Bruhat Decomposition of Classical Groups

Daniel Rademacher

October 18, 2022

Algebra and Representation Theory
RWTH AACHEN UNIVERSITY

Classical Groups

Special Linear Group

Let $d \in \mathbb{N}$, $q = p^f$ a prime power and define

$$\mathrm{SL}(d, q) := \{a \in \mathrm{GL}(d, q) \mid \det(a) = 1\}.$$

Sp, SU, SO, Omega

Let $d \in \mathbb{N}$, $q = p^f$ a prime power and define

Sp, SU, SO, Omega

Let $d \in \mathbb{N}$, $q = p^f$ a prime power and define

1) $\mathrm{Sp}(d, q) := \{a \in \mathrm{SL}(d, q) \mid a b_{Sp} a^{Tr} = b_{Sp}\}$ for d even and $b_{Sp} \in \mathrm{GL}(d, q)$.

Sp, SU, SO, Omega

Let $d \in \mathbb{N}$, $q = p^f$ a prime power and define

1) $\mathrm{Sp}(d, q) := \{a \in \mathrm{SL}(d, q) \mid a b_{Sp} a^{Tr} = b_{Sp}\}$ for d even and $b_{Sp} \in \mathrm{GL}(d, q)$.
2) $\mathrm{SU}(d, q) := \{a \in \mathrm{SL}(d, q^2) \mid a b_{SU} \overline{a}^{Tr} = b_{SU}\}$ for $b_{SU} \in \mathrm{GL}(d, q)$.

Sp, SU, SO, Omega

Let $d \in \mathbb{N}$, $q = p^f$ a prime power and define

1) $\mathrm{Sp}(d,q) := \{a \in \mathrm{SL}(d,q) \mid a b_{Sp} a^{Tr} = b_{Sp}\}$ for d even and $b_{Sp} \in \mathrm{GL}(d,q)$.
2) $\mathrm{SU}(d,q) := \{a \in \mathrm{SL}(d,q^2) \mid a b_{SU} \bar{a}^{Tr} = b_{SU}\}$ for $b_{SU} \in \mathrm{GL}(d,q)$.
3) $\mathrm{SO}^+(d,q) := \{a \in \mathrm{SL}(d,q) \mid a b_{SO^+} a^{Tr} = b_{SO^+}\}$ for d even, $b_{SO^+} \in \mathrm{GL}(d,q)$ and $p > 2$. Set $\Omega^+(d,q) = \mathrm{SO}^+(d,q)'$.
4) $\mathrm{SO}^-(d,q) := \{a \in \mathrm{SL}(d,q) \mid a b_{SO^-} a^{Tr} = b_{SO^-}\}$ for d even, $b_{SO^-} \in \mathrm{GL}(d,q)$ and $p > 2$. Set $\Omega^-(d,q) = \mathrm{SO}^-(d,q)'$.
5) $\mathrm{SO}^\circ(d,q) := \{a \in \mathrm{SL}(d,q) \mid a b_{SO^\circ} a^{Tr} = b_{SO^\circ}\}$ for d odd, $b_{SO^\circ} \in \mathrm{GL}(d,q)$ and $p > 2$. Set $\Omega^\circ(d,q) = \mathrm{SO}^\circ(d,q)'$.

Bruhat Decomposition

Definition

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 holds:

Definition

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 holds:

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

(B, N) Pair

Definition

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 holds:

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

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

Definition

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 holds:

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

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

3.) The group $W = N/H$ is generated by a set S of elements w_i of order 2 for $i \in I$ and $I \neq \emptyset$ an index set,

(B, N) Pair

Definition

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 holds:

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

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

3.) The group $W = N/H$ is generated by a set S of elements w_i of order 2 for $i \in I$ and $I \neq \emptyset$ an index set,

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.) $n_i B n_i \neq B$.

Definition

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

$$G = BWB.$$

Definition

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

$$G = BWB.$$

Theorem

Let $G \leq \mathrm{GL}(d, q)$ be a classical group in its natural representation, $B \leq G$ the subgroup of lower triangular matrices and $W \leq G$ the subgroup of monomial matrices. Then we can decompose G as

$$G = BWB.$$

Idea of Algorithm

Let $G \leq \mathrm{GL}(d, q)$ be a classical group in its natural representation.
Then we want to compute for $a \in G$

Let $G \leq \mathrm{GL}(d, q)$ be a classical group in its natural representation. Then we want to compute for $a \in G$

1) the Bruhat Decomposition and

Let $G \leq \mathrm{GL}(d, q)$ be a classical group in its natural representation. Then we want to compute for $a \in G$

1) the Bruhat Decomposition and
2) an expression of a as a word in the Leedham-Green O'Brien (LGO) standard generators.

Write a matrix $a \in G$ in terms of the LGO standard generators.
We proceed as follows:

Write a matrix $a \in G$ in terms of the LGO standard generators. We proceed as follows:

1.) Constructively find two unitriangular matrices u_1, u_2 such that $u_1 a u_2 = w$ where w is a monomial matrix. (Bruhat decomposition)

Write a matrix $a \in G$ in terms of the LGO standard generators.
We proceed as follows:

1.) Constructively find two unitriangular matrices u_1, u_2 such that $u_1 a u_2 = w$ where w is a monomial matrix. (Bruhat decomposition)
2.) Constructively find a monomial matrix w'' such that $(w'')^{-1} w = h$ where h is a diagonal matrix.

Write a matrix $a \in G$ in terms of the LGO standard generators. We proceed as follows:

1.) Constructively find two unitriangular matrices u_1, u_2 such that $u_1 a u_2 = w$ where w is a monomial matrix. (Bruhat decomposition)
2.) Constructively find a monomial matrix w'' such that $(w'')^{-1} w = h$ where h is a diagonal matrix.
3.) Write h as a word in terms of the LGO standard generators.

Write a matrix $a \in G$ in terms of the LGO standard generators.
We proceed as follows:

1.) Constructively find two unitriangular matrices u_1, u_2 such that $u_1 a u_2 = w$ where w is a monomial matrix. (Bruhat decomposition)
2.) Constructively find a monomial matrix w'' such that $(w'')^{-1} w = h$ where h is a diagonal matrix.
3.) Write h as a word in terms of the LGO standard generators.

Then we have

$$u_1^{-1} w'' h u_2^{-1} = a.$$

First Step

Constructively find two unitriangular matrices u_1, u_2 such that $u_1 a u_2 = w$ where w is a monomial matrix. (Bruhat decomposition)

Strategy for the first step

First Step

Constructively find two unitriangular matrices u_1, u_2 such that $u_1 a u_2 = w$ where w is a monomial matrix. (Bruhat decomposition)

The main idea for the first step is to use some kind of Gaussian algorithm. For this, new matrices for elementary row and column operations have to be constructed for each classical group.

$E_{i,j}(\alpha)$

We define $E_{i,j}(\alpha) \in \mathbb{F}^{d \times d}$ for $i, j \in \{1, \ldots, d\}$ with $i \neq j$ and $\alpha \in \mathbb{F}^*$ as follows:

$$E_{i,j}(\alpha) = I_d + \begin{array}{cc} \begin{array}{ccccc} & & j\text{-th} & & \end{array} & \\ \begin{pmatrix} 0 & \ldots & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ 0 & \ldots & \alpha & \ldots & 0 \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & \ldots & 0 \end{pmatrix} & i\text{-th} \end{array}$$

The matrices $E_{i,j}(\alpha)$ for $i, j \in \{1, \ldots, d\}$ with $i \neq j$ and $\alpha \in \mathbb{F}^*$ are transvections.

Transvection

$$E_{3,1}(3) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \in \mathrm{SL}(6,7)$$

$$S_{3,1}(3) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -3 & 0 & 1 \end{pmatrix} \in \mathrm{Sp}(6,7)$$

We demonstrate the algorithm for:

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

We demonstrate the algorithm for:

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

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

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

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

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

Example of the Algorithm

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

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

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

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

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

By multiplying the matrices we obtain the Bruhat decomposition

$$w = S_{3,2}(3)S_{4,1}(2)S_{3,1}(3)S_{2,1}(4)aS_{4,3}(1)S_{4,2}(3)S_{4,1}(3).$$

First Step

Constructively find two unitriangular matrices u_1, u_2 such that $u_1 a u_2 = w$ where w is a monomial matrix. (Bruhat decomposition)

Strategy for the second step

First Step

Constructively find two unitriangular matrices u_1, u_2 such that $u_1 a u_2 = w$ where w is a monomial matrix. (Bruhat decomposition)

Second Step

Given a monomial matrix $w \in G$. Constructively find a monomial matrix $w'' \in G$ such that $(w'')^{-1} w = h$ where $h \in G$ is a diagonal matrix.

Let $N \leq G$ be the subgroup of monomial matrices.
Then we can form a homomorphism

$$\Phi \colon N \to S_d$$

such that for the standard basis $e_1, \ldots, e_d$ of $\mathbb{F}_q^d$, $w \in N$ permutes the spans $\langle e_1 \rangle, \ldots, \langle e_d \rangle$ in the same way as $\Phi(w)$.

Strategy for the second step

Let $S' \subseteq S$ be the subset of monomial matrices of the LGO standard generators. Then

$$\Phi(s) \in S_d$$

for $s \in S'$ and $\Phi(S') \leq S_d$.
For a monomial matrix $w \in G$ we can express $\Phi(w) = w'$ as a word in the generators $\Phi(S')$.

Evaluating the word of the permutations with the matrices S' yields an element $w'' \in G$ with $\Phi(w'') = \Phi(w)$. Then

$$h := (w'')^{-1} \cdot w$$

is a diagonal matrix.

Second Step

Given a monomial matrix $w \in G$. Constructively find a monomial matrix $w'' \in G$ such that $(w'')^{-1}w = h$ where $h \in G$ is a diagonal matrix.

Second Step

Given a monomial matrix $w \in G$. Constructively find a monomial matrix $w'' \in G$ such that $(w'')^{-1}w = h$ where $h \in G$ is a diagonal matrix.

Third Step

Now we have a diagonal matrix $h \in G$. It remains to write h as a word in terms of the LGO standard generators. Then we have

$$w = w'' \cdot h$$

where $u_1 a u_2 = w$.

GAP Session

Example

```
gap> g := PseudoRandom(SL(10,5));;
```

Example

```
gap> g := PseudoRandom(SL(10,5));;
gap> rL := BruhatDecompositionSL(LGOStandardGensSL(10,5), g);
#I  returns an SLP to generate u1, u2, p_sign, diag
#I  Memory Usage is: 22 memory slots in UnipotentDecomposition
#I  and additional:  7 memory slots in PermSLP
#I  and additional:  3 memory slots in DiagonalDecomposition
#I  The Total Memory Usage is: 32 memory slots

[ <straight line program>,
[ < mutable compressed matrix 10x10 over GF(5) >,
  < mutable compressed matrix 10x10 over GF(5) >,
  < mutable compressed matrix 10x10 over GF(5) >,
  < mutable compressed matrix 10x10 over GF(5) > ] ]
```

Example

```
gap> Display(g);
 . 1 . 3 3 1 . 3 1 3
 . 1 2 4 . 3 . 4 3 2
 1 4 3 . 3 1 3 . 1 4
 2 . 4 1 3 2 . 4 3 .
 3 1 3 2 1 4 2 3 1 4
 3 2 2 4 . 1 . 2 1 4
 4 3 . 4 1 . 2 1 2 1
 1 . 3 . . 4 3 . 4 1
 3 . 2 2 3 3 . 1 2 1
 . 3 . 4 1 2 4 2 2 .
```

```
gap> Display(L[2,1]);
 1 . . . . . . . . .
 1 1 . . . . . . . .
 . 3 1 . . . . . . .
 3 3 . 1 . . . . . .
 . 3 . . 1 . . . . .
 2 3 3 4 3 1 . . . .
 3 2 . . 4 . 1 . . .
 1 1 4 . 3 1 2 1 . .
 . 1 3 3 3 4 1 1 1 .
 . 1 3 2 . 1 4 3 . 1
```

```
gap> Display(L[2,2]);
 1 . . . . . . . . .
 1 1 . . . . . . . .
 2 2 1 . . . . . . .
 3 2 4 1 . . . . . .
 2 2 2 4 1 . . . . .
 3 4 2 . 2 1 . . . .
 1 3 1 1 4 1 1 . . .
 2 3 2 4 . 1 1 1 . .
 4 3 3 2 1 1 2 2 1 .
 2 2 2 2 3 . . . 3 1
```

```
gap> Display(L[2,1]*g*L[2,2]);
 . . . . . . . . . 3
 . . . . . . . . 4 .
 . . . . . . . 2 . .
 . . . . . 4 . . . .
 . . . . . . 2 . . .
 . . . 2 . . . . . .
 . . . . 4 . . . . .
 . . 3 . . . . . . .
 . 2 . . . . . . . .
 4 . . . . . . . . .
```

Example

```
gap> Display(L[2,3]);
 . . . . . . . . . 1
 . . . . . . . . 4 .
 . . . . . . . 1 . .
 . . . . . 1 . . . .
 . . . . . . 1 . . .
 . . . 4 . . . . . .
 . . . . 4 . . . . .
 . . 4 . . . . . . .
 . 1 . . . . . . . .
 4 . . . . . . . . .
```

```
gap> Display((L[2,1]*g*L[2,2])^(-1)*L[2,3]);
 1 . . . . . . . . .
 . 3 . . . . . . . .
 . . 3 . . . . . . .
 . . . 2 . . . . . .
 . . . . 1 . . . . .
 . . . . . 4 . . . .
 . . . . . . 3 . . .
 . . . . . . . 3 . .
 . . . . . . . . 1 .
 . . . . . . . . . 2
```

Example

```
gap> Display(L[2,4]);
 1 . . . . . . . . .
 . 2 . . . . . . . .
 . . 2 . . . . . . .
 . . . 3 . . . . . .
 . . . . 1 . . . . .
 . . . . . 4 . . . .
 . . . . . . 2 . . .
 . . . . . . . 2 . .
 . . . . . . . . 1 .
 . . . . . . . . . 3
```

Example

```
gap> Display((L[2,1]*g*L[2,2])^(-1)*L[2,3]*L[2,4]);
 1 . . . . . . . . .
 . 1 . . . . . . . .
 . . 1 . . . . . . .
 . . . 1 . . . . . .
 . . . . 1 . . . . .
 . . . . . 1 . . . .
 . . . . . . 1 . . .
 . . . . . . . 1 . .
 . . . . . . . . 1 .
 . . . . . . . . . 1
```

Example

```
gap> r:=ResultOfStraightLineProgram(L[1], StGens(10,5));;
```

Example

```
gap> r:=ResultOfStraightLineProgram(L[1], StGens(10,5));;
gap> Display(r[1]^(-1)*r[3]*r[4]*r[2]^(-1));
 . 1 . 3 3 1 . 3 1 3
 . 1 2 4 . 3 . 4 3 2
 1 4 3 . 3 1 3 . 1 4
 2 . 4 1 3 2 . 4 3 .
 3 1 3 2 1 4 2 3 1 4
 3 2 2 4 . 1 . 2 1 4
 4 3 . 4 1 . 2 1 2 1
 1 . 3 . . 4 3 . 4 1
 3 . 2 2 3 3 . 1 2 1
 . 3 . 4 1 2 4 2 2 .
```

```
gap> r:=ResultOfStraightLineProgram(L[1], StGens(10,5));;
gap> Display(r[1]^(-1)*r[3]*r[4]*r[2]^(-1));
 . 1 . 3 3 1 . 3 1 3
 . 1 2 4 . 3 . 4 3 2
 1 4 3 . 3 1 3 . 1 4
 2 . 4 1 3 2 . 4 3 .
 3 1 3 2 1 4 2 3 1 4
 3 2 2 4 . 1 . 2 1 4
 4 3 . 4 1 . 2 1 2 1
 1 . 3 . . 4 3 . 4 1
 3 . 2 2 3 3 . 1 2 1
 . 3 . 4 1 2 4 2 2 .
gap> Display(r[1]^(-1)*r[3]*r[4]*r[2]^(-1) = g);
true
```

Thank you for your attention!