Finding normal subgroups of even order

Max Neunhöffer

University of St Andrews

Columbus, 8.10.2009

The problem

Problem

Let $1 < N \lhd G = \langle g_1, \ldots, g_k \rangle$ be a finite group and N be a normal subgroup.

The problem

Problem

Let $1 < N \lhd G = \langle g_1, \ldots, g_k \rangle$ be a finite group and N be a normal subgroup.
Produce a non-trivial element of N as a word in the g_i

The problem

Problem

Let $1 < N \lhd G = \langle g_1, \ldots, g_k \rangle$ be a finite group and N be a normal subgroup.
Produce a non-trivial element of N as a word in the g_i

- Assume no more knowledge about G or N.

The problem

Problem

Let $1 < N \triangleleft G = \langle g_1, \ldots, g_k \rangle$ be a finite group and N be a normal subgroup.
Produce a non-trivial element of N as a word in the g_i

- Assume no more knowledge about G or N.
- I shall tell you soon why we want to do this.

The problem

Problem

Let $1 < N \lhd G = \langle g_1, \ldots, g_k \rangle$ be a finite group and N be a normal subgroup.
Produce a non-trivial element of N as a word in the g_i

- Assume no more knowledge about G or N.
- I shall tell you soon why we want to do this.
- We are looking for a randomised algorithm.

The problem

Problem

Let $1 < N \lhd G = \langle g_1, \ldots, g_k \rangle$ be a finite group and N be a normal subgroup.
Produce a non-trivial element of N as a word in the g_i

- Assume no more knowledge about G or N.
- I shall tell you soon why we want to do this.
- We are looking for a randomised algorithm.
- Assume we can generate uniformly distributed random elements in G.

The problem

Problem

Let $1 < N \lhd G = \langle g_1, \ldots, g_k \rangle$ be a finite group and N be a normal subgroup.
Produce a non-trivial element of N as a word in the g_i with "high probability".

- Assume no more knowledge about G or N.
- I shall tell you soon why we want to do this.
- We are looking for a randomised algorithm.
- Assume we can generate uniformly distributed random elements in G.

The problem

Problem

Let $1 < N \lhd G = \langle g_1, \ldots, g_k \rangle$ be a finite group and N be a normal subgroup.
Produce a non-trivial element of N as a word in the g_i with "high probability".

- Assume no more knowledge about G or N.
- I shall tell you soon why we want to do this.
- We are looking for a randomised algorithm.
- Assume we can generate uniformly distributed random elements in G.
- "High probability" means for the moment "higher than $1/[G : N]$".

Matrix groups . . .

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n\times n} \mid M \text{ invertible}\}$$

Matrix groups ...

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Matrix groups ...

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

Matrix groups ...

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n} (q^i - 1)$

Matrix groups . . .

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n}(q^i - 1)$

What do we want to determine about G?

- The group order $|G|$

Matrix groups . . .

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n\times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n}(q^i - 1)$

What do we want to determine about G?

- The group order $|G|$
- Membership test: Is $M \in \mathrm{GL}_n(\mathbb{F}_q)$ in G?

Matrix groups ...

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n} (q^i - 1)$

What do we want to determine about G?

- The group order $|G|$
- Membership test: Is $M \in \mathrm{GL}_n(\mathbb{F}_q)$ in G?
- Homomorphisms $\varphi : G \to H$?

Matrix groups ...

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n\times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n} (q^i - 1)$

What do we want to determine about G?

- The group order $|G|$
- Membership test: Is $M \in \mathrm{GL}_n(\mathbb{F}_q)$ in G?
- Homomorphisms $\varphi : G \to H$?
- Kernels of homomorphisms? Is G simple?

Matrix groups ...

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n \times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n}(q^i - 1)$

What do we want to determine about G?

- The group order $|G|$
- Membership test: Is $M \in \mathrm{GL}_n(\mathbb{F}_q)$ in G?
- Homomorphisms $\varphi : G \to H$?
- Kernels of homomorphisms? Is G simple?
- Comparison with known groups

Matrix groups ...

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n\times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n}(q^i - 1)$

What do we want to determine about G?

- The group order $|G|$
- Membership test: Is $M \in \mathrm{GL}_n(\mathbb{F}_q)$ in G?
- Homomorphisms $\varphi : G \to H$?
- Kernels of homomorphisms? Is G simple?
- Comparison with known groups
- (Maximal) subgroups?

Matrix groups . . .

Let $\mathbb{F}_q$ be the field with q elements and

$$\mathrm{GL}_n(\mathbb{F}_q) := \{M \in \mathbb{F}_q^{n\times n} \mid M \text{ invertible}\}$$

Given: $M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q)$

Then the M_i generate a group $G \leq \mathrm{GL}_n(\mathbb{F}_q)$.

It is finite, we have $|\mathrm{GL}_n(\mathbb{F}_q)| = q^{n(n-1)/2} \prod_{i=1}^{n}(q^i - 1)$

What do we want to determine about G?

- The group order $|G|$
- Membership test: Is $M \in \mathrm{GL}_n(\mathbb{F}_q)$ in G?
- Homomorphisms $\varphi : G \to H$?
- Kernels of homomorphisms? Is G simple?
- Comparison with known groups
- (Maximal) subgroups?
- . . .

Constructive recognition

Problem

Let $\mathbb{F}_q$ be the field with q elements and

$$M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q).$$

Find for $G := \langle M_1, \ldots, M_k \rangle$:

- The group order $|G|$ and
- an algorithm that, given $M \in \mathrm{GL}_n(\mathbb{F}_q)$,
    - decides, whether or not $M \in G$, and,
    - if so, expresses M as word in the M_i.

Constructive recognition

Problem

Let $\mathbb{F}_q$ be the field with q elements and

$$M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q).$$

Find for $G := \langle M_1, \ldots, M_k \rangle$:

- The group order $|G|$ and
- an algorithm that, given $M \in \mathrm{GL}_n(\mathbb{F}_q)$,
  - decides, whether or not $M \in G$, and,
  - if so, expresses M as word in the M_i.
- The runtime should be bounded from above by a polynomial in n, k and $\log q$.

Constructive recognition

Problem

Let $\mathbb{F}_q$ be the field with q elements and

$$M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q).$$

Find for $G := \langle M_1, \ldots, M_k \rangle$:

- The group order $|G|$ and
- an algorithm that, given $M \in \mathrm{GL}_n(\mathbb{F}_q)$,
  - decides, whether or not $M \in G$, and,
  - if so, expresses M as word in the M_i.
- The runtime should be bounded from above by a polynomial in n, k and $\log q$.
- A Monte Carlo Algorithm is enough.

Constructive recognition

Problem

Let $\mathbb{F}_q$ be the field with q elements and

$$M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q).$$

Find for $G := \langle M_1, \ldots, M_k \rangle$:

- The group order $|G|$ and
- an algorithm that, given $M \in \mathrm{GL}_n(\mathbb{F}_q)$,
  - decides, whether or not $M \in G$, and,
  - if so, expresses M as word in the M_i.
- The runtime should be bounded from above by a polynomial in n, k and $\log q$.
- A Monte Carlo Algorithm is enough. (Verification!)

Constructive recognition

Problem

Let $\mathbb{F}_q$ be the field with q elements and

$$M_1, \ldots, M_k \in \mathrm{GL}_n(\mathbb{F}_q).$$

Find for $G := \langle M_1, \ldots, M_k \rangle$:

- The group order $|G|$ and
- an algorithm that, given $M \in \mathrm{GL}_n(\mathbb{F}_q)$,
  - decides, whether or not $M \in G$, and,
  - if so, expresses M as word in the M_i.
- The runtime should be bounded from above by a polynomial in n, k and $\log q$.
- A Monte Carlo Algorithm is enough. (Verification!)

If this problem is solved, we call

$\langle M_1, \ldots, M_k \rangle$ recognised constructively.

What is a reduction?

Let $G := \langle M_1, \ldots, M_k \rangle \leq \mathrm{GL}_n(\mathbb{F}_q)$.

What is a reduction?

Let $G := \langle M_1, \ldots, M_k \rangle \leq \mathrm{GL}_n(\mathbb{F}_q)$.

A reduction is a group homomorphism

$$\begin{array}{rccc} \varphi : & G & \to & H \\ & M_i & \mapsto & P_i \end{array} \qquad \text{for all } i$$

with the following properties:

What is a reduction?

Let $G := \langle M_1, \ldots, M_k \rangle \leq \mathrm{GL}_n(\mathbb{F}_q)$.

A reduction is a group homomorphism

$$\begin{array}{rccc} \varphi : & G & \to & H \\ & M_i & \mapsto & P_i \qquad \text{for all } i \end{array}$$

with the following properties:

- $\varphi(M)$ is explicitly computable for all $M \in G$

What is a reduction?

Let $G := \langle M_1, \ldots, M_k \rangle \le \mathrm{GL}_n(\mathbb{F}_q)$.

A reduction is a group homomorphism

$$\begin{array}{rcccl} \varphi & : & G & \to & H \\ & & M_i & \mapsto & P_i \qquad \text{for all } i \end{array}$$

with the following properties:

- $\varphi(M)$ is explicitly computable for all $M \in G$
- φ is surjective: $H = \langle P_1, \ldots, P_k \rangle$

What is a reduction?

Let $G := \langle M_1, \ldots, M_k \rangle \leq \mathrm{GL}_n(\mathbb{F}_q)$.

A reduction is a group homomorphism

$$\begin{array}{rcccl} \varphi & : & G & \to & H \\ & & M_i & \mapsto & P_i \qquad \text{for all } i \end{array}$$

with the following properties:

- $\varphi(M)$ is explicitly computable for all $M \in G$
- φ is surjective: $H = \langle P_1, \ldots, P_k \rangle$
- H is in some sense "smaller"
- or at least "easier to recognise constructively"

What is a reduction?

Let $G := \langle M_1, \ldots, M_k \rangle \leq \mathrm{GL}_n(\mathbb{F}_q)$.

A reduction is a group homomorphism

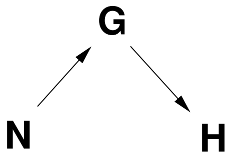
$$\begin{array}{rccc} \varphi : & G & \to & H \\ & M_i & \mapsto & P_i \end{array} \qquad \text{for all } i$$

with the following properties:

- $\varphi(M)$ is explicitly computable for all $M \in G$
- φ is surjective: $H = \langle P_1, \ldots, P_k \rangle$
- H is in some sense "smaller"
- or at least "easier to recognise constructively"
- e.g. $H \leq S_m$ or $H \leq \mathrm{GL}_{n'}(\mathbb{F}_{q'})$ with $n' \log q' < n \log q$

Recursion: composition trees

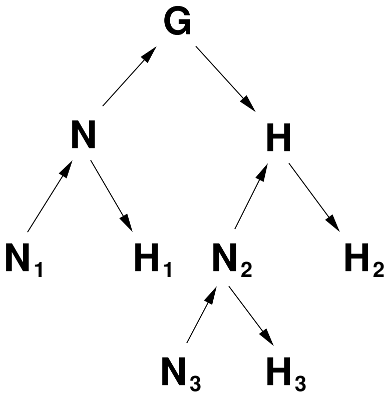
We get a tree:



Up arrows: inclusions
Down arrows: homomorphisms

Recursion: composition trees

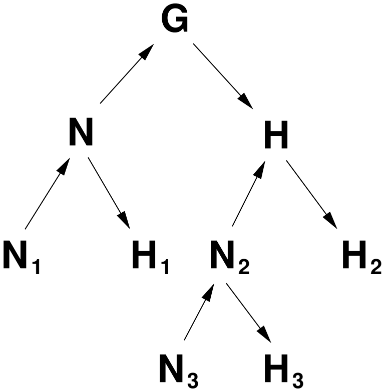
We get a tree:



Up arrows: inclusions
Down arrows: homomorphisms

Recursion: composition trees

We get a tree:



Up arrows: inclusions
Down arrows: homomorphisms

Old idea, substantial improvements are still being made

Reduction in the imprimitive case

One case, in which we want to find a reduction, is:

Reduction in the imprimitive case

One case, in which we want to find a reduction, is:

Situation

Let $G \leq \mathrm{GL}_n(\mathbb{F}_q)$ acting linearly on $V := \mathbb{F}_q^{1\times n}$, such that V is irreducible.

Reduction in the imprimitive case

One case, in which we want to find a reduction, is:

Situation

Let $G \leq \mathrm{GL}_n(\mathbb{F}_q)$ acting linearly on $V := \mathbb{F}_q^{1\times n}$, such that V is irreducible. Assume there is N with $Z(G) < N \triangleleft G$ such that

$$V|_N = W_1 \oplus W_2 \oplus \cdots \oplus W_k,$$

all W_i are invariant under N, and

Reduction in the imprimitive case

One case, in which we want to find a reduction, is:

Situation

Let $G \leq \mathrm{GL}_n(\mathbb{F}_q)$ acting linearly on $V := \mathbb{F}_q^{1\times n}$, such that V is irreducible. Assume there is N with $Z(G) < N \lhd G$ such that

$$V|_N = W_1 \oplus W_2 \oplus \cdots \oplus W_k,$$

all W_i are invariant under N, and G permutes the W_i transitively. Then there is a reduction $\varphi : G \to S_k$.

Reduction in the imprimitive case

One case, in which we want to find a reduction, is:

Situation

Let $G \leq \mathrm{GL}_n(\mathbb{F}_q)$ acting linearly on $V := \mathbb{F}_q^{1\times n}$, such that V is irreducible. Assume there is N with $Z(G) < N \lhd G$ such that

$$V|_N = W_1 \oplus W_2 \oplus \cdots \oplus W_k,$$

all W_i are invariant under N, and G permutes the W_i transitively. Then there is a reduction $\varphi : G \to S_k$.

We can compute the reduction once N is found.

Reduction in the imprimitive case

One case, in which we want to find a reduction, is:

Situation

Let $G \leq \mathrm{GL}_n(\mathbb{F}_q)$ acting linearly on $V := \mathbb{F}_q^{1\times n}$, such that V is irreducible. Assume there is N with $Z(G) < N \lhd G$ such that

$$V|_N = W_1 \oplus W_2 \oplus \cdots \oplus W_k,$$

all W_i are invariant under N, and G permutes the W_i transitively. Then there is a reduction $\varphi : G \to S_k$.

We can compute the reduction once N is found.

Since we can compute normal closures, our initial problem is exactly, what we need to do.

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ *with* $2 \mid |N|$.

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ *with* $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ *with* $x^2 = 1$.

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ *with* $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ *with* $x^2 = 1$.

Then, for $C := C_G(x)$*, we have:*

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ *with* $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ *with* $x^2 = 1$.

Then, for $C := C_G(x)$, *we have:*

- $1 < C \cap N \trianglelefteq C$ *and*
- $2 \mid |C \cap N|$.

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ *with* $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ *with* $x^2 = 1$.

Then, for $C := C_G(x)$*, we have:*

- $1 < C \cap N \trianglelefteq C$ *and*
- $2 \mid |C \cap N|$.

Proof: We have $xNx = N$ and $|N|$ is even.

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ with $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ with $x^2 = 1$.

Then, for $C := C_G(x)$, we have:

- $1 < C \cap N \trianglelefteq C$ *and*
- $2 \mid |C \cap N|$.

Proof: We have $xNx = N$ and $|N|$ is even. The orbits of $\langle x \rangle$ on N have lengths 1 and 2, so there must be an even number of orbits of length 1. ∎

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ with $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ with $x^2 = 1$.

Then, for $C := C_G(x)$, we have:

- $1 < C \cap N \trianglelefteq C$ *and*
- $2 \mid |C \cap N|$.

Proof: We have $xNx = N$ and $|N|$ is even. The orbits of $\langle x \rangle$ on N have lengths 1 and 2, so there must be an even number of orbits of length 1. ∎

In particular, $C \cap N$ contains an involution.

Finding even order normal subgroups

Theorem

Let $1 < N \trianglelefteq G$ with $2 \mid |N|$.

Let $1 \neq x \in G \setminus Z(G)$ with $x^2 = 1$.

Then, for $C := C_G(x)$, we have:

- $1 < C \cap N \trianglelefteq C$ *and*
- $2 \mid |C \cap N|$.

Proof: We have $xNx = N$ and $|N|$ is even. The orbits of $\langle x \rangle$ on N have lengths 1 and 2, so there must be an even number of orbits of length 1. ■

In particular, $C \cap N$ contains an involution.

That is, we can replace (N, G) with $(C \cap N, C)$ and use the statement again, provided we find another non-central involution.

Finding $N \lhd G$

We want to find an N with $1 < N \unlhd G$ and $2 \mid |N|$, or conclude that there is none.

Finding $N \lhd G$

We want to find an N with $1 < N \unlhd G$ and $2 \mid |N|$, or conclude that there is none.

Algorithm 1: InvolutionDescent

Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.

Finding $N \lhd G$

We want to find an N with $1 < N \unlhd G$ and $2 \mid |N|$, or conclude that there is none.

Algorithm 1: InvolutionDescent

Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.

Finding $N \lhd G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

Algorithm 1: InvolutionDescent

Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.

Finding $N \lhd G$

We want to find an N with $1 < N \unlhd G$ and $2 \mid |N|$, or conclude that there is none.

Algorithm 1: InvolutionDescent

Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.
4. Let D be the group generated by all central involutions we found.

Finding $N \lhd G$

We want to find an N with $1 < N \unlhd G$ and $2 \mid |N|$, or conclude that there is none.

Algorithm 1: InvolutionDescent

Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.
4. Let D be the group generated by all central involutions we found.
5. For all $1 \neq x \in D$: Test if $\langle x^G \rangle \neq G$.

Finding $N \triangleleft G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

Algorithm 1: InvolutionDescent

Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.
4. Let D be the group generated by all central involutions we found.
5. For all $1 \neq x \in D$: Test if $\langle x^G \rangle \neq G$.
6. If no normal closure is properly contained, conclude that G does not contain such an $|N|$ as assumed.

Finding $N \lhd G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

Algorithm 1: InvolutionDescent

Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.
4. Let D be the group generated by all central involutions we found.
5. For all $1 \neq x \in D$: Test if $\langle x^G \rangle \neq G$.
6. If no normal closure is properly contained, conclude that G does not contain such an $|N|$ as assumed.

We find involutions by powering up random elements.

✓

Involution centralisers

How can we compute the centraliser of an involution?

Involution centralisers

How can we compute the centraliser of an involution?

The following method by John Bray does the job:

Algorithm 2: InvolutionCentraliser

Input: $G = \langle g_1, \ldots, g_k \rangle$ and an involution $x \in G$.
initialise *gens* $:= [x]$
repeat
$y := \text{RandomElement}(G)$
$c := x^{-1}y^{-1}xy$ **and** $o := \text{Order}(c)$

Involution centralisers

How can we compute the centraliser of an involution?

The following method by John Bray does the job:

Algorithm 2: INVOLUTIONCENTRALISER

Input: $G = \langle g_1, \ldots, g_k \rangle$ and an involution $x \in G$.
initialise *gens* $:= [x]$
repeat
  $y :=$ RANDOMELEMENT(G)
  $c := x^{-1}y^{-1}xy$ **and** $o :=$ ORDER(c)
  if o is even **then**
    append $c^{o/2}$ and $(x^{-1}yxy^{-1})^{o/2}$ to *gens*
  else
    append $z := y \cdot c^{(o-1)/2}$ to *gens*

Involution centralisers

How can we compute the centraliser of an involution?

The following method by John Bray does the job:

Algorithm 2: InvolutionCentraliser

Input: $G = \langle g_1, \ldots, g_k \rangle$ and an involution $x \in G$.
initialise *gens* $:= [x]$
repeat
  $y := \textsc{RandomElement}(G)$
  $c := x^{-1}y^{-1}xy$ **and** $o := \textsc{Order}(c)$
  if o is even **then**
    append $c^{o/2}$ and $(x^{-1}yxy^{-1})^{o/2}$ to *gens*
  else
    append $z := y \cdot c^{(o-1)/2}$ to *gens*
until o was odd often enough or gens long enough
return *gens*

Involution centralisers

How can we compute the centraliser of an involution?

The following method by John Bray does the job:

Algorithm 2: INVOLUTIONCENTRALISER

Input: $G = \langle g_1, \ldots, g_k \rangle$ and an involution $x \in G$.
initialise *gens* $:= [x]$
repeat
  $y :=$ RANDOMELEMENT(G)
  $c := x^{-1}y^{-1}xy$ **and** $o :=$ ORDER(c)
  if o is even **then**
    append $c^{o/2}$ and $(x^{-1}yxy^{-1})^{o/2}$ to *gens*
  else
    append $z := y \cdot c^{(o-1)/2}$ to *gens*
until o was odd often enough or gens long enough
return *gens*

Note: If $xy = yx$ then $c = 1_G$ and $o = 1$ and $z = y$.

Involution centralisers

How can we compute the centraliser of an involution?

The following method by John Bray does the job:

Algorithm 2: InvolutionCentraliser

Input: $G = \langle g_1, \ldots, g_k \rangle$ and an involution $x \in G$.
initialise *gens* $:= [x]$
repeat
    $y :=$ RandomElement(G)
    $c := x^{-1}y^{-1}xy$ **and** $o :=$ Order(c)
    if o is even **then**
        append $c^{o/2}$ and $(x^{-1}yxy^{-1})^{o/2}$ to *gens*
    else
        append $z := y \cdot c^{(o-1)/2}$ to *gens*
until o was odd often enough or gens long enough
return *gens*

Note: If $xy = yx$ then $c = 1_G$ and $o = 1$ and $z = y$.
And: If o is odd, then z is uniformly distributed in $C_G(x)$.

Finding $N \lhd G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

Algorithm 1: INVOLUTIONDESCENT

Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.
4. Let D be the group generated by all central involutions we found.
5. For all $1 \neq x \in D$: Test if $\langle x^G \rangle \neq G$.
6. If no normal closure is properly contained, conclude that G does not contain such an $|N|$ as assumed.

Finding $N \lhd G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

Algorithm 1: InvolutionDescent

Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.
4. Let D be the group generated by all central involutions we found.
5. For all $1 \neq x \in D$: Test if $\langle x^G \rangle \neq G$.
6. If no normal closure is properly contained, conclude that G does not contain such an $|N|$ as assumed.

How do we test if we have a proper normal subgroup?

Finding $N \lhd G$

We want to find an N with $1 < N \trianglelefteq G$ and $2 \mid |N|$, or conclude that there is none.

Algorithm 1: InvolutionDescent

Initialise $H := G$. Then

1. Find a non-central involution $x \in H$. If none found, goto 4.
2. Compute its involution centraliser $C := C_H(x)$.
3. Replace H with C and goto 1.
4. Let D be the group generated by all central involutions we found.
5. For all $1 \neq x \in D$: Test if $\langle x^G \rangle \neq G$.
6. If no normal closure is properly contained, conclude that G does not contain such an $|N|$ as assumed.

How do we test if we have a proper normal subgroup?

What if D is large?

Blind descent (Babai, Beals)

Let $1 \neq x, y \in G$ and G non-abelian.

Blind descent (Babai, Beals)

Let $1 \neq x, y \in G$ and G non-abelian.

Assume at least one of x, y is contained in a non-trivial proper normal subgroup.

Blind descent (Babai, Beals)

Let $1 \neq x, y \in G$ and G non-abelian.

Assume at least one of x, y is contained in a non-trivial proper normal subgroup.

We do not know which!

Blind descent (Babai, Beals)

Let $1 \neq x, y \in G$ and G non-abelian.

Assume at least one of x, y is contained in a non-trivial proper normal subgroup.

We do not know which!

Aim: Produce $1 \neq z \in G$ that is contained in a non-trivial proper normal subgroup.

Blind descent (Babai, Beals)

Let $1 \neq x, y \in G$ and G non-abelian.

Assume at least one of x, y is contained in a non-trivial proper normal subgroup.

We do not know which!

Aim: Produce $1 \neq z \in G$ that is contained in a non-trivial proper normal subgroup.

Algorithm 3: BLINDDESCENT

1. Consider $c := [x, y] := x^{-1}y^{-1}xy$,
   if $c \neq 1$, we take $z := c$.

Blind descent (Babai, Beals)

Let $1 \neq x, y \in G$ and G non-abelian.

Assume at least one of x, y is contained in a non-trivial proper normal subgroup.

We do not know which!

Aim: Produce $1 \neq z \in G$ that is contained in a non-trivial proper normal subgroup.

Algorithm 3: BlindDescent

1. Consider $c := [x, y] := x^{-1}y^{-1}xy$,
   if $c \neq 1$, we take $z := c$.
2. If $c = 1$, the elements x and y commute.
   If $x \in Z(G)$, take $z := x$.

Blind descent (Babai, Beals)

Let $1 \neq x, y \in G$ and G non-abelian.

Assume at least one of x, y is contained in a non-trivial proper normal subgroup.

We do not know which!

Aim: Produce $1 \neq z \in G$ that is contained in a non-trivial proper normal subgroup.

Algorithm 3: BlindDescent

1. Consider $c := [x, y] := x^{-1}y^{-1}xy$,
   if $c \neq 1$, we take $z := c$.
2. If $c = 1$, the elements x and y commute.
   If $x \in Z(G)$, take $z := x$.
3. Compute generators $\{y_i\}$ for $Y := \langle y^G \rangle$.
   - If some $c_i := [x, y_i] \neq 1$, then take $z := c_i$ as in 1.

Blind descent (Babai, Beals)

Let $1 \neq x, y \in G$ and G non-abelian.

Assume at least one of x, y is contained in a non-trivial proper normal subgroup.

We do not know which!

Aim: Produce $1 \neq z \in G$ that is contained in a non-trivial proper normal subgroup.

Algorithm 3: BlindDescent

1. Consider $c := [x, y] := x^{-1}y^{-1}xy$, if $c \neq 1$, we take $z := c$.
2. If $c = 1$, the elements x and y commute. If $x \in Z(G)$, take $z := x$.
3. Compute generators $\{y_i\}$ for $Y := \langle y^G \rangle$.
   - If some $c_i := [x, y_i] \neq 1$, then take $z := c_i$ as in 1.
   - Otherwise $x \in C_G(Y)$ but $x \notin Z(G)$, thus $Y \neq G$, we take $z := y$.

Combining Algorithms 1 and 3

Algorithm 4: FINDELMOFEVENNORMALSUBGROUP

Let $G = \langle g_1, \ldots, g_k \rangle \leq \mathrm{GL}(d, q)$.

1. Use Algorithm INVOLUTIONDESCENT to produce candidate elements.
   (If there are too many central involutions, select some randomly.)
2. Use BLINDDESCENT to combine them.

Combining Algorithms 1 and 3

Algorithm 4: FINDELMOFEVENNORMALSUBGROUP

Let $G = \langle g_1, \ldots, g_k \rangle \leq \mathrm{GL}(d, q)$.

1. Use Algorithm INVOLUTIONDESCENT to produce candidate elements.
   (If there are too many central involutions, select some randomly.)
2. Use BLINDDESCENT to combine them.
3. If any of the candidates is in a proper normal subgroup, then the result will be.

Combining Algorithms 1 and 3

Algorithm 4: FINDELMOFEVENNORMALSUBGROUP

Let $G = \langle g_1, \ldots, g_k \rangle \leq \mathrm{GL}(d, q)$.

1. Use Algorithm INVOLUTIONDESCENT to produce candidate elements.
   (If there are too many central involutions, select some randomly.)
2. Use BLINDDESCENT to combine them.
3. If any of the candidates is in a proper normal subgroup, then the result will be.

- One non-trivial group element is returned.

Combining Algorithms 1 and 3

Algorithm 4: FINDELMOFEVENNORMALSUBGROUP

Let $G = \langle g_1, \ldots, g_k \rangle \leq \mathrm{GL}(d, q)$.

1. Use Algorithm INVOLUTIONDESCENT to produce candidate elements.
   (If there are too many central involutions, select some randomly.)
2. Use BLINDDESCENT to combine them.
3. If any of the candidates is in a proper normal subgroup, then the result will be.

- One non-trivial group element is returned.
- The algorithm is Monte Carlo and could return a wrong result.

✓

Examples

This approach works well in many important cases:

G	N	time
$A_{20} \wr A_{30}$	$A_5^{\times 30}$	120
$\mathrm{SL}(3,3) \wr A_{10} < \mathrm{GL}(30,3)$	$\mathrm{SL}(3,3)^{\times 10}$	724
$\mathrm{Sp}(6,3) \otimes 2.\mathrm{O}(7,3) < \mathrm{GL}(48,3)$ (computing projectively)	$\mathrm{Sp}(6,3) \otimes 1$ or $1 \otimes 2.\mathrm{O}(7,3)$	645
$6.\mathrm{Suz} < \mathrm{GL}(12,25)$	central 2	227
S_{100}	A_{100}	165
A_{100}	—	148
$\mathrm{PSL}(10,5)$	—	1248
$\mathrm{PGL}(10,5)$	$\mathrm{PSL}(10,5)$	1260

(here we have averaged over 10 runs, times in ms)

Examples

This approach works well in many important cases:

G	N	time
$A_{20} \wr A_{30}$	$A_5^{\times 30}$	120
$\mathrm{SL}(3,3) \wr A_{10} < \mathrm{GL}(30,3)$	$\mathrm{SL}(3,3)^{\times 10}$	724
$\mathrm{Sp}(6,3) \otimes 2.\mathrm{O}(7,3) < \mathrm{GL}(48,3)$ (computing projectively)	$\mathrm{Sp}(6,3) \otimes 1$ or $1 \otimes 2.\mathrm{O}(7,3)$	645
$6.\mathrm{Suz} < \mathrm{GL}(12,25)$	central 2	227
S_{100}	A_{100}	165
A_{100}	—	148
$\mathrm{PSL}(10,5)$	—	1248
$\mathrm{PGL}(10,5)$	$\mathrm{PSL}(10,5)$	1260

(here we have averaged over 10 runs, times in ms)

The success rate was 100% in all cases (using 200 runs).

✓

Reductions for imprimitive matrix groups

Situation

Let $G \leq \mathrm{GL}_n(\mathbb{F}_q)$ acting linearly on $V := \mathbb{F}_q^{1\times n}$, such that V is irreducible. Assume there is N with $Z(G) < N \lhd G$ such that

$$V|_N = W_1 \oplus W_2 \oplus \cdots \oplus W_k,$$

all W_i are invariant under N, and G permutes the W_i transitively. Then there is a reduction $\varphi : G \to S_k$.

Reductions for imprimitive matrix groups

Situation

Let $G \leq \mathrm{GL}_n(\mathbb{F}_q)$ acting linearly on $V := \mathbb{F}_q^{1\times n}$, such that V is irreducible. Assume there is N with $Z(G) < N \lhd G$ such that

$$V|_N = W_1 \oplus W_2 \oplus \cdots \oplus W_k,$$

all W_i are invariant under N, and G permutes the W_i transitively. Then there is a reduction $\varphi : G \to S_k$.

We use Algorithm FindElmOfEvenNormalSubgroup, for the result x, do:

- compute the normal closure $M := \langle x^G \rangle$,
- use the MeatAxe to check whether $V|_M$ is reducible,
- if $x \in N$, we find a reduction.

What can go wrong?

What can go wrong?

Actually, lots of things!

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.
- An order computation could take unpleasantly long.

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.
- An order computation could take unpleasantly long.
- There could be no non-central involutions.

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.
- An order computation could take unpleasantly long.
- There could be no non-central involutions.
- There could be extremely many central involutions.

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.
- An order computation could take unpleasantly long.
- There could be no non-central involutions.
- There could be extremely many central involutions.
- We could get an involution centraliser wrong.

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.
- An order computation could take unpleasantly long.
- There could be no non-central involutions.
- There could be extremely many central involutions.
- We could get an involution centraliser wrong.
- We might not find all non-central involutions.

What can go wrong?

Actually, lots of things!

- We could have trouble to find elements of even order.
- An order computation could take unpleasantly long.
- There could be no non-central involutions.
- There could be extremely many central involutions.
- We could get an involution centraliser wrong.
- We might not find all non-central involutions.
- G might not have an even order normal subgroup.

✓