Fast Solvers for Dense Linear Systems

Manuel Kauers

RISC-Linz, Austria
Example

Suppose you have given a sequence $a_n$ of rational numbers, say

\[
\frac{25}{24}, \frac{3898}{4213}, \frac{4774398}{5383247}, \frac{445394100}{509117429}, \frac{1875780301068}{2147400656503}, \frac{445092169340}{507340266747}, \ldots
\]
Example

Suppose you have given a sequence \( a_n \) of rational numbers, say

\[
\begin{align*}
25 & \quad 3898 & \quad 4774398 & \quad 445394100 & \quad 1875780301068 & \quad 445092169340 \\
24 & \quad 4213 & \quad 5383247 & \quad 509117429 & \quad 2147400656503 & \quad 507340266747 \\
& & & & & \cdots 
\end{align*}
\]

Suppose you suspect that \( a_n \) can be written as

\[
a_n = \text{rat}(n, H_n, H_n^{(2)}, H_n^{(3)}),
\]

for some rational function \( \text{rat} \).
Example

Suppose you have given a sequence $a_n$ of rational numbers, say

$$
\frac{25}{24}, \frac{3898}{4213}, \frac{4774398}{5383247}, \frac{445394100}{509117429}, \frac{1875780301068}{2147400656503}, \frac{445092169340}{507340266747}, \cdots
$$

Suppose you suspect that $a_n$ can be written as

$$
a_n = \text{rat}(n, H_n, H_n^{(2)}, H_n^{(3)}),
$$

for some rational function $\text{rat}$.

How could you discover such a rational function?
Example

Suppose you have given a sequence $a_n$ of rational numbers, say

\[
\frac{25}{24}, \frac{3898}{4213}, \frac{4774398}{5383247}, \frac{445394100}{509117429}, \frac{1875780301068}{2147400656503}, \frac{445092169340}{507340266747}, \cdots
\]

Suppose you suspect that $a_n$ can be written as

\[
a_n = \text{rat}(n, H_n, H_n^{(2)}, H_n^{(3)}),
\]

for some rational function $\text{rat}$.

How could you discover such a rational function?

Make an *ansatz*!
Example

Find constants $c_i \in \mathbb{Q}$ such that

$$a_n = \frac{c_1 + c_2 n + c_3 H_n + c_4 H_n^{(2)} + c_5 H_n^{(3)}}{c_6 + c_7 n + c_8 H_n + c_9 H_n^{(2)} + c_{10} H_n^{(3)}},$$
Example

Find constants \( c_i \in \mathbb{Q} \) such that

\[
a_n = \frac{c_1 + c_2 n + c_3 H_n + c_4 H_n^{(2)} + c_5 H_n^{(3)}}{c_6 + c_7 n + c_8 H_n + c_9 H_n^{(2)} + c_{10} H_n^{(3)}},
\]

i.e.,

\[
0 = c_1 + c_2 n + c_3 H_n + c_4 H_n^{(2)} + c_5 H_n^{(3)} - c_6 a_n - c_7 n a_n - c_8 H_n a_n - c_9 H_n^{(2)} a_n - c_{10} H_n^{(3)} a_n.
\]
Example

Find constants \( c_i \in \mathbb{Q} \) such that

\[
a_n = \frac{c_1 + c_2 n + c_3 H_n + c_4 H_n^{(2)} + c_5 H_n^{(3)}}{c_6 + c_7 n + c_8 H_n + c_9 H_n^{(2)} + c_{10} H_n^{(3)}},
\]

i.e.,

\[
0 = c_1 + c_2 n + c_3 H_n + c_4 H_n^{(2)} + c_5 H_n^{(3)}
- c_6 a_n - c_7 na_n - c_8 H_n a_n - c_9 H_n^{(2)} a_n - c_{10} H_n^{(3)} a_n.
\]

By plugging in \( n = 1, \ldots, 10 \) we get a dense linear system:

\[
\begin{pmatrix}
* & \cdots & *
\end{pmatrix}
\begin{pmatrix}
c_1 \\
\vdots \\
* & \cdots & *
\end{pmatrix}
= \begin{pmatrix}
c_1 \\
\vdots \\
c_{10}
\end{pmatrix}
= \begin{pmatrix}
0 \\
\vdots \\
0
\end{pmatrix}
\]
Example

This system has no solution.
Example

This system has no solution. Try a bigger ansatz:

\[ a_n = \frac{c_1 + \cdots + c_{15} n H_n H_n^{(2)} + \cdots + c_{30} n^2 (H_n^{(3)})^2}{c_{31} + \cdots + c_{45} n H_n H_n^{(2)} + \cdots + c_{60} n^2 (H_n^{(3)})^2}. \]
This system has no solution. Try a bigger ansatz:

\[ a_n = \frac{c_1 + \cdots + c_{15} n H_n H_n^{(2)} + \cdots + c_{30} n^2 (H_n^{(3)})^2}{c_{31} + \cdots + c_{45} n H_n H_n^{(2)} + \cdots + c_{60} n^2 (H_n^{(3)})^2}. \]

This leads to a system of size $60 \times 60$. 
Example

This system has no solution. Try a bigger ansatz:

\[ a_n = \frac{c_1 + \cdots + c_{15} n H_n H_n^{(2)} + \cdots + c_{30} n^2 (H_n^{(3)})^2}{c_{31} + \cdots + c_{45} n H_n H_n^{(2)} + \cdots + c_{60} n^2 (H_n^{(3)})^2}. \]

This leads to a system of size $60 \times 60$.

This system has a solution that corresponds to the closed form

\[
a_n = \left( (n + 3) H_n^2 + (2n + 3) H_n + (3n - 2) H_n^{(2)} H_n \\
+ (2n - 5) H_n^{(2)} + (n^2 + n - 3) H_n^{(3)} \\
+ (2n + 17) H_n^{(2)} H_n^{(3)} \right) / \left( 3n H_n^2 + (5n - 3) (H_n^{(2)})^2 \\
+ (6n + 5) (H_n^{(3)})^2 + (2n + 3) H_n^{(2)} + (7n - 5) H_n^{(3)} + 1 \right).
\]
Example

If there had not been a closed form at this point, we would have included cubic terms.
Example

If there had not been a closed form at this point, we would have included cubic terms.
The corresponding system would have been of size $160 \times 160$. 
Example

If there had not been a closed form at this point, we would have included cubic terms.

The corresponding system would have been of size $160 \times 160$.

The ugliest coefficient in this system would have been

$$9088325990386948470389868516199168966990698285202785767343132181522286868617842975740915627396600$$
$$7730965168605149385484475180035408435641902208677547085204403335118857901897921641508178647778278$$
$$95909364390545421753413156253428209138837436110103838070623827935592261678649929665160556677324$$
$$470873903641969510610033133866940362732235659419731684490438598259310108067614923918149572568852$$
$$463851315094097859434813883995756702579167128186328425670763241523886987083882016038071001636239$$
$$8827208185243969798419944563915280900867392963158106739766875263686972140779111507428570965825294$$
$$88925782759834228359564261186266665141843600586071958087703197746205189825787434923775654359633$$
$$14286580952543563670321455343283561699103990557348463417946008951275339383137217001034464084815$$
$$860074912527360333164889060007697392681240306838092094762240357437235301741257767771407557323331$$
$$98776514572024833132166748245392570781813055455442682338791285775275321/608071561520469263771864$$
$$91290020834051934122846232586665407095464878138276116083104729247559497016887693122971333361460$$
$$61752442615806230156283258610417579989603569611861748499212232349202704257338492766228143557$$
$$93839333646648563621353792212331512388593804234253494348937490551827553484761723686376518648743$$
$$3653876954168616008527135363644901210659942227293962109497647475233184372489732847890966566597135$$
$$44968623505997946055797174971204081295783848890368179505936580460893257023388718806123574709$$
$$88328253436342979074837271666110797383830372828145835447655486477224385836362983346375210030954$$
$$25043000357911856966334806802111130194010187489701556977700464998893774708829983347785295119355949$$
$$072698400685882490079977153154387203675657429903671982942691774960800951099556416364355824981174$$
$$95467031086106550727068112770708081706636636703709841624760002521355747824458767885526659062092840$$
$$5585081746477547520000000000000000000
Example

If there had not been a closed form at this point, we would have included cubic terms.

The corresponding system would have been of size $160 \times 160$.

The total size of the system would have been 7.5 Megabytes.
Example

If there had not been a closed form at this point, we would have included cubic terms.

The corresponding system would have been of size $160 \times 160$. The total size of the system would have been 7.5 Megabytes. And this was only a toy example...
Problem

Given: a matrix $A \in \mathbb{Q}^{n \times n}$
Find: all $x \in \mathbb{Q}^n$ such that $A \cdot x = 0$. 
Problem

Given: a matrix $A \in \mathbb{Q}^{n \times n}$
Find: all $x \in \mathbb{Q}^n$ such that $A \cdot x = 0$.

This can be done with Gaussian elimination.
Problem

Given: a matrix $A \in \mathbb{Q}^{n \times n}$
Find: all $x \in \mathbb{Q}^n$ such that $A \cdot x = 0$.

This can be done with Gaussian elimination.

But this is very slow...
Problem

*Given:* a matrix $A \in \mathbb{Q}^{n \times n}$

*Find:* all $x \in \mathbb{Q}^n$ such that $A \cdot x = 0$.

This can be done with Gaussian elimination.

But this is very slow...

*Observation:* This seems to be exponential.
**Problem**

*Given:* a matrix $A \in \mathbb{Q}^{n \times n}$

*Find:* all $x \in \mathbb{Q}^n$ such that $A \cdot x = 0$.

This can be done with Gaussian elimination.

But this is very slow...

*Observation:*
This seems to be exponential.

*Ex:* expected runtime for solving a $300 \times 300$ system: $10^{33}$ years.
Problem

*Given:* a matrix $A \in \mathbb{Q}^{n \times n}$

*Find:* all $x \in \mathbb{Q}^n$ such that $A \cdot x = 0$.

This can be done with Gaussian elimination.

But this is very slow...

*Observation:*
This seems to be exponential.

*Ex:* expected runtime for solving a $300 \times 300$ system: $10^{33}$ years.
(If you are 100,000 times faster, you still have to wait $10^{27}$ years.)
Problem

Why is this?
Problem

Why is this? Gaussian elimination should run in polynomial time.
Problem

Why is this? Gaussian elimination should run in polynomial time. Indeed it does, but let’s have a closer look:
Problem

Why is this? Gaussian elimination should run in polynomial time.

Indeed it does, but let’s have a closer look:

\[
\begin{pmatrix}
\frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\
\frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} \\
\frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14}
\end{pmatrix}
\]
Why is this? Gaussian elimination should run in polynomial time. Indeed it does, but let’s have a closer look:

\[
\begin{pmatrix}
\frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\
0 & \frac{1}{168} & \frac{8}{945} & \frac{1}{105} \\
0 & \frac{1}{198} & \frac{16}{2145} & \frac{2}{231}
\end{pmatrix}
\]
Problem

Why is this? Gaussian elimination should run in polynomial time.

Indeed it does, but let’s have a closer look:

\[
\begin{pmatrix}
\frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\
0 & \frac{1}{168} & \frac{8}{945} & \frac{1}{105} \\
0 & 0 & \frac{2}{1216215} & \frac{1}{291060}
\end{pmatrix}
\]
Problem

Why is this? Gaussian elimination should run in polynomial time.
Indeed it does, but let’s have a closer look:

\[
\begin{pmatrix}
\frac{2}{3648645} & \frac{1}{2432430} & 0 & -\frac{211}{510810300} \\
0 & \frac{1}{102162060} & 0 & -\frac{4}{297972675} \\
0 & 0 & \frac{2}{1216215} & \frac{1}{291060}
\end{pmatrix}
\]
Problem

Why is this? Gaussian elimination should run in polynomial time. Indeed it does, but let’s have a closer look:

\[
\begin{pmatrix}
\frac{1}{186376544704350} & 0 & 0 & \frac{1}{677732889834000} \\
0 & \frac{1}{102162060} & 0 & -\frac{4}{297972675} \\
0 & 0 & \frac{2}{1216215} & \frac{1}{291060}
\end{pmatrix}
\]
Problem

Why is this? Gaussian elimination should run in polynomial time.
Indeed it does, but let’s have a closer look:

\[
\begin{pmatrix}
1 & 0 & 0 & \frac{11}{40} \\
0 & 1 & 0 & -\frac{48}{35} \\
0 & 0 & 1 & \frac{117}{56}
\end{pmatrix}
\]
Problem

Why is this? Gaussian elimination should run in \textit{polynomial time}.

Indeed it does, but let’s have a closer look:

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & \frac{11}{40} \\
0 & 1 & 0 & 0 & -\frac{48}{35} \\
0 & 0 & 1 & 0 & \frac{117}{56}
\end{pmatrix}
\]

Solution: \(\left( \frac{11}{40}, -\frac{48}{35}, \frac{117}{56}, -1 \right)\)
Problem

Why is this? Gaussian elimination should run in polynomial time.
Indeed it does, but let’s have a closer look:

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & \frac{11}{40} \\
0 & 1 & 0 & 0 & -\frac{48}{35} \\
0 & 0 & 1 & 0 & \frac{117}{56}
\end{pmatrix}
\]

Solution: \((\frac{11}{40}, -\frac{48}{35}, \frac{117}{56}, -1)\)

Ugliest intermediate coefficient: \(\frac{1}{186376544704350}\)
Problem

Why is this? Gaussian elimination should run in polynomial time. Indeed it does, if numbers could be multiplied in constant time.
Problem

Why is this? Gaussian elimination should run in polynomial time. Indeed it does, if numbers could be multiplied in constant time. But in $\mathbb{Q}$, this time depends on the bitsize of the number.
Problem

Why is this? Gaussian elimination should run in \textit{polynomial time}. Indeed it does, if numbers could be multiplied in \textit{constant time}. But in \( \mathbb{Q} \), this time depends on the \textit{bitsize} of the number. The bitsize of the coefficients \textbf{doubles} at each elimination step.
Problem

Why is this? Gaussian elimination should run in polynomial time. Indeed it does, if numbers could be multiplied in constant time. But in \( \mathbb{Q} \), this time depends on the \textit{bitsize} of the number. The bitsize of the coefficients \textit{doubles} at each elimination step. Therefore, we have
Why is this? Gaussian elimination should run in polynomial time. Indeed it does, if numbers could be multiplied in constant time. But in \( \mathbb{Q} \), this time depends on the bitsize of the number. The bitsize of the coefficients doubles at each elimination step. Therefore, we have

- exponential "bit complexity" despite of the
Problem

Why is this? Gaussian elimination should run in polynomial time. Indeed it does, if numbers could be multiplied in constant time. But in \( \mathbb{Q} \), this time depends on the \textit{bitsize} of the number. The bitsize of the coefficients \textit{doubles} at each elimination step. Therefore, we have

- exponential \textit{“bit complexity”} despite of the
- polynomial \textit{“arithmetic complexity”}.
Problem

Why is this? Gaussian elimination should run in polynomial time. Indeed it does, if numbers could be multiplied in constant time. But in $\mathbb{Q}$, this time depends on the bitsize of the number. The bitsize of the coefficients doubles at each elimination step. Therefore, we have

- exponential “bit complexity” despite of the
- polynomial “arithmetic complexity”.

What to do?
Problem

Why is this? Gaussian elimination should run in polynomial time. Indeed it does, if numbers could be multiplied in constant time.

But in $\mathbb{Q}$, this time depends on the bitsize of the number. The bitsize of the coefficients doubles at each elimination step. Therefore, we have

- exponential "bit complexity" despite of the
- polynomial "arithmetic complexity".

What to do? Goal: Find ways to avoid expression swell.
Technique I: Gauss-Bareiss Elimination
Gauss-Bareiss Elimination

This is applicable to integer matrices.
Gauss-Bareiss Elimination

This is applicable to integer matrices.

Let $A = ((a_{i,j}))$ be such a matrix.
This is applicable to integer matrices.

Let $A = ((a_{i,j}))$ be such a matrix.

$$
\begin{pmatrix}
a_{1,1} & a_{1,2} & * & * & * \\
a_{2,1} & a_{2,2} & * & * & * \\
a_{3,1} & a_{3,2} & * & * & * \\
a_{4,1} & a_{4,2} & * & * & * \\
a_{5,1} & a_{5,2} & * & * & * \\
\end{pmatrix}
$$
Gauss-Bareiss Elimination

This is applicable to integer matrices.

Let $A = ((a_{i,j}))$ be such a matrix.

$$
\begin{pmatrix}
  a_{1,1} & a_{1,2} & \ast\ast & \ast\ast & \ast\ast \\
  0 & a_{1,1}a_{2,2} - a_{1,2}a_{2,1} & \ast\ast & \ast\ast & \ast\ast \\
  0 & a_{1,1}a_{3,2} - a_{1,2}a_{3,1} & \ast\ast & \ast\ast & \ast\ast \\
  0 & a_{1,1}a_{4,2} - a_{1,2}a_{4,1} & \ast\ast & \ast\ast & \ast\ast \\
  0 & a_{1,1}a_{5,2} - a_{1,2}a_{5,1} & \ast\ast & \ast\ast & \ast\ast \\
\end{pmatrix}
$$
This is applicable to integer matrices.

Let $A = ((a_{i,j}))$ be such a matrix.

$$
\begin{pmatrix}
  a_{1,1} & a_{1,2} & \ast & \ast & \ast & \ast & \ast & \ast \\
  0 & a_{1,1}a_{2,2} - a_{1,2}a_{2,1} & \ast & \ast & \ast & \ast & \ast & \ast \\
  0 & 0 & \ast & \ast & \ast & \ast & \ast & \ast \\
  0 & 0 & \ast & \ast & \ast & \ast & \ast & \ast \\
  0 & 0 & \ast & \ast & \ast & \ast & \ast & \ast
\end{pmatrix}
$$
Gauss-Bareiss Elimination

This is applicable to integer matrices.

Let $A = ((a_{i,j}))$ be such a matrix.

\[
\begin{pmatrix}
a_{1,1} & a_{1,2} & \star & \star & \star & \star & \star & \star \\
0 & a_{1,1}a_{2,2} - a_{1,2}a_{2,1} & \star & \star & \star & \star & \star & \star \\
0 & 0 & \star & \star & \star & \star & \star & \star \\
0 & 0 & \star & \star & \star & \star & \star & \star \\
0 & 0 & \star & \star & \star & \star & \star & \star \\
\end{pmatrix}
\]

**Thm.** All elements in the remaining matrix are divisible by $a_{1,1}$. 
Gauss-Bareiss Elimination

This is applicable to integer matrices.

Let $A = ((a_{i,j}))$ be such a matrix.

$$
\begin{pmatrix}
   a_{1,1} & a_{1,2} & \cdots & \cdots & \cdots \\
   0 & a_{1,1}a_{2,2} - a_{1,2}a_{2,1} & \cdots & \cdots & \cdots \\
   0 & 0 & \cdots & \cdots & \cdots \\
   0 & 0 & \cdots & \cdots & \cdots \\
   0 & 0 & \cdots & \cdots & \cdots \\
\end{pmatrix}
$$

**Thm.** All elements in the remaining matrix are divisible by $a_{1,1}$.

**Ex.** $\cdots = a_{1,1}( -a_{1,4}a_{2,2}a_{4,1} + a_{1,2}a_{2,4}a_{4,1} + a_{1,4}a_{2,1}a_{4,2} - a_{1,1}a_{2,4}a_{4,2} - a_{1,2}a_{2,1}a_{4,4} + a_{1,1}a_{2,2}a_{4,4})$
Gauss-Bareiss Elimination

This is applicable to integer matrices.

Let $A = ((a_{i,j}))$ be such a matrix.

In general, all entries in the submatrix of step $i$ are divisible by the pivot of step $i - 2$. 
This is applicable to integer matrices.

Let $A = ((a_{i,j}))$ be such a matrix.

In general, all entries in the submatrix of step $i$ are divisible by the pivot of step $i - 2$.

*Keep on dividing out the old pivots!*
Gauss-Bareiss Elimination

This is applicable to integer matrices.

Let $A = ((a_{i,j}))$ be such a matrix.

In general, all entries in the submatrix of step $i$ are divisible by the pivot of step $i - 2$.

*Keep on dividing out the old pivots!*

This division takes some time, but the resulting reduction in expression swell is worth it.
Gauss-Bareiss Elimination

This is applicable to integer matrices.

Let $A = ((a_{i,j}))$ be such a matrix.

In general, all entries in the submatrix of step $i$ are divisible by the pivot of step $i - 2$.

*Keep on dividing out the old pivots!*

This division takes some time, but the resulting reduction in expression swell is worth it.

In fact, the resulting algorithm as only polynomial *bit complexity*. 
Gauss-Bareiss Elimination

This is applicable to integer matrices.

Let $A = ((a_{i,j}))$ be such a matrix.

This technique is useless for rational matrices.
Gauss-Bareiss Elimination

This is applicable to integer matrices.

Let $A = ((a_{i,j}))$ be such a matrix.

This technique is useless for rational matrices.

Given a matrix over $\mathbb{Q}$, we could clear denominators to obtain a matrix over $\mathbb{Z}$.
**Gauss-Bareiss Elimination**

This is applicable to integer matrices.

Let \( A = ((a_{i,j})) \) be such a matrix.

This technique is useless for rational matrices.

Given a matrix over \( \mathbb{Q} \), we could clear denominators to obtain a matrix over \( \mathbb{Z} \).

But this will lead to an explosion in the bitsize of the coefficients.
This is applicable to integer matrices.

Let $A = ((a_{i,j}))$ be such a matrix.

This technique is useless for rational matrices.

Given a matrix over $\mathbb{Q}$, we could clear denominators to obtain a matrix over $\mathbb{Z}$.

But this will lead to an explosion in the bitsize of the coefficients.

We need another idea here.
Technique II: Homomorphic Images
**Homomorphic Images**

*Idea:* Perform the computation in an algebraic domain where all elements have the same bitsize.
Homomorphic Images

*Idea:* Perform the computation in an algebraic domain where all elements have the same bitsize.

Let $p$ be a prime number, e.g., $p = 7$ or $p = 2147483647$. 
**Homomorphic Images**

**Idea:** Perform the computation in an algebraic domain where all elements have the same bitsize.

Let $p$ be a prime number, e.g., $p = 7$ or $p = 2147483647$.

Let $\mathbb{Z}_p := \{0, 1, 2, 3, \ldots, p - 1\}$. 
**Homomorphic Images**

**Idea:** Perform the computation in an algebraic domain where all elements have the same bitsize.

Let $p$ be a prime number, e.g., $p = 7$ or $p = 2147483647$.
Let $\mathbb{Z}_p := \{0, 1, 2, 3, \ldots, p - 1\}$.
Define $+$ and $\cdot$ on $\mathbb{Z}_p$ via

$$a + b := (a + b) \mod p \quad a \cdot b := (a \cdot b) \mod p \quad (a, b \in \mathbb{Z}_p)$$
Homomorphic Images

Idea: Perform the computation in an algebraic domain where all elements have the same bitsize.

Let $p$ be a prime number, e.g., $p = 7$ or $p = 2147483647$.

Let $\mathbb{Z}_p := \{0, 1, 2, 3, \ldots, p - 1\}$.

Define $+$ and $\cdot$ on $\mathbb{Z}_p$ via

$$a + b := (a + b) \mod p \quad a \cdot b := (a \cdot b) \mod p \quad (a, b \in \mathbb{Z}_p)$$

Example: $4 + 5 = 2$ and $4 \cdot 5 = 6$ in $\mathbb{Z}_7$.  

Homomorphic Images

Idea: Perform the computation in an algebraic domain where all elements have the same bitsize.

Let $p$ be a prime number, e.g., $p = 7$ or $p = 2147483647$.

Let $\mathbb{Z}_p := \{0, 1, 2, 3, \ldots, p - 1\}$.

Define $+$ and $\cdot$ on $\mathbb{Z}_p$ via

\[
a + b := (a + b) \mod p \quad a \cdot b := (a \cdot b) \mod p \quad (a, b \in \mathbb{Z}_p)
\]

Example: $4 + 5 = 2$ and $4 \cdot 5 = 6$ in $\mathbb{Z}_7$.

The algebraic domain $\mathbb{Z}_p$ is called a finite field of characteristic $p$. 
Homomorphic Images

The domains $\mathbb{Z}$ and $\mathbb{Z}_p$ are closely related:
Homomorphic Images

The domains $\mathbb{Z}$ and $\mathbb{Z}_p$ are closely related:

Let $m: \mathbb{Z} \to \mathbb{Z}_p$ be the map $a \mapsto a \mod p$. 
Homomorphic Images

The domains $\mathbb{Z}$ and $\mathbb{Z}_p$ are closely related:

Let $m: \mathbb{Z} \rightarrow \mathbb{Z}_p$ be the map $a \mapsto a \mod p$.

Then

$$m(a + b) = m(a) + m(b), \quad m(a \cdot b) = m(a) \cdot m(b) \quad (a, b \in \mathbb{Z}).$$
Homomorphic Images

The domains $\mathbb{Z}$ and $\mathbb{Z}_p$ are closely related:
Let $m: \mathbb{Z} \rightarrow \mathbb{Z}_p$ be the map $a \mapsto a \mod p$.
Then

$$m(a + b) = m(a) + m(b), \quad m(a \cdot b) = m(a) \cdot m(b) \quad (a, b \in \mathbb{Z}).$$

The map $m$ is called a \textit{homomorphism}.
The domains $\mathbb{Z}$ and $\mathbb{Z}_p$ are closely related:

Let $m: \mathbb{Z} \rightarrow \mathbb{Z}_p$ be the map $a \mapsto a \mod p$.

Then

$$m(a + b) = m(a) + m(b), \quad m(a \cdot b) = m(a) \cdot m(b) \quad (a, b \in \mathbb{Z}).$$

The map $m$ is called a **homomorphism**.

We can extend $m$ from $\mathbb{Z}$ to rational numbers by mapping $u/v \in \mathbb{Q}$ to the solution of $m(v) \cdot x = m(u)$ in $\mathbb{Z}_p$. 
Homomorphic Images

The domains $\mathbb{Z}$ and $\mathbb{Z}_p$ are closely related:

Let $m: \mathbb{Z} \rightarrow \mathbb{Z}_p$ be the map $a \mapsto a \mod p$.

Then

$$m(a + b) = m(a) + m(b), \quad m(a \cdot b) = m(a) \cdot m(b) \quad (a, b \in \mathbb{Z}).$$

The map $m$ is called a homomorphism.

We can extend $m$ from $\mathbb{Z}$ to rational numbers by mapping $u/v \in \mathbb{Q}$ to the solution of $m(v) \cdot x = m(u)$ in $\mathbb{Z}_p$.

This will be possible whenever $p \nmid v$ (otherwise $m(v) = 0$.)
Homomorphic Images

The domains $\mathbb{Z}$ and $\mathbb{Z}_p$ are closely related:

Let $m: \mathbb{Z} \rightarrow \mathbb{Z}_p$ be the map $a \mapsto a \mod p$.

Then

$$m(a + b) = m(a) + m(b), \quad m(a \cdot b) = m(a) \cdot m(b) \quad (a, b \in \mathbb{Z}).$$

The map $m$ is called a homomorphism.

We can extend $m$ from $\mathbb{Z}$ to rational numbers by mapping $u/v \in \mathbb{Q}$ to the solution of $m(v) \cdot x = m(u)$ in $\mathbb{Z}_p$.

This will be possible whenever $p \nmid v$ (otherwise $m(v) = 0$.)

Example: $m(4/3) = 6$ in $\mathbb{Z}_7$, because $3 \cdot 6 = 4$ in $\mathbb{Z}_7$. 
Homomorphic Images

Global strategy:

\[ A \in \mathbb{Q}^{n \times n} \]
Global strategy:

\[ A \in \mathbb{Q}^{n \times n} \]

\[ \Downarrow \]

\[ m(A) \in \mathbb{Z}_p^{n \times n} \]
Homomorphic Images

Global strategy:

\[ A \in \mathbb{Q}^{n \times n} \]

\[ m(A) \in \mathbb{Z}_p^{n \times n} \xrightarrow{\text{Gauss in } \mathbb{Z}_p} m(x) \in \mathbb{Z}_p^n \]
Homomorphic Images

Global strategy:

\[ A \in \mathbb{Q}^{n \times n} \]

\[ m(A) \in \mathbb{Z}_p^{n \times n} \quad \text{Gauss in } \mathbb{Z}_p \quad m(x) \in \mathbb{Z}_p^n \]

\[ x \in \mathbb{Q}^n \]
Global strategy:

\[ A \in \mathbb{Q}^{n \times n} \]

\[ m(A) \in \mathbb{Z}_p^{n \times n} \]

Gauss in \( \mathbb{Z}_p \) \[ \rightarrow \]

\[ m(x) \in \mathbb{Z}_p^n \]

• Feature: Gaussian elimination in \( \mathbb{Z}_p \) has polynomial bit complexity.
Global strategy:

\[ A \in \mathbb{Q}^{n \times n} \quad \xrightarrow{\text{Gauss in } \mathbb{Z}_p} \quad m(x) \in \mathbb{Z}_p^n \]

- **Feature**: Gaussian elimination in \( \mathbb{Z}_p \) has polynomial bit complexity.
- **Problem**: \( m \) is not invertible. How to “lift” \( m(x) \) to \( x \)?
Homomorphic Images

- **Problem**: $m$ is not invertible. How to “lift” $m(x)$ to $x$?
Problem: $m$ is not invertible. How to “lift” $m(x)$ to $x$?

To do: Given $a \in \mathbb{Z}_p$, find $u/v \in \mathbb{Q}$ with $m(u/v) = a$. 

Homomorphic Images
Problem: $m$ is not invertible. How to “lift” $m(x)$ to $x$?

To do: Given $a \in \mathbb{Z}_p$, find $u/v \in \mathbb{Q}$ with $m(u/v) = a$.

One possible solution is $a/1$. 
Problem: $m$ is not invertible. How to “lift” $m(x)$ to $x$?

To do: Given $a \in \mathbb{Z}_p$, find $u/v \in \mathbb{Q}$ with $m(u/v) = a$.

One possible solution is $a/1$.

We want the solution $u/v$ where $\max(|u|, |v|)$ is minimal.
Homomorphic Images

- **Problem:** $m$ is not invertible. How to “lift” $m(x)$ to $x$?
- **To do:** Given $a \in \mathbb{Z}_p$, find $u/v \in \mathbb{Q}$ with $m(u/v) = a$.
- One possible solution is $a/1$.
- We want the solution $u/v$ where $\max(|u|, |v|)$ is minimal.
- **Example:** For $a = 3$, $p = 7$, we want to obtain $-1/2$. 
Homomorphic Images

▶ Problem: $m$ is not invertible. How to “lift” $m(x)$ to $x$?
▶ To do: Given $a \in \mathbb{Z}_p$, find $u/v \in \mathbb{Q}$ with $m(u/v) = a$.
▶ One possible solution is $a/1$.
▶ We want the solution $u/v$ where $\max(|u|, |v|)$ is minimal.
▶ Example: For $a = 3$, $p = 7$, we want to obtain $-1/2$.
▶ Example: For $a = 209510601$, $p = 2147483647$, we want to obtain $53/41$. 
Problem: $m$ is not invertible. How to “lift” $m(x)$ to $x$?

To do: Given $a \in \mathbb{Z}_p$, find $u/v \in \mathbb{Q}$ with $m(u/v) = a$.

One possible solution is $a/1$.

We want the solution $u/v$ where $\max(|u|, |v|)$ is minimal.

Example: For $a = 3$, $p = 7$, we want to obtain $-1/2$.

Example: For $a = 209510601$, $p = 2147483647$, we want to obtain $53/41$.

There is an efficient way to compute $u, v$ for given $a, p$ with a modified version of the Euclidean algorithm.
Problem: $m$ is not invertible. How to “lift” $m(x)$ to $x$?

To do: Given $a \in \mathbb{Z}_p$, find $u/v \in \mathbb{Q}$ with $m(u/v) = a$.

One possible solution is $a/1$.

We want the solution $u/v$ where $\max(|u|, |v|)$ is minimal.

Example: For $a = 3$, $p = 7$, we want to obtain $-1/2$.

Example: For $a = 209510601$, $p = 2147483647$, we want to obtain $53/41$.

There is an efficient way to compute $u, v$ for given $a, p$ with a modified version of the Euclidean algorithm.

This is called rational reconstruction.
Homomorphic Images

*Theorem.* This works.
Homomorphic Images

*Theorem.* This works.

More precisely:


**Homomorphic Images**

*Theorem.* This works.

More precisely:

*Theorem.* If $A \in \mathbb{Q}^{n \times n}$ and $p$ is a *sufficiently large* prime, then the rational reconstruction $x$ of a solution $m(x)$ of $m(A)$ in $\mathbb{Z}_p$ is a solution of $A$ in $\mathbb{Q}$. 
Theorem. This works.

More precisely:

Theorem. If $A \in \mathbb{Q}^{n \times n}$ and $p$ is a sufficiently large prime, then the rational reconstruction $x$ of a solution $m(x)$ of $m(A)$ in $\mathbb{Z}_p$ is a solution of $A$ in $\mathbb{Q}$.

What means “sufficiently large”? 
Homomorphic Images

**Theorem.** This works.

More precisely:

**Theorem.** If $A \in \mathbb{Q}^{n \times n}$ and $p$ is a *sufficiently large* prime, then the rational reconstruction $x$ of a solution $m(x)$ of $m(A)$ in $\mathbb{Z}_p$ is a solution of $A$ in $\mathbb{Q}$.

What means “sufficiently large”?

The prime $p$ has to be about twice as large as the largest numerator or denominator in the solution vector $x \in \mathbb{Q}^n$. 
Homomorphic Images

\textit{Theorem.} This works.

More precisely:

\textit{Theorem.} If $A \in \mathbb{Q}^{n \times n}$ and $p$ is a \textit{sufficiently large} prime, then the rational reconstruction $x$ of a solution $m(x)$ of $m(A)$ in $\mathbb{Z}_p$ is a solution of $A$ in $\mathbb{Q}$.

What means “sufficiently large”?

The prime $p$ has to be about twice as large as the largest numerator or denominator in the solution vector $x \in \mathbb{Q}^n$.

This might be too large to be efficient. We prefer to compute with small primes.
**Homomorphic Images**

**Idea:** Instead of one big prime $p$, compute with several small primes $p_1, p_2, \ldots, p_k$. 
**Homomorphic Images**

*Idea:* Instead of one big prime $p$, compute with several small primes $p_1, p_2, \ldots, p_k$.

Then we get several homomorphic images, $m_1(x), \ldots, m_k(x)$ of the solution $x$, one image for each of the primes.
**Homomorphic Images**

*Idea:* Instead of one big prime \( p \), compute with several small primes \( p_1, p_2, \ldots, p_k \).

Then we get several homomorphic images, \( m_1(x), \ldots, m_k(x) \) of the solution \( x \), one image for each of the primes.

There is a simple way to combine these images to one (big) image \( m(x) \) in \( \mathbb{Z}_{p_1 p_2 \cdots p_k} \), called *Chinese Remaindering:*
Homomorphic Images

Idea: Instead of one big prime $p$, compute with several small primes $p_1, p_2, \ldots, p_k$.

Then we get several homomorphic images, $m_1(x), \ldots, m_k(x)$ of the solution $x$, one image for each of the primes.

There is a simple way to combine these images to one (big) image $m(x)$ in $\mathbb{Z}_{p_1 p_2 \cdots p_k}$, called Chinese Remaindering:

If $\gcd(p, q) = 1$ then we can find $s, t$ with $sp + tq = 1$. 
Homomorphic Images

Idea: Instead of one big prime $p$, compute with several small primes $p_1, p_2, \ldots, p_k$.

Then we get several homomorphic images, $m_1(x), \ldots, m_k(x)$ of the solution $x$, one image for each of the primes.

There is a simple way to combine these images to one (big) image $m(x)$ in $\mathbb{Z}_{p_1 p_2 \cdots p_k}$, called **Chinese Remaindering**:

If $\gcd(p, q) = 1$ then we can find $s, t$ with $sp + tq = 1$.

Let $a \in \mathbb{Z}_p$, $b \in \mathbb{Z}_q$. 
**Homomorphic Images**

**Idea:** Instead of one big prime $p$, compute with several small primes $p_1, p_2, \ldots, p_k$.

Then we get several homomorphic images, $m_1(x), \ldots, m_k(x)$ of the solution $x$, one image for each of the primes.

There is a simple way to combine these images to one (big) image $m(x)$ in $\mathbb{Z}_{p_1 p_2 \cdots p_k}$, called **Chinese Remaindering**:

If $\gcd(p, q) = 1$ then we can find $s, t$ with $sp + tq = 1$.

Let $a \in \mathbb{Z}_p$, $b \in \mathbb{Z}_q$.

Consider $c = a + (b - a)sp = a + (b - a)(1 - tq)$. 
**Homomorphic Images**

**Idea:** Instead of one big prime $p$, compute with several small primes $p_1, p_2, \ldots, p_k$.

Then we get several homomorphic images, $m_1(x), \ldots, m_k(x)$ of the solution $x$, one image for each of the primes.

There is a simple way to combine these images to one (big) image $m(x)$ in $\mathbb{Z}_{p_1p_2\cdots p_k}$, called *Chinese Remaindering*:

If $\gcd(p, q) = 1$ then we can find $s, t$ with $sp + tq = 1$.

Let $a \in \mathbb{Z}_p$, $b \in \mathbb{Z}_q$.

Consider $c = a + (b - a)sp = a + (b - a)(1 - tq)$.

Then $c = a \mod p$ and $c = b \mod q$. 
Homomorphic Images

Idea: Instead of one big prime $p$, compute with several small primes $p_1, p_2, \ldots, p_k$.

Then we get several homomorphic images, $m_1(x), \ldots, m_k(x)$ of the solution $x$, one image for each of the primes.

There is a simple way to combine these images to one (big) image $m(x)$ in $\mathbb{Z}_{p_1p_2\cdots p_k}$, called **Chinese Remaindering**:

**Example:** If $a = 3$ in $\mathbb{Z}_7$ and $b = 4$ in $\mathbb{Z}_{11}$, then $(-3) \cdot 7 + 2 \cdot 11 = 1$
and $c = 3 + (4 - 3)(-3)7 = -18 = 59$ in $\mathbb{Z}_{77}$. 
Homomorphic Images

Algorithm: For primes $p_k = p_1, p_2, p_3, \ldots$ do
**Homomorphic Images**

**Algorithm:** For primes $p_k = p_1, p_2, p_3, \ldots$ do

- Solve the system $Ax = 0$ in $\mathbb{Z}_{p_k}$, obtaining an image $m_k(x)$. 
Homomorphic Images

Algorithm: For primes $p_k = p_1, p_2, p_3, \ldots$ do

- Solve the system $Ax = 0$ in $\mathbb{Z}_{p_k}$, obtaining an image $m_k(x)$.
- Combine all images $m_1(x), \ldots, m_k(x)$ to a big image $m(x)$. 
Algorithm: For primes \( p_k = p_1, p_2, p_3, \ldots \) do

- Solve the system \( Ax = 0 \) in \( \mathbb{Z}_{p_k} \), obtaining an image \( m_k(x) \).
- Combine all images \( m_1(x), \ldots, m_k(x) \) to a big image \( m(x) \).
- Apply rational reconstruction to recover a preimage \( x \) from \( m(x) \).
Homomorphic Images

**Algorithm:** For primes $p_k = p_1, p_2, p_3, \ldots$ do

- Solve the system $Ax = 0$ in $\mathbb{Z}_{p_k}$, obtaining an image $m_k(x)$.
- Combine all images $m_1(x), \ldots, m_k(x)$ to a big image $m(x)$.
- Apply rational reconstruction to recover a preimage $x$ from $m(x)$.
- If $Ax = 0$ in $\mathbb{Q}$, stop.
**Algorithm:** For primes $p_k = p_1, p_2, p_3, \ldots$ do

- Solve the system $Ax = 0$ in $\mathbb{Z}_{p_k}$, obtaining an image $m_k(x)$.
- Combine all images $m_1(x), \ldots, m_k(x)$ to a big image $m(x)$.
- Apply rational reconstruction to recover a preimage $x$ from $m(x)$.
- If $Ax = 0$ in $\mathbb{Q}$, stop.
- Otherwise, proceed with the next prime.
Homomorphic Images

**Algorithm:** For primes $p_k = p_1, p_2, p_3, \ldots$ do

- Solve the system $Ax = 0$ in $\mathbb{Z}_{p_k}$, obtaining an image $m_k(x)$.
- Combine all images $m_1(x), \ldots, m_k(x)$ to a big image $m(x)$.
- Apply rational reconstruction to recover a preimage $x$ from $m(x)$.
- If $Ax = 0$ in $\mathbb{Q}$, stop.
- Otherwise, proceed with the next prime.

**Cool:** The images $m_1(x), \ldots, m_k(x)$ can be computed independently *in parallel*, each prime on a separate processor.
In total, we get a bit complexity of $dn^2 + dn^3/N$ with
In total, we get a *bit complexity* of \(dn^2 + dn^3/N\) with

- \(n\) the size of the matrix,
Homomorphic Images

In total, we get a *bit complexity* of $dn^2 + dn^3 / N$ with

- $n$ the size of the matrix,
- $d$ the length of the output,
In total, we get a \textit{bit complexity} of $dn^2 + dn^3/N$ with

- $n$ the size of the matrix,
- $d$ the length of the output,
- $N$ the number of processors.
Homomorphic Images

In total, we get a *bit complexity* of $dn^2 + dn^3/N$ with

- $n$ the size of the matrix,
- $d$ the length of the output,
- $N$ the number of processors.

This allows to crack much larger systems in a reasonable time, even on a single processor machine.
**Homomorphic Images**

*Feature:* This technique extends to linear systems with polynomial coefficients:

\[
\begin{align*}
A & \in \mathbb{Q}[t]^{n \times n} \\
m(A) & \in \mathbb{Z}_p[t]^{n \times n} \\
M(m(A)) & \in \mathbb{Z}_p^{n \times n}
\end{align*}
\]

\[
\begin{align*}
x & \in \mathbb{Q}[t]^{n} \\
m(x) & \in \mathbb{Z}_p[t]^{n} \\
M(m(x)) & \in \mathbb{Z}_p^{n}
\end{align*}
\]
Concluding Remarks
Concluding Remarks
Concluding Remarks

- Linear systems can be solved in polynomial time.
Concluding Remarks

- Linear systems can be solved in polynomial time. Seriously.
Concluding Remarks

- Linear systems can be solved in *polynomial time*. Seriously.
- Matrix sizes of up to $2000 \times 2000$ are feasible on a laptop, at least if the solution has a reasonable bitsize.
Concluding Remarks

- Linear systems can be solved in polynomial time. Seriously.
- Matrix sizes of up to $2000 \times 2000$ are feasible on a laptop, at least if the solution has a reasonable bitsize.
- The algorithms presented in this talk are known since long.
Concluding Remarks

- Linear systems can be solved in polynomial time. Seriously.
- Matrix sizes of up to $2000 \times 2000$ are feasible on a laptop, at least if the solution has a reasonable bitsize.
- The algorithms presented in this talk are known since long.
- Modern algorithms are even faster than this. (But also more difficult.)
Concluding Remarks

- Linear systems can be solved in polynomial time. Seriously.
- Matrix sizes of up to $2000 \times 2000$ are feasible on a laptop, at least if the solution has a reasonable bitsize.
- The algorithms presented in this talk are known since long.
- Modern algorithms are even faster than this. (But also more difficult.)
- In applications, special knowledge about a matrix should always be taken into account (sparsity, structure, ...) before a general purpose algorithm is applied.