Zariski density and algorithms for infinite linear groups

Alla Detinko

University of St Andrews

Groups St Andrews 2017 in Birmingham

11 August 2017

1. Computing with infinite linear groups: set up

Given a subgroup $G \leq \mathrm{GL}(n, \mathbb{F})$, $\mathbb{F}$ is an infinite field.

1. Computing with infinite linear groups: set up

Given a subgroup $G \leq \mathrm{GL}(n, \mathbb{F})$, $\mathbb{F}$ is an infinite field.

Aim: Design practical methods, algorithms, and software for computing in this class of groups.

1. Computing with infinite linear groups: set up

Given a subgroup $G \leq \mathrm{GL}(n, \mathbb{F})$, $\mathbb{F}$ is an infinite field.

Aim: Design practical methods, algorithms, and software for computing in this class of groups.

Motivation:

- Applications in mathematics and further afield.

1. Computing with infinite linear groups: set up

Given a subgroup $G \leq \mathrm{GL}(n, \mathbb{F})$, $\mathbb{F}$ is an infinite field.

Aim: Design practical methods, algorithms, and software for computing in this class of groups.

Motivation:

- Applications in mathematics and further afield.
- Convenient way to represent groups in computer.

1. Computing with infinite linear groups: set up

Given a subgroup $G \leq \mathrm{GL}(n, \mathbb{F})$, $\mathbb{F}$ is an infinite field.

Aim: Design practical methods, algorithms, and software for computing in this class of groups.

Motivation:

- Applications in mathematics and further afield.
- Convenient way to represent groups in computer.

Obstacle: Lack of methods for computing in this class of groups.

How to represent a *linear* group in computer?

Methods.

- Finite set of matrices: finitely generated groups.

How to represent a *linear* group in computer?

Methods.

- Finite set of matrices: finitely generated groups.
- Finite set of polynomials: linear algebraic groups.

How to represent a *linear* group in computer?

Methods.

- Finite set of matrices: finitely generated groups.
- Finite set of polynomials: linear algebraic groups.

How to represent a *finitely generated* linear group in computer?

Given $G = \langle g_1, \ldots, g_r \rangle \leq \mathrm{GL}(n, \mathbb{F})$, $\mathbb{F}$ is a (infinite) field.

How to represent a *linear* group in computer?

Methods.

- Finite set of matrices: finitely generated groups.
- Finite set of polynomials: linear algebraic groups.

How to represent a *finitely generated* linear group in computer?

Given $G = \langle g_1, \ldots, g_r \rangle \leq \mathrm{GL}(n, \mathbb{F})$, $\mathbb{F}$ is a (infinite) field.

Aim: *symbolic* representation of G over an arbitrary infinite field.

How to represent a *linear* group in computer?

Methods.

- Finite set of matrices: finitely generated groups.
- Finite set of polynomials: linear algebraic groups.

How to represent a *finitely generated* linear group in computer?

Given $G = \langle g_1, \ldots, g_r \rangle \leq \mathrm{GL}(n, \mathbb{F})$, $\mathbb{F}$ is a (infinite) field.

Aim: *symbolic* representation of G over an arbitrary infinite field.

Method.

- G is defined over a finitely generated extension of the prime subfield of $\mathbb{F}$.

How to represent a *linear* group in computer?

Methods.

- Finite set of matrices: finitely generated groups.
- Finite set of polynomials: linear algebraic groups.

How to represent a *finitely generated* linear group in computer?

Given $G = \langle g_1, \ldots, g_r \rangle \leq \mathrm{GL}(n, \mathbb{F})$, $\mathbb{F}$ is a (infinite) field.

Aim: *symbolic* representation of G over an arbitrary infinite field.

Method.

- G is defined over a finitely generated extension of the prime subfield of $\mathbb{F}$.
- G is defined over a finitely generated integral domain $R \subset \mathbb{F}$.

Finite approximation.

Finite approximation.

Given $G \leq \mathrm{GL}(n, R)$, R is a finitely generated integral domain. For an ideal $\rho \leq R$ let the φ_ρ denote the natural homomorphism
$\mathrm{GL}(n, R) \to \mathrm{GL}(n, R/\rho)$.

Finite approximation.

Given $G \leq \mathrm{GL}(n, R)$, R is a finitely generated integral domain. For an ideal $\rho \leq R$ let the φ_ρ denote the natural homomorphism
$\mathrm{GL}(n, R) \to \mathrm{GL}(n, R/\rho)$.

Fact: G is residually finite and approximated by matrix groups of degree n over finite fields R/ρ, ρ is maximal.

Finite approximation.

Given $G \leq \mathrm{GL}(n, R)$, R is a finitely generated integral domain. For an ideal $\rho \leq R$ let the φ_ρ denote the natural homomorphism
$\mathrm{GL}(n, R) \to \mathrm{GL}(n, R/\rho)$.

Fact: G is residually finite and approximated by matrix groups of degree n over finite fields R/ρ, ρ is maximal.

Reason: R is approximated by finite fields R/ρ, i.e. for any non-zero $a \in R$ there exists a maximal ideal ρ which does not contain a.

Finite approximation.

Given $G \leq \mathrm{GL}(n, R)$, R is a finitely generated integral domain. For an ideal $\rho \leq R$ let the φ_ρ denote the natural homomorphism
$\mathrm{GL}(n, R) \to \mathrm{GL}(n, R/\rho)$.

Fact: G is residually finite and approximated by matrix groups of degree n over finite fields R/ρ, ρ is maximal.

Reason: R is approximated by finite fields R/ρ, i.e. for any non-zero $a \in R$ there exists a maximal ideal ρ which does not contain a.

Advantage:
Reduction to computing with matrix groups over finite fields.

Example: virtually solvable groups.

Given a finitely generated $G \leq \mathrm{GL}(n, \mathbb{F})$. *One maximal ideal* is enough for the following.

Example: virtually solvable groups.

Given a finitely generated $G \leq \mathrm{GL}(n, \mathbb{F})$. *One maximal ideal* is enough for the following.

Recognition algorithms:

Example: virtually solvable groups.

Given a finitely generated $G \leq \mathrm{GL}(n, \mathbb{F})$. *One maximal ideal* is enough for the following.

Recognition algorithms:

- Testing whether G is finite.

Example: virtually solvable groups.

Given a finitely generated $G \leq \mathrm{GL}(n, \mathbb{F})$. *One maximal ideal* is enough for the following.

Recognition algorithms:

- Testing whether G is finite.
- Testing whether G is (virtually) solvable (computational realization of the *Tits alternative*);

Example: virtually solvable groups.

Given a finitely generated $G \leq \mathrm{GL}(n, \mathbb{F})$. *One maximal ideal* is enough for the following.

Recognition algorithms:

- Testing whether G is finite.
- Testing whether G is (virtually) solvable (computational realization of the *Tits alternative*);
- Testing whether G is (virtually) nilpotent, abelian-by-finite, central-by-finite, etc.

Example: virtually solvable groups.

Given a finitely generated $G \leq \mathrm{GL}(n, \mathbb{F})$. *One maximal ideal* is enough for the following.

Recognition algorithms:

- Testing whether G is finite.
- Testing whether G is (virtually) solvable (computational realization of the *Tits alternative*);
- Testing whether G is (virtually) nilpotent, abelian-by-finite, central-by-finite, etc.

Structural investigation. If G is solvable-by-finite we can:

Example: virtually solvable groups.

Given a finitely generated $G \leq \mathrm{GL}(n, \mathbb{F})$. *One maximal ideal* is enough for the following.

Recognition algorithms:

- Testing whether G is finite.
- Testing whether G is (virtually) solvable (computational realization of the *Tits alternative*);
- Testing whether G is (virtually) nilpotent, abelian-by-finite, central-by-finite, etc.

Structural investigation. If G is solvable-by-finite we can:

- Compute completely reducible part of G, including testing whether G is completely reducible or unipotent.

Example: virtually solvable groups.

Given a finitely generated $G \leq \mathrm{GL}(n, \mathbb{F})$. *One maximal ideal* is enough for the following.

Recognition algorithms:

- Testing whether G is finite.
- Testing whether G is (virtually) solvable (computational realization of the *Tits alternative*);
- Testing whether G is (virtually) nilpotent, abelian-by-finite, central-by-finite, etc.

Structural investigation. If G is solvable-by-finite we can:

- Compute completely reducible part of G, including testing whether G is completely reducible or unipotent.
- Compute Prüfer and torsion free ranks of G.

Example: virtually solvable groups.

Given a finitely generated $G \leq \mathrm{GL}(n, \mathbb{F})$. *One maximal ideal* is enough for the following.

Recognition algorithms:

- Testing whether G is finite.
- Testing whether G is (virtually) solvable (computational realization of the *Tits alternative*);
- Testing whether G is (virtually) nilpotent, abelian-by-finite, central-by-finite, etc.

Structural investigation. If G is solvable-by-finite we can:

- Compute completely reducible part of G, including testing whether G is completely reducible or unipotent.
- Compute Prüfer and torsion free ranks of G.
- Do structural investigation of finite groups over $\mathbb{F}$ via construction of an isomorphic copy over a finite field.

Example: virtually solvable groups.

Given a finitely generated $G \leq \mathrm{GL}(n, \mathbb{F})$. *One maximal ideal* is enough for the following.

Recognition algorithms:

- Testing whether G is finite.
- Testing whether G is (virtually) solvable (computational realization of the *Tits alternative*);
- Testing whether G is (virtually) nilpotent, abelian-by-finite, central-by-finite, etc.

Structural investigation. If G is solvable-by-finite we can:

- Compute completely reducible part of G, including testing whether G is completely reducible or unipotent.
- Compute Prüfer and torsion free ranks of G.
- Do structural investigation of finite groups over $\mathbb{F}$ via construction of an isomorphic copy over a finite field.

Joint with Dane Flannery and Eamonn O'Brien (see MAGMA implementation).

2. **Next step: from finite to strong approximation**

(joint work with Alexander Hulpke and Dane Flannery)

2. **Next step: from finite to strong approximation**

(joint work with Alexander Hulpke and Dane Flannery)

Groups with free non-abelian subgroups: challenges

2. **Next step: from finite to strong approximation**

(joint work with Alexander Hulpke and Dane Flannery)

Groups with free non-abelian subgroups: challenges

- Ubiquity of non solvable-by-finite groups: a finitely generated linear group most likely is not solvable-by-finite (see e.g. D. Epstein, 1971; R. Aoun, 2011).

2. **Next step: from finite to strong approximation**

(joint work with Alexander Hulpke and Dane Flannery)

Groups with free non-abelian subgroups: challenges

- Ubiquity of non solvable-by-finite groups: a finitely generated linear group most likely is not solvable-by-finite (see e.g. D. Epstein, 1971; R. Aoun, 2011).
- Undecidable basic algorithmic problems.

2. **Next step: from finite to strong approximation**

(joint work with Alexander Hulpke and Dane Flannery)

Groups with free non-abelian subgroups: challenges

- Ubiquity of non solvable-by-finite groups: a finitely generated linear group most likely is not solvable-by-finite (see e.g. D. Epstein, 1971; R. Aoun, 2011).
- Undecidable basic algorithmic problems.
- Lack of computational methods: to proceed with non solvable-by-finite groups *one ideal* may not be enough.

Why dense subgroups?

Why dense subgroups?

- Approach to computing: each finitely generated linear group H is a subgroup of a linear algebraic group $\mathcal{G}$; and without loss of generality H is (Zariski) dense in $\mathcal{G}$.

Why dense subgroups?

- Approach to computing: each finitely generated linear group H is a subgroup of a linear algebraic group $\mathcal{G}$; and without loss of generality H is (Zariski) dense in $\mathcal{G}$.
- Algorithms for dense subgroups are in high demand, particularly due to applications of in number theory, topology, physics, etc. (cf. P. Sarnak, *Notes on thin matrix groups*, 2012).

Dense, arithmetic, and thin subgroups

Given a finitely generated dense subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$, $n > 2$.
In particular, H can be *arithmetic*, i.e. of finite index in $\mathrm{SL}(n, \mathbb{Z})$.
Dense non-arithmetic subgroups are called *thin*.

Dense, arithmetic, and thin subgroups

Given a finitely generated dense subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$, $n > 2$.
In particular, H can be *arithmetic*, i.e. of finite index in $\mathrm{SL}(n, \mathbb{Z})$.
Dense non-arithmetic subgroups are called *thin*.

Facts:

- Each dense subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$ is contained in the unique 'minimal' arithmetic overgroup $\mathrm{cl}(H)$ (*arithmetic closure* of H).

Dense, arithmetic, and thin subgroups

Given a finitely generated dense subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$, $n > 2$.
In particular, H can be *arithmetic*, i.e. of finite index in $\mathrm{SL}(n, \mathbb{Z})$.
Dense non-arithmetic subgroups are called *thin*.

Facts:

- Each dense subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$ is contained in the unique 'minimal' arithmetic overgroup $\mathrm{cl}(H)$ (*arithmetic closure* of H).
- $\mathrm{SL}(n, \mathbb{Z})$, $n \geq 3$, satisfies the *congruence subgroup property* (CSP), i.e. each arithmetic subgroup H of $\mathrm{SL}(n, \mathbb{Z})$, $n \geq 3$, contains the kernel of a homomorphism $\varphi_m : \mathrm{SL}(n, \mathbb{Z}) \to \mathrm{SL}(n, \mathbb{Z}_m)$ for some $m \in \mathbb{N}$: the *principal congruence subgroup* of level m (PCS).

Dense, arithmetic, and thin subgroups

Given a finitely generated dense subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$, $n > 2$.
In particular, H can be *arithmetic*, i.e. of finite index in $\mathrm{SL}(n, \mathbb{Z})$.
Dense non-arithmetic subgroups are called *thin*.

Facts:

- Each dense subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$ is contained in the unique 'minimal' arithmetic overgroup $\mathrm{cl}(H)$ (*arithmetic closure* of H).
- $\mathrm{SL}(n, \mathbb{Z})$, $n \geq 3$, satisfies the *congruence subgroup property* (CSP), i.e. each arithmetic subgroup H of $\mathrm{SL}(n, \mathbb{Z})$, $n \geq 3$, contains the kernel of a homomorphism $\varphi_m : \mathrm{SL}(n, \mathbb{Z}) \to \mathrm{SL}(n, \mathbb{Z}_m)$ for some $m \in \mathbb{N}$: the *principal congruence subgroup* of level m (PCS).
- H contains the unique maximal principal congruence subgroup Γ_M. The level M of Γ_M is the *level* of H.

Dense, arithmetic, and thin subgroups

Given a finitely generated dense subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$, $n > 2$.
In particular, H can be *arithmetic*, i.e. of finite index in $\mathrm{SL}(n, \mathbb{Z})$.
Dense non-arithmetic subgroups are called *thin*.

Facts:

- Each dense subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$ is contained in the unique 'minimal' arithmetic overgroup $\mathrm{cl}(H)$ (*arithmetic closure* of H).
- $\mathrm{SL}(n, \mathbb{Z})$, $n \geq 3$, satisfies the *congruence subgroup property* (CSP), i.e. each arithmetic subgroup H of $\mathrm{SL}(n, \mathbb{Z})$, $n \geq 3$, contains the kernel of a homomorphism $\varphi_m : \mathrm{SL}(n, \mathbb{Z}) \to \mathrm{SL}(n, \mathbb{Z}_m)$ for some $m \in \mathbb{N}$: the *principal congruence subgroup* of level m (PCS).
- H contains the unique maximal principal congruence subgroup Γ_M. The level M of Γ_M is the *level* of H.
- If H is dense the *level* of H is defined as the level of $\mathrm{cl}(H)$.

Strong approximation

Questions.

(1) To which extent do congruence images define a linear group G?

Strong approximation

Questions.

(1) To which extent do congruence images define a linear group G?

(2) Can we compute all congruence images of G?

Strong approximation

Questions.

(1) To which extent do congruence images define a linear group G?

(2) Can we compute all congruence images of G?

Example. There are infinite index subgroups $H < \mathrm{SL}(n, \mathbb{Z})$ such that $\overline{H} \equiv \mathrm{SL}(3, \mathbb{Z})$ mod m, $\forall m \in \mathbb{N}$.

Strong approximation

Questions.

(1) To which extent do congruence images define a linear group G?

(2) Can we compute all congruence images of G?

Example. There are infinite index subgroups $H < \mathrm{SL}(n, \mathbb{Z})$ such that $H \equiv \mathrm{SL}(3, \mathbb{Z}) \bmod m, \forall m \in \mathbb{N}$.

Theorem. Let $H \leq \mathrm{SL}(n, \mathbb{Z})$ be dense.

- H surjects onto $\mathrm{SL}(n, p)$ for all but a finite number of primes p (Strong Approximation Theorem).

Strong approximation

Questions.

(1) To which extent do congruence images define a linear group G?

(2) Can we compute all congruence images of G?

Example. There are infinite index subgroups $H < \mathrm{SL}(n, \mathbb{Z})$ such that $H \equiv \mathrm{SL}(3, \mathbb{Z}) \bmod m, \forall m \in \mathbb{N}$.

Theorem. Let $H \leq \mathrm{SL}(n, \mathbb{Z})$ be dense.

- H surjects onto $\mathrm{SL}(n, p)$ for all but a finite number of primes p (Strong Approximation Theorem).
- H surjects onto $\mathrm{SL}(n, p)$ iff p does not divide the level M of H (besides small exceptions for $n = 3, 4$, $p = 2$).

Main algorithms

Given a finitely generated subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$. We can:

Main algorithms

Given a finitely generated subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$. We can:

- Test density of H.

Main algorithms

Given a finitely generated subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$. We can:

- Test density of H.
  Method: a number of Monte-Carlo and deterministic algorithms.

Main algorithms

Given a finitely generated subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$. We can:

- Test density of H.
  Method: a number of Monte-Carlo and deterministic algorithms.

- For dense H compute the set of all primes p for which H does not surjects onto $\mathrm{SL}(n, p)$ (computational realization of the *strong approximation theorem*).

Main algorithms

Given a finitely generated subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$. We can:

- Test density of H.
  Method: a number of Monte-Carlo and deterministic algorithms.

- For dense H compute the set of all primes p for which H does not surjects onto $\mathrm{SL}(n, p)$ (computational realization of the *strong approximation theorem*).
  Method: based on classification results for subgroups of $\mathrm{SL}(n, p)$ (e.g., classification of maximal subgroups of $\mathrm{SL}(n, p)$).

Main algorithms

Given a finitely generated subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$. We can:

- Test density of H.
  Method: a number of Monte-Carlo and deterministic algorithms.

- For dense H compute the set of all primes p for which H does not surjects onto $\mathrm{SL}(n, p)$ (computational realization of the *strong approximation theorem*).
  Method: based on classification results for subgroups of $\mathrm{SL}(n, p)$ (e.g., classification of maximal subgroups of $\mathrm{SL}(n, p)$).

- Compute the level M of a dense (in particular, arithmetic) subgroup H; compute $\mathrm{cl}(H)$; and find all congruence images of dense subgroup H.

Main algorithms

Given a finitely generated subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$. We can:

- Test density of H.
  Method: a number of Monte-Carlo and deterministic algorithms.

- For dense H compute the set of all primes p for which H does not surjects onto $\mathrm{SL}(n, p)$ (computational realization of the *strong approximation theorem*).
  Method: based on classification results for subgroups of $\mathrm{SL}(n, p)$ (e.g., classification of maximal subgroups of $\mathrm{SL}(n, p)$).

- Compute the level M of a dense (in particular, arithmetic) subgroup H; compute $\mathrm{cl}(H)$; and find all congruence images of dense subgroup H.
  Method: computing in $\mathrm{GL}(n, \mathbb{Z}_m)$.

Main algorithms

Given a finitely generated subgroup $H \leq \mathrm{SL}(n, \mathbb{Z})$. We can:

- Test density of H.
  Method: a number of Monte-Carlo and deterministic algorithms.

- For dense H compute the set of all primes p for which H does not surjects onto $\mathrm{SL}(n, p)$ (computational realization of the *strong approximation theorem*).
  Method: based on classification results for subgroups of $\mathrm{SL}(n, p)$ (e.g., classification of maximal subgroups of $\mathrm{SL}(n, p)$).

- Compute the level M of a dense (in particular, arithmetic) subgroup H; compute $\mathrm{cl}(H)$; and find all congruence images of dense subgroup H.
  Method: computing in $\mathrm{GL}(n, \mathbb{Z}_m)$.

Knowing M we can proceed to algorithms for *arithmetic subgroups*.

Computing with arithmetic subgroups: sample algorithms.

Computing with arithmetic subgroups: sample algorithms.

Let H be an arithmetic subgroup of $\Gamma_n := \mathrm{SL}(n, \mathbb{Z})$.

Computing with arithmetic subgroups: sample algorithms.

Let H be an arithmetic subgroup of $\Gamma_n := \mathrm{SL}(n, \mathbb{Z})$.

`IsIn`(H, g): membership test of $g \in \Gamma_n$ in H.

Computing with arithmetic subgroups: sample algorithms.

Let H be an arithmetic subgroup of $\Gamma_n := \mathrm{SL}(n, \mathbb{Z})$.

`IsIn`(H, g): membership test of $g \in \Gamma_n$ in H.

`Index`$(\Gamma_n : H)$: computing the index.

Computing with arithmetic subgroups: sample algorithms.

Let H be an arithmetic subgroup of $\Gamma_n := \mathrm{SL}(n, \mathbb{Z})$.

`IsIn`(H, g): membership test of $g \in \Gamma_n$ in H.

`Index`$(\Gamma_n : H)$: computing the index.

`IsSL`(H): test whether $H = \mathrm{SL}(n, \mathbb{Z})$.

Computing with arithmetic subgroups: sample algorithms.

Let H be an arithmetic subgroup of $\Gamma_n := \mathrm{SL}(n, \mathbb{Z})$.

`IsIn`(H, g): membership test of $g \in \Gamma_n$ in H.

`Index`$(\Gamma_n : H)$: computing the index.

`IsSL`(H): test whether $H = \mathrm{SL}(n, \mathbb{Z})$.

`IsSubnormal`(H): tests whether H is subnormal in Γ_n.

Computing with arithmetic subgroups: sample algorithms.

Let H be an arithmetic subgroup of $\Gamma_n := \mathrm{SL}(n, \mathbb{Z})$.

`IsIn`(H, g): membership test of $g \in \Gamma_n$ in H.

`Index`$(\Gamma_n : H)$: computing the index.

`IsSL`(H): test whether $H = \mathrm{SL}(n, \mathbb{Z})$.

`IsSubnormal`(H): tests whether H is subnormal in Γ_n.

`Normalizer`(H): returns a generating set of $N_{\Gamma_n}(H)$.

Computing with arithmetic subgroups: sample algorithms.

Let H be an arithmetic subgroup of $\Gamma_n := \mathrm{SL}(n, \mathbb{Z})$.

`IsIn`(H, g): membership test of $g \in \Gamma_n$ in H.

`Index`$(\Gamma_n : H)$: computing the index.

`IsSL`(H): test whether $H = \mathrm{SL}(n, \mathbb{Z})$.

`IsSubnormal`(H): tests whether H is subnormal in Γ_n.

`Normalizer`(H): returns a generating set of $N_{\Gamma_n}(H)$.

`NormalClosure`(H): returns a generating set of $\langle H \rangle^{\Gamma_n}$, H is any finitely generated subgroup of Γ_n.

Orbit-stabilizer problem.

Given an arithmetic subgroup H of $\mathrm{SL}(n, \mathbb{Z})$, and vectors $u, v \in \mathbb{Q}^n$.

Orbit-stabilizer problem.

Given an arithmetic subgroup H of $\mathrm{SL}(n, \mathbb{Z})$, and vectors $u, v \in \mathbb{Q}^n$.

`Orbit`(H, u, v): tests whether $\exists\, g \in H$ such that $g(u) = v$ and find a g if such exists.

Orbit-stabilizer problem.

Given an arithmetic subgroup H of $\mathrm{SL}(n, \mathbb{Z})$, and vectors $u, v \in \mathbb{Q}^n$.

`Orbit`(H, u, v): tests whether $\exists\, g \in H$ such that $g(u) = v$ and find a g if such exists.

`Stabilizer`(H, u): returns a generating set of $\mathrm{Stab}_H(u)$.

N.B.: `Stabilizer`(H, u) is finitely generated.

Orbit-stabilizer problem.

Given an arithmetic subgroup H of $\mathrm{SL}(n,\mathbb{Z})$, and vectors $u, v \in \mathbb{Q}^n$.

`Orbit`(H, u, v): tests whether $\exists\, g \in H$ such that $g(u) = v$ and find a g if such exists.

`Stabilizer`(H, u): returns a generating set of $\mathrm{Stab}_H(u)$.

N.B.: `Stabilizer`(H, u) is finitely generated.

Method: solution of orbit-stabilizer problem for the congruence image $\varphi_M(H)$ acting on $\mathbb{Z}_M^n$ and the principal congruence subgroup $\Gamma_M \leq H$ acting on $\mathbb{Z}^n$.

Orbit-stabilizer problem.

Given an arithmetic subgroup H of $\mathrm{SL}(n,\mathbb{Z})$, and vectors $u, v \in \mathbb{Q}^n$.

`Orbit`(H, u, v): tests whether $\exists\, g \in H$ such that $g(u) = v$ and find a g if such exists.

`Stabilizer`(H, u): returns a generating set of $\mathrm{Stab}_H(u)$.

N.B.: `Stabilizer`(H, u) is finitely generated.

Method: solution of orbit-stabilizer problem for the congruence image $\varphi_M(H)$ acting on $\mathbb{Z}_M^n$ and the principal congruence subgroup $\Gamma_M \leq H$ acting on $\mathbb{Z}^n$.

Remark: We justified decidability of the above problems (cf results by Grunewald & Segal).

Sample application: computing with monodromy groups.

Let

$$U := \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ d & d & 1 & 0 \\ 0 & -k & -1 & 1 \end{bmatrix}, \quad T := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with $d, k \in \mathbb{Z}$. Then $\mathrm{G}(d,k) = \langle U, T \rangle \leq \mathrm{Sp}(4, \mathbb{Z})$ is the monodromy group of a generalized hypergeometric ordinary differential equation.

For 14 pairs (d,k) the group $\mathrm{G}(d,k)$ is a monodromy group associated to Calabi-Yau threefolds; seven of these are arithmetic while the rest are thin.

Sample application: computing with monodromy groups.

Let

$$U := \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ d & d & 1 & 0 \\ 0 & -k & -1 & 1 \end{bmatrix}, \quad T := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with $d, k \in \mathbb{Z}$. Then $\mathrm{G}(d,k) = \langle U, T \rangle \leq \mathrm{Sp}(4, \mathbb{Z})$ is the monodromy group of a generalized hypergeometric ordinary differential equation.

For 14 pairs (d,k) the group $\mathrm{G}(d,k)$ is a monodromy group associated to Calabi-Yau threefolds; seven of these are arithmetic while the rest are thin.

Problem (D. van Straten et. al.).
Find an arithmetic subgroup $\hat{\mathrm{G}}(d,k)$ of $\mathrm{Sp}(4,\mathbb{Z})$ which contains $\mathrm{G}(d,k)$, and compute the index $|\mathrm{Sp}(4,\mathbb{Z}) : \hat{\mathrm{G}}(d,k)|$.

(d,k)	M	index	t(sec)
$(1,3)$	2	6	3.910
$(1,2)$	2	10	3.306
$(2,3)$	8	$2^6 \cdot 3 \cdot 5$	4.797
$(3,4)$	$2^2 \cdot 3^2$	$2^9 \cdot 3^5 \cdot 5^2$	7.155
$(4,4)$	2^6	$2^{20} \cdot 3^2 \cdot 5$	8.064
$(6,5)$	$2^3 \cdot 3^2$	$2^{10} \cdot 3^6 \cdot 5^2$	9.988
$(9,6)$	$2 \cdot 3^5$	$2^8 \cdot 3^{14} \cdot 5^2$	10.671
$(5,5)$	$2 \cdot 5^3$	$2^8 \cdot 3^3 \cdot 5^8 \cdot 13$	10.312
$(2,4)$	2^4	$2^{11} \cdot 3^2 \cdot 5$	5.106
$(1,4)$	2^2	$2^5 \cdot 5$	3.515
$(16,8)$	2^{10}	$2^{40} \cdot 3^2 \cdot 5$	16.841
$(12,7)$	$2^5 \cdot 3^2$	$2^{17} \cdot 3^6 \cdot 5^2$	21.446
$(8,6)$	2^7	$2^{24} \cdot 3^2 \cdot 5$	10.771
$(4,5)$	2^5	$2^{13} \cdot 3 \cdot 5$	7.605