Efficient Generation of Random Nonsingular Matrices

Dana Randall *

Abstract

We present an efficient algorithm for generating an $n \times n$ nonsingular matrix uniformly over a finite field. This algorithm is useful for several cryptographic and checking applications. Over GF[2] our algorithm runs in expected time $M(n) + O(n^2)$, where $M(n)$ is the time needed to multiply two $n \times n$ matrices, and the expected number of random bits it uses is $n^2 + 3$. (Over other finite fields we use $n^2 + O(1)$ random field elements on average.) This is more efficient than the standard method for solving this problem, both in terms of expected running time and the expected number of random bits used. The standard method is to generate random $n \times n$ matrices until we produce one with nonzero determinant. In contrast, our technique *directly* produces a random matrix *guaranteed* to have non-zero determinant. We also introduce efficient algorithms for related problems such as uniformly generating singular matrices or matrices with fixed determinant.

1 Introduction

There are several algorithms that depend on efficient methods for generating random nonsingular matrices according to the uniform distribution. For example, the program checker that Blum, Luby and Rubinfeld use to check the rank of a matrix uses $2k$ random nonsingular matrices, where k is the confidence parameter [1]. Random nonsingular systems of linear equations are used in Shamir's knapsack-based cryptosystem in order to allow the receiver to decrypt messages [2]. A third application is the family of trapdoor functions provided in the McEliece Cryptosystem, where a random nonsingular matrix over GF[2] is needed to construct the public key [3].

We consider two resources for analyzing the efficiency of our algorithm: the total expected running time and the expected number of random bits. A simple algorithm that can be used to find a random nonsingular matrix is to randomly choose any matrix over the finite field and then test whether it has nonzero determinant. If we are unlucky and the matrix is not invertible, we reject it and repeat the process. Unfortunately this simple method is

*Research supported in part by NSF grant CCR88-13632 and in part by an AT&T Ph.D. Fellowship.
Computer Science Division, 571 Evans Hall, University of California at Berkeley, Berkeley, California 94720.

potentially inefficient, both in terms of time and of the number of random bits used. We can test whether the determinant of the matrix is nonzero in $O(M(n))$ time, where $M(n)$ is the time it takes to multiply two $n \times n$ matrices, but the constant is fairly large. This step dominates the running time of the algorithm. Also, each time we fail we need to generate another n^2 random field elements and in general random bits are difficult to generate and should be used sparingly. In GF[2], for example, the probability that a matrix is nonsingular is less than .3, so on average we calculate the determinant more than three times and we consume more than $3n^2$ random bits before the algorithm successfully terminates.

In this paper we present a more efficient technique for generating nonsingular matrices. There are two parts to the algorithm. The first part is probabilistic and on average takes $O(n^2)$ time. If we are working in a finite field $\mathcal{F}$, then the average number of random field elements we need is less than $n^2 + 3$. In particular, if we work over $GF[2]$ this gives an upper bound for the number of bits we need to generate a nonsingular matrix of 0's and 1's. An information theoretical lower bound shows that we require at least $n^2 - 1$ random field elements, so this is optimal to within an additive constant. The second part of the algorithm is a single matrix multiplication which takes time $M(n)$. Thus, the total expected running time is $M(n) + O(n^2)$.

2 Nonsingular Matrix Generation

2.1 Background and Notation

Let $\mathcal{M}^n$ represent the set of $n \times n$ nonsingular matrices. $\mathcal{M}^n_v$ is the set of $n \times n$ nonsingular matrices with first row v. If M is an $n \times n$ matrix, then $M[i,j]$ denotes the $(i,j)^{th}$-minor. Let e_r be the vector with a 1 in the r^{th} position and 0's elsewhere.

We use the notation "choose $a \in_u S$" to mean "choose a from the set S according to the uniform distribution." A running time of "$\mathbf{E}[\,\cdot\,]$" means "the *expected* running time is $[\,\cdot\,]$".

Let $\rho = |\mathcal{F}|$. The number of nonsingular $n \times n$ matrices is $(\rho^n - 1)(\rho^n - \rho)...(\rho^n - \rho^{n-1})$ which we can see by considering the nonsingular matrix as n linearly independent vectors in $\mathcal{F}^n$. There are $\rho^n - 1$ choices for the first vector in the space, since all nonzero vectors will work. The i^{th} vector chosen can be anything outside the subspace spanned by the first $i-1$ vectors, so there are $\rho^n - \rho^{i-1}$ choices. Thus, the probability that a random $n \times n$ matrix is nonsingular is:

$$\frac{\Pi_{i=1}^n (\rho^n - \rho^{i-1})}{\rho^{n^2}}.$$

We show by a simple information theoretical argument that at least $n^2 - 1$ random field elements are required to generate a random nonsingular matrix when random field elements are our only source of randomness. Taking the first two terms of the inclusion/exclusion expansion, we find the number of nonsingular matrices is greater than

$$\rho^{n^2}\left(1 - \sum_{i=0}^{n-1} \frac{\rho^i}{\rho^n}\right) = \rho^{n^2}\left(1 - \frac{\rho^n - 1}{\rho^n(\rho - 1)}\right)$$

$$> \rho^{n^2}\left(\frac{\rho-2}{\rho-1}\right).$$

Taking the log base ρ we find that we need n^2-1 random field elements, for $\rho > 2$.

The argument for $\rho = 2$ is similar. The number of nonsingular matrices over GF[2] is

$$\prod_{i=0}^{n-1}\left(2^n-2^i\right) = \left(2^{n-1}\right)\left(\prod_{i=0}^{n-2}(2^n-2^i)\right)$$

$$> \left(2^{n-1}\right)\left(2^{n(n-1)}\right)\left(1-\sum_{i=0}^{n-2}\frac{2^i}{2^n}\right) > \frac{2^{n^2}}{4},$$

which proves that we need at least n^2-1 random bits.

2.2 The Algorithm

The following lemmas show that to generate a random nonsingular matrix with first row v it is sufficient to uniformly generate a nonsingular matrix with first row e_r, where r is any coordinate where v is nonzero, and then multiply by a transformation matrix which takes e_r into v (i.e. the identity with the r^{th} row replaced with v). To uniformly generate a matrix with first row e_r we can pick random field elements for the rest of the r^{th} column, and choose the $(r,r)^{th}$ minor to be a random $n-1 \times n-1$ nonsingular matrix. This allows us to reduce the problem of uniformly generating an $n \times n$ nonsingular matrix to that of generating an $n-1 \times n-1$ nonsingular matrix and takes one matrix multiplication. Although this suggests an algorithm that requires n matrix multiplications, we will show that we can use the sparse structure of these transformation matrices to multiply them trivially, so we actually only need one matrix multiplication.

Lemma 1 *: Fix any $v \in \mathcal{F}^n \backslash \{0^n\}$, and let r be any coordinate where v is nonzero. Let T be the $n \times n$ identity matrix with the r^{th} row replaced by v. We define f_T as follows:*

$$f_T : \mathcal{M}^n_{e_r} \longrightarrow \mathcal{M}^n_v$$

$$f_T(M) = M * T.$$

Then f_T is a bijection.

Proof: Since T is nonsingular it has an inverse, T^{-1}. Define $g_T : \mathcal{M}^n_v \longrightarrow \mathcal{M}^n_{e_r}$, $g_T(M) = M * T^{-1}$. Then $g_T = f_T^{-1}$ and since g_T is defined on all of $\mathcal{M}^n_v$, this gives us the bijection. ■

This bijection shows that there are an equal number of nonsingular matrices with any fixed nonzero first row.

Lemma 2 : *For any r, we can uniformly generate an element $A \in \mathcal{M}^n_{e_r}$ using the following technique:*

1. *The first row of A is e_r.*
2. *For all $2 \leq i \leq n$, $A_{i,r} \in_u \mathcal{F}$.*
3. *Let the $(1,r)^{th}$-minor $= A'$, where $A' \in_u \mathcal{M}^{n-1}$ is an $n-1 \times n-1$ nonsingular matrix.*

Proof: Each matrix generated is nonsingular since $A_{1,r}$ is the only nonzero entry of the first row of A and therefore $\det(A) = \pm \det(A') \neq 0$.

If $A \in \mathcal{M}^n_{e_r}$ then $A[1,r]$ must be nonsingular so we can generate it by this method. Furthermore, since we pick the r^{th} row and the $(1,r)^{th}$ minor uniformly and each choice defines a unique matrix, we will generate each matrix with equal probability. ■

The algorithm that follows constructs matrices A and T recursively, where A is a nonsingular matrix that assigns some e_r to the top row of a minor at each stage of the iteration, and T is the linear transformation that maps this to a random nonsingular matrix over the field. At each stage we assign values to a row and a column of each matrix, and we call the function recursively on the two minors defined by these rows and columns until we have assigned all the elements of the matrices. The product $A * T$ is the output.

NonSing (n):
```
begin
     Generate (A, T, n);
     return (A * T).
end.
```

Generate (A, T, n):
```
begin
     if n = 1 then
          let A_{1,1} = 1;
          choose T_{1,1} ∈_u F\{0};
     else
          choose v ∈_u F^n\{0^n} such that v ≠ 0^n;
          let r be the first nonzero coordinate of v;
          /* Assign Values to A:  */
               let the first row of A = e_r;
               for all 2 ≤ i ≤ n, choose A_{i,r} ∈_u F;
          /* Assign Values to T:  */
               let the r^{th} row of T = v;
               for all 1 ≤ i ≤ n, where i ≠ r, T_{i,r} = 0;
          Generate (A[1,r], T[r,r], n-1).
end.
```

Choosing r to be the coordinate of the *first* nonzero entry of v is arbitrary, but this simplifies the transformation matrix T and enforces a canonical representation of the matrices A and T for each choice of v.

The efficiency of the algorithm comes from reducing the problem of generating random $n \times n$ nonsingular matrices to the problem of generating nonzero vectors of size k, for all $k \leq n$. We save random bits since each time we generate a vector which is identically zero we need only regenerate that one vector. In the standard algorithm where we generate $n \times n$ matrices until we find one with nonzero determinant, we have to regenerate the *entire* matrix each time we fail. Furthermore, it is easy to check whether a vector is identically zero, unlike determining whether the determinant of a matrix is zero, so we speed up the running time.

Theorem 1 *: The output $A * T$ of* `NonSing(n)` *is a random nonsingular matrix generated according to the uniform distribution. The number of random field elements used by the algorithm is $\mathbf{E}[n^2 + O(1)]$ and the running time is $M(n) + \mathbf{E}[O(n^2)]$.*

Proof: We show that our algorithm uniformly generates a nonsingular $n \times n$ matrix over $\mathcal{F}$ by induction on n.

If $n = 1$, then $a_{11} = 1$ and $t_{11} \in_u \mathcal{F} \backslash \{0\}$, so their product is a random nonsingular 1 x 1 matrix.

Applying the inductive hypothesis, we assume that we can uniformly generate $n-1 \times n-1$ matrices. We use `Generate` to generate matrices A and T and we want to show that their product is a random nonsingular matrix.

Since $T_{i,r} = 0$, for $i \neq r$, we can factor T into matrices T_1 and T_2 as follows: T_1 is the identity matrix with the $(r,r)^{th}$-minor replaced by the $(r,r)^{th}$ minor of T. T_2 is T with the $(r,r)^{th}$-minor replaced by the identity matrix I_{n-1}. Then $T_1 * T_2 = T and A * T = A * T_1 * T_2$.

Because the first row of A is e_r, the product of A and T_1 is the matrix A with the $(1,r)^{th}$-minor replaced by $A[1,r] * T[r,r]$, which we know from the inductive hypothesis to be a random $n-1 \times n-1$ nonsingular matrix. Therefore $A * T_1$ is a nonsingular matrix with first row e_r, and from lemma 2 we know that it is generated uniformly.

By lemma 1 $(A * T_1) * T_2$ is a random nonsingular matrix with first row v, and since v was chosen uniformly and is independent of $A * T_1$, $A * T$ is a random nonsingular matrix.

To analyze the expected number of random bits we consider the i^{th} stage of the algorithm. At this stage we choose $i-1$ random elements from $\mathcal{F}$ and one nonzero vector of length i. The total expected number of random elements this takes is

$$(i-1) + i\left(1 + \frac{1}{\rho^i} + \frac{1}{\rho^{2i}} + \ldots\right)$$

$$= (i-1) + i\left(\frac{\rho^i}{\rho^i - 1}\right).$$

Therefore the total expected number of random field elements needed by the algorithm is

$$\sum_{i=1}^{n}\left((i-1) + \frac{i\rho^i}{\rho^i - 1}\right)$$

$$< \frac{n(n-1)}{2} + \frac{n(n+1)}{2} + c_\rho$$
$$= n^2 + c_\rho,$$

where c_ρ is a constant dependent on ρ. In GF[2], $c_2 = 3$, and since a random field element is represented by a single bit, on average $n^2 + 3$ random bits are sufficient. For larger fields c_ρ will be smaller than 3.

This also gives us an upper bound of $\mathbf{E}[O(n^2)]$ for the expected time for `Generate` to generate matrices A and T, so the total running time of the algorithm is dominated by the matrix multiplication and takes time $M(n) + \mathbf{E}[O(n^2)]$. ■

3 Other Results

The algorithm can be easily modified to address several related problems.

Corollary 1.1 : *We can uniformly generate a matrix and its inverse in time* $2M(n) + \mathbf{E}[O(n^2)]$.

Proof: We can use `Generate` to generate A and T. T is upper triangular and A is easily decomposed into a lower triangular matrix and a permutation matrix. By inverting each of these matrices in time $O(n^2)$, we can construct $A * T$ and $T^{-1} * A^{-1}$ in time $2M(n) + \mathbf{E}[O(n^2)]$. ■

Corollary 1.2 : *We can uniformly generate matrices with given nonzero determinant. In particular, given any* $d \in \mathcal{F} \backslash \{0\}$ *we can uniformly generate an* $n \times n$ *matrix with determinant* d *in time* $M(n) + \mathbf{E}[O(n^2)]$.

Proof: This follows from the way we generate the matrices A and T. When we choose the 1×1 transformation matrix $T_{1,1}$ in the final call to `Generate` we can choose it to be $d * (\prod_{i=2}^{n}(T_{i,i})^{-1})$. The determinant of the matrix we generate is d and is chosen uniformly since there are exactly $|\mathcal{F}|-1$ choices for $T_{1,1}$ in each case and exactly one gives us determinant d. ■

The problem of generating a random singular matrix is only slightly more complicated than generating a random nonsingular matrix. We can determine the fraction of singular matrices with first row 0^n and flip a biased coin to assign the first row 0^n with this probability. If the first row is zero we randomly choose each of the remaining elements of the matrix from the field. If the first row is nonzero we use the same trick to reduce the problem to that of finding an $n-1 \times n-1$ singular matrix. The algorithm follows:

Sing (n):

```
begin
     Sing_Gen (A, T, n);
     return (A * T).
end.
```

Sing_Gen (A, T, n):

```
begin
     if n = 1 then
          let A_{1,1} = 0;
          let T_{1,1} = 1;

     else
          let t be the fraction of singular matrices
                    whose first row is 0^n;

          with probability t:
               let the first row of A = 0^n;
               for 2 ≤ i ≤ n, 1 ≤ j ≤ n, choose A_{i,j} ∈_u F;
               let T = I_n;

          with probability 1 - t:
               choose v ∈_u F^n\{0^n} such that v ≠ 0^n;
               let r be the first nonzero component of v;
               /* Assign Values to A: */
                    let the first row of A = e_r;
                    for all 2 ≤ i ≤ n, choose A_{i,r} ∈_u F;
               /* Assign Values to T: */
                    let the r^{th} row of T = v;
                    for all 1 ≤ i ≤ n, where i ≠ r, T_{i,r} = 0;
               Sing_Gen (A[1, r], T[r, r], n - 1).
end.
```

Theorem 2 : *The output $A * T$ of* Sing(A, T, n) *has determinant zero and is chosen uniformly from the set of singular matrices. The running time is $M(n) + \mathbf{E}[O(n^2)]$ and the number of random bits we use is $n^2 + \mathbf{E}[O(n)]$.*

Proof: We can easily determine the value of t. Let ρ be the size of $\mathcal{F}$. Then the number of nonsingular matrices of size n

$$|\mathcal{M}^n| = (\rho^n - 1)(\rho^n - \rho)...(\rho^n - \rho^{n-1}).$$

Then the number of singular matrices is

$$\sigma = \rho^{n^2} - |\mathcal{M}^n|,$$

and the fraction which have first row 0^n is

$$t = \frac{\rho^{n(n-1)}}{\sigma}.$$

The rest of the proof is similar to the proof of Theorem 1.

The expected number of random bits to simulated the biased coin in each iteration is constant, so the total number of random elements of the field necessary is $\mathbf{E}[n^2 + O(n)]$ and the expected running time is $M(n) + \mathbf{E}[O(n^2)]$. $\blacksquare$

References

[1] Blum, M., Luby, M., and Rubinfeld, R., "Self-Testing/Correcting with Applications to Numerical Problems", *STOC 1990.*

[2] McEliece, R., "A Public-Key Cryptosystem Based on Algebraic Coding Theory", *Deep Space Network Progress Report 42-44, Pasadena JPL, January 1978.*

[3] Shamir, A., "The Strongest Knapsack-Based Cryptosystem?", *Manuscript from The Weizmann Institute, Rehovot, Israel, August 1982.*