O}^\circ(n, q)$. □

Corollary 5.7: Let $n, f \in \mathbb{N}$, $q = p^f$ a prime power, n odd, $n = 2m + 1$ and $\mathbb{F}$ be a field. Then

$$\kappa^\circ\colon \mathrm{SL}(m, q) \hookrightarrow \mathrm{SO}^\circ(n, q), a \mapsto \mathrm{diag}(a, 1, J_m a^{-\mathrm{T}}J_m)$$

is a monomorphism.

Proof. For a matrix $a \in \mathrm{SL}(m, q)$ we have $\det(a) = 1$, $\det(a^{-\mathrm{T}}) = 1$, $\det(J_m) = \pm 1$ and hence

$$\begin{aligned}\det(\mathrm{diag}(a, 1, J_m a^{-\mathrm{T}}J_m)) &= \det(a)\cdot\det(1)\cdot\det(J_m a^{-\mathrm{T}}J_m)\\ &= \det(a)\cdot 1\cdot\det(a^{-\mathrm{T}})\cdot\det(J_m)^2\\ &= 1.\end{aligned}$$

Therefore, the result directly follows from the previous lemma. □

5.2 Siegel Transformations in SO°

The focus of this chapter lies on transformations in the orthogonal group of circle type. We describe transformations, prove that they lie in the special orthogonal group of circle type and check whether they are Siegel transformations. In contrast to the plus type, not all constructed matrices for Algorithm 6 are Siegel transformations.

Remark 5.8: For the remainder of this chapter, we fix some notations of the previous subsection. First, we set

$$\mathrm{SO}^\circ(n,q) \coloneqq \{a \in \mathrm{SL}(n,q) \mid aJ^\circ_{n,q}a^{\mathrm{T}} = J^\circ_{n,q}\}.$$

We also want to investigate κ° of Corollary 5.7 in more detail. Therefore, we set

$$\kappa^\circ\colon \mathrm{SL}(m,q) \hookrightarrow \mathrm{SO}^\circ(n,q), a \mapsto \mathrm{diag}(a, 1, J_m a^{-\mathrm{T}} J_m).$$

As for the plus type, we start by studying the image of monomial matrices under κ.

Lemma 5.9: Let $a \in \mathrm{SL}(m,q)$ be a monomial matrix. Then $\kappa^\circ(a)$ is a monomial matrix.

Proof. The monomial matrices form a group under matrix multiplication. Therefore, a^{-1} is a monomial matrix and so is $(a^{-1})^{\mathrm{T}} = a^{-\mathrm{T}}$. Multiplying a monomial matrix by J_m from the left side induces a permutation of the rows and, hence, $J_m a^{-\mathrm{T}}$ is again a monomial matrix. Multiplying a monomial matrix by J_m from the right side induces a permutation of the columns and, hence, $(J_m a^{-\mathrm{T}})J_m$ is again a monomial matrix and so is $J_m a^{-\mathrm{T}} J_m$. Clearly, $\mathrm{diag}(a, 1, J_m a^{-\mathrm{T}} J_m)$ is a monomial matrix. □

We continue as for the plus type and investigate the image of transvections under κ°. Here, $E_{i,j}$ is as defined in Definition 3.1.

Lemma 5.10: Let $I_m + \imath \cdot E_{i,j} = I_{i,j}(\imath) \in \mathrm{SL}(m,q)$ for $\imath \in \mathbb{F}_q$, $i, j \in \{1, \ldots, m\}$ and $j < i$ be a lower-unitriangular matrix. Then $\kappa^\circ(I_{i,j}(\imath))$ is a lower-unitriangular matrix.

Proof. It suffices to check that $J_m \cdot (I_{i,j}(\imath))^{-\mathrm{T}} \cdot J_m$ is a lower-unitriangular matrix. We see that $(I_{i,j}(\imath))^{-\mathrm{T}} = I_{j,i}(-\imath)$ is an upper-unitriangular matrix. Moreover, we have that $J_m \cdot I_{j,i}(-\imath) \cdot J_m = I_{n+1-j,n+1-i}(-\imath)$. It is $j < i$ if and only if $n+1-j > n+1-i$. Hence, $J_m \cdot (I_{i,j}(\imath))^{-\mathrm{T}} \cdot J_m$ is a lower-unitriangular matrix. □

For the orthogonal group of circle type, we also need transformations with entries in the lower left block and with entries in the $(m+1)$th row or column.

Example 5.11: Let $f \in \mathbb{N}$ and $q = p^f$ a prime power. First, we take a look at matrices with a non-zero entry in the lower left block. Consider for example the matrix

$$a = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ a_{4,1} & 0 & 0 & 1 & 0 \\ 0 & a_{5,2} & 0 & 0 & 1 \end{pmatrix} \in \mathbb{F}_q^{5\times 5}.$$

We notice that

$$a \cdot J_{5,q}^{\circ} \cdot a^{\mathrm{T}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -2^{-1} & 0 & 0 \\ 0 & 1 & 0 & 0 & a_{4,1}+a_{5,2} \\ 1 & 0 & 0 & a_{4,1}+a_{5,2} & 0 \end{pmatrix}.$$

Thus, $a \in \mathrm{SO}^\circ(5,q)$ if and only if $a_{4,1} = -a_{5,2}$. This is similar to the plus type.
For the circle type, we also have to look at matrices which have an entry in the central row or column. With the previous knowledge, we could define, for example, the matrix

$$b = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \imath & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\imath & 0 & 1 \end{pmatrix} \in \mathbb{F}_q^{5\times 5}$$

for $\imath \in \mathbb{F}_q^*$. We notice that

$$b \cdot J_{5,q}^{\circ} \cdot b^{\mathrm{T}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -2^{-1} & 0 & 2^{-1}\imath^2 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 2^{-1}\imath^2 & 0 & -2^{-1}\imath^2 \end{pmatrix}.$$

Therefore, $b \in \mathrm{SO}^\circ(5,q)$ if and only if $\imath = 0$. Since we want to use matrices with entries in the central row and column, we can try to fix this by choosing

$$c = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \imath & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \jmath & 0 & 2\imath & 0 & 1 \end{pmatrix} \in \mathbb{F}_q^{5\times 5}.$$

In this case we notice that

$$c \cdot J_{5,q}^{\circ} \cdot c^{\mathrm{T}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & \jmath - 2\imath^2 + \jmath \end{pmatrix}.$$

Hence, $c \in \mathrm{SO}^{\circ}(5, q)$ if and only if $\imath$ and $\jmath$ satisfy the condition $\jmath - 2\imath^2 + \jmath = 2 \cdot (\jmath - \imath^2) = 0$. This holds if $\jmath = \imath^2$.

We prove what we have seen exemplary in general.

Theorem 5.12: Let $n, m, f \in \mathbb{N}$, $n = 2m + 1$ and $q = p^f$ a prime power. Define

$$T_{i,j}(\imath) \coloneqq I_n + E_{i,j}(\imath) + E_{n-j+1,n-i+1}(-\imath)$$

for $i, j \in \{1, \ldots, n\} - \{m + 1\}$, $j < i$, $i + j \neq n + 1$ and $\imath \in \mathbb{F}_q$. Then $T_{i,j}(\imath) \in \mathrm{SO}^{\circ}(n, q)$.

Proof. We use Corollary 5.4 and check that $T_{i,j}(\imath) \cdot J_{n,q}^{\circ} \cdot T_{i,j}(\imath)^{\mathrm{T}} = J_{n,q}^{\circ}$. We have

$$\begin{aligned} T_{i,j}(\imath) \cdot J_{n,q}^{\circ} \cdot T_{i,j}(\imath)^{\mathrm{T}} &= (I_n + E_{i,j}(\imath) + E_{n-j+1,n-i+1}(-\imath)) \cdot J_{n,q}^{\circ} \cdot T_{i,j}(\imath)^{\mathrm{T}} \\ &= (J_{n,q}^{\circ} + E_{i,n-j+1}(\imath) + E_{n-j+1,i}(-\imath)) \cdot T_{i,j}(\imath)^{\mathrm{T}} \\ &= (J_{n,q}^{\circ} + E_{i,n-j+1}(\imath) - E_{n-j+1,i}(\imath)) \cdot (I_n + E_{j,i}(\imath) - E_{n-i+1,n-j+1}(\imath)) \\ &= J_{n,q}^{\circ} + E_{i,n-j+1}(\imath) - E_{n-j+1,i}(\imath) + E_{n-j+1,i}(\imath) - E_{i,n-j+1}(\imath) \\ &= J_{n,q}^{\circ}. \end{aligned}$$

Hence, $T_{i,j}(\imath) \in \mathrm{SO}^{\circ}(n, q)$. □

Theorem 5.13: Let $n, m, f \in \mathbb{N}$, $n = 2m + 1$ and $q = p^f$ a prime power. Define

$$T_{i,j}(\imath) \coloneqq I_n + E_{i,j}(\imath) + E_{n-j+1,n-i+1}(2\imath) + E_{n-j+1,j}(\imath^2)$$

for $j \in \{1, \ldots, m\}$, $i = m + 1$ and $\imath \in \mathbb{F}_q$. Moreover, we define

$$T_{i,j}(\imath) \coloneqq I_n + E_{i,j}(\imath) + E_{n-j+1,n-i+1}(\tfrac{1}{2}\imath) + E_{i,n-i+1}((\tfrac{1}{2}\imath)^2)$$

for $i \in \{m + 2, \ldots, n\}$, $j = m + 1$ and $\imath \in \mathbb{F}_q$. Then $T_{i,j}(\imath) \in \mathrm{SO}^{\circ}(n, q)$.

Proof. We use Corollary 5.4 and check that $T_{i,j}(\imath) \cdot J^\circ_{n,q} \cdot T_{i,j}(\imath)^{\mathrm{T}} = J^\circ_{n,q}$. We have

$$\begin{aligned}
T_{i,j}(\imath) \cdot J^\circ_{n,q} \cdot T_{i,j}(\imath)^{\mathrm{T}} &= (I_n + E_{i,j}(\imath) + E_{n-j+1,n-i+1}(2\imath) + E_{n-j+1,j}(\imath^2)) \cdot J^\circ_{n,q} \cdot T_{i,j}(\imath)^{\mathrm{T}} \\
&= (J^\circ_{n,q} + E_{i,n-j+1}(\imath) + E_{n-j+1,i}(-\imath) + E_{n-j+1,n-j+1}(\imath^2)) \cdot T_{i,j}(\imath)^{\mathrm{T}} \\
&= (J^\circ_{n,q} + E_{i,n-j+1}(\imath) + E_{n-j+1,i}(-\imath) + E_{n-j+1,n-j+1}(\imath^2)) \\
&\quad \cdot (I_n + E_{j,i}(\imath) + E_{n-i+1,n-j+1}(2\imath) + E_{j,n-j+1}(\imath^2)) \\
&= J^\circ_{n,q} + E_{i,n-j+1}(\imath) - E_{n-j+1,i}(\imath) + E_{n-j+1,n-j+1}(\imath^2) \\
&\quad + E_{n-j+1,i}(\imath) - E_{i,n-j+1}(\imath) + E_{n-j+1,n-j+1}(\imath^2) \\
&\quad - 2E_{n-j+1,n-j+1}(\imath^2) \\
&= J^\circ_{n,q}.
\end{aligned}$$

Hence, $T_{i,j}(\imath) \in \mathrm{SO}^\circ(n,q)$. The second equation directly follows from the first one. □

We use the following notation to work with transformations in the special orthogonal group of circle type. Note that depending on i and j, the matrices defined above are selected appropriately.

Definition 5.14: Let $n, m, f \in \mathbb{N}$, $n = 2m+1$ and $q = p^f$ a prime power. For $\imath \in \mathbb{F}_q - \{0\}$, $i, j \in \{1, \dots, n\}$, $j < i$ and $i + j \neq n+1$ we set

$$T_{i,j}(\imath) \coloneqq \begin{cases} I_n + E_{i,j}(\imath) + E_{n-j+1,n-i+1}(-\imath), & \text{if } i,j \neq m+1, \\ I_n + E_{i,j}(\imath) + E_{n-j+1,n-i+1}(2\imath) + E_{n-j+1,j}(\imath^2), & \text{if } i = m+1, \\ I_n + E_{i,j}(\imath) + E_{n-j+1,n-i+1}(\frac{1}{2}\imath) + E_{i,n-i+1}((\frac{1}{2}\imath)^2), & \text{else.} \end{cases}$$

Finally, we want to check whether these matrices are Siegel transformations. First, we assume that $i, j \neq m+1$. The proof is similar to the one for the plus type.

Lemma 5.15: Let $n, m, f \in \mathbb{N}$, $n = 2m+1$ and $q = p^f$ a prime power. $T_{i,j}(\imath)$ is a Siegel transformation for $\imath \in \mathbb{F}_q - \{0\}$, $i, j \in \{1, \dots, n\} - \{m+1\}$, $j < i$ and $i + j \neq n+1$.

Proof. We choose $\nu, w \in \mathbb{F}_q^n$ where $\nu = \imath e_i$ and $w = e_{n-j+1} \in \langle \nu \rangle^\perp$. Notice that $\Phi(w,w) = 0$ which yields

$$\begin{aligned}
\rho_{\nu,w}(x) &= x + (x^T J^\circ_n e_{n-j+1})\imath e_i - (x^T J^\circ_n \imath e_i) e_{n-j+1} \\
&= x + \imath x_j e_i - \imath x_{n-i+1} e_{n-j+1} \\
&= (x_1, \dots, x_{i-1}, x_i + \imath x_j, x_{i+1}, \dots, x_{n-j+2}, x_{n-j+1} - \imath x_{n-i+1}, x_{n-j}, \dots, x_n)^T.
\end{aligned}$$

Moreover,

$$T_{i,j}(\imath)x = (x_1, \dots, x_{i-1}, x_i + \imath x_j, x_{i+1}, \dots, x_{n-j+2}, x_{n-j+1} - \imath x_{n-i+1}, x_{n-j}, \dots, x_n)^T.$$

□

Remark 5.16: $T_{i,j}(\imath)$ is not always a Siegel transformation for $\imath \in \mathbb{F}_q - \{0\}$, $i = m+1$, $j \in \{1, \ldots, n\}$, $j < i$ and $i + j \neq n + 1$. Consider for example $T_{3,1}(1) \in \mathrm{SO}^\circ(5, 3)$. [GAP] can be used to test that there are no $\nu, w \in \mathbb{F}_3^5$ such that $\rho_{\nu,w} = T_{3,1}(1)$. Therefore, we call $T_{i,j}(\imath)$ a transformation instead of a Siegel transformation from now on.

5.3 Algorithm for the Bruhat Decomposition in SO°

We now describe an algorithm to compute the Bruhat decomposition for elements of $\mathrm{SO}^\circ(n,q)$.

For the orthogonal group of circle type and our choice of the bilinear form, we have the anisotropic 1-dimensional space in the middle of the matrix. The rows and columns that cross the anisotropic space are more difficult to modify and require special consideration. That is the reason why we have also defined special transformations for operations on these columns and rows (see Definition 5.14).
Due to the group of monomial matrices in the $\mathrm{SO}^\circ(n,q)$, we can assume that our pivot element never is in the central row. Otherwise, we could eliminate the elements below and would get a monomial matrix which is not contained in the orthogonal group. After selecting the pivot element the algorithm always considers the central row or column first and eliminates the element if it is not zero. However, we can first clear out the entire column and then look at the central element of the row. The algorithm then proceeds as in the case of SO^+.

We state the algorithm UnitriangularDecompositionSOCircle in pseudo code and Example 5.17 demonstrates a computation.

Algorithm 6: UnitriangularDecompositionSOCircle

Input: $a \in \mathrm{SO}^\circ(n,q)$.
Output: A monomial matrix $w \in \mathrm{SO}^\circ(n,q)$ and two lower unitriangular matrices $u_1, u_2 \in \mathrm{SO}^\circ(n,q)$ such that $w = u_1 \cdot a \cdot u_2$.

function UnitriangularDecompositionSOCircle(a)
- $u_1 := I_n$, $u_2 := I_n$;
- **for** $c \in [n, \ldots, \frac{n-1}{2}]$ **do**
  - Search first entry $i \in [1, \ldots, n]$ with $a_{i,c} \neq 0$;
  - Set $r := i$ and piv $:= a_{r,c}$;
  - **if** $a_{\frac{n+1}{2},c}$ is not zero **then**
    - Set $i = \frac{n+1}{2}$ and $\alpha := -\frac{a_{i,c}}{\text{piv}}$;
    - **if** $r + i \neq n+1$ **then**
      - $a_{n-r+1,-} := a_{n-r+1,-} + \alpha^2 a_{r,-}$; $u_{1n-r+1,-} := u_{1n-r+1,-} + \alpha^2 u_{1r,-}$;
      - $a_{n-r+1,-} := a_{n-r+1,-} + 2\alpha a_{n-i+1,-}$; $u_{1n-r+1,-} := u_{1n-r+1,-} + 2\alpha u_{1n-i+1,-}$;
      - $a_{i,-} := a_{i,-} + \alpha a_{r,-}$; $u_{1i,-} := u_{1i,-} + \alpha u_{1r,-}$;
  - **for** $i \in [r+1, \ldots, n]$ **do**
    - $\alpha := -\frac{a_{i,c}}{\text{piv}}$;
    - **if** $r + i \neq n+1$ **then**
      - $a_{i,-} := a_{i,-} + \alpha a_{r,-}$; $u_{1i,-} := u_{1i,-} + \alpha u_{1r,-}$;
      - $a_{n-r+1,-} := a_{n-r+1,-} - \alpha a_{n-i+1,-}$; $u_{1n-r+1,-} := u_{1n-r+1,-} - \alpha u_{1n-i+1,-}$;

if $a_{r,\frac{n+1}{2}}$ is not zero **then**
  Set $j = \frac{n+1}{2}$ and $\alpha := -\frac{a_{r,j}}{\text{piv}}$;
  if $r + j \neq n + 1$ **then**
    $a_{-,n-c+1} := a_{-,n-c+1} + (\frac{1}{2}\alpha)^2 a_{-,c}$; $u_{2-,n-c+1} := u_{2-,n-c+1} + (\frac{1}{2}\alpha)^2 u_{2-,c}$;
    $a_{-,n-c+1} := a_{-,n-c+1} + \frac{1}{2}\alpha a_{-,n-j+1}$; $u_{2-,n-c+1} := u_{2-,n-c+1} + \frac{1}{2}\alpha u_{2-,n-j+1}$;
    $a_{-,j} := a_{-,j} + \alpha a_{-,c}$; $u_{2-,j} := u_{2-,j} + \alpha u_{2-,c}$;
for $j \in [c-1, \ldots, 1]$ **do**
  $\alpha := -\frac{a_{r,j}}{\text{piv}}$;
  if $c + j \neq n + 1$ **then**
    $a_{-,j} := a_{-,j} + \alpha a_{-,c}$; $u_{2-,j} := u_{2-,j} + \alpha u_{2-,c}$;
    $a_{-,n-c+1} := a_{-,n-c+1} - \alpha a_{-,n-j+1}$; $u_{2-,n-c+1} := u_{2-,n-c+1} - \alpha u_{2-,n-j+1}$;
return $[a, u_1, u_2]$;

The difference to the algorithm for the orthogonal group of plus type should become clearer through the following example.

Example 5.17: We demonstrate how Algorithm 6 works using the matrix

$$a := \begin{pmatrix} 0 & 5 & 5 & 3 & 4 \\ 2 & 5 & 6 & 4 & 5 \\ 4 & 1 & 4 & 1 & 2 \\ 1 & 3 & 4 & 5 & 3 \\ 5 & 0 & 1 & 3 & 6 \end{pmatrix} \in \mathrm{SO}^\circ(5, 7).$$

First, we start with the fifth column, pick the entry $a_{1,5} = 4$ and clear out the entry in the middle of that column.

$$\begin{pmatrix} 0 & 5 & 5 & 3 & 4 \\ 2 & 5 & 6 & 4 & 5 \\ 4 & 1 & 4 & 1 & 2 \\ 1 & 3 & 4 & 5 & 3 \\ 5 & 0 & 1 & 3 & 6 \end{pmatrix} \xrightarrow{\cdot^L T_{3,1}(3)} \begin{pmatrix} 0 & 5 & 5 & 3 & 4 \\ 2 & 5 & 6 & 4 & 5 \\ 4 & 2 & 5 & 3 & 0 \\ 1 & 3 & 4 & 5 & 3 \\ 1 & 2 & 0 & 1 & 5 \end{pmatrix}$$

We continue as for the plus type and clear everything below this element in the fifth column.

$$\begin{pmatrix} 0 & 5 & 5 & 3 & 4 \\ 2 & 5 & 6 & 4 & 5 \\ 4 & 2 & 5 & 3 & 0 \\ 1 & 3 & 4 & 5 & 3 \\ 1 & 2 & 0 & 1 & 5 \end{pmatrix} \xrightarrow{\cdot^L T_{2,1}(4)} \begin{pmatrix} 0 & 5 & 5 & 3 & 4 \\ 2 & 4 & 5 & 2 & 0 \\ 4 & 2 & 5 & 3 & 0 \\ 1 & 3 & 4 & 5 & 3 \\ 4 & 4 & 5 & 2 & 0 \end{pmatrix} \xrightarrow{\cdot^L T_{4,1}(1)} \begin{pmatrix} 0 & 5 & 5 & 3 & 4 \\ 2 & 4 & 5 & 2 & 0 \\ 4 & 2 & 5 & 3 & 0 \\ 1 & 1 & 2 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We observe the same effect we had in the plus type of the orthogonal group where the last row also automatically has the desired form. We continue the algorithm and clear the first row with $a_{1,5} = 4$. Again, we start by clearing out the entry in the middle.

$$\begin{pmatrix} 0 & 5 & 5 & 3 & 4 \\ 2 & 4 & 5 & 2 & 0 \\ 4 & 2 & 5 & 3 & 0 \\ 1 & 1 & 2 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^R T_{5,3}(4)} \begin{pmatrix} 5 & 5 & 0 & 3 & 4 \\ 5 & 4 & 5 & 2 & 0 \\ 0 & 2 & 5 & 3 & 0 \\ 5 & 1 & 2 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Analogously to the columns we can clear out the rest of the first row.

$$\begin{pmatrix} 5 & 5 & 0 & 3 & 4 \\ 5 & 4 & 5 & 2 & 0 \\ 0 & 2 & 5 & 3 & 0 \\ 5 & 1 & 2 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^R T_{5,4}(1)} \begin{pmatrix} 0 & 5 & 0 & 0 & 4 \\ 1 & 4 & 5 & 2 & 0 \\ 5 & 2 & 5 & 3 & 0 \\ 4 & 1 & 2 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^R T_{5,2}(4)} \begin{pmatrix} 0 & 0 & 0 & 0 & 4 \\ 0 & 4 & 5 & 2 & 0 \\ 0 & 2 & 5 & 3 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The first and fifth row as well as the first and fifth column are in the desired form at this step and we can, thus, continue with the next column. We select $a_{2,4} = 2$ and clear out the entry in the middle of the fourth column.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 4 \\ 0 & 4 & 5 & 2 & 0 \\ 0 & 2 & 5 & 3 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^L T_{3,2}(2)} \begin{pmatrix} 0 & 0 & 0 & 0 & 4 \\ 0 & 4 & 5 & 2 & 0 \\ 0 & 3 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Finally, we can clear the rest of the second row with $a_{2,4} = 2$ and this completes the example.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 4 \\ 0 & 4 & 5 & 2 & 0 \\ 0 & 3 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot^R T_{4,3}(1)} \begin{pmatrix} 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix} =: h.$$

Therefore,

$$h = T_{3,2}(2)T_{4,1}(1)T_{2,1}(4)T_{3,1}(3)aT_{5,3}(4)T_{5,4}(1)T_{5,2}(4)T_{4,3}(1)$$

where h is a monomial matrix. Hence, we have computed the Bruhat decomposition of a.

We also want to show the correctness of the algorithm. But before we can do this, we have to prove a result that is similar to the one in Lemma 4.18. In Chapter 5.4 we conclude that every monomial matrix $a \in \mathrm{SO}^\circ(n,q)$ has a non-zero entry $a_{\frac{n+1}{2},\frac{n+1}{2}}$. Hence, we can exclude this special case.

Lemma 5.18: Let $n, m, f \in \mathbb{N}$, $n = 2m+1$, $q = p^f$ a prime power, $j \in \{1,\dots,n\} - \{m+1\}$ and $a \in \mathrm{SO}^\circ(n,q)$ such that there exists $k \in \{1,\dots,n\} - \{m+1\}$ with $a_{k,j} \neq 0$ and $a_{k,i} = 0$ for all $i \in \{1,\dots,n\} - \{j, n-j+1\}$. Then

$$a_{i,n-j+1} = \begin{cases} a_{k,j}^{-1}, & \text{if } i = n-k+1, \\ 0, & \text{else.} \end{cases}$$

Proof. We set $t := n - j + 1$. Since $a \in \mathrm{SO}^\circ(n,q)$, we know by Corollary 5.4 that $a \cdot J^\circ_{n,q} \cdot a^{\mathrm{T}} = J^\circ_{n,q}$. Hence,

$$\begin{aligned} & a \cdot J^\circ_{n,q} \cdot a^{\mathrm{T}} \\ = & \begin{pmatrix} * & \dots & * & a_{1,t} & * & \dots & * & * & * & \dots & * \\ * & \dots & * & a_{2,t} & * & \dots & * & * & * & \dots & * \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ * & \dots & * & a_{k-1,t} & * & \dots & * & * & * & \dots & * \\ 0 & \dots & 0 & a_{k,t} & 0 & \dots & 0 & a_{k,j} & 0 & \dots & 0 \\ * & \dots & * & a_{k+1,t} & * & \dots & * & * & * & \dots & * \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ * & \dots & * & a_{n,t} & * & \dots & * & * & * & \dots & * \end{pmatrix} \cdot J^\circ_{n,q} \cdot a^{\mathrm{T}} \\ = & \begin{pmatrix} * & \dots & * & * & * & \dots & * & a_{1,t} & * & \dots & * \\ * & \dots & * & * & * & \dots & * & a_{2,t} & * & \dots & * \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ * & \dots & * & * & * & \dots & * & a_{k-1,t} & * & \dots & * \\ 0 & \dots & 0 & a_{k,j} & 0 & \dots & 0 & a_{k,t} & 0 & \dots & 0 \\ * & \dots & * & * & * & \dots & * & a_{k+1,t} & * & \dots & * \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ * & \dots & * & * & * & \dots & * & a_{n,t} & * & \dots & * \end{pmatrix} \cdot \begin{pmatrix} * & \dots & * & 0 & * & \dots & * \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ * & \dots & * & 0 & * & \dots & * \\ a_{1,t} & \dots & a_{k-1,t} & a_{k,t} & a_{k+1,t} & \dots & a_{n,t} \\ * & \dots & * & 0 & * & \dots & * \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ * & \dots & * & 0 & * & \dots & * \\ * & \dots & * & a_{k,j} & * & \dots & * \\ * & \dots & * & 0 & * & \dots & * \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ * & \dots & * & 0 & * & \dots & * \end{pmatrix} \end{aligned}$$

$$= \begin{pmatrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 \\ a_{k,j}a_{1,t} + a_{k,t}a_{1,j} & a_{k,j}a_{2,t} + a_{k,t}a_{2,j} & \dots & a_{k,j}a_{n-1,t} + a_{k,t}a_{n-1,j} & a_{k,j}a_{n,t} + a_{k,t}a_{n,j} \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \end{pmatrix}$$
$$= J^\circ_{n,q}.$$

Now, we have $(a \cdot J^\circ_{n,q} \cdot a^{\mathrm{T}})_{k,k} = a_{k,j}a_{k,t} + a_{k,t}a_{k,j} = 2a_{k,j}a_{k,t} = 0$. Since $2 \neq 0$ and $a_{k,j} \neq 0$, it follows that $a_{k,t} = 0$. Therefore, the claim follows. □

We show the same result for a column with one non-zero entry. Notice that we can also exclude the special case here.

Lemma 5.19: Let $n, m, f \in \mathbb{N}$, $n = 2m+1$, $q = p^f$ a prime power, $j \in \{1, \dots, n\} - \{m+1\}$ and $a \in \mathrm{SO}^\circ(n, q)$ such that there exists $k \in \{1, \dots, n\} - \{m+1\}$ with $a_{k,j} \neq 0$ and $a_{i,j} = 0$ for all $i \in \{1, \dots, n\} - \{k, n-k+1\}$. Then

$$a_{n-k+1,i} = \begin{cases} a_{k,j}^{-1}, & \text{if } i = n-j+1, \\ 0, & \text{else.} \end{cases}$$

Proof. We know by Corollary 5.4 that $aJ^\circ_{n,q}a^{\mathrm{T}} = J^\circ_{n,q}$. Moreover, we see that

$$(aJ^\circ_{n,q}a^{\mathrm{T}})^{-1} = (J^\circ_{n,q})^{-1} \iff a^{-\mathrm{T}}(J^\circ_{n,q})^{-1}a^{-1} = (J^\circ_{n,q})^{-1}.$$

$(J^\circ_{n,q})^{-1}$ is the Gram-matrix for a basis $\{e_1, f_1, \dots, e_m, f_m, w'\}$ such that (e_i, f_i) is a hyperbolic pair for $i \in \{1, \dots, m\}$, $w'e_i = w'f_i = 0$ for $i \in \{1, \dots, m\}$ and $w'w' = (-2^{-1})^{-1}$. Since w as in Corollary 5.4 and w' are anisotropic, there exists an isometry between $\langle w \rangle$ and $\langle w' \rangle$ and, therefore, an isometry between $\{e_1, f_1, \dots, e_m, f_m, w\}$ and $\{e_1, f_1, \dots, e_m, f_m, w'\}$ by Witt's Lemma. Hence, $a^{-\mathrm{T}} \in \mathrm{SO}^\circ(n, q)$ and, thus, $a^{\mathrm{T}} \in \mathrm{SO}^\circ(n, q)$. Therefore, the claim follows by Lemma 5.18. □

With the results so far, we can already prove the correctness of the algorithm UnitriangularDecompositionSOCircle.

Theorem 5.20: The algorithm UnitriangularDecompositionSOCircle (Algorithm 6) works correctly and terminates.

Proof. Let $a \in \mathrm{SO}^\circ(n,q)$. We start with column n and pick the least i such that $a_{i,n} \neq 0$. Such an entry must exist since $\det(a) = 1$. By Theorem 5.12 we can use the matrices $T_{i,j}$ to clear all entries $a_{k,n}$ for $k \in \{i+1,\ldots,n\}$ and $k+i \neq n+1$. Therefore, everything above and below the entry $a_{i,n}$ in column n is 0 except for $a_{n-i+1,n}$ and we only used lower-unitriangular matrices. Again, we use Theorem 5.12 and clear every entry $a_{i,k}$ left from $a_{i,n}$ for $k \in \{1,\ldots,n-1\}$ and $k+i \neq n+1$. Therefore, the row i also contains only one non-zero entry and $a_{i,1}$. By Lemma 5.18 and Lemma 5.19 it follows that $a_{n-i+1,n} = a_{i,1} = 0$ and that row $n-i+1$ as well as column $n-n+1 = 1$ contain only one non-zero entry. Hence, we can ensure that over the course of the algorithm one non-zero entry remains in the cleared row and column, respectively. Therefore, the claim follows iteratively. □

Remark 5.21: Note that for the previous proof we used the same exclusion as in Lemma 5.18 and Lemma 5.19.

5.4 Implementation of the Bruhat Decomposition in SO°

In this subsection, we compute an upper bound of the number of operations and memory slots for a constructed MSLP using Algorithm 6 UnitriangularDecompositionSOCircle. We also describe procedures for converting monomial matrices into diagonal matrices and represent diagonal matrices with an MSLP (this corresponds to the second and third step as described in Remark 3.4).

As for the plus type, we use the LGO standard generators [LGO]. For the circle type, however, we only have 6 instead of 9 generators which are presented in the next definition.

Definition 5.22: Let q be a prime power, say $q = p^f$, $n, m \in \mathbb{N}$ with $5 \leq n = 2m + 1$, $\omega \in \mathbb{F}_q$ a primitive element and $x, y \in \mathbb{N}_0$ such that $q - 1 = 2^y \cdot x$ where x is odd. The Leedham-Green-O'Brien standard generators [LGO] of $\mathrm{SO}^\circ(n, q)$ are the following:

$$\widetilde{s} = \begin{pmatrix} s_0 & 0 \\ 0 & I_{n-3} \end{pmatrix} \text{ for } s_0 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix},$$

$$\widetilde{t} = \begin{pmatrix} t_0 & 0 \\ 0 & I_{n-3} \end{pmatrix} \text{ for } t_0 = \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix},$$

$$\widetilde{\delta} = \mathrm{diag}(\omega^2, \omega^{-2}, 1, \ldots, 1),$$

$$\widetilde{u} = \begin{pmatrix} u_0 & 0 \\ 0 & I_{n-5} \end{pmatrix} \text{ for } u_0 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{pmatrix},$$

$$\widetilde{v} = \begin{cases} \begin{pmatrix} 0 & 0 & I_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & I_{n-5} \\ I_2 & 0 & 0 & 0 \end{pmatrix}, & m \text{ odd}, \\ \begin{pmatrix} 0 & 0 & I_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & I_{n-5} \\ -I_2 & 0 & 0 & 0 \end{pmatrix}, & m \text{ even and} \end{cases}$$

$$\widetilde{\sigma} = \mathrm{diag}(\omega^x, \omega^{-x}, 1, \ldots, 1).$$

Leedham-Green and O'Brien use a basis in their paper where the anisotropic space is in the top left 4×4 block. Hence, we transform the basis with the matrix

$$T_n^{\mathrm{SO}^\circ} = \begin{pmatrix} e_1 & e_n & e_{\frac{n+1}{2}} & e_2 & e_{n-1} & \dots & e_{\frac{n+1}{2}-1} & e_{\frac{n+1}{2}+1} \end{pmatrix} \in \mathrm{GL}(n,q)$$

where $\{e_1, \dots, e_n\}$ is the standard basis of $\mathbb{F}_q^n$ and n odd.

Example 5.23: For $n = 7$ we have

$$T_7^{\mathrm{SO}^\circ} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

After a basis transformation with $T_n^{\mathrm{SO}^\circ}$, we obtain the generating set below.

Definition 5.24: Let q be a prime power, say $q = p^f$, $n, m \in \mathbb{N}$ with $5 \leq n = 2m + 1$, $\omega \in \mathbb{F}_q$ a primitive element and $x, y \in \mathbb{N}_0$ such that $q - 1 = 2^y \cdot x$ where x is odd. The Leedham-Green-O'Brien standard generators [LGO] of $\mathrm{SO}^\circ(n,q)$ transformed with $T_n^{\mathrm{SO}^\circ}$ are the following:

A transposition:
$$s = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & I_{m-1} & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & I_{m-1} & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix},$$

A transformation:
$$t = I_n + 2 \cdot E_{1,\frac{n+1}{2}} + E_{1,n} + E_{\frac{n+1}{2},n},$$

A diagonal matrix:
$$\delta = \mathrm{diag}(\omega^2, 1, \dots, 1, \omega^{-2}),$$

Two 2-cycles:
$$u = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_{n-4} & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

Product of two cycles of length m:

$$v = \begin{cases} \begin{pmatrix} 0 & I_{m-1} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & I_{m-1} & 0 \end{pmatrix}, & m \text{ odd}, \\ \begin{pmatrix} 0 & I_{m-1} & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & I_{m-1} & 0 \end{pmatrix}, & m \text{ even}, \end{cases}$$

A diagonal matrix: $\sigma = \mathrm{diag}(\omega^x, 1, \ldots, 1, \omega^{-x})$.

Example 5.25: We get the following generating set for $n = 7$ and $q = 7$:

$$s = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad t = \begin{pmatrix} 1 & 0 & 0 & 2 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \delta = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 \end{pmatrix},$$

$$u = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad v = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad \sigma = \begin{pmatrix} 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6 \end{pmatrix}.$$

For the orthogonal group of circle type, we have to construct two special matrices to generate the transformations from Algorithm 6. First, we want to write the special matrix $\mathrm{diag}(\omega, 1, \ldots, 1, \omega^{-1}) \in \mathrm{SO}^\circ(n, q)$ as a word in the LGO standard generators.

Lemma 5.26: Let q be a prime power, say $q = p^f$, $5 \leq n = 2m+1$ a positive odd integer, $\omega \in \mathbb{F}_q$ a primitive element and $x, y \in \mathbb{N}_0$ such that $q - 1 = 2^y \cdot x$ where x is odd. Then we have that

$$\delta_* \coloneqq \operatorname{diag}(\omega, 1, \ldots, 1, \omega^{-1}) \in \mathrm{SO}^\circ$$

and δ_* can be written in the LGO standard generators as

$$\delta_* = \begin{cases} \sigma, & \text{if } x = 1, \\ \delta^{\frac{1-x}{2}}\sigma, & \text{else.} \end{cases}$$

Proof. If $x = 1$, we clearly have

$$\sigma = \operatorname{diag}(\omega^x, 1, \ldots, 1, \omega^{-x}) = \operatorname{diag}(\omega, 1, \ldots, 1, \omega^{-1}) \in \mathrm{SO}^\circ.$$

If $x \neq 1$, we set $i \coloneqq \frac{1-x}{2}$ and get

$$\begin{aligned} &\delta^{\frac{1-x}{2}}\sigma \\ &= \operatorname{diag}(\omega^{2i}, 1, \ldots, 1, \omega^{-2i}) \cdot \operatorname{diag}(\omega^x, 1, \ldots, 1, \omega^{-x}) \\ &= \operatorname{diag}(\omega^{2i+x}, 1, \ldots, 1, \omega^{-(2i+x)}) \in \mathrm{SO}^\circ\,. \end{aligned}$$

Moreover, $2i + x = 2\frac{1-x}{2} + x = 1 - x + x = 1$. Therefore, the claim follows. □

Notice that all LGO standard generators of $\mathrm{SO}^\circ(n, q)$ do not have any similarity to the matrix $T_{2,1}(\imath)$. Hence, we have to construct a formula which describes matrices of the form $T_{i,j}(\imath)$ where $i, j \notin \{m+1\}$, $i + j \neq n+1$ and $i, j \in \{2, \ldots, m\}$ or $i, j \in \{m+2, \ldots, n-1\}$.

Lemma 5.27: Let q be a prime power, say $q = p^f$, $5 \leq n = 2m+1$ a positive odd integer and $\omega \in \mathbb{F}_q$ a primitive element. Then $T_{1,n-1}(1)$ can be written in the LGO standard generators as

$$T_{1,n-1}(1) = v^{-1}t^{-1}vt^j v^{-1}tvt^{-j}$$

where $j \equiv 2^{-1} \bmod q$.

Proof. First notice that the conjugation with v induces a permutation such that

$$\begin{aligned} v^{-1}tv &= I_n + 2 \cdot E_{2,m+1} + E_{2,n-1} + E_{m+1,n-1} \text{ and} \\ v^{-1}t^{-1}v &= I_n - 2 \cdot E_{2,m+1} + E_{2,n-1} - E_{m+1,n-1}. \end{aligned}$$

This yields

$$v^{-1}t^{-1}vt^jv^{-1}tvt^{-j} = (I_n - 2\cdot E_{2,m+1} + E_{2,n-1} - E_{m+1,n-1})t^j \\ (I_n + 2\cdot E_{2,m+1} + E_{2,n-1} + E_{m+1,n-1})t^{-j}.$$

We now have

$$v^{-1}t^{-1}vt^j = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 2j & 0 & \dots & 0 & 0 & j^2 \\ 0 & 1 & 0 & \dots & 0 & -2 & 0 & \dots & 0 & 1 & \jmath \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & -1 & j \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 1 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{pmatrix}$$

$$v^{-1}tvt^{-j} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & -2j & 0 & \dots & 0 & 0 & j^2 \\ 0 & 1 & 0 & \dots & 0 & 2 & 0 & \dots & 0 & 1 & \jmath \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & -j \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 1 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{pmatrix}$$

where $\jmath = q - 2j$. Therefore,

$$v^{-1}t^{-1}vt^jv^{-1}tvt^{-j} = I_n + 2j\cdot E_{1,n-1} - 2j\cdot E_{2,n}.$$

Hence, the claim follows if we choose $j \equiv 2^{-1} \bmod q$. □

Now, we have everything to describe the transformations of $\mathrm{SO}^\circ(n,q)$.

Theorem 5.28: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $5 \leq n = 2m + 1$ and $\omega \in \mathbb{F}_q$ a primitive element of $\mathbb{F}_q$. Let $(s, t, \delta, u, v, \sigma)$ be the LGO standard generators as in Definition 5.24 and δ_* as in Lemma 5.26. We set

$$u_1 \coloneqq u,$$
$$u_i \coloneqq v^{-1} u_{i-1} v.$$

Then all matrices of Algorithm UnitriangularDecompositionSOCircle (Algorithm 6) are a product of the LGO standard generators as shown in the following tables where $j \equiv 2^{-1} \bmod q$:

Entry in the bottom left block.

Matrix	**Word in terms of the LGO generators**
$T_{n,2}(\omega^\ell)$	$(\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2)(s^{-1}u^{-1})^2 v^{-1} t^{-1} v t^j v^{-1} t v t^{-j} (us)^2 (\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2)^{-1}$
$T_{n,i}(\omega^\ell)$	$u v^{-1} T_{n,i-1}(\omega^\ell) v u^{-1}$
$T_{j,i}(\omega^\ell)$	$u_{n-j+1}^{-1} T_{j-1,i}(\omega^\ell) u_{n-j+1}$

Entry on the center row and column.

Matrix	**Word in terms of the LGO generators**
$T_{n,m+1}(\omega^\ell)$	$(\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2) s^{-1} t^{-j} s (\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2)^{-1}$
$T_{i,m+1}(\omega^\ell)$	$v^{-1} T_{i-1,m+1}(\omega^\ell) v$

Entry in the top left or bottom right block.

Matrix	**Word in terms of the LGO generators**
$T_{2,1}(\omega^\ell)$	$(\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2) s^{-1} u^{-1} (v^{-1} t^{-1} v t^j v^{-1} t v t^{-j})^{-1} u s (\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2)^{-1}$
$T_{i+1,i}(\omega^\ell)$	$v^{-1} T_{i,i-1}(\omega^\ell) v$
$T_{i,j}(\omega^\ell)$	$u_j T_{i,j+1}(\omega^\ell) u_j^{-1}$

Proof. This follows directly from Lemma 5.27, Lemma 5.26 of this thesis and [DR, Theorem 4.35]. □

Finally, we can estimate the length and the amount of memory slots for an MSLP, which computes the Bruhat decomposition for an element in the orthogonal group of circle type.

Theorem 5.29: Let $q = p^f$ with p prime, $f \geq 1$, $n = 2m+1$ an odd integer with $n \geq 5$ and $a \in \mathrm{SO}^\circ(n,q)$. Set $b = 17 + 3f + m$ and

$$\begin{aligned}\lambda =& \frac{5q-1}{2} + 38f + n + 1 + (n^2 - 2n)(f + 2\log_2(q) + 2\log_2(m-2) + n - 1) \\ &+ n(f + \log_2(q) + \log_2(m-2) + 1).\end{aligned}$$

There exists a b-MSLP S of length at most λ such that if S is evaluated with memory containing the list $\{s, t, \delta, u, v, \sigma\}$, then S outputs memory containing (u_1, u_2) with $a = u_1 w u_2$ the Bruhat decomposition of a.

Proof. We use the same technique as in Theorem 4.27 which means that we obtain $T_{2,1}(\beta)$ for $\beta \in \mathbb{F}_q$ through

$$T_{2,1}(\beta) = T_{2,1}(\sum_{i=0}^{f-1} k_i \omega^i) = \prod_{i=0}^{f-1} T_{2,1}(\omega^i)^{k_i}.$$

Hence, we just calculate the transformations $T_{2,1}(\omega^0), T_{2,1}(\omega^1), \ldots, T_{2,1}(\omega^{f-1})$ and store them. We use the same method for transformations with non-zero entries in the lower left block and the middle row and columns.
Therefore, we start by computing these transformations. First, we look at the transformations $T_{2,1}(\omega^i)$ for $i \in \{0, \ldots, f-1\}$ and use the formula of Theorem 5.28. For this, we need δ_* which we can compute via Lemma 5.26 with at most $1 + \frac{q-1}{2}$ operations. Moreover, we need at most $q + 7$ operations for the formula of Lemma 5.27. We need 6 operations for $\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2$ (notice that we compute $\delta_*^{\ell+1}$ by one multiplication of δ_*^ℓ and δ_*) and thus 12 operations in total for

$$(\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2) s^{-1} u^{-1} (v^{-1} t^{-1} v t^j v^{-1} t v t^{-j})^{-1} u s (\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2)^{-1}.$$

Since we do this f times, we need $12f$ operations. Analogously, it follows that we need $16f$ operations for

$$T_{n,2}(\omega^i) = (\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2)(s^{-1}u^{-1})^2 v^{-1} t^{-1} v t^j v^{-1} t v t^{-j} (us)^2 (\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2)^{-1}$$

for $i \in \{0, \ldots, f-1\}$ and $10f$ operations for

$$T_{n,\frac{n+1}{2}}(\omega^i) = (\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2) s^{-1} t^{-j} s (\delta_*^{-\ell} v^{-2} \delta_*^\ell v^2)^{-1}$$

for $i \in \{0, \ldots, f-1\}$. Moreover, we store the u_i of Theorem 4.26 for $i \in \{1, \ldots, m-2\}$ and need $2(m-3) = n - 7$ operations to calculate them. In total, we need $\frac{5q-1}{2} + 38f + n + 1$ operations for the initialization.
Now, we start with Algorithm 6. We consider a matrix with a non-zero entry in every position. We need at most $f - 1 + 2f\log_2(p) = f - 1 + 2\log_2(q)$ operations to calculate a transformation $T_{2,1}(\beta)$, $T_{n,2}(\beta)$ or $T_{n,\frac{n+1}{2}}(\beta)$. To shift $T_{n,2}(\beta)$ by an index j we need at most

$3+2\log_2(m-2)$ operations and by an index i we need at most $2(m-1) = 2m+1-1-2 = n-3$ operations. We need to do this at most $\frac{n^2-2n}{2}$ times. If we sum up these results, we obtain

$$\tfrac{n^2-2n}{2}(f + 2\log_2(q) + 2\log_2(m-2) + n - 1)$$

operations. Analogously, it follows that we need at most

$$\begin{aligned}&\frac{n^2-2n}{2}(f - 1 + 2\log_2(q) + 2 + 2\log_2(m-2) + 2(m-1))\\ \leq &\frac{n^2-2n}{2}(f + 2\log_2(q) + 2\log_2(m-2) + n)\end{aligned}$$

operations for all transformations in the top left and lower right block and

$$n(f - 1 + 2\log_2(q) + 2 + 2\log_2(m-2)) = n(f + \log_2(q) + \log_2(m-2) + 1)$$

operations for all transformations on the middle row and column, yielding the claimed maximum total length of the MSLP which is

$$\tfrac{5q-1}{2}+38f+n+1+(n^2-2n)(f+2\log_2(q)+2\log_2(m-2)+n-1)+n(f+\log_2(q)+\log_2(m-2)+1).$$

We need 12 memory slots for the LGO standard generators and their inverse elements, 2 memory slots for repeated squaring, 2 memory slots for the results u_1 and u_2, 1 memory slot for the current transformation, 1 additional memory slot to save some calculations and 1 memory slot for δ_*.
Moreover, we need f memory slots for the matrices $T_{2,1}(\omega^i)$ for $i \in \{0, \ldots, f-1\}$, f memory slots for the matrices $T_{n,2}(\omega^i)$ for $i \in \{0, \ldots, f-1\}$, f memory slots for the matrices $T_{n,\frac{n+1}{2}}(\omega^i)$ for $i \in \{0, \ldots, f-1\}$ and $m-2$ memory slots for the remaining u_i for $i \in \{1, \ldots, m-2\}$. Thus, we need

$$12 + 2 + 2 + 1 + 1 + 1 + f + f + f + m - 2 = 17 + 3f + m$$

memory slots altogether. □

This concludes step 1 as described in Remark 3.4. The next goal is to transform a monomial matrix in $\mathrm{SO}^\circ(n,q)$ into a diagonal matrix in $\mathrm{SO}^\circ(n,q)$ and to write this in terms of the LGO standard generators.

Let $M_{\mathrm{SO}^\circ} \leq \mathrm{SO}^\circ(n,q)$ be the subgroup of monomial matrices of $\mathrm{SO}^\circ(n,q)$. We use the same map into S_n as for the plus type which means we define

$$\varphi_{\mathrm{SO}^\circ} \colon M_{\mathrm{SO}^\circ} \to S_n$$

such that $g \in M_{\mathrm{SO}^\circ}$ permutes the subspaces $\langle e_1\rangle, \ldots, \langle e_n\rangle$ in the same way how $\varphi_{\mathrm{SO}^\circ}(g)$ permutes $1, \ldots, n$. Definitely, $\varphi_{\mathrm{SO}^\circ}(M_{\mathrm{SO}^\circ})$ is generated by the images of the standard generators

s, u, v from Definition 5.24 since these are non-trivial monomial matrices. We set $m := \frac{n-1}{2}$ and get the following generating set:

$$\varphi_{\mathrm{SO}^\circ}(M_{\mathrm{SO}^\circ}) = \langle (1,n), (1,2)(n-1,n), (1,2,\ldots,m)(m+2,n,n-1,\ldots,m+3)\rangle =: G_{\mathrm{SO}^\circ}$$

This is the same group as for the special unitary matrices of odd dimension. Therefore, we summarize the algorithm for monomial matrices from [DR].

Let $\pi \in G_{\mathrm{SO}^\circ}$. We want to find an MSLP that outputs π as a word in these generators when evaluated with the input $(1,n), (1,2)(n-1,n)$ and $(1,2,\ldots,m)(m+2,n,n-1,\ldots,m+3)$. Let us assume that the inverse of the last element is also given as input. Note that $(m+1)^\pi = m+1$ for all $\pi \in G_{\mathrm{SO}^\circ}$.
First, we divide $\Omega = \{1,\ldots,n\} - \{m+1\}$ into two sets, namely $\Omega_1 = \{1,\ldots,m\}$ and $\Omega_2 = \{m+2,\ldots,n\}$. We know that $S_m = S_{\Omega_1}$ is generated by the permutations $(1,2)$ and $(1,2,\ldots,m)$ and, therefore, we can also write every element in S_m by $(1,2)(n-1,n)$ and $(1,2,\ldots,m)(m+2,n,n-1,\ldots,m+3)$ if we ignore the cycle in S_{Ω_2}. We describe how we can do that later. Our first aim is to factorize π into $\pi = g_1^{-1}\pi_1\pi_2 g_2^{-1}$ where $g_1, g_2 \in G_{\mathrm{SO}^\circ}$, $\pi_1 \in S_{\Omega_1}$ and $\pi_2 \in S_{\Omega_2}$.
For this let $\pi = z_1 \ldots z_k$ where $z_i \in S_n$ is a non-trivial cycle. We now iterate over π cycle by cycle. Therefore, let $i \in \{1,\ldots,k\}$ and $g_1, g_2 = \mathrm{Id}_{S_n}$.
Case 1: The largest moved point of z_i is smaller than $m+1$ or the smallest moved point of z_i is greater than $m+1$:
In this case, we have nothing to do since z_i has the required form. We can continue with the next cycle.
Case 2: Let $\imath$ be the largest moved point of z_i and $(n-\imath+1)^{z_i} = (n-\imath+1)$:
Since we are not in the first case, we know that z_i moves a point in Ω_1 and in Ω_2. We know that z_i stabilizes $(n-\imath+1)$ and, therefore, $(n-\imath+1)$ is not in z_i. We use the transposition $(n-\imath+1,\imath) = (1,n)^{(1,2,\ldots,m-1)(m+1,n,n-1,\ldots,m+2)^{n-\imath+1}}$ by conjugation on z_i. Moreover, we store this action by $g_1 \leftarrow (n-\imath+1,\imath)g_1$ and $g_2 \leftarrow g_2(n-\imath+1,\imath)$. Again, we go through the whole procedure for this cycle.
Case 3: Let $\imath$ be the largest moved point of z_i. Then z_i moves $\imath$ and $n-\imath+1$:
In this case, we split z_i in two cycles.
Let $z_i = (\imath_1,\ldots,\imath_{\ell_1},\imath,\imath_{\ell_1+1},\ldots,\imath_{\ell_1+\ell_2},n-\imath+1,\imath_{\ell_1+\ell_2+1},\ldots,\imath_{\ell_1+\ell_2+\ell_3})$. Again, we use $(n-\imath+1,\imath) = (1,n)^{(1,2,\ldots,m-1)(m+1,n,n-1,\ldots,m+2)^{n-\imath+1}}$, but this time by left multiplication. We have

$$\begin{aligned}
&(n-\imath+1,\imath)\cdot z_i\\
=&(n-\imath+1,\imath)\cdot(\imath_1,\ldots,\imath_{\ell_1},\imath,\imath_{\ell_1+1},\ldots,\imath_{\ell_1+\ell_2},n-\imath+1,\imath_{\ell_1+\ell_2+1},\ldots,\imath_{\ell_1+\ell_2+\ell_3})\\
=&(n-\imath+1,\imath_{\ell_1+1},\ldots,\imath_{\ell_1+\ell_2})(\imath,\imath_{\ell_1+\ell_2+1},\ldots,\imath_{\ell_1+\ell_2+\ell_3},\imath_1,\ldots,\imath_{\ell_1}).
\end{aligned}$$

We store this by $g_1 \leftarrow (n-\imath+1,\imath)g_1$. Again, we go through the same procedure for both new cycles.

In the end, we have that $g_1 \pi g_2 = \pi_1 \pi_2$ where $g_1, g_2 \in G_{\mathrm{SO}^\circ}$, $\pi_1 \in S_{\Omega_1}$ and $\pi_2 \in S_{\Omega_2}$. Since $\pi_1 \pi_2 \in G_{\mathrm{SO}^\circ}$, we have that $\pi_1 \pi_2 \in \langle (1,2)(n-1,n), (1,2,\ldots,m)(m+2,n,n-1,\ldots,m+3) \rangle$. Now, we can write π_1 as a word of $(1,2)$ and $(1,2,\ldots,m)$. Clearly, we also get π_2, otherwise we would have to multiply $(1,2)(n-1,n)$ or $(1,2,\ldots,m)(m+2,n,n-1,\ldots,m+3)$ from the left side or the right side and, therefore, destroy π_1. That is a contradiction to

$$\pi_1 \pi_2 \in \langle (1,2)(n-1,n), (1,2,\ldots,m)(m+2,n,n-1,\ldots,m+3) \rangle.$$

It remains to write π_1 as a word of $(1,2)$ and $(1,2,\ldots,m)$. How to do this is described in [NPP, Chapter 3.3]. Therefore, we only restate the result. Let $\pi \in S_m$ be a permutation. Moreover, we set $s_1 \coloneqq (1,2)$ and $v_1 \coloneqq (1,m,m-1,\ldots,2) = (1,2,\ldots,m)^{-1}$. We have

$$\pi = \left(\prod_{i=1}^{m} v_i^{\pi_{i-1}(i)-i} \right)^{-1}$$

where

$$\begin{aligned} v_i &\coloneqq (i,m,m-1,\ldots,i+1), \\ \pi_0 &\coloneqq \pi && \text{and} \\ \pi_i &\coloneqq \pi_{i-1} \cdot v_i^{\pi_{i-1}(i)-i} && \text{for } 1 \leq i. \end{aligned}$$

Furthermore, v_i is computed recursively from s_{i-1} and v_{i-1} via

$$v_i \coloneqq s_{i-1} v_{i-1} \qquad \text{where } s_i \coloneqq (i,i+1),$$

with s_i computed recursively via

$$s_i \coloneqq v_1 s_{i-1} v_1^{-1}.$$

Hence, we can write every element of G_{SO° in terms of the generators above.

We give the algorithm MonomialSOCircle in pseudo code and demonstrate a computation in Example 5.30.

Algorithm 7: MonomialSOCircle

Input: $a \in M_{\mathrm{SO}^\circ}(n,q)$.
Output: MSLP S which outputs the permutation $\varphi_{\mathrm{SO}^\circ}(a)$ with the generators of G_{SO°

```
function MonomialSOCircle(a)
    π := φ_SO°(a);
    c_p := Cycle of permutation π;
    u_1 := (), u_2 := ();                    // Neutral element of S_n.
    Add instructions to MSLP S;
```

while cycle $c \in c_p$ exists s.t. c operates non-trivial on $\{1,\dots,m\}$ and $\{m+2,\dots,n\}$ **do**
  $k \coloneqq \text{LargestMovedPoint}(c)$;
  if $(n-k+1)^c = (n-k+1)$ **then**
    // We are in Case 2.
    $\pi \coloneqq \pi^{(k,n-k+1)}$; $c_p \coloneqq$ Cycle of permutation π;
    $u_1 \coloneqq (k, n-k+1)u_1$;
    $u_2 \coloneqq u_2(k, n-k+1)$;
    Add instructions to S;
  else
    // We are in Case 3.
    $\pi \coloneqq (k, n-k+1)\pi$; $c_p \coloneqq$ Cycle of permutation π;
    $u_1 \coloneqq (k, n-k+1)u_1$;
    Add instructions to S;
// $u_1^{-1}\pi u_2^{-1}$ is the starting permutation. With S we encode u_1 and u_2.
// Moreover, $\pi = \pi_1\pi_2$, $\pi_1 \in S_{\Omega_1}$, $\pi_2 \in S_{\Omega_2}$.
$\pi_1 \coloneqq$ Cycles of permutation π with largest moved point smaller or equal than m;
Initialize as in [NPP, Chapter 3.3];
for $i \in [1,\dots,m-1]$ **do**
  $e \coloneqq i^{\pi_1} - i$;
  $\pi_1 \coloneqq \pi_1 v_i^e$;
  $v_{i+1} \coloneqq s_i v_i$;
  $s_{i+1} \coloneqq s_i^{v_1^{-1}}$;
  Add instructions to S;
return S;

To better understand how and why this algorithm works, we give an example.

Example 5.30: We consider the matrix

$$a \coloneqq \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix} \in \mathrm{SO}^\circ(13,5)$$

with $\varphi_{\mathrm{SO}^\circ}(a) = (1,13)(2,11)(3,12)(4,8,5,10,6,9) =: \pi = \tilde{\pi}$. We iterate over π cycle by cycle. For the initialization we set $u_1 = (), u_2 = (), \Omega_1 = \{1,\dots,6\}$ and $\Omega_2 = \{8,\dots,13\}$. We start with $(1,13)(2,11)(3,12)(4,8,5,10,6,9)$. Notice that $1^{(1,13)} = 13$ and $13^{(1,13)} = 1$. Hence, $(1,13)$ does neither lie in S_{Ω_1} nor S_{Ω_2}. The largest moved point of $(1,13)$ is $k = 13$ and $n-k+1 = 13-13+1 = 1$. Since $1^{(1,13)} \neq 1$ we are in the third case. Therefore, we set

$$\pi \leftarrow (1,13)\cdot\pi = (1,13)\cdot(1,13)(2,11)(3,12)(4,8,5,10,6,9) = (2,11)(3,12)(4,8,5,10,6,9),$$
$$u_1 \leftarrow (1,13)\cdot u_1 = (1,13)\cdot() = (1,13).$$

We consider the next cycle of $\pi = (2,11)(3,12)(4,8,5,10,6,9)$. With the same argument as above, we are not in the first case. The largest moved point of $(2,11)$ is $k = 11$ and $n-k+1 = 13-11+1 = 3$. Since $3^{(2,11)} = 3$ we are in the second case. Hence, we have to conjugate π with $(3,11)$

$$\pi \leftarrow \pi^{(3,11)} = (2,11)(3,12)(4,8,5,10,6,9)^{(3,11)} = (2,3)(4,8,5,10,6,9)(11,12),$$
$$u_1 \leftarrow (3,11)\cdot u_1 = (3,11)\cdot(1,13) = (1,13)(3,11),$$
$$u_2 \leftarrow u_2\cdot(3,11) = ()\cdot(3,11) = (3,11).$$

The next cycle is $(2,3)(4,8,5,10,6,9)(11,12)$. We have $(2,3) \in S_{\Omega_1}$, which means that we are in the first case. We look at the next cycle $(2,3)(4,8,5,10,6,9)(11,12)$. The largest moved point of $(4,8,5,10,6,9)$ is $k = 10$ and $n-k+1 = 4$. Since $4^{(4,8,5,10,6,9)} = 8$, we are in the third case. We multiply with $(4,10)$ from the left side and obtain

$$\pi \leftarrow (4,10)\cdot\pi = (4,10)\cdot(2,3)(4,8,5,10,6,9)(11,12) = (2,3)(4,6,9)(5,10,8)(11,12),$$
$$u_1 \leftarrow (4,10)\cdot u_1 = (4,10)\cdot(1,13)(3,11) = (1,13)(3,11)(4,10).$$

The next cycle is $(2,3)(4,6,9)(5,10,8)(11,12)$. The largest moved point is $k = 9$ and $n-k+1 = 5$. Since $5^{(4,6,9)} = 5$, we are in the second case. We conjugate π with $(5,9)$. This yields

$$\pi \leftarrow \pi^{(5,9)} = (2,3)(4,6,9)(5,10,8)(11,12)^{(3,11)} = (2,3)(4,6,5)(9,10,8)(11,12),$$
$$u_1 \leftarrow (5,9)\cdot u_1 = (5,9)\cdot(1,13)(3,11)(4,10) = (1,13)(3,11)(4,10)(5,9),$$
$$u_2 \leftarrow u_2\cdot(5,9) = (3,11)\cdot(5,9) = (3,11)(5,9).$$

Notice that $\pi = (2,3)(4,6,5)(9,10,8)(11,12) = u_1\tilde{\pi}u_2$ and that now every cycle is in the first case. We set $\pi_1 := (2,3)(4,6,5)$ and are now in the position to apply the algorithm from [NPP, Chapter 3.3] on π_1. We start with

$$\pi_1 := (2,3)(4,6,5),$$
$$v := (1,6,5,4,3,2) = v_1 \text{ and}$$
$$s := (1,2).$$

We know that the algorithm needs five iterations and, thus, we can traverse them.

- Iteration 1:

$$\begin{aligned} i &\leftarrow 1 \\ p &\leftarrow 1^{\pi_1} - 1 = 0 \\ \pi_1 &\leftarrow \pi_1 v^p = \pi_1 v^0 = \pi_1 \\ v &\leftarrow sv = (1,2)(1,6,5,4,3,2) = (2,6,5,4,3) \\ s &\leftarrow v_1 s v_1^{-1} = (2,3) \end{aligned}$$

- Iteration 2:

$$\begin{aligned} i &\leftarrow 2 \\ p &\leftarrow 2^{\pi_1} - 2 = 3 - 2 = 1 \\ \pi_1 &\leftarrow \pi_1 v^p = \pi_1 v^1 = (3,6,4,5) \\ v &\leftarrow sv = (2,3)(2,6,5,4,3) = (3,6,5,4) \\ s &\leftarrow v_1 s v_1^{-1} = (3,4) \end{aligned}$$

- Iteration 3:

$$\begin{aligned} i &\leftarrow 3 \\ p &\leftarrow 3^{\pi_1} - 3 = 6 - 3 = 3 \\ \pi_1 &\leftarrow \pi_1 v^p = \pi_1 v^3 = (4,6,5) \\ v &\leftarrow sv = (2,3)(2,6,5,4,3) = (4,6,5) \\ s &\leftarrow v_1 s v_1^{-1} = (4,5) \end{aligned}$$

- Iteration 4:

$$\begin{aligned} i &\leftarrow 4 \\ p &\leftarrow 4^{\pi_1} - 4 = 6 - 4 = 2 \\ \pi_1 &\leftarrow \pi_1 v^p = \pi_1 v^2 = () \\ v &\leftarrow sv = (2,3)(2,6,5,4,3) = (4,5) \\ s &\leftarrow v_1 s v_1^{-1} = (5,6) \end{aligned}$$

- Iteration 5:

$$\begin{aligned} i &\leftarrow 5 \\ p &\leftarrow 5^{\pi_1} - 5 = 5 - 5 = 0 \\ \pi_1 &\leftarrow \pi_1 v^p = \pi_1 v^0 = \pi_1 \\ v &\leftarrow sv = (2,3)(2,6,5,4,3) = () \\ s &\leftarrow v_1 s v_1^{-1} = (1,6) \end{aligned}$$

At this point we have successfully represented $(1,13)(2,11)(3,12)(4,8,5,10,6,9)$ as a word in the generators.

We also want to know the length and memory quota of an MSLP for this algorithm.

Theorem 5.31: Let $n, m \in \mathbb{N}$, $n = 2m+1$, $\pi \in G_{\mathrm{SO}^\circ}$, $b = 11$ and

$$\lambda = (2^{\lceil \log_2(n) \rceil} - 1)(4\log_2(n) + 7) + 2n\log_2(n) + 4n.$$

There exists a b-MSLP S of length at most λ such that if S is evaluated with memory containing $\{s, u, v, v^{-1}\}$, then S outputs memory containing a monomial matrix $g \in \mathrm{SO}^\circ$ such that $\varphi_{\mathrm{SO}^\circ}(g) = \pi$.

Proof. In the worst case, we have to split the permutation into a product of 2-cycles. Therefore, we are in the third case at most $2^{\lceil \log_2(n) \rceil} - 1$ times. For this we use

$$(1, 2, \ldots, m)(m+2, n, n-1, \ldots, m+3)^{n-\imath+1}.$$

Calculating this via repeated squaring needs [NPP, Section 2.2(ii)] at most

$$2\log_2(n)$$

MSLP instructions. We conjugate $(1,n)$ with $(1,2,\ldots,m)(m+1,n,n-1,\ldots,m+2)^{n-\imath+1}$ and multiply the result from the left side. This costs 3 MSLP instructions. Therefore, we need at most

$$(2^{\lceil \log_2(n) \rceil} - 1)(2\log_2(n) + 3)$$

MSLP instructions. In the worst, case all 2-cycles are in the second case. Therefore, we have to perform the second case at most $2^{\lceil \log_2(n) \rceil} - 1$ times. Analogously to the third case, we calculate

$$(1,n)^{(1,2,\ldots,m)(m+1,n,n-1,\ldots,m+2)^{n-\imath+1}}$$

and this needs at most $2\log_2(n)+2$ MSLP instructions. This time, we conjugate our permutation and this needs 2 more instructions. Hence, we need at most

$$(2^{\lceil \log_2(n)\rceil}-1)(2\log_2(n)+4)$$

MSLP instructions. Finally, we use the algorithm from [NPP, Section 3.3]. This needs at most $2n\log_2(n)+4n$ MSLP instructions, yielding the claimed maximum total length of the MSLP which is

$$(2^{\lceil \log_2(n)\rceil}-1)(4\log_2(n)+7)+2n\log_2(n)+4n.$$

In addition to storing the input X, the memory quota for the MSLP is as follows. Two memory slots are needed to store the permutations we multiply from the left and right side. Additionally, we need three memory slots for the algorithm from [NPP, Section 3.3]. Two memory slots are needed for repeated squaring [NPP, Section 2.2(ii)]. Thus, we need

$$4+2+3+2=11$$

memory slots in total. □

Finally, we are in the third step as described in Remark 3.4 and have to write a diagonal matrix in SO° in terms of the LGO standard generators. We observe the following property of diagonal matrices in SO° which helps us to solve the problem at hand and is similar to the result for the plus type.

Lemma 5.32: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $a = \mathrm{diag}(a_1, \ldots, a_n) \in \mathrm{SO}^\circ(n,q)$ a diagonal matrix and $n = 2m+1$. Then $a_i = a_{n-i+1}^{-1}$ for $i \in \{1, \ldots, n\} - \{m+1\}$ and $a_{m+1} = 1$.

Proof. We use Corollary 5.4 and know that $a \cdot J^\circ_{n,q} \cdot a^{\mathrm{T}} = J^\circ_{n,q}$. We have

$$\begin{aligned}
a \cdot J^\circ_{n,q} \cdot a^{\mathrm{T}} &= a \cdot J^\circ_{n,q} \cdot a \\
&= \mathrm{diag}(a_1, \ldots, a_n) \cdot J^\circ_{n,q} \cdot \mathrm{diag}(a_1, \ldots, a_n) \\
&= \begin{pmatrix}
0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & a_1 \\
0 & 0 & \ldots & 0 & 0 & 0 & \ldots & a_2 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \ldots & 0 & 0 & a_m & \ldots & 0 & 0 \\
0 & 0 & \ldots & 0 & -\frac{1}{2}a_{m+1} & 0 & \ldots & 0 & 0 \\
0 & 0 & \ldots & a_{m+2} & 0 & 0 & \ldots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & a_{n-1} & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 \\
a_n & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0
\end{pmatrix} \cdot \mathrm{diag}(a_1, \ldots, a_n)
\end{aligned}$$

$$
\begin{aligned}
&= \begin{pmatrix}
0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & a_1a_n \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & a_2a_{n-1} & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \dots & 0 & 0 & a_ma_{m+2} & \dots & 0 & 0 \\
0 & 0 & \dots & 0 & -\frac{1}{2}a_{m+1}^2 & 0 & \dots & 0 & 0 \\
0 & 0 & \dots & a_{m+2}a_m & 0 & 0 & \dots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & a_{n-1}a_2 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\
a_na_1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0
\end{pmatrix} \\
&= J_{n,q}^{\circ}.
\end{aligned}
$$

Now, $\det(a) = a_1a_2 \dots a_ma_{m+1}a_{m+2} \dots a_{n-1}a_n = a_{m+1} = 1$ since $a_i = a_{n-i+1}^{-1}$ for $i \in \{1, \dots, n\} - \{m+1\}$. Hence, the claim follows. □

Since the result is similar to the one of the plus type, it is advisable to work with similar matrices. We define these matrices for the circle type in the next definition and show how these can be expressed as words in the LGO standard generators.

Definition 5.33: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m+1$ and $\omega \in \mathbb{F}_q$ a primitive element. We set

$$h_j^{\mathrm{SO}^\circ} \coloneqq \mathrm{diag}(\underbrace{1, \dots, 1}_{j-1}, \omega, 1, \dots, 1, \omega^{-1}, \underbrace{1, \dots, 1}_{j-1})$$

for $j \in \{1, \dots, m\}$.

Since the generators of the plus and circle type differ from each other, we obviously need new formulas to describe them. The next lemma shows how we can write the matrices of the previous definition in terms of the LGO standard generators.

Lemma 5.34: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m+1$ and $\omega \in \mathbb{F}_q$ a primitive element. We have

$$
\begin{aligned}
h_1^{\mathrm{SO}^\circ} &= \delta^{(q-x)/2}\sigma \text{ and} \\
h_j^{\mathrm{SO}^\circ} &= v^{-1}h_{j-1}^{\mathrm{SO}^\circ}v.
\end{aligned}
$$

Proof. We have

$$\begin{aligned}&\delta^a\sigma\\ &= \mathrm{diag}(\omega^{2a}, 1, \ldots, 1, \omega^{-2a}) \cdot \mathrm{diag}(\omega^x, 1, \ldots, 1, \omega^{-x})\\ &= \mathrm{diag}(\omega^{2a+x}, 1, \ldots, 1, \omega^{-(2a+x)}).\end{aligned}$$

We wish that $2a + x \equiv_{q-1} 1$. We choose $a \coloneqq \frac{q-x}{2}$ and get

$$2 \cdot \tfrac{q-x}{2} + x = q - x + x = q \equiv_{q-1} 1.$$

Clearly, we obtain $h_1^{\mathrm{SO}^\circ}$. Now, we use $\varphi_{\mathrm{SO}^\circ}$ and notice that

$$\varphi_{\mathrm{SO}^\circ}(v) = (1, 2, \ldots, m)(m+2, n, n-1, \ldots, m+3).$$

Hence, we showed that $v^{-1}h_{j-1}^{\mathrm{SO}^\circ}v$ induces the desired permutation on the diagonal entries. □

Since diagonal matrices of this type always have a 1 in the middle, $h_j^{\mathrm{SO}^\circ}$ for $j \in \{1, \ldots, m\}$ are already sufficient.

Theorem 5.35: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m + 1$ and $\omega \in \mathbb{F}_q$ a primitive element. Let $a = \mathrm{diag}(a_1, \ldots, a_n) \in \mathrm{SO}^\circ(n, q)$ be a diagonal matrix. Then a is a product of the $h_j^{\mathrm{SO}^\circ}$.

Proof. We know by Lemma 5.32 that the matrix a has the form

$$a = \mathrm{diag}(a_1, \ldots, a_n) = \mathrm{diag}(a_1, \ldots, a_m, 1, a_m^{-1}, \ldots, a_1^{-1}).$$

For $j \in \{1, \ldots, m\}$ we choose $k_j \in \{1, \ldots, q\}$ such that $\omega^{k_j} = a_j$. We show that

$$a = \prod_{j=1}^{m} (h_j^{\mathrm{SO}^\circ})^{k_j}.$$

We have

$$\begin{aligned}\prod_{j=1}^{m} (h_j^{\mathrm{SO}^\circ})^{k_j} &= (h_1^{\mathrm{SO}^\circ})^{k_1} \ldots (h_m^{\mathrm{SO}^\circ})^{k_m}\\ &= \mathrm{diag}(\omega, 1, \ldots, 1, \omega^{-1})^{k_1} \ldots \mathrm{diag}(1, \ldots, 1, \omega, 1, \omega^{-1}, 1, \ldots, 1)^{k_m}\\ &= \mathrm{diag}(\omega^{k_1}, 1, \ldots, 1, (\omega^{-1})^{k_1}) \ldots \mathrm{diag}(1, \ldots, 1, \omega^{k_m}, 1, (\omega^{-1})^{k_m}, 1, \ldots, 1)\\ &= \mathrm{diag}(a_1, 1, \ldots, 1, a_1^{-1}) \ldots \mathrm{diag}(1, \ldots, 1, a_m, 1, (a_m)^{-1}, 1, \ldots, 1)\\ &= \mathrm{diag}(a_1, \ldots, a_m, 1, a_m^{-1}, \ldots, a_1^{-1})\\ &= a.\end{aligned}$$

Hence, the claim follows. □

We give the algorithm DiagonalSOCircle in pseudo code. It is identical to the one of the plus type except for the running index of the for-loop and the matrices $h_j^{\mathrm{SO}^\circ}$.

Algorithm 8: DiagonalSOCircle

Input: $a \in \mathrm{SO}^\circ(n,q)$ a diagonal matrix.
Output: An MSLP S such that a is the output of the evaluation of S with the LGO standard generators.

```
function DiagonalSOCircle(a)
    Write q = 2^y x with x odd; s := (q-x)/2;
    Add instructions to MSLP S to generate (δ)^s σ;          // h_1^{SO°}.
    for i ∈ [1, ..., (n-1)/2] do
        a_i := DiscreteLogarithm(a_{i,i}, ω);
        Add instructions to S for (h_i^{SO°})^{a_i} and h_{i+1}^{SO°} = v^{-1} h_i^{SO°} v;
    return S;
```

An example for this algorithm is given in Example 4.46 by inserting a 1 in the middle.

Finally, we calculate the maximum length and the number of memory slots of an MSLP, which computes a diagonal matrix from the orthogonal group of circle type.

Theorem 5.36: Let $n, f, m \in \mathbb{N}$, $q = p^f$ a prime power, $n = 2m+1$ and $\omega \in \mathbb{F}_q$ a primitive element. Let $\lambda = m + 2\log_2(q) + n - 2 + (n-1)\log_2(q)$ and $b = 16$. Let $h \in \mathrm{SO}^\circ(n,q)$ be a diagonal matrix given in the form of $h = \mathrm{diag}(\omega^{k_1}, \ldots, \omega^{k_n})$. There exists a b-MSLP S of length at most λ such that if S is evaluated with memory containing the set $X = \{s, s^{-1}, t, t^{-1}, \delta, \delta^{-1}, v, v^{-1}, u, u^{-1}, \sigma, \sigma^{-1}\}$, then S outputs final memory containing h.

Proof. The MSLP writes h as described in Theorem 5.35. The $h_j^{\mathrm{SO}^\circ}$ for $j \in \{1, \ldots, m\}$ are computed in terms of the LGO standard generators as written in Lemma 5.34. The construction of the $h_j^{\mathrm{SO}^\circ}$ via Lemma 5.34 costs $2\log_2(q) + 2(m-1) + 1 = 2\log_2(q) + n - 2$ MSLP instructions:

- $h_1^{\mathrm{SO}^\circ}$ needs at most $2\log_2(\frac{q-x}{2}) + 1 \leq 2\log_2(q) + 1$ multiplications.
- $h_{j+1}^{\mathrm{SO}^\circ}$ arises from $h_j^{\mathrm{SO}^\circ}$ by conjugating with v which needs 2 multiplications.

Raising each $h_j^{\mathrm{SO}^\circ}$ up to its k_jth power can be done by repeated squaring [NPP, Section 2.2(ii)] which costs at most

$$2\log_2(k_j) \leq 2\log_2(q)$$

MSLP instructions. Since we have to do this m times, the computation of all $(h_j^{\mathrm{SO}^\circ})^{k_j}$ needs at most

$$m(2\log_2(q)) = (n-1)\log_2(q)$$

instructions. Moreover, we have to multiply the $(h_j^{\mathrm{SO}^\circ})^{k_j}$ together which requires

$$m - 1 < m = \tfrac{n-1}{2}$$

instructions. Overall, this results in a maximum total length of the MSLP of

$$m + 2\log_2(q) + n - 2 + (n-1)\log_2(q).$$

The memory quota for this MSLP is as follows:

- 12 memory slots are needed to store the input X,
- 1 memory slot is needed to multiply the $(h_j^{\mathrm{SO}^\circ})^{k_j}$ together,
- 1 memory slot is needed to store $h_j^{\mathrm{SO}^\circ}$ because the recursion for $h_j^{\mathrm{SO}^\circ}$ refers back to $h_{j-1}^{\mathrm{SO}^\circ}$ and v, hence, each $h_j^{\mathrm{SO}^\circ}$ can be written into the slot from which it is computed and
- 2 memory slots are needed for computing $(h_j^{\mathrm{SO}^\circ})^{k_j}$ via repeated squaring [NPP, Section 2.2(ii)].

Thus, we need

$$12 + 2 + 1 + 1 = 16$$

memory slots in total. □

Remark 5.37: Just like at the end of Chapter 4, we would like to compare this group to the other classical groups which reveals lots of similarities to the unitary group of odd dimension. For example, the permutation group that is generated by the images of the monomial matrices is equal. Nevertheless, this requires a separate implementation since the LGO standard generators do not match. Another commonality is that the entries in the $(m+1)$th row and $(m+1)$th column take a special role and must be considered separately. However, we can also use transformations of the form $T_{i,j}$ with $i + j = n + 1$ in the unitary group, which are also required in the Bruhat decomposition algorithm in contrast to Algorithm 6. As in Remark 4.48, the reader is refered to the authors Bachelor's thesis *Bruhat Decomposition in Unitary and Symplectic Groups over Finite Fields* [DR] for further details.

References

[NPP] A. C. Niemeyer, T. Popiel and C. E. Praeger, *Straight-line programs with memory and matrix Bruhat decomposition*. CoRR abs/1305.5617 (May 2017)

[LGO] C. R. Leedham-Green and E. A. O'Brien, *Constructive recognition of classical groups in odd characteristic*. Journal of Algebra 322 (2009), 833–881

[Taylor] Donald E. Taylor, *The geometry of the classical groups*, volume 9 of Sigma Series in Pure Mathematics (Heldermann Verlag, Berlin, 1992)

[BHGO] Henrik Bäärnhielm, Derek Holt, C.R. Leedham-Green, and E.A. O'Brien, *A practical model for computation with matrix groups*. J. Symbolic Comput. 68 (2015), part 1, 27–60

[DR] Daniel Rademacher, *Bruhat Decomposition in Unitary and Symplectic Groups over Finite Fields*. Bachelor's thesis (2019)

[MA] Aschbacher, Michael, *On finite groups of component type*. Illinois J. Math. 19 (1975), no. 1, 87–115. doi:10.1215/ijm/1256050927.

[GAP] The GAP Group, *GAP – Groups, Algorithms, Programming - a System for Computational Discrete Algebra, Version 4.11.0*. GAP (September 2020)

[Magma] W. Bosma, J. Cannon and C. Playoust, *The Magma algebra system. I. The user language*. J. Symbolic Comput. 24 (1997) 235–265.

Acknowledgements

My thanks go to the many people who have supported me during my Master's thesis in numerous ways. I thank Prof. Dr. Alice Niemeyer for the opportunity to write this thesis about such an interesting topic with her and the great support. I am grateful to Dr. Wilhelm Plesken for also dealing with my Master's thesis and for the wonderful collaboration. I also thank M. Sc. Dominik Bernhardt who is a patient, motivated and very supportive supervisor. In addition, I am grateful to the entire Chair of Algebra and Representation theory for the very pleasant and fun atmosphere.

Moreover, I would also like to thank all of the people who accompanied me in my mathematics studies. Therefore, I like to express my special thanks to Anna Sucker and Lucas Wollenhaupt who spent this time with me and proofread this thesis.
Many thanks also to Tobias Rademacher, Simon Berger, Sven Argo, Niko Molke, Paul Geuchen and Hannah Hamacher for reading this thesis. I am also grateful to my parents, Gudrun and Helge Rademacher, who made my studies possible.

Furthermore, without naming individuals, I thank my private environment for the support.

Appendix

An MSLP for Example 4.46:

$$
\begin{aligned}
&[[1,1],[2,1],[3,1],[4,1],[5,1],[6,1],[7,1],[8,1],[9,1],[1,-1],\\
&\ [2,-1],[3,-1],[4,-1],[5,-1],[6,-1],[7,-1],[8,-1],[9,-1],[1,0],\\
&\ [1,0],[[1,0],21],[[1,0],22],[[1,0],23],[[5,0],19],[[5,1],20],\\
&\ [[19,1,20,1],19],[[20,2],20],[[19,1,20,1],19],[[20,2],20],\\
&\ [[19,1,20,1],19],[[20,2],20],[[20,2],20],[[19,1,20,1],19],\\
&\ [[20,2],20],[[19,1,9,1],22],[[6,0],19],[[6,1],20],\\
&\ [[19,1,20,1],19],[[20,2],20],[[19,1,20,1],19],[[20,2],20],\\
&\ [[19,1,20,1],19],[[20,2],20],[[20,2],20],[[19,1,20,1],19],\\
&\ [[20,2],20],[[19,1,22,1],22],[[8,1],21],[[22,0],19],[[22,1],20],\\
&\ [[20,2],20],[[19,1,20,1],19],[[20,2],20],[[20,2],20],\\
&\ [[19,1,20,1],19],[[20,2],20],[[23,1,19,1],23],\\
&\ [[21,-1,22,1,21,1],22],[[22,0],19],[[22,1],20],[[19,1,20,1],19],\\
&\ [[20,2],20],[[20,2],20],[[19,1,20,1],19],[[20,2],20],[[20,2],20],\\
&\ [[20,2],20],[[19,1,20,1],19],[[20,2],20],[[23,1,19,1],23],\\
&\ [[21,-1,22,1,21,1],22],[[22,0],19],[[22,1],20],[[20,2],20],\\
&\ [[19,1,20,1],19],[[20,2],20],[[19,1,20,1],19],[[20,2],20],\\
&\ [[19,1,20,1],19],[[20,2],20],[[19,1,20,1],19],[[20,2],20],\\
&\ [[23,1,19,1],23],[[21,-1,22,1,21,1],22],[23,1]]
\end{aligned}
$$