On the Complexity of Matrix Reduction over Finite Fields

Daniel Andrén, Lars Hellström, Klas Markström*

April 18, 2007

Abstract

In this paper we study the complexity of matrix elimination over finite fields in terms of row operations, or equivalently in terms of the distance in the the Cayley graph of $\mathrm{GL}_n(\mathbb{F}_q)$ generated by the elementary matrices.

We present an algorithm called *striped matrix elimination* which is asymptotically faster than traditional Gauss–Jordan elimination. The new algorithm achieves a complexity of $\mathcal{O}\left(n^2/\log_q n\right)$ row operations, and $\mathcal{O}\left(n^3/\log_q n\right)$ operations in total, thanks to being able to eliminate many matrix positions with a single row operation. We also bound the average and worst-case complexity for the problem, proving that our algorithm is close to being optimal, and show related concentration results for random matrices.

Next we present the results of a large computational study of the complexities for small matrices and fields. Here we determine the exact distribution of the complexity for matrices from $\mathrm{GL}_n(\mathbb{F}_q)$, with n and q small. Finally we consider an extension from finite fields to finite semifields of the matrix reduction problem. We give a conjecture on the behaviour of a natural analogue of GL_n for semifields and prove this for a certain class of semifields.

MSC: 15A33, 65F05, 68W30, 68Q17, 68-04
Keywords: Matrix reduction, Complexity, Finite fields, Semifields

1 Introduction

One of the most basic facts of linear algebra is that any invertible matrix can be written as a product of so-called elementary matrices, or equivalently that any invertible matrix can be reduced to the identity matrix using row operations. This is a fact used in many proofs and also the workhorse behind many numerical methods. For numerical algorithms an essential feature is that row reduction of a matrix can be performed in time polynomial in the matrix side, e.g. using Gaussian elimination in $\mathcal{O}(n^3)$ element operations for an $n \times n$ matrix, or (at least asymptotically) even faster using methods based on fast matrix multiplication [1].

*Department of Mathematics and Mathematical Statistics, Umeå universitet, SE-901 87 Umeå, Sweden

The aim of this paper is to consider the complexity of matrix reduction from a different point of view. We wish to consider only methods based on row operations, and primarily use the number of such as the complexity of the problem. One computational reason for focusing on row operations is that they on existing processors can often be implemented far more efficiently than straight line programs in general, in particular if each word of memory stores multiple matrix elements, although we shall not make use of any particular computer architecture in our analysis of the problem here. The fast methods for general matrix reduction based on fast matrix multiplication can not be expressed in terms of row operations so the two complexity measures are essentially different. See Chapter 16 of [2] for a good overview of the connections between matrix multiplication and other matrix problems.

In terms of row operations, the worst-case complexity of Gauss–Jordan elimination is n^2 (i.e., one row operation per matrix element), and this gives an elementary upper bound on the complexity of matrix reduction. For matrices over infinite fields such as $\mathbb{C}$ this bound turns out to be optimal, however when we consider matrices over a finite field it is in general possible to do better, at least for large enough matrices. We will give both lower and upper bounds of the order $\mathcal{O}\left(n^2/\log n\right)$ for the worst-case complexity of row reduction of an $n \times n$ matrix over any given finite field, and the constants of these bounds are only a factor 2 apart. Our upper bound comes from a new algorithm for matrix elimination called *striped matrix elimination* which, for small field sizes, guarantees that we can on the average eliminate more than one off-diagonal position per row operation. We also show that most matrices require a number of operations which is close to the worst case.

Finally we will report the result of a large scale computational effort to determine the exact reduction complexity for small values of n and the field size q. Here we have also investigated what happens when the base field is replaced by a semifield. These computations touch upon a very interesting construction problem, attributed to Wigderson in [5], namely to explicitly construct a family of matrices over $\mathbb{Z}_2$ whose row reduction requires a super-linear number of row operations; see also [23] for a survey of related problems. While we do not have such a construction we have found all the extremal matrices for small n.

A natural point of view when considering our result is in terms of the Cayley graph for $\mathrm{GL}_n(\mathbb{F}_q)$ with the elementary matrices as the set of generators; in this setting the complexity of row reduction is exactly the diameter of this Cayley graph. Our results thus give close to optimal bounds for the diameter of this Cayley graph and also a fast algorithm which can find a near optimal expression for any element of the group as product of its generators. Here our results form an interesting parallel to the work of Riley and Kassabov [7, 21] on constant size generating sets for $\mathrm{SL}_n(\mathbb{Z}_k)$, and the non-algorithmic diameter bounds of [13] for finite simple groups.

1.1 Notation and some facts about $\mathrm{GL}(n, q)$

In this paper we will use $\mathrm{GL}_n(F)$ to denote the group of invertible $n \times n$-matrices over a field F, and we will also use the short form $\mathrm{GL}(n, q) = \mathrm{GL}_n(\mathbb{F}_q)$, where $\mathbb{F}_q$ is the finite field of order q.

Let us first state some basic facts about $\mathrm{GL}(n, q)$; see [22] for a good textbook reference on this.

1.1.1 The size of $\mathrm{GL}(n,q)$

Using some basic q-combinatorics we can write the number of elements in $\mathrm{GL}(n,q)$ as

$$|\mathrm{GL}(n,q)| = \prod_{i=0}^{n-1}(q^n - q^i) = q^{n^2}\prod_{i=0}^{n-1}(1-q^{i-n}) = q^{n^2}C(n,q), \tag{1}$$

for some constant $C(n,q)$. Here $C(n,q)$ can be interpreted as the probability that a matrix with random entries from $\mathbb{F}_q$ will be invertible.

For a fixed n and increasing q the product $C(n,q)$ approaches 1 at a speed proportional to q^{-1}, and for a fixed q and increasing n the product $C(n,q)$ will converge to a value $C(\infty,q) < 1$ quite rapidly. Using a theorem of Euler [4] we can find the asymptotic value for $C(\infty,q)$ as

$$C(\infty,q) = \prod_{i=1}^{\infty}(1-q^{-i}) = 1 + \sum_{i=1}^{\infty}(-1)^i\left(q^{-\omega(i)} + q^{-\omega(-i)}\right) \tag{2}$$

where $\omega(m) = \frac{1}{2}\left(3m^2+m\right)$. Note that the alternating signs of the sparse series make it possible to compute good bounds for the series, should they be needed. As q increases $C(\infty,q)$ will monotonely increase to 1, and is already above 0.9 for $q = 11$.

1.1.2 Elementary matrices and the Cayley graph

The three basic types of row operations on a matrix M from $\mathrm{GL}(n,q)$ are

1. Adding a non-zero multiple of one row of M to another. There are $(q-1)n(n-1) = 2(q-1)\binom{n}{2}$ such operations.
2. Interchanging two rows of M. There are $\binom{n}{2}$ such operations.
3. Multiplying a row by a non-zero, non-identity constant. There are $(q-2)n$ such operations.

Each of these operations can also be expressed in terms of multiplying the matrix M with an elementary matrix from $\mathrm{GL}(n,q)$. Let us denote the set of elementary matrices from $\mathrm{GL}(n,q)$ by $S(n,q)$, or simply S when there is no risk for confusion. We have that $|S| = \mathcal{O}(qn^2)$

We can now consider the Cayley graph $\mathrm{Cay}(\mathrm{GL}(n,q),S)$ for our chosen set S of generators. (See [11] for definitions and general facts about Cayley graphs.) A series of row operations correspond exactly to a walk on this Cayley graph and our aim is to find a short path to the vertex corresponding to the identity matrix. Note that since Cayley graphs are vertex transitive the problem of finding a shortest path between two vertices can always be reduced to that of finding a shortest path from a general vertex to the identity vertex.

Given a matrix M let $\mathcal{D}(M)$ denote the smallest number of row operations needed to reduce M to the identity matrix I,

$$\mathcal{D}(M) = \min\{k \mid \exists E_1,\ldots,E_k \in S \text{ such that } E_1\cdots E_kM = I\},$$

or equivalently the distance between the two vertices in the Cayley graph. We define the *complexity* of row reduction for matrices in $\mathrm{GL}(n,q)$ to be

$$\mathcal{D}(n,q) = \max_{M \in \mathrm{GL}(n,q)} \mathcal{D}(M),$$

which in turn is the radius, and diameter, of the Cayley graph.

2 Lower bounds for the diameter

Before considering matrices from $\mathrm{GL}(n,q)$ let us first note that Gaussian elimination is optimal for matrices from $\mathrm{GL}_n(\mathbb{C})$. To see this we note that the set of matrices which can be expressed as a product of k elementary matrices form a variety of dimension k; see e.g. [6] for background on algebraic geometry. Since $\mathrm{GL}_n(\mathbb{C})$ cannot be written as a finite union of lower dimensional varieties, and has dimension n^2, we get our lower bound. In fact this short argument also shows that the set of matrices with complexity less than n^2 have measure zero.

2.1 Fixed q and growing n

We first find a lower bound on the diameter of $\mathrm{Cay}(\mathrm{GL}(n,q),S)$ for a fixed value of q. The proof is a simple enumerative calculation much in the style of the lower bounds for addition chains in e.g. [19].

Theorem 2.1. For a fixed value of q,

$$\mathcal{D}(n,q) \geqslant \frac{n^2}{2\log_q n + 1 + \log_q\left(A_{n,q}\left(1-\frac{1}{n}\right)\right)} + \mathcal{O}\left(\frac{1}{\log n}\right) \sim \frac{n^2}{2\log_q n}, \qquad (3)$$

where $A_{n,q} = \frac{n}{n-1} - \frac{n+3}{2q(n-1)}$

Proof. The degree r of the Cayley graph $\mathrm{Cay}(\mathrm{GL}(n,q),S)$ can be written as $r = qn(n-1)A_{n,q}$. In a free group with r generators, the number of elements generated by products of at most k generators is $\frac{r^{k+1}-1}{r-1} = r^k B_{r,k}$ where $1 < B_{r,k} < 2$. This is clearly an overestimate in our case, since our Cayley graph has quite a lot of cycles, but it will be sufficient to get our bound.

If $\mathcal{D}(n,q) < k$ then $r^k B_{r,k} > q^{n^2} C(n,q)$. Taking the logarithm of this inequality, for our r, and simplifying we get

$$k\left(2\log n + \log\left(q\left(1-\frac{1}{n}\right)A_{n,q}\right)\right) + \log B_{r,k} > n^2 \log q + \log C(n,q),$$

$$k > \frac{n^2}{2\log_q n + 1 + \log_q\left(A_{n,q}\left(1-\frac{1}{n}\right)\right)} + \frac{\log C(n,q) - \log B_{r,k}}{2\log n + \log\left(qA_{n,q}\left(1-\frac{1}{n}\right)\right)},$$

and since $\log C(n,q)$ is bounded the theorem follows. □

We thus find that the lower bound for the complexity of matrix reduction over a finite field differs from that over $\mathbb{C}$. This could of course be a result of our rather crude proof method but as we will see in the next section this is not

the case. Our upper bound for the complexity differs only by a multiplicative constant.

Given the very simple proof of this bound it is natural to ask if the result can be sharpened. In the proof we treat $\mathrm{GL}(n,q)$ as if it were a free non-commutative group with the same number of generators, but for large n most elements of $S(n,q)$ commute with each other, and even those elements that do not typically satisfy some small nontrivial identity. In the Cayley graph view of $\mathrm{GL}(n,q)$, these small relations correspond to short cycles in the graph. In order to improve the bound on the diameter one would wish to reduce the estimate on the number of distinct elements at distance k by taking these cycles into account. A quick calculation shows however that it does not suffice to consider only cycles of length $\mathcal{O}(1)$, as in that case there is an improvement but not in the dominant term of the bound. Using cycles of a length that grows with n complicates matters significantly and we will leave this potential improvement as an open problem.

Another open problem here is to prove that $\mathcal{D}(n,q)$ is monotonely increasing in n. This seems intuitively obvious, and as we shall see later is true for small n and q. A good start is of course to observe that the map

$$A \mapsto \begin{pmatrix} A & 0 \\ 0 & 1 \end{pmatrix} : \mathrm{GL}(n,q) \longrightarrow \mathrm{GL}(n+1,q)$$

is an embedding (as an induced subgraph) of one Cayley graph into the next, as path lengths are preserved by this map and a shortest path from some $A \in \mathrm{GL}(n,q)$ to the identity remains a shortest path in the image of the embedding. Unfortunately we have not yet found a proof ruling out the possibility that there is a path from the vertex $\left(\begin{smallmatrix} A & 0 \\ 0 & 1 \end{smallmatrix}\right)$ to the identity which does not stay within the embedded $\mathrm{GL}(n,q)$ and is shorter than any path that does. In the related problem of addition chains (see Subsection 3.3), it turns out that an increased workspace actually may make a difference for the problem complexity, but we suspect that one extra row in the matrix will not be sufficient for that effect to arise.

2.2 Fixed n and growing q

For a fixed value of n and growing q we find a quite different situation. Here there is an upper bound of n^2 operations, provided by ordinary Gauss–Jordan elimination, and as we shall see this complexity will be reached once q is large enough. As q continues to increase the situation will become even more like that for matrices over $\mathbb{C}$ as we reach a value of q beyond which a majority of matrices will require n^2 operations. In the theorems below our bounds are general integers whereas q must be a prime power, however by the sharper versions of Bertrand's postulate, see e.g. [15], there is always a prime power close to the bound.

Theorem 2.2. For $q \geqslant \left(3\binom{n}{2}\right)^{n^2-1}$ and $n \geqslant 3$ more than $\frac{1}{2}|\mathrm{GL}(n,q)|$ of the matrices M in $\mathrm{GL}(n,q)$ have $\mathcal{D}(M) = n^2$.

More generally, if $n^2 - a \geqslant 2$ and $\left(3\binom{n}{2}\right)^{\frac{n^2}{a}-1} \leqslant q$ then more than $\frac{1}{2}|\mathrm{GL}(n,q)|$ of the matrices M in $\mathrm{GL}(n,q)$ satisfy $\mathcal{D}(M) \geqslant n^2 - a$.

Proof. Let us to the contrary assume that at least half the elements in $\mathrm{GL}(n,q)$ can be written as products of at most k generators. As in the proof of Theorem 2.1 we get

$$B_{r,k}(2q)^k\binom{n}{2}^k(A_{n,q})^k > \frac{1}{2}q^{n^2}C(n,q)$$

We can now divide both sides by q^k and, letting $a = n^2 - k$, we get

$$2\frac{B_{r,k}}{C(n,q)}2^{n^2-a}\binom{n}{2}^{n^2-a}(A_{n,q})^{n^2-a} > q^a.$$

Taking an ath root gives

$$\left(2\frac{B_{r,k}}{C(n,q)}\right)^{1/a}2^{\frac{n^2}{a}-1}(A_{n,q})^{\frac{n^2}{a}-1}\binom{n}{2}^{\frac{n^2}{a}-1} > q$$

and finally, using $n \geqslant 3$ and $q \geqslant 2$ to bound $A_{n,q}, B_{r,k}$ and $C(n,q)$, we find that if $n^2 - a \geqslant 2$ then

$$3^{\frac{n^2}{a}-1}\binom{n}{2}^{\frac{n^2}{a}-1} > q$$

Setting $a = 1$ in the last inequality shows that if $q \geqslant 3^{n^2-1}\binom{n}{2}^{n^2-1}$ then the $n^2 - 1$ first levels cannot contain half the vertices of the graph. □

In the final step of the proof we could instead of bounding the base of the exponential part by 3, have used a slightly smaller function depending on n and q, but since this bound is unlikely to be very good we preferred this simpler form instead.

In view of the last result it is natural to make the following definitions.

Definition 2.3. Let $q_s(n)$ denote the smallest prime power such that if $q \geqslant q_s(n)$ then there are at least $\frac{1}{2}\big|\mathrm{GL}(n,q)\big|$ matrices M in $\mathrm{GL}(n,q)$ with $\mathcal{D}(M) = n^2$. Let $q_a(n)$ be the smallest prime power such that if $q \geqslant q_a(n)$ then $\mathcal{D}(n,q) = n^2$.

Clearly $q_s(n)$ is larger than $q_a(n)$, so our previous bound holds for $q_a(n)$ as well. Our next step will be to find a lower bound for q_s.

Theorem 2.4. For all n,

$$q_s(n) > \frac{3}{2}\binom{n}{2} \tag{4}$$

Proof. We will prove the bound by demonstrating that for $q \leqslant \frac{3}{2}\binom{n}{2}$ the set Z of invertible matrices with a zero entry outside the main diagonal satisfies $|Z| \geqslant \frac{1}{2}\big|\mathrm{GL}(n,q)\big|$. A matrix from the set Z can be reduced to the identity matrix by gaussian elimination using at most $n^2 - 1$ row operations, since the zero entry already has the right value.

Let A_0 denote the set of matrices with no zero entries outside the main diagonal and let A_1 be the set of singular matrices. We have that

$$\begin{aligned}|Z| = |\mathbb{F}_q^{n\times n}\backslash A_0\backslash A_1| \geqslant q^{n^2} - |A_0| - |A_1| = q^{n^2} - q^n(q-1)^{n^2-n} - q^{n^2}\big(1 - C(n,q)\big) = \\ = q^{n^2}C(n,q) - q^{n^2}(1-1/q)^{n^2-n} = q^{n^2}\left(C(n,q) - (1-1/q)^{n^2-n}\right)\end{aligned} \tag{5}$$

In order to make Z as large as required we thus need to find q such that

$$C(n,q) - (1-1/q)^{n^2-n} \geqslant \frac{1}{2}C(n,q)$$

or equivalently

$$\frac{1}{2}C(n,q) \geqslant (1-1/q)^{n^2-n}. \tag{6}$$

The bound (4) is trivial for $n < 3$. For $q = 2$ and $n \geqslant 3$ the right hand side of (6) is bounded from above by $2^{-6} < 0.02$ whereas the left hand side is bounded from below by $C(\infty, 2) > 0.25$. For $q \geqslant 3$ we have $C(\infty, q) > 0.56$ and thus the inequality will be satisfied if

$$0.28 \geqslant (1-1/q)^{n^2-n}.$$

Taking logarithms we get $\ln 0.28 \geqslant (n^2-n)\ln(1-1/q)$, and this in turn follows from $\ln 0.28 \geqslant -(n^2-n)/q$. We observe that $-1/\ln 0.28 > 0.75$ and thus (6) is fulfilled whenever $q \leqslant \frac{3}{4}(n^2-n) = \frac{3}{2}\binom{n}{2}$. Therefore $q_s(n) > \frac{3}{2}\binom{n}{2}$. □

By considering larger singular submatrices we have sketched a proof for a result of the form: For $n \geqslant N_k$ we have that $q_s(n) \geqslant \Omega(n^k)$. However, it seems that the proof would be lengthy, require technical assumptions, and the result is in our opinion likely to be far from optimal, so we do not include this extension.

The gap between the two bounds given here, going from $\mathcal{O}(n^2)$ to $\mathcal{O}\left(n^{2n^2-2}\right)$, is enormous and it would be desirable to find bounds which are more comparable, without complicated technical assumptions. Another property that we currently do not a have a proof for is that $\mathcal{D}(n,q)$ is monotone in q. Once again this seems intuitively obvious.

2.3 Concentration

For matrices over $\mathbb{C}$ we have seen that almost all matrices have a complexity of n^2, and that the set of matrices with complexity k only have dimension k, giving us a very strong form of concentration for the complexity of a random matrix. For matrices over $\mathbb{F}_q$ Theorem 2.2 gives a similar type of result for a fixed n and sufficiently large values of q. However when q is fixed and we consider larger matrices Theorem 2.2 no longer tells us anything. Using the Azuma–Hoeffding inequality we can prove another concentration result. See [14] for a nice treatment of Doob-martingales and the Azuma–Hoeffding inequality.

Theorem 2.5. Let M be a matrix taken uniformly at random from $\mathrm{GL}(n,q)$ and let $\mathrm{E}(n,q)$ be the expected value of $\mathcal{D}(M)$. Then for all $t \geqslant 0$,

$$\Pr\left(\left|\mathrm{E}(n,q) - \mathcal{D}(M)\right| \geqslant tn^{\frac{3}{2}}\right) \leqslant 2e^{-t^2/2}. \tag{7}$$

Proof. First let us note that if two matrices M and M' differ in exactly one row then their complexities can differ by at most n, since one can be transformed to the other using at most n row-operations. Thus we find that the complexity is a Lipschitz-function with Lipschitz constant n.

Let us now consider the random variables X_k obtained by conditioning $\mathcal{D}(M)$ on the values of the first k rows in the random matrix M. The sequence $X_0, \ldots, X_n$ now form a Doob-martingale with Lipschitz constant n, and applying the Azuma–Hoeffding inequality we get our theorem. □

From the proof of Theorem 2.1 and the upper bound in the next section we can also conclude that $\mathrm{E}(n,q) = \Theta\left(n^2/\log_q n\right)$. Thus the complexity of a random matrix from $\mathrm{GL}(n,q)$ is again strongly concentrated around its expectation. A natural problem would be to determine

$$E_q = \lim_{n\to\infty} \frac{E(n,q)\log_q n}{n^2},$$

for a fixed q, if this limit exists. The bounds of this and the following section imply that

$$\frac{1}{2} \leqslant \liminf_{n\to\infty} \frac{E(n,q)\log_q n}{n^2} \leqslant \limsup_{n\to\infty} \frac{E(n,q)\log_q n}{n^2} \leqslant 1.$$

3 An algorithmic upper bound: Striped matrix elimination

When reducing an invertible matrix A over some finite field $\mathbb{F}_q$ to the identity, it is possible to outperform the Gauss–Jordan algorithm by carefully choosing the row operations to eliminate several matrix elements at each step. The asymptotic performance is within a factor 2 of the theoretical lower bound $n^2/\left(2\log_q n\right)$ row operations.

The fundamental difference between the new algorithm and the classical Gauss–Jordan algorithm is that where the latter proceeds with one column at a time, the former at the same level processes a 'stripe' of columns; hence the 'striped' above. The width s of these stripes is a parameter that needs to be tuned for optimal performance, but for $s \sim \log_q n$ one can get the performance quoted above. What one uses is basically that there can be at most q^s distinct subrows within a stripe of width s, so if the matrix side $n \gg q^s$ then all matrix elements of the other $n - q^s$ rows that fall within a particular stripe can be eliminated in only $n - q^s$ row operations. This reduces the amount of work involved approximately by a factor s, from s operations per row and stripe to 1 operation, and thus leads to the quoted asymptotic complexity of $n^2/\log_q n$. Although this argument is not a proof, it explains the basic reason why the goal is attainable.

3.1 Row operations description

For a first more rigorous analysis and description, we will restrict ourselves to counting full-width elementary row operations. Each of these require n field operations, and will turn out to be dominant also in a more careful analysis. The two parameters n (matrix side) and s (stripe width) are given and may be arbitrary. The outermost loop in the algorithm is over the stripes of the matrix in pretty much the same way as the outermost loop in the Gauss–Jordan algorithm is over the columns of the matrix. A high level description of the algorithm can be given as:

For the processing of stripe k, which consists of columns $s(k-1)+1$ through sk, the input consists of an invertible matrix A which has $a_{i,i} = 1$ for $1 \leqslant i \leqslant sk - s$ and $a_{i,j} = 0$ for all $1 \leqslant i \leqslant sk - s$ and $j \neq i$, i.e., the diagonal elements

Algorithm 1 Striped Matrix Reduction

```
1: procedure REDUCE(A)                          ▷ A is a n × n matrix.
2:     Block A into stripes of s consecutive columns.
3:     for all stripes A_k but the last do
4:         REDUCESTRIPE(A, k)
5:     end for
6:     GAUSSIANELIMINATION(the last stripe of A)
7: end procedure
```

of the previous stripes are all 1 and the off-diagonal elements of the previous stripes are all 0.

$$\begin{pmatrix} 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

This matrix is to be transformed via elementary row operations to such a state that also the kth stripe has all ones on the diagonal and is zero otherwise.

The first step of processing stripe k is to apply elementary row operations so that the submatrix of rows and columns $sk - s + 1$ through sk is reduced to the identity. Provided that this submatrix is invertible, this can be done using the ordinary Gauss–Jordan algorithm in at most s^2 row operations, but it may become necessary to pivot in some other rows into the $sk-s+1$ through sk range to ensure that this submatrix is invertible. The maximal cost for those pivots is another s row operations. Thus $s^2 + s$ row operations suffice for transforming the above matrix to this:

$$\begin{pmatrix} 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 1 & 0 & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 1 & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

The second step of processing stripe k is where most of the work is done. Here a fixed row is picked as "cursor row"; for simplicity we may assume that the cursor row is row n. The remaining subrows in this stripe, i.e., 1 through $sk - s$ and $sk+1$ through $n-1$, will be zeroed by subtracting the cursor row from them, at a cost of one row operation for each of these $n - s - 1$ rows. Obviously this would not work if the value of the cursor row stayed the same throughout, and it is quite possible that all the q^s distinct subrows of s elements from $\mathbb{F}_q$ occur somewhere in this stripe, so during the second step the cursor subrow will have to assume all the q^s distinct values a subrow can assume. Each row operation which changes the cursor row will be to add some multiple of a row in the range

$sk - s + 1$ through sk to it, and matters can be arranged so that each of the $q^s - 1$ transitions from one of the q^s states of the cursor row to the next requires only one such row operation, giving a total of $n - s - 1 + q^s - 1 = n + q^s - s - 2$ row operations for the second step.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \cdot & \cdot & \cdot & \cdot \\ 0 & 1 & 0 & 0 & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 1 & 0 & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 1 & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

In the third and final step, the part of the cursor row that falls within the stripe is zeroed. This requires at most s row operations. The grand total for processing a stripe is thus $n + q^s + s^2 + s - 2$ row operations.

At the end of the matrix, the second step above cannot be carried out because the suggested cursor row n may be one of the $sk - s + 1$ through sk that should be reduced in the first step and then remain fixed throughout the second and third. The easy way to deal with this is to use the above striped procedure for stripes 1 through $k = \lfloor (n-1)/s \rfloor$ and then the ordinary Gauss–Jordan algorithm for the remaining $n - ks \leqslant s$ columns. The cost for this is at most $n(n - ks) \leqslant ns$ row operations. The total number of full-width elementary row operations required for the algorithm is thus

$$T(n,s) = \left\lfloor \frac{n-1}{s} \right\rfloor (n + q^s + s^2 + s - 2) + n\Big(n - s\lfloor (n-1)/s \rfloor\Big) \leqslant$$
$$\leqslant \frac{n^2 + nq^s}{s} + 2ns + n.$$

Since much of the point of the algorithm is to make use of repetitions of subrows, it seems silly to make $s > \log_q n$, and indeed for $s \sim (1+\varepsilon)\log_q n$ for any $\varepsilon > 0$ one would find that $nq^s/s = \mathcal{O}(n^{2+\varepsilon}/\log_q n)$, which is worse than the Gauss–Jordan algorithm. However for $s = \lfloor \log_q n \rfloor$ one gets

$$T\big(n, \lfloor \log_q n \rfloor\big) \leqslant \frac{n^2 + nq^{\lfloor \log_q n \rfloor}}{\lfloor \log_q n \rfloor} + 2n\lfloor \log_q n \rfloor + n \leqslant$$
$$\leqslant \frac{2n^2}{\log_q n - 1} + 2n \log_q n + n \sim \frac{2n^2}{\log_q n}.$$

This is already within a factor 4 of the theoretical lower bound, but it is possible to do even better by choosing s so that $q^s \sim n/\log_q n$, e.g. a suitably rounded

$\log_q n - \log_q \log_q n$. In this case

$$\begin{aligned}
T(n,s) &\leqslant \frac{n^2}{s} + \frac{nq^s}{s} + 2ns + n = \\
&= \frac{n^2}{\log_q n} \frac{1}{1 - \frac{\log_q \log_q n}{\log_q n}} + \frac{n^2}{\log_q n \,(\log_q n - \log_q \log_q n)} + \mathcal{O}(n \log_q n) = \\
&= \frac{n^2}{\log_q n} \Big(1 + \mathcal{O}\left(\tfrac{\log_q \log_q n}{\log_q n}\right)\Big) + \mathcal{O}\bigg(\frac{n^2}{(\log_q n)^2}\bigg) = \\
&= \frac{n^2}{\log_q n} + \mathcal{O}\bigg(\frac{n^2 \log_q \log_q n}{(\log_q n)^2}\bigg) \sim \frac{n^2}{\log_q n},
\end{aligned}$$

improving the constant of the asymptotically dominant term by half of that factor 4 to a mere 2 from the theoretical lower bound.

3.2 Full description

A full complexity analysis must also include the decision of which row operations to apply, and we also want to make sure that this can be done in an efficient way so that the total number of operations is dominated by the cost of the row operations. In order to do that we will in this section consider the time complexity of all parts of the algorithm rather than just the number of row operations.

The most mysterious part of the last section, in terms of decisions, is probably the second step, where one has to find a route through the space of possible subrows that never intersects itself and furthermore know which rows have stripe subrows equal to the current cursor subrow. There is however a simple solution to this.

Beginning with the problem of finding the route, one may observe that the problem is to visit all the q^s vectors with s elements from a set of size q, without repetitions, and without being allowed to change more than one vector element at each step. These are the restrictions that a Gray code counter has to satisfy, so one may simply let the route be given by a Gray code. See [10] for a thorough survey of Gray codes. Suppose $G_e \colon \{1, \ldots, q^s\} \longrightarrow \mathbb{F}_q^s$ is some Gray encoding function and $G_d \colon \mathbb{F}_q^s \longrightarrow \{1, \ldots, q^s\}$ the corresponding decoding function G_e^{-1}. Let us also define a function $f(l)$ as $f(l) = l + q^s$ if $l < G_d(a_{n,sk-s+1}, \ldots, a_{n,sk})$ and $f(l) = l$ otherwise. This function will be used to shift the Gary code appropriately. The reduction of a stripe in the high level description of our algorithm can now be carried out as described in Algorithm 2.

The G_d function can be computed in $\mathcal{O}(s)$ time, so line 5 has complexity $\mathcal{O}(ns)$. The input for the sort in line 6 has size $\mathcal{O}(ns)$, so its time complexity is $\mathcal{O}\big(ns \log(ns)\big) = \mathcal{O}(ns \log n)$. Since the row operation on line 13 is executed least $n - s - 1$ times and these row operations have time complexity $\mathcal{O}(n)$, it is clear that the corresponding $\mathcal{O}(n^2)$ in loop 7 dominates the two previous steps.

Since the loop on line 8 may perform s row operations and line 13 never more than one, it may appear as though the loop 8 inside loop 7 should be what dominates the complexity of the above, but thanks to the Gray code the average number of row operations performed by loop 8 is one or less. This can be seen by observing that that the Hamming distance between $G_e(l_1)$ and $G_e(l_2)$ is always bounded from above by $|l_1 - l_2|$.

Algorithm 2 Reduce stripe

1: **procedure** ReduceStripe(A, k) ▷ $a_{i,j}$ denotes the current element in position (i, j) of the matrix A.
2: Find s linearly independent rows in stripe k.
3: Pivot those rows into rows $s(k-1)+1$ through sk.
4: Apply Gaussian elimination to rows $s(k-1)+1$ through sk.
5: Form the list L of all pairs

$$\big(i, (f \circ G_d)(a_{i,s(k-1)+1}, \ldots, a_{i,sk})\big)$$

where $i \in \{1, \ldots, n-1\} \setminus \{sk-k+1, \ldots, sk\}$. ▷ f and G_d are defined on p. 11.
6: Sort the list L by the second element.
7: **for all** $(i, l) \in L$, in the order they appear **do**
8: **for all** j from $s(k-1)+1$ to sk inclusive **do**
9: **if** $a_{i,j} \neq a_{n,j}$ **then**
10: Add $a_{i,j} - a_{n,j}$ times row j to row n.
11: **end if**
12: **end for**
13: Add -1 times row n to row i.
14: **end for**
15: **for all** j from $s(k-1)+1$ to sk inclusive **do**
16: Add $-a_{n,j}$ times row j to row n.
17: **end for**
18: **end procedure**

There are many different Gray codes and if some structural information about the matrices one intends to reduce is available then this may be used to guide to choice of Gray code. A monotone Gray code could for example be used if the matrices are sparse, thus increasing the likelihood that the code will rapidly generate all present subrows. In practical implementations it may be convenient to choose the Gray code so that $G_e(1) = \mathbf{0}$ and use $f(l) = -l$ instead of the f specified above, as that will make the all zeroes vector $\mathbf{0}$ the final value for the cursor subrow and thus provides an easy condition for aborting the loop on line 7 when only rows for which nothing needs to be done remain; effectively this swaps the order of the second and third steps by unifying the loop on line 15 with the loop on line 8. This requires some care however, as there is no guarantee that there was any all zeroes subrow in the stripe to begin with; the loop on line 8 only changes the cursor subrow value to match values of other subrows actually present in the matrix. In no all zeroes subrow is present , the modified algorithm may actually end up using s row operations more than Algorithm 2.

Using a standard Gray code here is optimal for $q = 2$ but for larger q we can do even better. In the description above we construct each element of $\mathbb{F}_q^s$ and just subtract the given element from the rows in which it appears. However for $q > 2$ we can instead choose to subtract a multiple of the cursor row. This means that instead of having to let the cursor row assume all of the $q^s - 1$ nonzero subrow values it is sufficient to let it assume some multiple of each such value. In other words we can consider subrows which are multiples of each

other as being equivalent, which amounts to considering the elements of the finite projective space $\mathbb{P}^{s-1}_{\mathbb{F}_q}$ and seeking a Hamiltonian path through this set instead of $\mathbb{F}^s_q$. By doing this, the second term nq^s/s of the complexity estimate can be reduced to $n(q^s-1)/s(q-1)$, which is only $\frac{1}{q-1}$ of the previous value.

The other nontrivial decision in the algorithm is determining which rows to pivot in lines 2 and 3. One brute force method of doing this is to form the $(n-sk+s)\times s$ submatrix

$$\begin{pmatrix} a_{sk-s+1,sk-s+1} & \cdots & a_{sk-s+1,sk} \\ \vdots & \ddots & \vdots \\ a_{n,sk-s+1} & \cdots & a_{n,sk} \end{pmatrix}$$

and reduce it to row-echelon form with Gauss elimination, while keeping track of the permutations performed; the original s rows which are pivoted to the top s rows of this submatrix are then known to be linearly independent. The number of stripe-width row operations needed for this Gauss elimination is $\mathcal{O}(ns)$, using $\mathcal{O}(ns^2)$ field operations, which again is dominated by the $\mathcal{O}(n^2)$ field operations for the full-width row operations in the loop on line 7.

3.3 Optimality and a relation to addition chains

At the moment we do not know if this algorithm is optimal; in order to prove optimality we would have to improve the lower bound given earlier. There is a connection to a problem studied by several earlier authors which should be mentioned here. Let M be a $p_1\times p_2$ matrix with integer entries from $\{0,\ldots,N\}$. An addition chain for M is a sequence of vectors constructed by starting out with the zero vector and the p_2 standard unit vectors, then constructing each new vector as the sum of two earlier vectors and stopping once the p_1 row vectors of M are members of the sequence. This problem is now studied over the integers and not over a finite field but clearly an addition chain over $\mathbb{Z}$ also defines an addition chain over $\mathbb{Z}_p$ for each prime p.

Matrix reduction is not directly equivalent to addition chains but there is clearly a connection between the way our algorithm treats an individual stripe and an addition chain constructing the $n\times s$ submatrix defining the stripe from the s unit vectors in the final diagonal block. If the sequence of row operations performed on the stripe was done in reverse and all length s vectors so constructed were kept we would have an extended addition chain for the original stripe. Here extended addition chain means that we can also add a multiple of one vector to another.

Pippenger studied addition chains in several papers [17, 18, 19] and found the length of optimal chains for many ranges of the parameters. He used $L(p_1,p_2,N)$ to denote this length and found that for our relevant range of parameters,

$$L(p_1,p_2,1)\sim\frac{p_1p_2}{\log_2 p_1p_2}, \tag{8}$$

The cost in memory for an addition chain is the same as its cost in operations, plus the number of initial vectors. For an addition chain using at most n vectors in memory we find that $p_2=\mathcal{O}(\log n)$. The cost of constructing a $n\times\log_2 n$ stripe would thus be

$$L(n,\log_2 n,1)\sim\frac{n\log_2 n}{\log_2(n\log_2 n)}\sim n, \tag{9}$$

which turns out to be equivalent to the average cost per stripe for our algorithm on a matrix with elements from $\mathbb{F}_2$, given a good choice of s.

There are two differences between the pure addition chain problem and our treatment of individual stripes. First we are working over a field with non-zero characteristic, which in principle could make things easier for us, as any fast addition chain over $\mathbb{Z}$ can be adopted to fields of prime order but not necessarily vice versa. However for $q = 2$ Pippenger's lower bound proof works just as well in $\mathbb{Z}_2$ as it does for 0/1 matrices over $\mathbb{Z}$, so at least for $q = 2$ we cannot hope to improve on Pippenger's integer bound. For $p_1 = n$ and $\log n = o(p_2)$ we find that Pippenger's addition chains uses fewer, by a factor of 2, operations than our method but in this case they also use more than our restriction of n vectors in memory.

Second, we are actually more restricted than in the addition chain problem. When we process one stripe we must make sure not to undo our work in earlier stripes and we thus have an interaction between the stripes which is not present in the addition chain problem.

All in all we find that within the class of algorithms which perform reduction by processing a matrix one stripe a time, never returning to a previous stripe, our algorithm is optimal for $q = 2$.

3.4 A potential application: Integer factoring

In integer factoring algorithms such as the quadratic sieve, see e.g. [3], it is necessary to perform row reduction on certain very large matrices over $\mathbb{F}_2$ as one step of the algorithm. These matrices have a very distinct structure, being both very sparse and having most of their non-zero entries concentrated in the first $\sqrt{M}$ columns, where M is the total number of columns. Many different approaches to handling these matrices have been investigated, see [20] and [16]. These methods focus on handling the sparse part of the matrix in an efficient way, trying to save both memory and row operations by avoiding fill-in of the sparse part while it is processed. After these methods have processed the sparsest part of the matrix a dense part is left and is usually handled by ordinary Gauss–Jordan reduction. In [12] it is reported that for a certain, at the time very large, factoring instance a few hours of CPU time were used to reduce the sparse part of the matrix and that several weeks of CPU time were needed for the dense part.

Since our algorithm performs particularly well on dense matrices an interesting possibility would be to replace the Gauss–Jordan step in the last paragraph by striped matrix elimination. Conservatively estimating one multiplication and addition per microsecond, three weeks would correspond to a matrix with $n \approx 1.2 \cdot 10^4$, making $\log_2 n > 13$ and $s \approx 10$; a rather nice speed-up, and actually better than the vulgar estimate $n^{\log_2 8/7} \approx 6$ on what one may gain from Strassen's fast matrix multiplication for matrices of this size. If the overhead involved in our algorithm can be managed well in an actual implementation this would lead to a considerable speed-up of the processing of the dense part of these matrices.

4 The optimal results for small n and q

While our bounds for $\mathcal{D}(n,q)$ for a fixed q are quite good they are still far from giving us the exact values of $\mathcal{D}(n,q)$, and the bounds for growing q even more so. In order to determine the exact values for small n and q and get a more detailed picture of how the Cayley graph develops we have performed a computational investigation as well.

4.1 Experimental set-up

We wrote a C program which performs a breadth first search from the identity matrix. In the standard way the program keeps track of a 'state' of each vertex—whether the vertex is in the current level, the next level, some earlier level, or has not yet been seen—whereas there is no explicit representation of edges. In order to manage the larger graphs the vertex states were coded using only 2 bits of storage per vertex. This was done by allocating a large bit vector and then assigning to each matrix with nonzero columns a pair of consecutive bits within this vector; in Table 1 we can see the memory requirements for small n and q. As there are $q^n - 1$ different values for a nonzero column, a unique position for a matrix could be computed simply by interpreting its n columns as n digits in base $q^n - 1$. This use of column vectors also simplified implementing the row operations, as we could tabulate the function saying "row operation i transforms the column with number j to the column with number k" and thus compute the positions of all neighbours of a given vertex through elementary arithmetic and table look-ups.

	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$
$q=2$	<1	<1	<1	<1	15	$1.2 \cdot 10^5$
$q=3$	<1	<1	<1	194	$3.4 \cdot 10^7$	
$q=4$	<1	<1	<1	$2.6 \cdot 10^5$		
$q=5$	<1	<1	36			
$q=7$	<1	<1	7725			
$q=8$	<1	<1				
$q=9$	<1	<1				
$q=11$	<1	<1				
$q=13$	<1	3				
$q=16$	<1	16				
$q=17$	<1	28				
$q=19$	<1	76				
$q=23$	<1	420				
$q=25$	<1	889				

Table 1: Memory requirement for the Cayley graph in gigabytes.

As the search progresses the program outputs the number of vertices that belong to each distance class in the graph; these data are shown in tables 3–7. An unusual feature is that some graphs have one very large distance class (see e.g. Table 4) that can account for well over half the vertices of that graph. In these cases we could save a large amount of work by first searching forward and then backward. During the first phase (forward search) the program constructs

the next distance class by applying all generators in our set S to each element in the current distance class. During the second phase (backward search), which starts when a predetermined distance class K is reached, the program instead applies the generators to all matrices not yet encountered in order to see if they have a neighbour in the previously constructed distance class; in this phase the processing of a vertex stops as soon as a neighbour in the lower distance class is found.

Our program was also parallelised using OpenMP. The computations were performed on three different SGI Origin machines. The largest case was the computation for $\mathrm{GL}(3, 23)$ which in total used over 430 gigabytes of RAM, running on over 400 processors for several days and accumulating a run time of 4.1 CPUyears.

The main obstacle to proceeding to even larger graphs was the amount of RAM available. Our program will access different parts of the RAM in a very unpredictable way so a fast shared RAM is essential for this search, and few current computers have shared RAMs larger than 512 Gb.

4.2 Results

In Table 2 we have listed the diameters for all Cayley graphs which could be reached with the computational resources available to us. In tables 3–7 we have listed the sizes of the distance classes in the Cayley graphs, i.e., the number of vertices at a given distance from the identity matrix.

	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
$q=2$	2	4	7	10	13
$q=3$	3	6	9	12	
$q=4$	4	7	11		
$q=5$	4	7	11		
$7 \leqslant q \leqslant 23$	4	8			

Table 2: Diameter of the Cayley graph

As we can see from Table 2 the diameter is monotone in both n and q, as we expected, and we state this as a conjecture.

Conjecture 4.1. $\mathcal{D}(n, q)$ is monotone in both n and q.

If we look at the diameters of the graphs for $q = 2$ in Table 2 we find that the main term of our asymptotic upper bound $\frac{n^2}{\log_2 n}$ in fact agrees remarkably well with the exact values for small n, predicting $4, 5, 8, 10, 13$ as the first few diameters. The sizes of the distance classes also agrees well with our concentration result. For $n = 4, 5$, tables 5 and 6, we see an exponential like drop in the size of the distance class as we move away from the largest one.

If we assume that Conjecture 4.1 is true, Table 3 gives us a complete display of the phenomena expected to appear as q increases for a fixed n. For $q = 2, 3$ the diameter of $\mathcal{D}(2, q)$ is still less than 4. For $q = 4$ we find the first matrices requiring 4 row operations, and we have $q_a(2) = 4$. As q continues to increase the last distance class continues to grow and for $q = 13$ it contains more than half of the vertices, giving us $q_s(2) = 13$. We have performed computations for larger q than those shown in this table as well, and as q continues to increase

$n = 2$										
Level	$q = 2$	$q = 3$	$q = 4$	$q = 5$	$q = 7$	$q = 8$	$q = 9$	$q = 11$	$q = 13$	$q = 16$
0	1	1	1	1	1	1	1	1	1	1
1	3	7	11	15	23	27	31	39	47	59
2	2	23	54	103	239	326	431	679	983	1542
3		17	110	313	1249	2034	3161	6385	11257	22106
4			4	48	504	1140	2136	6096	13920	37492

Table 3: Size of the distance classes for $n = 2$

a larger and larger proportion of the vertices belongs to the last distance class, just as expected.

$n = 3$							
Level	$q = 2$	$q = 3$	$q = 4$	$q = 5$	$q = 7$	$q = 8$	$q = 9$
0	1	1	1	1	1	1	1
1	9	18	27	36	54	63	72
2	38	182	404	728	1634	2216	2900
3	78	1156	3968	9894	33968	53772	81058
4	42	4287	26046	93813	512307	952710	1671753
5		5130	92950	545628	5245120	11675814	24409482
6		458	57846	802306	21204546	63663690	171433796
7			198	35594	6785764	39018738	141869106
8					734	12708	187512

$n = 3$						
Level	$q = 11$	$q = 13$	$q = 16$	$q = 17$	$q = 19$	$q = 23$
0	1	1	1	1	1	1
1	90	108	135	144	162	198
2	4526	6512	10160	11564	14630	21842
3	158844	275006	536516	653178	930548	1700256
4	4152327	8702397	21299670	27845457	44776143	100461507
5	78654196	202545280	629821474	888997108	1623054880	4509563860
6	798714832	2706903968	11370766138	17699483800	37684754564	134745058836
7	1232098786	6628001012	47971230510	83878534274	228871419044	1215099678186
8	10492398	179983508	4163519396	8707753322	36587912588	364987070066

Table 4: Size of the distance classes for $n = 3$

So far the exact values all agreed well with our expectations, however the data for $n = 3$ came as a surprise to us. As q increases from 2 to 7 the diameter of the Cayley graph rapidly grows from 4 to 8, however once that diameter has been reached the graphs seem very reluctant to rise any further. The computation for $n = 3$ was pushed to higher and higher q in the hope of being able to find the value of $q_a(3)$, probably the last $q_a(n)$ for which this is computationally feasible, but as the table shows we have not succeeded. When q was increased a larger and larger part of the vertices was found in the last three, and later the last two, distance classes, but not a single vertex appeared at distance 9. It is quite possible that there is some underlying algebraic property preventing matrices at large distance when q is small, relative n, but so far we have not found one. It would be interesting to find shaprper bounds for $q_a(n)$. By Theorem 2.4 we

know that $q_s(3)$ must be at least 15, but here this lower bound seems to be much lower than the actual value.

For our larger n, tables 5–7, the graphs look similar to the asymptotic picture for a fixed q, i.e., we see a diameter noticeably less than n^2, the largest distance classes are several steps in from the extremal ones, and most matrices are concentrated close to the largest class.

$n = 4$				
Level	$q = 2$	$q = 3$	$q = 4$	$q = 5$
0	1	1	1	1
1	18	34	50	152
2	167	665	1439	1964
3	1010	9370	30318	49902
4	3918	100139	503842	966269
5	8572	794654	6654868	15407781
6	6301	4305691	67436357	203070282
7	173	12199038	470243499	2124066449
8		6778876	1567458540	16691702892
9		72652	845884998	75238909564
10			2886870	48695739370
11			18	148496842

Table 5: Size of the distance classes for $n = 4$

$n = 5$		
Level	$q = 2$	$q = 3$
0	1	1
1	30	55
2	475	1735
3	5230	40735
4	43004	775109
5	265000	12302561
6	1176535	162811445
7	3336505	1761590085
8	4334920	14842741840
9	837280	86149921538
10	380	245807452970
11		126205337589
12		623498577

Table 6: Size of the distance classes for $n = 5$

4.3 Extremal matrices

As mentioned in the introduction it would be of great interest to be able to construct an explicit family of $n \times n$ matrices which cannot be row reduced to the identity matrix in a linear number of row operations. So far *constructing* even a family which requires cn operations for some large constant c seems challenging, despite the fact that almost surely $\mathcal{D}(M_n) = \Omega(n^2/\log_q n)$ for $\{M_n\}_{n=1}^{\infty}$ any

Level	$n = 6\ q = 2$
0	1
1	45
2	1075
3	18195
4	240934
5	2589042
6	22779975
7	161946260
8	893603745
9	3517544498
10	8207684400
11	6816796888
12	535485765
13	18937

Table 7: Size of the distance classes for $n = 6$

infinite sequence where $M_n \in \mathrm{GL}(n, q)$ is chosen independently and uniformly at random.

We have not been able to construct a family of the kind described, but for small n and q our program can produce the full set of extremal matrices, i.e. matrices M such that $\mathcal{D}(M) = \mathcal{D}(n, q)$. In Table 8 we have listed a few examples for $q = 2$. For each size we have picked matrices with minimal and maximal number of non-zero entries. For both $n = 3$ and $n = 4$, but not for $n = 5$, we also found that $J - I$ is extremal, where J is the all ones matrix. Note that since $\mathcal{D}(M) = \mathcal{D}(M^{-1})$ the inverses of all these matrices are also extremal matrices.

So, two natural problems still remain

Problem 4.2. *Find an explicit construction for a sequence of matrices* $\{M_n\}$*, where* M_n *has side* n*, such that* $n = o\left(\mathcal{D}(M_n)\right)$

Problem 4.3. *What is the complexity of computing* $\mathcal{D}(M)$*?*

5 Matrices over semifields

As we have seen, one impediment to the experimental side of our work has been the lack of computers with enough RAM to handle the very large graphs that are involved when q grows, so we have examined also other ways to vary the base field. One is to consider other algebraic structures than fields as domains from which to fetch the matrix elements. Such a change of domain also helps elucidate to which extent properties of the elimination problem depend on the algebraic structure of the chosen domain, rather than just its size.

Row operations are defined in terms of addition and multiplication, so those two operations are indispensable, which means the thing replacing the field will at least have to be some kind of ring. Furthermore the Gauss–Jordan algorithm, which provides the constant upper bound on the graph diameter, requires that there exists multiplicative inverses, so the thing replacing $\mathbb{F}_q$ should still be some kind of field. The first generalisation of the (commutative) field concept

is the *skew field* or noncommutative division ring, but Wedderburn's theorem says all finite division rings are commutative, so we have to go even further to get anything new.

The next generalisation are the semifields [9], where one has abandoned also associativity. Finite semifields are plenty, but the only ones small enough to currently allow any experiments for $n > 2$ are those with 16 elements, which have been completely enumerated by Kleinfeld [8]. Our experiments here have not yet exhausted the possibilities, but they point to some interesting features.

5.1 Row operations and their graph

First it must be observed that the problem statement is no longer as straightforward as when one works over a field. The basic problem is that since a semifield T is not associative, matrix multiplication will not be associative either, and hence $\mathrm{GL}_n(T)$ is typically not a group. This does not prevent defining a "Cayley graph" whose set of vertices is $\mathrm{GL}_n(T)$ and where two vertices are adjacent if one is mapped to the other by an elementary row operation, so that is still what we do, but it means parts of the theory of Cayley graphs need no longer apply, e.g. one can no longer prove transitivity of these graphs simply by using multiplication (on the right) by a matrix as the explicit automorphism.

Having made that decision, there is next the problem of defining the elementary row operations. Since T is not commutative the operations of 'multiply from the left' and 'multiply from the right' are distinct, which in principle means one can close to double the number of row operations by considering the distinct left and right forms of the 'add multiple of row to row' and 'multiply row by' operations (this would not quite double their number though, since 1 still commutes

$n = 2$	$\begin{pmatrix}1&1\\1&0\end{pmatrix}\begin{pmatrix}0&1\\1&1\end{pmatrix}$
$n = 3$	$\begin{pmatrix}1&1&1\\1&0&0\\0&1&0\end{pmatrix}\begin{pmatrix}0&1&1\\1&0&1\\1&1&1\end{pmatrix}\begin{pmatrix}1&1&0\\1&1&1\\0&1&1\end{pmatrix}$
$n = 4$	$\begin{pmatrix}1&1&1&0\\1&1&0&1\\0&1&0&0\\1&0&0&0\end{pmatrix}\begin{pmatrix}1&1&1&0\\1&1&0&0\\0&0&1&1\\0&1&1&1\end{pmatrix}\begin{pmatrix}1&0&1&1\\1&1&0&1\\0&1&1&1\\1&1&1&1\end{pmatrix}$
$n = 5$	$\begin{pmatrix}0&1&0&0&1\\0&0&1&1&1\\1&1&0&1&1\\1&0&0&0&1\\0&1&1&0&0\end{pmatrix}\begin{pmatrix}1&0&1&1&1\\1&1&1&1&0\\1&1&1&1&1\\0&1&1&1&1\\1&1&1&0&1\end{pmatrix}\begin{pmatrix}1&1&1&1&1\\1&1&0&1&1\\1&1&1&0&1\\1&1&1&1&0\\1&0&1&1&1\end{pmatrix}$

Table 8: Some extremal matrices for $q = 2$

with everything and thus would have identical left and right forms). Here we have chosen to only include the left forms however, as these are the ones which can be represented as multiplication by an elementary matrix, even if that matrix multiplication is not associative. Note that this nonassociativity means that the composition of two row operations need no longer be representable as multiplication by the product of the two corresponding elementary matrices.

A more subtle issue arises with 'multiply row by' operations, in that the set of these no longer need to be closed under inversion. Over a field the inverse operation to 'multiply by r' is 'multiply by r^{-1}', but the proof that this returns an arbitrary element x to itself is $r^{-1}(rx) = (r^{-1}r)x = x$ which uses associativity, and hence it will not hold in a general semifield T. Again there is a possibility to add 'divide by r' as another class of elementary row operations, but for simplicity we have instead chosen to drop the 'multiply row by' operations entirely. Seemingly this should mean the set of vertices in our Cayley graph are more like some kind of $\mathrm{SL}_n(T)$ than a $\mathrm{GL}_n(T)$, but in fact the number of vertices is closer to what one would expect from $\mathrm{GL}_n(T)$, so the distinction between the two groups probably breaks down somewhere along the generalisation to semifields. The 'add multiple of row to row' operations do not similarly get into problems, because here it is addition that needs to be associative, which it still is.

Definition 5.1. Let $M(n, T)$ denote the graph whose set of vertices is the set of $n \times n$ matrices with elements from T and where two vertices are adjacent if there exists an 'add to row i the multiple $r \in T$ of row j' or 'interchange rows i and j' operation that changes one vertex into the other. Denote by $\mathrm{GL}_n^\dagger(T)$ the set of vertices in the component of $M(n, T)$ that contains the identity matrix I.

We have only studied $\mathrm{GL}_n^\dagger(T)$ for the five semifields (known as T_{24}, T_{25}, T_{35}, T_{45}, and T_{50}; for exact definitions of these we refer to [8]) of order 16 that are $\mathbb{F}_4$-algebras, but there it displays a very striking pattern.

Observation 5.2. For any $T \in \{T_{24}, T_{25}, T_{35}, T_{45}, T_{50}\}$,

$$\left|\mathrm{GL}_2^\dagger(T)\right| = 64260 = (q^2-1)(q^2-p), \tag{10}$$

$$\left|\mathrm{GL}_3^\dagger(T)\right| = 68367499200 = (q^3-1)(q^3-p)(q^3-p^2), \tag{11}$$

where $q = 16 = |T|$ and $p = 4 = |\mathbb{F}_4|$.

Proof. For (10), add up the columns in Table 9. For (11), do the same for Table 10. □

There exists a very direct interpretation of the products $\prod_{i=0}^{n-1}(q^n - p^i)$ as the number of n-tuples of elements of T^n that are linearly independent over $\mathbb{F}_4$. This indicates that this may be precisely an alternative characterisation of $\mathrm{GL}_n^\dagger(T)$, so we state this as a conjecture.

Conjecture 5.3. If T is a semifield with centre $K \neq T$, then for all $n \geqslant 2$,

$$\mathrm{GL}_n^\dagger(T) = \left\{A \in T^{n\times n} \,\middle|\, \text{the columns of } A \text{ are linearly independent over } K\right\}. \tag{12}$$

Below, we are able to prove this conjecture for $T \in \{T_{24}, T_{25}, T_{35}, T_{45}, T_{50}\}$, but leave it open for other semifields. It should be straightforward to verify for $|T| = q = 16$ and $n \in \{2, 3\}$, but is beyond the current computers for $n > 3$ or $n = 3$ and $q > 16$. Luckily, for any given semifield T, the conjecture can be decided from the case $n = 2$!

$n = 2$					
Level	T_{24}	T_{25}	T_{35}	T_{45}	T_{50}
0	1	1	1	1	1
1	31	31	31	31	31
2	478	478	478	478	478
3	6230	6140	5782	6336	6216
4	36093	37600	33095	38138	37116
5	21396	19994	24849	19268	20409
6	31	16	24	8	9

Table 9: Size of the distance classes for $n = 2$ and semifields

$n = 3$					
Level	T_{24}	T_{25}	T_{35}	T_{45}	T_{50}
0	1	1	1	1	1
1	93	93	93	93	93
2	5666	5666	5666	5666	5666
3	251292	251022	249660	251034	251826
4	9832323	9765156	9613359	9829032	9869298
5	370707801	360824106	351848817	371064842	372641521
6	11832755153	11321169174	11018482054	11831033570	11894787095
7	56036224680	56495526044	56734090686	56050421202	55980687767
8	117722191	179957938	253208864	104893760	109255933

Table 10: Size of the distance classes for $n = 3$ and semifields

5.2 Characterisation of reducible matrices

Why do we above require the *columns* to be linearly independent? Wouldn't it be more natural to use the condition that the *rows* must be linearly independent? It may certainly feel that way, but this latter condition in fact does not characterise $\mathrm{GL}_n^\dagger(T)$, as is easy to see. Let T and K be as above, and let $\alpha \in T \setminus K$ be arbitrary. Then the rows of the matrix

$$\begin{pmatrix} 0 & 1 \\ 0 & \alpha \end{pmatrix}$$

are linearly independent over K, and it is easy to see that no row operation can change a zero column to a nonzero column. As all columns of the identity I are nonzero, it follows that the above matrix is not in $\mathrm{GL}_n^\dagger(T)$.

What then about the transpose

$$\begin{pmatrix} 0 & 0 \\ 1 & \alpha \end{pmatrix}$$

of the above matrix—shouldn't that also be disqualified from belonging to $\mathrm{GL}_n^\dagger(T)$ by its zero *row*? Actually this need not be the case, as K-linear spans of elements of T^n need not be closed under multiplication by elements of $T \setminus K$. It is fairly easy to find a multiple of the $(1, \alpha)$ row that is K-linearly independent from $(1, \alpha)$ (any $T \setminus K$ multiple will do) and adding this multiple of the second row to the first row will make the two rows K-linearly independent. The

property of rows being K-linearly dependent or not is simply not preserved by row operations. The situation for columns is more familiar.

Lemma 5.4. Let T be a semifield with centre K. If $A \in T^{m\times n}$ is a matrix whose columns are linearly independent over K, then any matrix A' that arises from A by applying one elementary row operation also has columns that are linearly independent over K. Consequently the set

$$\left\{A \in T^{n\times n} \,\middle|\, \text{the columns of } A \text{ are linearly independent over } K\right\} \tag{13}$$

is closed under elementary row operations and

$$\mathrm{GL}_n^\dagger(T) \subseteq \left\{A \in T^{n\times n} \,\middle|\, \text{the columns of } A \text{ are linearly independent over } K\right\}. \tag{14}$$

Proof. Consider first how elementary row operations act on individual columns of a matrix. Multiplying a scalar $\alpha \in T$ by some matrix element $a_{i,j} \in T$ is a K-linear operation, so all elementary row operations are, when applied to a single column, K-linear operators $T^m \longrightarrow T^m$. As such the elementary row operations must preserve any linear dependency that exists between the columns of A. The elementary row operations are furthermore all invertible. Hence an elementary row operation can, when applied to the matrix A, neither destroy nor create any K-linear dependency between its columns.

It follows that the property of an element of $T^{n\times n}$ to have K-linearly independent columns is preserved by elementary row operations, and hence the set (13) of all matrices with this property is closed under such operations. Finally, $\mathrm{GL}_n^\dagger(T)$ is by definition the smallest set of matrices that contains the identity matrix and is closed under elementary row operations, whence it must be contained within the set (13) as (14) claims. □

From the counting results in Observation 5.2 it now follows that Conjecture 5.3 indeed holds for $T \in \{T_{24}, T_{25}, T_{35}, T_{45}, T_{50}\}$ and $n \in \{2, 3\}$, but the cases $n > 2$ can in fact always be reduced to the case $n = 2$.

Lemma 5.5. Let T be a semifield with centre K. If

$$\mathrm{GL}_2^\dagger(T) = \left\{A \in T^{2\times 2} \,\middle|\, \text{the columns of } A \text{ are linearly independent over } K\right\} \tag{15}$$

then equality in (12) holds also for $n > 2$. More precisely, if $k \geqslant 4$ and every matrix $A \in T^{2\times 2}$ with K-linearly independent columns can be transformed into the identity matrix by a sequence of at most k row operations, then every matrix $A \in T^{n\times n}$ with K-linearly independent columns can be transformed into the identity matrix by a sequence of at most $n^2 + \binom{k-1}{2}$ row operations.

Proof. The proof consists simply of applying a modified (to handle the extra situations brought up because T is a semifield) Gauss–Jordan elimination procedure to a matrix A. Thus one proceeds column by column, transforming diagonal elements to 1s and off-diagonal elements to 0s. Observe that the property of the matrix A to have columns that are linearly independent over K by Lemma 5.4 is an invariant of the process.

Suppose the current column is column $j < n$, i.e., $a_{i,i} = 1$ for all $i < j$ and $a_{i,l} = 0$ for all $i < j$ and $l \neq i$. There are then a number of possibilities for

what to do, depending on the contents of column j, and they will be described in order of increasing difficulty. If the diagonal element $a_{j,j}$ of the current column is already 1 then one can go ahead with zeroing any off-diagonal ($i \neq j$) element just as in ordinary Gauss–Jordan elimination, i.e., by adding to row i the multiple $-a_{i,j}$ of row j. This costs at most $n-1$ row operations for the entire column.

If $a_{j,j} \neq 1$ and $a_{i,j} \neq 0$ for some $i > j$ then one additional row operation suffices to transform this situation into the previous one, yielding a total cost of n operations for this column. This first row operation is to add to row j some multiple r of a row i, where $i > j$ is such that $a_{i,j} \neq 0$ and r is the semifield element that solves $ra_{i,j} = 1 - a_{j,j}$. In an ordinary field or skew field that r can be calculated as $(1 - a_{j,j})a_{i,j}^{-1}$, but this need not be the case in a semifield, because $(ra_{i,j})a_{i,j}^{-1}$ is not necessarily the same thing as r. Since the multiplication table of a semifield still is a latin square however, there will be an r which solves this equation, and hence there is an 'add multiple of row i to row j' operation that changes $a_{j,j}$ to 1.

Even the case that $a_{j,j} \notin \{0, 1\}$ and $a_{i,j} = 0$ for all $i > j$ is when $j \leqslant n-2$ possible to handle in at most n row operations. The trick is to first *manufacture* a nonzero $a_{j+1,j}$ by adding row j to row $j+1$ and then proceed as in the previous case; row $j+1$ is then changed twice, but on the other hand row $j+2$ is not changed so the total still cannot exceed n. However for $j = n-1$ this may cost $n+1$ row operations, which we have to account for when summing up the total below.

What remains for columns $j < n$ is the case that $a_{i,j} = 0$ for all $i \geqslant j$. Had T been a field then $T = K$ and this possibility would have been ruled out by the fact that column j is not in the linear span of columns 1 through $j-1$, but when T is not a field then it remains an important possibility, at least for $j > 1$, so what can be done? Since the K-linear span of columns 1 through $j-1$ is the set of columns with zeroes in rows j through n and elements from K in rows 1 through $j-1$, it follows that $a_{i,j} \in T \setminus K$ for some $i < j$. Consider the 2×2 submatrix consisting of rows and columns i and j:

$$\begin{pmatrix} 1 & a_{i,j} \\ 0 & 0 \end{pmatrix}$$

This has K-linearly independent columns, whence there is a sequence of at most k elementary row operations that transforms it into the identity. Applying the same sequence of elementary row operations to rows i and j of A thus leaves column i unchanged while setting $a_{j,j} = 1$ and $a_{i,j} = 0$, after which the rest of the column can be zeroed in at most $j-2$ operations. The total cost for this column is thus $j + k - 2$ operations, which is $\leqslant n$ for all $j \leqslant n - k + 2$.

The last column $j = n$ is special in that there of course does not exist a row $j+1$ which one may add back to row j to make $a_{j,j} = 1$ as above, but one may still rely on the condition about $\mathrm{GL}_2^\dagger(T)$ and consider the 2×2 submatrix consisting of rows and columns i and n for some suitable $i < n$. If $a_{n,n} = 0$ then one chooses i as above, and otherwise the choice of i is arbitrary. The total number of row operations required for column n is at most $n + k - 2$, and after completing these then A has been transformed into the identity matrix. The

total number of row operations performed is thus at most

$$\sum_{j=1}^{n-2} \max\{n, j+k-2\} + \max\{n+1, n-1+k-2\} + n + k - 2 = n^2 + \binom{k-1}{2},$$

where the assumption that $k \geqslant 4$ is used for the second term. □

Since the conditions in this lemma are fulfilled for $T \in \{T_{24}, T_{25}, T_{35}, T_{45}, T_{50}\}$, it is now established that Conjecture 5.3 is true for these semifields. Verifying this condition also for other semifields is probably rather easy, but we shall not do so here as that matter is more about structure theory for semifields than it is about row operations on matrices. It may furthermore be observed that the condition of Lemma 5.5 is clearly false for $T = \mathbb{F}_q$ whenever $q > 3$, as then the determinant $\det A = \pm 1$ for all $A \in \mathrm{GL}_n^\dagger(\mathbb{F}_q)$ despite there being invertible matrices with other values of $\det A$. If something like a determinant can be defined over semifields, then apparently that cannot be as predictably affected by row operations as the determinant over a field is.

The form of the bound in Lemma 5.5—n^2 plus a constant that depends only on T—is also strict enough for the complexity arguments in Section 3 to go through even for matrices over a semifield; the parts of the algorithm using Gauss–Jordan elimination of submatrices can instead be carried out as in Lemma 5.5 without any change in the order of complexity, and are thus still dominated by other parts of the algorithm. Hence the asymptotic result that the diameter is $\leqslant \big(1 + o(1)\big) n^2 / \log_q n$ continues to hold for $\mathrm{GL}_n^\dagger(T)$. While this result (as far as we can tell) does not have anything like the immediate computational applications of its counterpart over a field, it does remain a fairly tight bound on the diameter of a large family of explicitly constructible graphs.

Acknowledgements

The substantial amount of computer time used for the larger cases of the exact computational results were made possible by a special grant by NWO/NCF, The Netherlands, as a gesture in the European spirit and in remembrance of Anna Lind. We are also grateful for the computer time granted at NTNU, Trondheim, Norway and at NSC, Linköping, Sweden.

We would like to thank Daniel Bernstein for providing the reference to the original works of Pippenger. Finally we would like to thank our referee for numerous suggestions and prompt replies.

References

[1] J.R. Bunch, John.E. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Math. Comp.*, 28:231–236, 1974.

[2] P. Bürgisser, M. Clausen, M. Amin Shokrollahi. Algebraic complexity theory, volume 315 of Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. Springer-Verlag, Berlin, 1997.

[3] R. Crandall, C. Pomerance. Prime numbers, A computational perspective. Springer, New York, second edition, 2005.

[4] L. Euler. De mirabilis proprietatibus numerorum pentagonalium. Acta Academiae Scientarum Imperialis Petropolitinae, 4:56–75, 1783.

[5] W. T. Gowers. Rough structure and classification. Geom. Funct. Anal., (Special Volume, Part I):79–117, 2000. GAFA 2000 (Tel Aviv, 1999).

[6] J. Harris. Algebraic geometry, volume 133 of Graduate Texts in Mathematics. Springer-Verlag, New York, 1995.

[7] M. Kassabov, T.R. Riley. Diameters of cayley graphs of $SL_n(Z/kZ)$, 2005. http://arXiv.org:math/0502221.

[8] E. Kleinfeld. Techniques for enumerating Veblen-Wedderburn systems. J. Assoc. Comput. Mach., 7:330–337, 1960.

[9] D.E. Knuth. Finite semifields and projective planes. J. Algebra, 2:182–217, 1965.

[10] D.E. Knuth. The Art of Computer Programming, Volume 4, Fascicle 2 : Generating All Tuples and Permutations. Wiley-Interscience , New York, first edition, 2005.

[11] J. Lauri, Raffaele Scapellato. Topics in graph automorphisms and reconstruction. Cambridge University Press, Cambridge, 2003.

[12] A. K. Lenstra, H. W. Lenstra, Jr., M. S. Manasse, J. M. Pollard. The factorization of the ninth Fermat number. Math. Comp., 61(203):319–349, 1993.

[13] M. W. Liebeck, A. Shalev. Diameters of finite simple groups: sharp bounds and applications. Ann. of Math. (2), 154:383–406, 2001.

[14] R. Motwani, P. Raghavan. Randomized algorithms. Cambridge University Press, Cambridge, 1995.

[15] J. Nagura. On the interval containing at least one prime number. Proc. Japan Acad., 28:177–181, 1952.

[16] A. M. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. In Advances in cryptology (Paris, 1984), volume 209 of Lecture Notes in Comput. Sci., pages 224–314. Springer, Berlin, 1985.

[17] N. Pippenger. On the evaluation of powers and related problems (preliminary version). In 17th Annual Symposium on Foundations of Computer Science (Houston, Tex., 1976), pages 258–263. IEEE Comput. Soc., Long Beach, Calif., 1976.

[18] N. Pippenger. The minimum number of edges in graphs with prescribed paths. Math. Systems Theory, 12:325–346, 1979.

[19] N. Pippenger. On the evaluation of powers and monomials. SIAM J. Comput., 9:230–250, 1980.

[20] C. Pomerance, J. W. Smith. Reduction of huge, sparse matrices over finite fields via created catastrophes. Experiment. Math., 1:89–94, 1992.

[21] T. R. Riley. Navigating in the Cayley graphs of $\mathrm{SL}_N(\mathbb{Z})$ and $\mathrm{SL}_N(\mathbb{F}_p)$, 2005.

[22] J. H. van Lint, R. M. Wilson. A course in combinatorics. Cambridge University Press, Cambridge, second edition, 2001.

[23] A. Wigderson. Arithmetic complexity - a survey (lecture notes), 2001.