Fast Solvers for Dense Linear Systems

Manuel Kauers*[a]

[a]Research Institute for Symbolic Computation (RISC)
Altenbergerstraße 69
A4040 Linz, Austria, Europe

It appears that large scale calculations in particle physics often require to solve systems of linear equations with rational number coefficients exactly. If classical Gaussian elimination is applied to a *dense* system, the time needed to solve such a system grows exponentially in the size of the system. In this tutorial paper, we present a standard technique from computer algebra that avoids this exponential growth: homomorphic images. Using this technique, big dense linear systems can be solved in a much more reasonable time than using Gaussian elimination over the rationals.

1. Motivation

Suppose we know the first terms of a sequence (a_n) of rational numbers, say,

$$(a_n) = 0, 1, \tfrac{7}{5}, \tfrac{29}{17}, \tfrac{73}{37}, \tfrac{437}{197}, \tfrac{169}{69}, \tfrac{1343}{503}, \tfrac{3001}{1041}, \tfrac{29809}{9649}, \ldots$$

and suppose we suspect that the sequence (a_n) admits a closed form representation in terms of harmonic numbers, say

$$a_n = \frac{p(n, H_n)}{q(n, H_n)} \quad (n \geq 0)$$

for some polynomials p, q. In order to find p, q, we can make an ansatz with undetermined coefficients for generic polynomials of a certain fixed degree d,

$$p = \sum_{i,j \leq d} c_{i,j,1} x^i y^j, \quad q = \sum_{i,j \leq d} c_{i,j,2} x^i y^j.$$

Then, equating the expression

$$q(n, H_n) a_n - p(n, H_n)$$

evaluated at $n = 1, 2, 3, \ldots$ to zero gives a system of linear constraints for the undetermined coefficients $c_{i,j,k}$.

Solutions of this system are candidates for closed forms of (a_n). For the sample values given in the beginning, making an ansatz with p and q of degree $d = 1$ and solving the corresponding linear system reveals the conjecture

$$a_n = \frac{n + H_n}{1 + H_n} \quad (n \geq 0)$$

If the system had not had a solution, we could have repeated the computation with a higher degree ansatz for p and q (linear polynomials will hardly ever suffice) and/or we could take further expressions like harmonic numbers of higher order $H_n^{(2)}, H_n^{(3)}, \ldots$ into account (n and H_n will hardly ever suffice).

Proceeding in this way very soon leads to rather big linear systems over the rationals, and so to a demand for solving such systems. Note that we cannot hope for sparsity in the systems arising in this approach; typically there is not a single zero coefficient. Therefore, we focus on *dense* linear systems and leave the large subject of solving *sparse* linear systems entirely aside. Our problem is the following:

GIVEN $A \in \mathbb{Q}^{n \times m}$ (dense),
FIND a basis of $\ker A$.

By $\ker A$ we denote the *kernel* (or *nullspace* or *solution space* or *solution*) of the matrix A, i.e., the linear subspace of all $x \in \mathbb{Q}^m$ with $Ax = 0$.

In principle, this problem can be solved with the usual Gaussian elimination. But there is a

*Partially supported by the Austrian Science Foundation (FWF) grants SFB F1305, P19462-N18 and P20162-N18.

catch when Gaussian elimination is applied to a dense matrix: When the jth column is being eliminated using some element $a_{i,j} \neq 0$ as a pivot, then all elements $a_{k,l}$ in the remaining submatrix are replaced by $a'_{k,l} := a_{k,l} - a_{k,j}a_{i,l}/a_{i,j}$. In general, the addition of two rational numbers p and q will give a rational number $p+q$ of about twice the bit size of the two summands, and so in the generic case, the entries $a'_{k,l}$ of the submatrix will have about twice as many digits than the elements $a_{k,l}$ from before the elimination. As n columns have to be eliminated in total, and the bitsize of the entries doubles at each step, it follows that the elimination consumes a time that is proportional to 2^n. This phenomenon is called *expression swell.*

Expression swell is not only a hypothetical worst case scenario, but arises for almost every dense matrix, including typical dense matrices arising in applications. For these, however, the expression swell is often only *intermediate,* i.e., elimination produces very long fractions which in the very end all cancel out to give solution vectors with reasonably sized coefficients. In the following, we explain two ways of avoiding this intermediate expression swell in Gaussian elimination. The first way is a fraction free elimination scheme, applicable to integer matrices. The second uses homomorphic images. None of these approaches is new, they both belong to the standard folklore in computer algebra. Pointers to the relevant literature are given in the end.

2. Fraction Free Gaussian Elimination

If $A \in \mathbb{Z}^{n\times m}$, then we may modify the elimination step in such a way that no fractions are introduced: Instead of $a'_{k,l} := a_{k,l} - a_{k,j}a_{i,l}/a_{i,j}$, we use the update formula $a'_{k,l} := a_{k,l}a_{i,j} - a_{k,j}a_{i,l}$. This is obviously correct as well because multiplying a matrix row by a nonzero element does not change the solution space. This scheme, however, still leads to an exponential expression swell, because a product pq has in general twice as many digits as the factors p or q.

It can now be shown that when the ith column is about to be eliminated, then all the entries from the remaining submatrix are divisible by the pivot that was used for eliminating the $(i-1)$st column. If we keep on dividing out these old pivots, we gain a considerable improvement on the expression swell. The full elimination algorithm is then as follows:

INPUT $A = ((a_{i,j})) \in \mathbb{Z}^{n\times m}$
OUTPUT $A \in \mathbb{Z}^{n\times m}$ in reduced echolon form

```
1  d = 1; r = 1;
2  for k from 1 to m do
3     if a_pk = 0 for all p then next k
4     Let p be such that a_pk ≠ 0
5     Exchange rows p and r
6     for i from r + 1 to n do
7        for j from k to m do
8           a_ij = (a_rk a_ij − a_rj a_ik)/d
9     d = a_rk; r = r + 1;
10 return A
```

This algorithm differs from the usual elimination algorithm only in the update formula in line 8. By theory, the division in this line will always yield an integer. Surprisingly enough, this simple modification turns the exponential runtime into a polynomial runtime.

3. Homomorphic Images

If the entries of A are not just integers but arbitrary rational numbers, then the fraction free approach is not directly applicable. Of course it is possible to first clear all denominators and then apply the fraction free elimination, but clearing denominators will in general already lead to an explosion of size of the entries.

We now turn to an independent approach which is applicable also to matrices with rational number entries. The idea is to perform the elimination in an algebraic domain where all elements have a certain fixed length, so that no expression swell can occur. These “algebraic domains” are finite fields.

3.1. Finite Fields

Let p be a prime number and consider the set $\mathbb{Z}_p := \{0, 1, \dots, p-1\}$. We define operations $\oplus$ and $\otimes$ on $\mathbb{Z}_p$ via

$$a \oplus b := (a+b) \bmod p, \quad a \otimes b := (a \cdot b) \bmod p$$

($a, b \in \mathbb{Z}_p$), where $+$ and $\cdot$ on the right refer to ordinary addition and multiplication of integers, and mod refers to the remainder upon (integral) division by p. We will write $+$ instead of $\oplus$ and $\cdot$ instead of $\otimes$ from now on, although the operations in $\mathbb{Z}_p$ must be carefully distinguished from those in $\mathbb{Z}$.

The set $\mathbb{Z}_p$ is called a *finite field.* Arithmetic in $\mathbb{Z}$ and $\mathbb{Z}_p$ is closely related: consider the map

$$m\colon \mathbb{Z} \to \mathbb{Z}_p, \quad m(a) := a \bmod p,$$

which maps any $a \in \mathbb{Z}$ to its remainder modulo p. Then

$$m(a+b) = m(a)+m(b), \quad m(a\cdot b) = m(a)\cdot m(b)$$

($a, b \in \mathbb{Z}$), i.e., addition and multiplication may be exchanged with application of m. (Maps with this property are called *homomorphisms.*)

The map m may be extended from $\mathbb{Z}$ to (most of) $\mathbb{Q}$ by defining that a fraction $u/v \in \mathbb{Q}$ ($u, v \in \mathbb{Z}$, $v \geq 1$) is mapped to the solution $x \in \mathbb{Z}_p$ of the linear equation $m(v)x = m(u)$ in $\mathbb{Z}_p$. Such a solution will exist whenever v is not a multiple of p. For example, in $\mathbb{Z}_7$, we have $m(4/3) = 6$ because $3 \cdot 6 = 4$ in $\mathbb{Z}_7$. Given $v \in \mathbb{Z}$, we can find $m(1/v)$ with the extended Euclidean algorithm, which, for the sake of completeness, we describe next.

3.2. The Extended Euclidean Algorithm

Let us temporarily forget about finite fields and consider two integers $a, b \in \mathbb{Z}$. The greatest common divisor $\gcd(a, b)$ of these numbers can be found with the Euclidean algorithm by repeatedly replacing the larger number (in absolute value) by its remainder upon division by the other number. The sequence of numbers computed in this way will eventually become zero, and the last nonzero number is precisely the desired gcd.

It can be shown that for every $a, b \in \mathbb{Z}$ there will exist numbers $s, t \in \mathbb{Z}$ such that

$$as + bt = \gcd(a, b).$$

These numbers are called *cofactors* or *Bezout coefficients,* and they can be computed by the extended Euclidean algorithm (EEA), as follows.

INPUT: $a, b \in \mathbb{Z}$
OUTPUT: $g, s, t \in \mathbb{Z}$ with $as+bt = g = \gcd(a, b)$.

1 $(g, s, t, g', s', t') := (a, 1, 0, b, 0, 1)$;
2 **while** $g' \neq 0$ **do**
3 $\quad q = g \text{ quo } g'$
4 $\quad (g, s, t, g', s', t')$
$\qquad := (g', s', t', g - qg', s - qs', t - qt')$;
5 **return** (g, s, t)

The token quo in line 3 refers to the integer quotient, e.g., $8 \text{ quo } 3 = 2$. As an example, applying the algorithm to $a = 34567$ and $b = 76543$ gives the following trace for (g, s, t):

34567	1	0
76543	0	1
34567	1	0
7409	-2	1
4931	9	-4
2478	-11	5
2453	20	-9
25	-31	14
3	3058	-1381
1	-24495	11062

Indeed, we have $\gcd(34567, 76543) = 1$ and

$$1 = (-24495) \cdot 34567 + 11062 \cdot 76543.$$

The EEA can be used for performing division in $\mathbb{Z}_p$. Let $a \in \mathbb{Z}_p$. Then $\gcd(a, p) = 1$ (because p is prime) and so we can find $s, t \in \mathbb{Z}$ with

$$1 = as + pt,$$

i.e., $1 = as \bmod p$, i.e. $m(1/a) = s$ if $m\colon \mathbb{Q} \to \mathbb{Z}_p$ is as before. For example, by the computation above $m(1/34567) = -24495 = 52048$ in $\mathbb{Z}_{76543}$. We will see next that the EEA can not only be used for mapping rational numbers to elements of $\mathbb{Z}_p$, but it can also be used for the opposite direction.

3.3. Rational Reconstruction

After having mapped a given matrix over $\mathbb{Q}$ to some finite field $\mathbb{Z}_p$ and after having solved the system in that field, we only have a solution that is valid for this field. If this solution is the image $m(x)$ of a "true" solution x, then we need a way to reconstruct a rational number $x \in \mathbb{Q}$ from its homomorphic image $m(x) \in \mathbb{Z}_p$.

Unfortunately, the map m is not invertible. If, for example, $p = 76543$ and $m(x) = 34567$ then we clearly have $m(34567) = 34567$, but also $m(111110) = 34567$ or $m(-1/24495) = 34567$. There are infinitely many rational numbers that are mapped to 34567 under m. Among those, we wish to determine the number u/v where $\max\{|u|, |v|\}$ is minimized. Intuitively, this means that we want to distribute the available information evenly among numerator u and denominator v. (In contrast, in the preimages $34567/1$ and $-1/24495$ all information is pressed into numerator and denominator, respectively.)

Consider the values of g, s, t during execution of the EEA algorithm applied to $a, b \in \mathbb{Z}$. The identity

$$g = as + bt$$

is true in every iteration. So $m(g/s) = a$ in $\mathbb{Z}_b$ for all pairs (g, s) in the trace of the algorithm. We will get a good rational preimage if we abort the iteration when g and s are approximately the same in size, this is when $|g| \approx |s| \approx \sqrt{b}$. (Note that we may assume $p = b > a \geq 0$ for our purpose.) So we obtain an algorithm for rational reconstruction by replacing line 2 of the EEA by

2 **while** $|g| > \sqrt{b}$ **do**

and returning g/s in the end.

For example, with $a = 34567$ and $p = b = 76543$ we find that $m(-25/37) = 34567$ in $\mathbb{Z}_{76543}$.

3.4. System Solving with a Big Prime

Now we turn to linear systems. Let $A \in \mathbb{Q}^{n\times m}$ and let p be a large prime number. Let $m\colon \mathbb{Q} \to \mathbb{Z}_p$ be as defined as above (if we later run into one of the few $x \in \mathbb{Q}$ for which $m(x)$ is undefined, then we discard p and try another prime. Almost all primes will be fine.) We map A to a matrix $m(A) \in \mathbb{Z}_p^{n\times m}$, then solve the system over $\mathbb{Z}_p$, obtaining a basis B_p of $\ker m(A) \subseteq \mathbb{Z}_p^m$, which will (with high probability) be the image $m(B)$ of a basis B of $\ker A \subseteq \mathbb{Q}^{n\times m}$:

$$\begin{tikzcd}
A \in \mathbb{Q}^{n\times m} \arrow[d, "\text{mod}"'] & \ker A \subseteq \mathbb{Q}^m \\
m(A) \in \mathbb{Z}_p^{n\times m} \arrow[r, "\text{Gauss}"] & \ker m(A) \subseteq \mathbb{Z}_p^{n\times m} \arrow[u, "\substack{\text{rational}\\\text{recon-}\\\text{struction}}"']
\end{tikzcd}$$

As an example, let

$$A = \begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 \\ 1/6 & 1/7 & 1/8 & 1/9 \\ 1/10 & 1/11 & 1/12 & 1/13 \end{pmatrix}$$

and $p = 10007$. Then

$$m(A) = \begin{pmatrix} 5004 & 3336 & 2502 & 4003 \\ 1668 & 7148 & 1251 & 1112 \\ 7005 & 3639 & 834 & 6928 \end{pmatrix}.$$

The kernel of this matrix in $\mathbb{Z}_p$ is generated by the vector $(4875, 617, 6772, 1)$. Applying rational reconstruction to this vector gives the solution candidate $b = (-\frac{8}{39}, \frac{77}{65}, \frac{87}{34}, 1)$. But this vector is not a solution as $Ab \neq 0$. Our prime was too small for the method to succeed.

Taking the bigger prime $p = 76543$, we get

$$m(A) = \begin{pmatrix} 38272 & 51029 & 19136 & 45926 \\ 63786 & 43739 & 9568 & 42524 \\ 22963 & 13917 & 31893 & 5888 \end{pmatrix}.$$

The kernel of this matrix in $\mathbb{Z}_p$ is generated by the vector $(9813, 60058, 48279, 1)$. Applying rational reconstruction to this vector gives the solution candidate $b = (-\frac{8}{39}, \frac{77}{65}, -\frac{128}{65}, 1)$. This vector is actually a solution, as $Ab = 0$.

It can happen that $m(A)$ has more solutions in $\mathbb{Z}_p$ than A has in $\mathbb{Q}$. For example, the system

$$\begin{pmatrix} 1 & 6 \\ 1 & 1 \end{pmatrix}$$

has obviously no solution in $\mathbb{Q}$, but it has the solution $(1, -1)$ in $\mathbb{Z}_5$, because $6 = 1$ in $\mathbb{Z}_5$. Primes p for which the nullspace of $m(A)$ in $\mathbb{Z}_p$ has a larger dimension than the nullspace of A in $\mathbb{Q}$ are called *unlucky.* Fortunately, large unlucky primes are very rare. In practice, they virtually never occur. (And if there occurs one, we discard it and take another prime.)

Note that 10007 is not an unlucky prime in this sense: the image of $(-\frac{8}{39}, \frac{77}{65}, -\frac{128}{65}, 1)$ in $\mathbb{Z}_{10007}$ is precisely the vector $(4875, 617, 6772, 1)$ that we obtained by solving the system in $\mathbb{Z}_{10007}$. It was the rational reconstruction that has failed to hit the right rational solution, because there are shorter preimages for the prime 10007. With increasing length of the chosen prime, all these preimages that are shorter than the rational solution will eventually disappear, so that the shortest preimage of m, as found by rational reconstruction, will coincide with the rational solution. As a rule, this will happen as soon as p exceeds the square of the largest numerator or denominator appearing in the solution vectors, in our example $128^2 = 16384$. Indeed, the reconstruction succeeds for the first prime beyond this number, $p = 16411$.

A bound for the coefficients in the solution vector in dependence of the entries of A can be obtained from bounds for determinants such as Hadamard's bound. However, most often these bounds will be too pessimistic on practical examples and would unnecessarily slow down the computation. It is better to do the computation with some large prime, and redo the computation with a prime of, say, twice the size if necessary. Eventually we will be using a prime that is large enough but, typically, still much smaller than the worst case bounds. A disadvantage of this procedure is that potentially many modular solutions are computed just to be thrown away. It would be much more economic if these results could be reused. This is indeed possible, as we will see next.

3.5. Chinese Remaindering

Let $p, q \in \mathbb{Z}$ with $\gcd(p, q) = 1$ but not necessarily prime. Let $m_p\colon \mathbb{Q} \to \mathbb{Z}_p$ and $m_q\colon \mathbb{Q} \to \mathbb{Z}_q$ be the modular maps and suppose that about some unknown rational number $x \in \mathbb{Q}$ we know its images $m_p(x) \in \mathbb{Z}_p$ and $m_q(x) \in \mathbb{Z}_q$. Our goal is to compute from this information the image of x in $\mathbb{Z}_{pq}$. This will enable us to combine different small modular solutions to a single big modular solution for which we then can proceed as before.

By $\gcd(p, q) = 1$, there exist numbers $s, t \in \mathbb{Z}$ such that $sp + tq = 1$, and such numbers can be found with the EEA. Now set

$$\begin{aligned} u &:= m_p(x) + (m_q(x) - m_p(x))sp \\ &= m_p(x) + (m_q(x) - m_p(x))(1 - tq) \end{aligned}$$

and observe that $u \bmod p = m_p(x)$ and $u \bmod q = m_q(x)$. It can be shown that u is the only number in $\mathbb{Z}_{pq}$ with this property, and therefore it must be the image of x in that domain. We have the following simple algorithm, which is known as the Chinese Remainder Algorithm:

INPUT: $a \in \mathbb{Z}_p$, $b \in \mathbb{Z}_q$ where $\gcd(p, q) = 1$.
OUTPUT: $u \in \mathbb{Z}_{pq}$ such that $u \bmod p = m_p(x)$ and $u \bmod q = m_q(x)$

Find s, t such that $sp + tq = 1$.
return $a + (b - a)sp$

3.6. System Solving with Small Primes

Instead of one big prime, we will now use several small primes. We solve the linear system for each prime, and then combine the results with Chinese remaindering. This has two advantages: First, we need not compute with very long primes, even if the output contains very long numbers. In particular, we can stick to primes that are short enough to fit into a processor word (typically 32 or 64 bit) so that fast hardware arithmetic can be exploited. Secondly, we need not estimate in advance the length of the numbers in the output, but instead keep on including new primes until their product is so long that rational reconstruction gives the right answer. This leads to the following algorithm, in which also the unlikely event of encountering unlucky primes is handled.

INPUT: $A \in \mathbb{Q}^{n\times m}$
OUTPUT: a basis B of $\ker A \subseteq \mathbb{Q}^m$.

$q = 0$
repeat
  Let p be a new prime.
  if $m_p(A) = \perp$ **then**
    next // discard unlucky prime.
  Let B_p be a basis of $\ker m_p(A) \subseteq \mathbb{Z}_p^m$.
  if $q = 0$ **or** $|B_p| < |B_q|$ **then**
    $q = p; B_q = B_p;$ // (re)initialize
  else if $|B_p| = |B_q|$ **then** // combine
    $B_q = \mathrm{CRA}(B_q, B_p, q, p);\quad q = q \cdot p$
    $B = \mathrm{RR}(B_q, q);$
until $A \cdot b = 0$ for all $b \in B$
return B

The condition $m_p(A) = \perp$ means that the homomorphic image of A in $\mathbb{Z}_p^{n\times m}$ does not exist, because some denominator in A is a multiple of p.

The token **next** is meant to break the current iteration and proceed directly to the next. The function CRA refers to the Chinese remainder algorithm as described in the previous section, applied to corresponding entries of B_q and B_p. The function RR refers to rational reconstruction as described in Section 3.3.

As an example, consider the matrix A from Section 3.4. For $p = 131$ we obtain the modular solution $b_p = (114, 108, 125, 1)$ of $m_p(A)$. Rational reconstruction turns b_p into $(-\frac{5}{8}, -\frac{7}{6}, -6, 1)$ which is not yet a solution over $\mathbb{Q}$. For the next prime, $p = 137$, we obtain the modular solution $b_p = (56, 115, 17, 1)$. With the Chinese remainder algorithm, we combine this solution with the previous one to the solution $(13345, 14911, 2483, 1)$ valid modulo $131 \cdot 137 = 17947$. Rational reconstruction turns this vector into $(-\frac{8}{39}, \frac{77}{65}, -\frac{128}{65}, 1)$. This is the correct solution and the algorithm stops.

It is important to normalize the modular solutions such as to ensure that modular solutions for different primes share the same preimage. If, in the example, we had chosen $2 \cdot (114, 108, 125, 1) = (97, 85, 119, 2)$ as the first solution (after all, the solution of a linear system is not unique), then the algorithm would never have terminated. To avoid this effect, we force one component to 1 in every modular solution (in the example, we took the fourth). Similarly, if the solution space is higher dimensional, we can bring the modular solution vectors to a form

$$(*, \dots, *, 0, \dots, 0, 1, 0, \dots, 0)$$

to ensure that different modular solutions correspond to the same preimage.

The complexity of the algorithm depends on the size of the input matrix as well as on the length of the numbers in the output. Let us consider a square matrix $A \in \mathbb{Q}^{n\times n}$ for simplicity. Application of m_p to the n^2 matrix elements requires time proportional to n^2, Chinese remaindering and rational reconstruction take time proportional to $n^2\ell$, where ℓ bounds the length of the numbers in the output. The runtime bottleneck is the computation of the modular solutions, each of which takes time proportional to n^3, together $n^3\ell$ (the number of primes needed is proportional to ℓ.) As the computations of solutions in $\mathbb{Z}_p$ are entirely independent of each other, they can be done in parallel on P processors. The overall complexity is then proportional to $n^2\ell + n^3\ell/P$.

4. Concluding Remarks

We have been focussing on matrices over $\mathbb{Q}$, i.e., matrices without any parameters. Similar techniques can be applied for matrixes over $\mathbb{Q}(x)$, i.e., matrices containing a single parameter x. Such matrices are reduced to several matrices over $\mathbb{Q}$ by evaluating x at several points (analogous to the reduction to several $\mathbb{Z}_p$). Their solutions are then combined via interpolation (analogous to Chinese remaindering), and Pade approximation (analogous to rational reconstruction) is used to turn interpolating polynomials into rational functions. For matrices over $\mathbb{Q}(x_1, \dots, x_n)$, the method can be applied recursively, giving, however, a complexity that is exponential in the number of parameters.

The algorithms described in this paper prove useful for solving large dense linear systems over rationals. We repeat that they are well known. Further details can be found in textbooks on computer algebra such as [1] and [4]. These algorithms should be used only for dense systems which have no special structure. Otherwise there are typically faster special purpose algorithms (see, e.g., [2]). Also for general dense matrices, modern algorithms (e.g., [3]) are faster than the ones presented here.

REFERENCES

1. Keith O. Geddes, Stephen R. Czapor, and George Labahn. *Algorithms for Computer Algebra.* Kluwer, 1992.
2. Victor Y. Pan. *Structured Matrices and Polynomials.* Birkhäuser, 2001.
3. Arne Storjohann and Gilles Villard. Computing the rank and a small nullspace basis of a polynomial matrix. In *Proceedings of ISSAC'05*, pages 309–316, 2005.
4. Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra.* Cambridge University Press, 1999.