Signature of Symmetric Rational Matrices and the Unitary Dual of Lie Groups

[Extended Abstract]

Jeffrey Adams*
Department of Mathematics
University of Maryland
College Park, MD, 20742 USA
jda@math.umd.edu

B.David Saunders†
Department of Computer and Information Sciences
University of Delaware
Newark, DE, 19716 USA
saunders@cis.udel.edu

Zhendong Wan†
Department of Computer and Information Sciences
University of Delaware
Newark, DE, 19716 USA
wan@cis.udel.edu

ABSTRACT
A key step in the computation of the unitary dual of a Lie group is the determination if certain rational symmetric matrices are positive semidefinite. The size of some of the computations dictates that high performance integer matrix computations be used. We explore the feasibility of this approach by developing three algorithms for integer symmetric matrix signature and studying their performance both asymptotically and by experimenting on a particular matrix family constructed from the exceptional Weyl group E8. We conclude that the computation is doable, with a parallel implementation needed for the largest representations.

Categories and Subject Descriptors
G.4 [**Mathematical Software**]: Algorithm Design and Analysis; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*

General Terms
Algorithms, Performance

Keywords
matrix signature, symmetric matrix, lie group

1. INTRODUCTION
We propose, analyze, and test several algorithms for computing the signature and for verifying or disproving specific definiteness properties of symmetric integer matrices. The latter is sometimes easier than the signature. Also there is often a considerable difference in the cost of verifying (certifying) a property and of certifying it's negation. This is primarily an experimental paper whose purpose is to assess the feasibility of exact integer linear algebra methods for signature and definiteness determinations that may be desirable in the study of Lie group representations. The combination of measurements and analysis of asymptotic growth rates of time and memory use we provide are for the purpose of predicting the cost of the larger computations of interest, so as to determine the most promising algorithms and the hardware resource needs. The algorithmic asymptotic complexities reported here are largely straightforward applications of known results. We have an interesting observation which allows efficient binary tree structured Chinese remaindering to be used with an early termination strategy, giving practical and asymptotic speedup when using early termination. We have measured the time and memory costs of three algorithms on some matrices arising in group representations, studied their relative merits, and developed formulas to predict the costs for large matrices.

In section 2, the motivation for this work in the study of Lie Group representations is developed. This application creates large problems straining our limits of time and memory.

In section 3 we present the proposed algorithms and discuss their mathematical basis and complexity. Also included is some discussion of unimplemented alternatives and of the space issues for large instances of the problems. In section 4, we apply these algorithms to the matrices from Lie group representation.

Finally, in section 5, experimental results are reported and, in section 6, conclusions drawn on the state of these problems.

The signature of a real symmetric matrix A is generally defined as $\sigma = p - n$, the number by which positive eigenvalues outnumber negative ones. For our purposes, define the *signature* to be the triple $\sigma(A) = (p, z, n)$, where p is the number of positive eigenvalues, z is the multiplicity of zero as an eigenvalue, and n is the number negative. The triple $(\sigma(A), \mathtt{rank}(A), \dim(A))$ and $\sigma(A)$ convey the same information. By the symmetry, all eigenvalues are real. A matrix is

*Supported by NSF grant DMS 0200851.
†Supported by NSF grants CCR 0098284 and 0112807.

ISSAC '05, KLMM, Chinese Academy of Sciences, Beijing, China

positive definite if the signature has pattern is $(*, 0, 0)$ and positive semidefinite if the pattern is $(*, *, 0)$.

2. LIE GROUP REPRESENTATIONS AND UNITARY DUALITY

We assume some familiarity with root systems and Weyl groups, for example see [9]. We begin with a formal construction. Let R be a root system with Weyl group W. Thus R is a finite subset of $V = \mathbb{R}^n$ satisfying certain properties; in particular for each $\alpha \in R$ the reflection $s_\alpha \in \mathrm{Hom}(V, V)$ takes R to itself. By definition W acts on V. We may choose simple roots $S = \{\alpha_1, \ldots, \alpha_n\}$ so that W is generated by $\{s_i = s_{\alpha_i} \mid i = 1, \ldots, n\}$, with relations $\mathrm{order}(s_i s_j) = m_{i,j}$ for certain $m_{i,j} \in \{2, 3, 4, 6\}$. For $\nu \in V$ define $\langle \nu, \alpha^\vee \rangle = 2(\nu, \alpha)/(\alpha, \alpha)$. Let V^+ be the set of $\nu \in V$ which are dominant, i.e. $\langle v, \alpha_i^\vee \rangle \geq 0$ $(1 \leq i \leq n)$.

Fix a finite dimensional representation (ρ, V_ρ) of W. Thus $\rho : W \to \mathrm{Hom}(V_\rho, V_\rho)$ is a group homomorphism. Fix an invariant Hermitian form $(,)_\rho$ on V_ρ, i.e. satisfying $(v_1, v_2)_\rho = (\rho(w)v_1, \rho(w)v_2)_\rho$ for all $v_1, v_2 \in V_\rho, w \in W$. Choose a matrix J_ρ so that $(v_1, v_2)_\rho = v_1 J_\rho v_2^t$.

For $\alpha \in R$ and $\nu \in V^+$ we define

$$A_\rho(\alpha, \nu) = \frac{1 + \langle \nu, \alpha^\vee \rangle \rho(s_\alpha)}{1 + \langle \nu, \alpha^\vee \rangle} \in \mathrm{Hom}(V_\rho, V_\rho). \qquad (1)$$

Let w_0 be the longest element of W and choose a reduced expression $w_0 = s_{\alpha_{i_N}} s_{\alpha_{i_{N-1}}} \ldots s_{\alpha_{i_1}}$ $(1 \leq j \leq N,\ 1 \leq i_j \leq n)$. Set $w^0 = 1$ and define $w^j = s_{\alpha_{i_j}} s_{\alpha_{i_{j-1}}} \ldots s_{\alpha_{i_1}}$ $(1 \leq j \leq N)$. For $\nu \in V$ define

$$A_\rho(\nu) = \prod_{j=1}^{n-1} A_\rho(\alpha_{i_j}, w^{j-1}\nu) \in \mathrm{Hom}(V_\rho, V_\rho)$$

$$\text{and } J_\rho(\nu) = A_\rho(\nu) J_\rho.$$

LEMMA 1.
1. *$A_\rho(\nu)$ is independent of the choice of reduced expression for w_0,*
2. *$A_\rho(0) = Id$ and $A_\rho(\delta) = 0$ (unless ρ is trivial) where δ is one-half the sum of the positive roots,*
3. *$\lim_{\nu \to \infty} A_\rho(\nu) = \rho(w_0)$,*
4. *$A_\rho(\nu)$ is invertible unless $\langle \nu, \alpha_i^\vee \rangle = 1$ for some i,*
5. *Assume $w_0 \nu = -\nu$. Then $(A_\rho(\nu)v_1, v_2)_\rho = (v_1, A_\rho(\nu)v_2)_\rho$ for all $v_1, v_2 \in V$, and $J_\rho(\nu)$ is symmetric.*

For proofs of this and other statements in this section see [2] and `http://atlas.math.umd.edu/papers`.

Let $\mathcal{H}$ be the affine Hecke algebra of W [9, Chapter 7]. Associated to ν is an irreducible spherical representation τ_ν.

LEMMA 2. *The representation τ_ν of $\mathcal{H}$ is unitary if and only if for every irreducible representation ρ of W, the operator $J_\rho(\nu)$ is positive semi–definite.*

It is also of interest to determine the signature of $J_\rho(\nu)$.

The denominator in (1) is a convenient normalization, which makes Lemma 1(2) and (3) hold. Since $\nu \in V^+$ it is positive and does not affect whether $J_\rho(\nu)$ is positive semi–definite.

The question of whether $J_\rho(\nu)$ is positive semi–definite only depends on whether $1 - \langle \nu, \alpha^\vee \rangle$ is positive, negative, or 0, for each $\alpha \in R$. Therefore the set of dominant parameters ν is decomposed into a finite number of facets, each one determined by an element of $\{+, 0, -\}^n$ (not every such n-tuple arises). The classification of the unitary representations τ_ν of $\mathcal{H}$ is therefore reduced to a finite calculation.

Each facet contains an element ν with rational coordinates. It is well known we may choose a basis of V_ρ so that $\rho(w)$ is a rational, or even an integral, matrix for all $w \in W$. For $n \leq 8$ the former has been carried out explicitly in [14], and the latter in [1] except for E_8. Then the matrices $A_\rho(\nu)$ and $J_\rho(\nu)$ will have rational entries. We may clear denominators when testing for positive semi–definiteness.

Now let G be a semi–simple group over $\mathbb{R}$ or a p–adic field $\mathbb{F}$, with root system R. Associated to ν is an irreducible spherical representation π_ν of G.

LEMMA 3. *If $\mathbb{F}$ is p–adic π_ν is unitary if and only if τ_ν is unitary.*

CONJECTURE 2. *If $\mathbb{F} = \mathbb{R}$ then π_ν is unitary if and only if τ_ν is unitary.*

Thus computing the unitary representations τ_ν of Lemma 2 tells us about a subset of the unitary dual of Lie groups (the "spherical" unitary dual).

We may as well assume that G is simple, or equivalently that R is irreducible. The irreducible root systems are of type A_n, B_n, C_n, D_n (classical) or E_6, E_7, E_8, F_4 or G_2 (exceptional).

The most interesting case is that of E_8. The Weyl group has order 696, 729, 600; it has 112 representations, the largest of which has dimension 7, 168. There are 1, 070, 716 facets.

In the classical case the classification of the unitary representations τ_ν, and Conjecture 2, are known [2]. Dan Barbasch and Dan Ciubotaru have also computed the unitary τ_ν in the exceptional case. Thus the problem which the calculation of $J_\rho(\nu)$ solves is already known. However this calculation is the prototype of a much more general calculation which will be needed to calculate the unitary dual of Hecke algebras, real and p–adic Lie groups.

The Atlas of Lie Groups and Representations is a project to compute the unitary dual by theoretical and computational means, see `http://atlas.math.umd.edu`. Information about what size computations are feasible is of great importance in the continuation of this project.

3. ALGORITHMS

For the rest of this paper we will consider the question of signatures and sign patterns of symmetric matrices in general, but constantly keeping in mind the operators generated from Lie group matrix representations.

The signature of a matrix can be determined from the characteristic polynomial or the polynomial with coefficients of leading principal minors. For a polynomial with only real roots, Descartes' rule can be used to determinate the number of positive roots, the number of negative roots. To facilitate discussion, a n-degree polynomial $f_0 + f_1 x + \dots f_n x^n$ is presented by a $(n+1)$-length vector $v = (f_0, f_1, \dots, f_n)$, and also a vector may be viewed as a polynomial, and sometimes, we inter-exchange them. So here, for a vector v, we define $\sigma(v) = (p, z, n)$, where p is the number of alternating successive nonzero pairs, and n is the number of constant successive nonzero pairs, z is the number of successive zeros at the beginning, i.e, the multiplicity of zero roots. To be precise, a pair of entries, (v_i, v_j) of $(v_0, v_1, \dots v_n)$ is *successive* if $i < j$ and the entries between them are zero. A successive pair of nonzero entries is *alternating*(*constant*) if their signs are opposite(same). Following [11], say a matrix of rank r has *generic rank profile* if the leading principal minors of dimensions 1 through r are nonzero.

THEOREM 1 (SIGNATURE THEOREM). *Let A be a real symmetric $n \times n$ matrix. The following hold:*

1. *Signature is invariant under congruence, that is, if Q is nonsingular then $\sigma(A) = \sigma(QAQ^T)$.*
2. $\sigma(A) = \sigma(\texttt{charpoly}(A))$.
3. *If A is in generic rank profile,*
$$\sigma(A) = \sigma((-1)^n m_n, \cdots, m_1, 1),$$
where m_i is the ith leading principal minor of A.
4. *A matrix in generic rank profile has a unique $A = LDL^T$ decomposition with unit lower triangular L and diagonal $D =$ diag$(d_1, \dots, d_n)$. If A has generic rank profile and rank r, then (starting with $n - r$ zeroes) $\sigma(A) = \sigma(0, \cdots, 0, (-1)^r \prod_{1 \le i \le r} d_i, \cdots, d_2 d_1, -d_1, 1)$.*

Proof. A good source for these fundamental facts is [7]. In particular the third statement is a theorem of Jacobi, [7, Chapter X, §3, theorem 2]. The fourth item follows since $d_i = m_i / m_{i-1}$. See [8, Chapter 4] for a good discussion of LDL^T decomposition. □

The generic rank profile condition assures that the m_i consist of nonzero entries followed by zeros with no intermingling of zero and nonzero values, and the same applies to the diagonal D of the LDL^T decomposition. Interestingly, again see [7], the signature can be recovered even when there are some scattered zeroes among the nonzero leading minors, hence something less than generic rank profile is needed. We will not pursue this point further.

Let us say that a vector v, such as the characteristic polynomial coefficients or leading principal minors with alternative sign is a σ-revealing vector for A if σ(A) $= \sigma$(v). The algorithms we propose all work by computing images of a signature revealing vector mod a series of primes, and construction of the integer vector via the Chinese Remainder Algorithm. If a rational answer is desired, a rational reconstruction can be used to reveal the answer at the final step. In this paper, we are interested in computing the characteristic polynomial, minimal polynomial and leading principal minors of an integer matrix, all of them can be represented as a vector of integers. Next, a generic Chinese remainder algorithm for such vectors with an early termination technique is described.

Algorithm: `GenCRA`
[Chinese Remainder Algorithm with early termination]
Input:
A, a symmetric integer matrix.
`reveal` (v, A, p), a function takes two inputs A, p, and compute integer answer mod p and store in in-out vector v. this function will return false if `reveal` can't compute integer answer mod p correctly, or true if it computes the integer answer mod p.
$\mathcal{P}$, a set of primes of size M, with minimal prime β.
Ω, the random sample size.

Output:
The revealed vector v with integer coefficients.

1. set $n := 0$, the length of output vector.
set list $\mathcal{L} := \emptyset$, list of pair (good prime, revealed answer mod that prime).
2. choose a random vector x of length n with entries independently and uniformly chosen from $[0, \Omega - 1]$.
3. Uniformly choose a prime p from $\mathcal{P}$, until p is not used before. Mark p as being used. call `reveal` (v_p, A, p).
If return value is false, goto statement 3.
If the length of v_p is lower than n, goto statement 3.
If the length of v_p is larger than n, set n to be this length, empty $\mathcal{L}$, append pair (p, v_p) to $\mathcal{L}$, goto statement 2.
If the length of v_p is equal to n, append pair (p, v_p) to $\mathcal{L}$. Use Chinese remainder algorithm to construct a certificate $c^{(i)}$ where i the size of $\mathcal{L}$, such that $c^{(i)} = x \cdot v_q \pmod q$, for each pair (q, v_q) in $\mathcal{L}$.
4. If $c^{(i)} \neq c^{(i-1)}$, then goto statement 3. Otherwise $c^{(i)} = c^{(i-1)}$, then the termination condition is met, return the vector v, which is constructed from pairs in $\mathcal{L}$ by the Chinese remainder algorithm, such that $v = v_q \pmod q$, for every pair (q, v_q) in $\mathcal{L}$. This construction can be done by using a divide-and-conquer method.

Note: 1. Here, using Chinese remainder algorithm to construct a number a from its remainders $r_1, \dots, r_n$ moduli a bunch of primes $p_1, \dots, p_n$,respectively. We need to normalize the final number a, such that a lies between $[-m/2, m/2)$, where $m = \prod_{1 \le i \le n} p_i$.
2. An early termination technique is used here to speedup the computation in practice. There is already some discussion about the early termination, for example [10]. But here, we use a different and more efficient termination technique. At each step, only one integer called "certificate" is constructed instead of the actual answer mod modulus so far,

and has almost the same probability. This technique is easy to adopt to other cases for example, solving a non-singular integer systems over the rationals.
3. More than one independently certificates can reduce the error probability. Since the error for each independently certificate are independent. In practice, we use two independently certificates.

THEOREM 2. *On the condition that the length of output is correct, the probability that output v is not correct is at most $\frac{1}{\Omega}+\frac{\log^2(2l\Omega B)}{\log^2(\beta)(\mathsf{M}-\mathsf{M}'-\log(2l\Omega B)/\log(\beta))}$, where l is the length of the correct answer, B is infinity norm of the correct answer, M', number of primes p, such that either* `reveal` *(v_p, A, p) returns false or v_p has length of less than l.*

Note: For a typical problem, such as characteristic polynomial, minimal polynomial, it is easy to choose a reasonable set of prime and random sample size, such that the error probability is very tiny like one in a million.

PROOF: Let α be the correct answer, and $c = x \cdot \alpha$. Under assumption, α is a vector of length l. Also Let pair (p_i, v_{p_i}) in the $\mathcal{L}$ in order. We define $\|t\|_\infty$ to be the maximal absolute value of entries of x, for any vector t. We call a prime p "a good prime" if `reveal` (v, A, p) returns true and the length of output vector v equals l, otherwise we call it "a bad prime".

Claim 1:If x is a random vector with entries independently and uniformly chosen from the integer set of $[0, \Omega - 1]$, then $\text{Prob}(|(|x \cdot \alpha) < \|\alpha\|_\infty) \leq \frac{1}{\Omega}$. This claim is true since there is at least one entry of α, whose absolute value is equal to $\|\alpha\|_\infty$, and if with loss of generosity, we assume $|\alpha_0| = \|\alpha\|_\infty$, then any $x_1, \ldots, x_{l-1}$, then there is at most one integer x_0, such that $|x \cdot \alpha| < \|\alpha\|_\infty$.

Claim 2: For an integer m, if a random prime p is uniformly chosen from a cardinality k subset of $\mathcal{P}$, then $\text{Prob}(p|m) \leq \frac{\log(|m|)}{\log(\beta)k}$. This claim is true since there is at most $\frac{\log(|m|)}{\log(\beta)}$ primes which are members of $\mathcal{P}$ and also divide m.

When the early termination first meets at $c_n = c_{n-1}$, let us look at the probability of $c \neq c_n$. We have $c_n = c_{n-1}$, that means $c - c_{n-1} = 0 \pmod{p_n}$, and moreover, $\|c - c_{n-1}\|_\infty \leq 2\|c\|_\infty \leq 2l\Omega B$.

For any fixed n, if $c \neq c_{n-1}$ and a good prime p_n is randomly uniformly chosen from $\mathcal{P}$, then $\text{Prob}(p_n \mid (c - c_{n-1})) \leq \frac{\log_2(2l\Omega B)}{\log(\beta)(\mathsf{M}-\mathsf{M}'-n)}$. If n is larger than $\frac{\log(2l\Omega B)}{\log(\beta)}$, then v must be equal to c. Since the constructed $v = c \pmod{m}$, and $m = \prod_{1 \leq i \leq n} p_i > 2\|c\|_\infty$,

So when the early termination condition meets, $\text{Prob}(c \neq c_{i-1}) \leq \frac{(\log(2l\Omega)^2}{(\log(\beta))^2(\mathsf{M}-\mathsf{M}'-\log(2l\Omega B)/\log(\beta))}$.

If both $\|c\|_\infty \geq \|\alpha\|_\infty$ and $c = c_n$ are true, then v must be equal to the correct answer. So the probability of $v \neq \alpha$ is at most $\frac{1}{\Omega}+\frac{\log^2(2l\Omega B)^2}{\log^2(\beta)(\mathsf{M}-\mathsf{M}'-\log(2l\Omega B)/\log(\beta))}$. $\square$

The following theorem gives additional flexibility in using the strategies for signature computation made available by `GenCRA`. For $n \times n$ matrix A let `shiftminpoly`(A) denote a function which returns $x^k \cdot$ minpoly of A for k such that the degree of the resulting polynomial is n.

THEOREM 3. *Let A be an $n \times n$ real symmetric matrix and let S be a set of nonzero integers of sufficient size that $\epsilon = n^2 \log(n)^2/|S|$ is as small as desired.*

1. *Let D be a diagonal matrix whose n diagonal entries are chosen uniformly at random from S, and let $B = DAD^T$. Then*

$$\mathit{Prob}(\texttt{shiftminpoly}(B) = \texttt{charpoly}(B) \geq 1 - \epsilon.$$

2. *Let Q be a butterfly matrix whose $n \log(n)$ defining entries are chosen uniformly at random from S, and let $B = QAQ^T$. Then*

$$\mathit{Prob}(B \text{ is in generic rank profile}) \geq 1 - \epsilon.$$

Proof: These preconditionings are discussed in detail in [4] $\square$

For the generic rank profile condition, The butterfly is chosen as preconditioner because the specified matrix QAQ^T can be computed in $\tilde{O}(n^2)$ time. For our purposes here we can afford preconditioning complexity up to $\tilde{O}(n^3)$ time. A general random matrix could be used for the preconditioner, or Toeplitz [12] or sparse preconditioners [4, Section 6]. It is of interest to keep the size of the entries of the resulting matrix as small as possible.

We build three algorithms on this theorem, two using the minimal polynomial (for blackbox and for dense matrices) and one using LDL^T decomposition.

Algorithm: `BBSM [BlackBox Signature by Minpoly]`
Input: Symmetric matrix A, in blackbox form.
Output: The signature $\sigma(A)$.

1. [Preconditioning may be necessary]
   Let q be a random prime.
   Let $r :=$ `rank`(A, q) and `minpoly`$(m_q(x), A, q)$.
   If $\deg(m_q) < n$ and $\deg(m_q) \leq r$, let $B := DAD$ [A blackbox],
   for a random diagonal matrix D with entries chosen from $[1, s]$ (or any set of s nonzero integers).
   Otherwise, let $B := A$.
   [Now B has the same signature as A and its charpoly is a shift of its minpoly.]
   Choose a set $\mathcal{P}$ of primes, and the sample size Ω, such that the error probability is as small as desired.
2. Return $\sigma(x^{\max(0,n-r-1)}\text{GenCRA}(B, \texttt{minpoly}(), \mathcal{P}, \Omega))$.

The `rank`(A, p) and `minpoly`(v, A, p) algorithms used are as in [15, 12, 4], for example. Here `minpoly`() and `rank`() run in time $\tilde{O}(nE)$, and are probabilistic with probability of error no more than $1/p$. But `rank`() will never return a value greater than the true rank and `minpoly`() always computes at least a factor of the true minimal polynomial of A mod p, its return status is true. (This is in addition to the possibility of p being unfaithful.) DAD is the blackbox whose matrix-vector product is formed as $y = D(A(Dx))$.

Algorithm: `DSM [Dense Signature by Minpoly]`
Input: Matrix A, in dense form.
Output: $\sigma(A)$.
Method:

Apply algorithm `BBSM`, except use `rank`(A, p) [6] and `minpoly`(v, A, p), [5], algorithms which are available for the explicit (dense) matrix representation. Then `rank`() and `minpoly`() are are deterministic eliminations running in time $O\tilde{}(n^3)$ and using $O(n^2d)$ memory Of course, in this case the DAD preconditioning is done explicitly (and cheaply). Again, of course, p may be unfaithful.

Especially for blackbox matrices, it is useful that minpoly computation can suffice, because we have faster algorithms for minpoly than for charpoly. Also note that the minpoly suffices for determining the sign pattern (not full signature) even without preconditioning. To be precise, the sign pattern of A is determined from $s = \sigma$(minpoly of A) by replacing positive entries in s by $+$. When the minpoly is of low degree or has small coefficients, this is a great savings. In general, though, BBSM is not a fully memory efficient algorithm because of the size of the σ-revealing vector. It is possible that the technique of [3] could be used to determine the signs using less memory and perhaps less time. This approach deserves further examination.

However, it is not necessary to compute the characteristic or minimal polynomial to answer signature and definiteness questions. Combining Theorem 1, part 3, 4, and theorem 3, part 2, we may work from the LDL^T decomposition of a matrix in generic rank profile. If the matrix should fail to have generic rank profile, this will be detected during the elimination because of the need for pivoting. A note on pivoting: Of course not every symmetric matrix has an LDL^T decomposition. We can use preconditioners Q to assure that QAQ^T has generic rank profile, and in some cases symmetric pivoting could be used instead to avoid the increased entry size caused by preconditioning. Also, the image mod p may fail to have generic rank profile even when the prime is faithful. We could modify the LU to successfully handle such cases, but it is a remote possibility and not worth the overhead. So we reject such primes as if unfaithful.

The LDL^T decomposition based signature algorithm assumes a procedure $\mathrm{lpm}(v, A, p)$ which computes the vector

$$((-i)^k \prod_{1 \le i \le k} D_{i,i}, \cdots, D_{1,1}D_{2,2}, -D_{1,1}, 1),$$

i.e, leading principal minors of A up to the the first which is zero mod p with alternative sign. Its return status is true, the k is equal to rank mod p, otherwise false. In any case, $k \le r$. The procedure lpm works by attempting, mod p, the LDL^T decomposition, stopping when a zero pivot is encountered. And the lpm meets the requirements of the σ-revealing function for GenCRA.

Algorithm: `DSLU`
[Dense matrix Signature by LU-decomposition]
Input: A, an integer symmetric $k \times k$ matrix.
Output: Signature $\sigma(A) = (p, z, n)$.

1. [Preconditioning may be necessary]
   Let q be a random prime.
   Let $\mathrm{lpm}(v_q, A, q)$, $r :=$ `rank`(A, q).
   If length of v_q less than $r + 1$, let $B := QAQ^T$,
      for a random integer matrix Q with entries chosen from 1..s (or any set of s nonzero integers).
      [Q may be a Toeplitz or Butterfly matrix for speed]
   Otherwise, let $B := A$.
   [Now B has the same signature as A and has generic rank profile.]
   Choose a set $\mathcal{P}$ of primes, and the sample size Ω, such that the error probability is as small as desired.
2. Return $\sigma(x^{n-r}\mathrm{GenCRA}(B, \mathrm{lpm}(), \mathcal{P}, \Omega))$.

THEOREM 4. *Let A be an integer blackbox $n \times n$ symmetric matrix, whose matrix-vector product costs e operations and whose entries would be of bit length at most d if they were constructed.*

Algorithms BBSM, DSM, and DSLU are Las Vegas probabilistic algorithm if the Hadamard bound is used and Monte Carlo if early termination is used in the remaindering. Even with early termination, DSM is Las Vegas if the computed integer minimal polynomial is checked by evaluation at A over the integers. This costs $O\tilde{}(n^4)$.Also a minimal polynomial verification by application to the identity could be done after BBSM making it Las Vegas for the sign pattern of the signature.

The BBSM expected run time is $O\tilde{}(neh)$, where h is $O\tilde{}(nd)$ if the Hadamard bound is used and h is the largest bit length of any charpoly coefficient if early termination is used. In either case, BBSM runs in expected time $O\tilde{}(n^2ed)$. The DSM and DSLU expected run time is $O\tilde{}(n^3h)$, where h is $O\tilde{}(nd)$ if the Hadamard bound is used and h is the largest bit length of any charpoly coefficient if early termination is used.

In particular if $e \in O\tilde{}(n)$, and $d \in \mathcal{O}(\log(n))$, then the BBSM expected run time is $O\tilde{}(n^3)$ and the DSM, DSLU expected times in $O\tilde{}(n^4)$.

Also, with either of the σ-revealing vectors, the entries tend to grow in proportion to their index. In particular the ith entry is either an $i \times i$ minor or sum of $i \times i$ minors, so is bounded by the Hadamard bound which is in $O\tilde{}(id)$. A heuristic to determine indefiniteness compute the first few vector entries, using many fewer remaindering steps than are required for the later entries. If the sign pattern fails to be constant or strictly alternating the matrix is indefinite. It is an open question whether an conjugacy preconditioning, $A \to QAQ^T$, could give that early entries indicate definiteness with high probability.

4. APPLICATION TO LIE MATRICES

The matrices from Lie group representation are rational matrices, our algorithms in the previous section focus on integer matrices. There are at least two alternative way to adapt. One is to compute the minimal polynomial, or the leading principal minors over the rationals. The rational answers need to be reconstructed at the GenCRA, and it is easy to adopt our GenCRA with early termination technique to this case. The other way is to multiply the matrices by the lcm of denominators of all entries. For these special constructed matrices from Lie group representation, the lcm of denominators of all entries is just a little larger than each individual one, so the latter is a better way. We found for some models the gcd of all the numerators of the dense representation are not trivial, so it could be removed. Here, we present blackbox algorithms to compute the lcm of denominators of all entries, and gcd of numerators of all entries, respectively.

Algorithm: `LDE`
Input:
  A, a rational matrix
  M, sample size
  n, number of trials
Output:
  d, the lcm of denominators of all entries of dense representation of A.

1. for i from 1 to n do
   Choose a random vector $x^{(i)}$ with entries independently and uniformly chosen from $[0, M]$
   $y^{(i)} = Ax^{(i)}$, apply $x^{(i)}$ to A
   $d^{(i)}$= the lcm of denominators of every entry of $y^{(i)}$.
2. $d := \text{lcm}(d^{(1)}, \cdots, d^{(i)})$
3. return d.

Algorithm: `GNE`
Input:
  A, an integer matrix
  M, sample size
  n, number of trials
Output:
  g, the gcd of all entries of dense representation of A.
Method:
  Apply algorithm *LDE* by replacing lcm with gcd and denominators with numerators.

The algorithm *LDE* always returns a factor of the true answer. The algorithm *GNE* always returns a multiple of the true answer. For both algorithms, for a rational matrix A, if d and g are the lcm of all denominators and gcd all numerators of entries of A, respectively. If let $A' = \frac{d}{g}A$, then A' is an integer matrix. For each individual trial, if all entries of $A'x^{(i)}$ are coprime, then $d^{(i)}$ is equal to the correct answer in both algorithms. Please see [13] for probability analysis. In both cases, the error probability is 2^{-n+1}, roughly speaking.

Each matrices from Lie group representation for E8 can be represented as $(\prod_{1 \le i \le 120} \frac{1}{r_i}(r_i Ops^{(i)} + I))F$, where $Ops^{(i)}$ is one of the eight sparse matrices called operators, F is also sparse matrix. The algorithm BBSM is very suitable for this case, and we also reduce them to product of 94 sparse matrices by explicitly computing and storing $r_i Ops^{(i)} + I$ as a sparse matrix, and multiplying it to left one if the computed sparse matrix is a diagonal matrix. By reducing number of sparse matrices, 20% run time was saved for BBSM algorithm. After multiplying these collapsed matrices by the lcm of denominators of all entries and dividing by gcd of numerators of all entries, these matrices can be handled by these algorithms discussed in previous section.

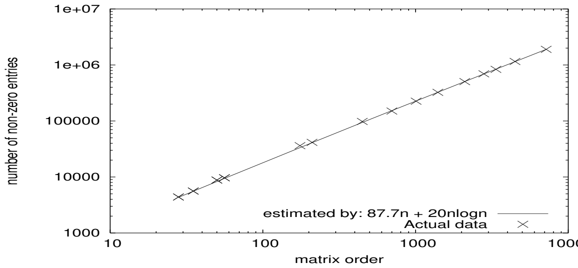
5. EXPERIMENTS

Our goal is to know which algorithm to use and how much resources it will take for computing the signature of the $J_n(f) = J_\rho(f)$, the matrix generated by the facet f for the model of dimension n of the group E8 . Let $J_n(f) = J_\rho(f)$ to emphasize the model dimension. In particular, we focus on the case when the facet is the challenge facet $\nu = 1/40(0, 1, 2, 3, 4, 5, 6, 23)$, and the goal is to verify (or disprove) that $J_n(\nu)$ is positive definite. The three signature algorithms we have discussed are BBSM, DSM, DSLU. We have computed full solutions by each model for the models Q2..Q12, Q22, Q32, Q42, Q52(dimension 1008), and we have computed single modular step costs for every 10th model on up to the largest, Q112(dimension 7168). The experiments involved individual run times up to about one day and verified, where the full computation was done, that $J_n(\nu)$ is indeed positive definite. All computations reported here were performed on a 3.2GHz Xeon processor with 6GB of main memory. Our purpose here is to determine the algorithm to use and resources needed for the full signature computation on examples of all dimensions (up to 7168) and to have tools to estimate the cost for matrices defined by other facets.

The blackbox representation of $J_n(\nu)$ is a product of 121 very sparse factors. Since the cost of computing the characteristic polynomial depends on the cost of applying A to a vector, we examined the number of nonzero elements in total in these factors.

The blackbox step cost is sensitive to the cost e of the single matrix vector product. To guess the expected growth of e with dimension, we measured the total number of nonzero entries in the factors of every 10th model up to Q112. Since the number of factors is fixed and they are extremely sparse we expect a linear or $n\log(n)$ term to dominate. The formula $e = 132.5900691n + 13.12471811n\ln(n)$, a least squares fit to the data, is plotted in figure 1 along with the data values. The fit is extremely good and we see that e is only slightly super-linear. Therefore we expect the runtime of characteristic polynomial computation mod a single prime to grow at a rate only slightly above quadratic in n.

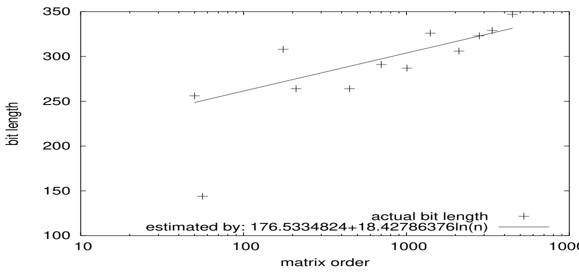
Figure 1: Blackbox number of non-zeroes



Each algorithm involves a series of steps mod primes p_i, each step being a computation of σ-revealing vector (a minpoly or diagonal of LDL^T-decomposition) mod p_i. The computation is finished when the CRA remaindering has given an image of the vector with modulus, $M = 2\prod p_i$, sufficiently large to recover the actual (signed) vector entries. If we know a bound d for the length of the entries of $J_n(\nu)$, we have by the Hadamard bound a maximal length $b = n(\log(n)/2 + d) + 1$ for M.

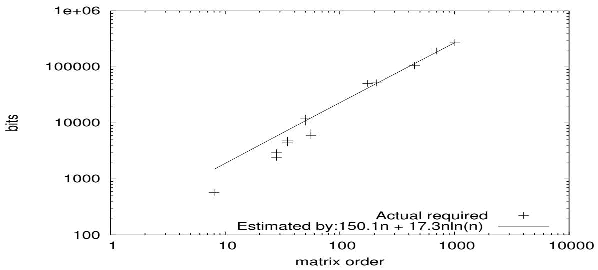
the Dense matrices are computed by applying the black box to the columns of the identity matrix (over $\mathbb{Z}$, not modularly), and so are sensitive to the length of the actual integer entries of the expanded (and dense). Let d be the length of the largest entry. We do not have a theory to predict how matrix entry bit length d may depend on the model dimension for a fixed facet. We have plotted the bit length d of the largest entry of $J_n(\nu)$ along with the fitted curve $d = 67.7 + 33.2\log(n)$ in figure 2. One sees the $\log(n)$ playing a stronger role than that for the non-zeroes in the blackbox, but the fit to the data is poorer and likely to have less predictive value. We noted that the n^2 entries of A all have about the same length, so $n^2 d$ accurately describes the storage required for A.

Figure 2: Bit length of entries



It will also be useful to use d in estimating the number of remaindering steps. The needed bit length b for the modulus M is at most 1 bit (for sign) more than the length of the Hadamard bound. The rows norms are no more than $\sqrt{n(e^d)^2}$, so their product has length bounded by $n(\log(n)/2 + d)$. Reasonably consistent with this prediction is the number of bits used when we ran the full algorithms on $J_n(\nu)$ for n up to 1008 (model Q52). The curve in figure 3 is a least squares fit, $b = 150.1n + 17.30nln(n)$, The number of bits for the charpoly coefficients (sums of minors) are expected to be slightly larger than for the entries of the leading principal minors vector, but most likely, the final term, determinant of $J_n(\nu)$, is dominant. The number of bits needed appears to be slightly super-linear, not quite as large as the worst case Hadamard bound level.

Figure 3: Bit length of final modulus

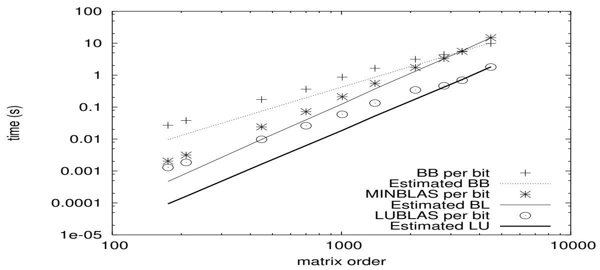


Next we consider the costs of the modular steps. Let step(s) denote the cost per step which varies with length, s, of the prime, but is essentially constant among primes of the same length. The total time is $b/s \times \text{step}(s)$ when using s bit primes. Thus we see the advantage of minimizing $\text{step}(s)/s$, the cost per bit. After discussing $\text{step}(s)/s$ and determining it's value, for each algorithm, we will know how to estimate the full runtime of the algorithms.

We continue to tune the implementation of the modular steps. We expect to be able to reduce the LU step cost by half, as an elimination better tailored to symmetric matrices can be done, while still taking advantage of block operations using BLAS. For the blackbox algorithm, block methods may be able to reduce the costs somewhat, and the DSM may perform relatively better than here if the dense `charpoly`() algorithm [5] is used rather than preconditioning. At any rate, let us see the performance of the modular steps of the three algorithms, BBSM, DAME, DSLU as currently implemented. Asymptotically, the BBSM step runtime, s, is expected to grow in proportion to ne. There was a good fit to $s = 61n^2\log(n)$ nanoseconds per bit. (recall that e has a $n\log(n)$ dominant term). . The algorithms DSM and DSLU depend on the construction of the full integer matrix of $J_n(\nu)$, which is done once. The step times consist of computing a modular image of that and proceeding with an elimination on the image. The implementation of the Krylov matrix construction in the `minpoly`() algorithm of DSM, uses a technique of recursive doubling to better exploit fast matrix multiplication, and gets a step time s which fits the formula $t = 1.8n^3 + 0.16n^3\log(n)$ nanoseconds per bit. $t = 0.37n^3\log(n)$ nanoseconds per bit. quite well. The DSLU step fits $t = 0.3838888362n^3$ nanosec/bit. But these formulas have to be worked to the bit cost level!!!

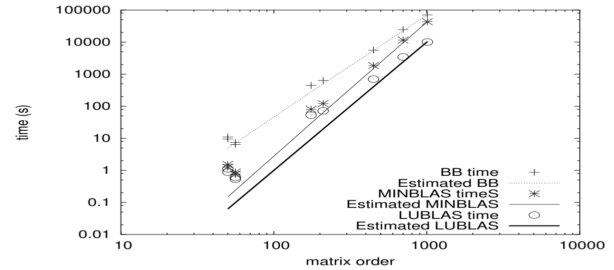
The log factor for DSM step can be removed, so it remains possible that it can become competitive or participate with BBSM in a heterogeneous distributed computation, running on the machines with larger memory. The time formula for the dense matrices should remain valid as long as the modular step fits in main memory. After than swapping would dramatically increase times. Assuming a memory capable of holding a 2GB virtual memory, this would allow for n^2 words, so $n < 2^{15}$. For all practical purposes, the blackbox step has no memory limitation.

Figure 4: Time per bit of modulus



The total runtime then involves the time per bit in the modular steps times the number of bits in the σ-revealing vector. The Chinese remaindering adds a cost that is similar for each but large enough to mute somewhat the effect of the differing step costs. However with the early termination technique of the previous section this remaindering is a smaller factor than in earlier timings. For instance, without the asymptotic improvement in the termination strategy our time for x was y, now it is z. The two dense algorithm also incur a lower order cost for the creation of the expanded integer matrix initially. The least squares fit formulas for the overall run times in nanoseconds are BBSM: $t = 10050n^3 \log(n)$, DSM: $t = 6.08n^4 \log(n)$, and DSLU: $t = 1.45n^4 \log(n)$.Theory predicts a second log factor in the BBSM time, coming from the blackbox e. However that effect is weak and the fit was poorer.

Figure 5: Total run time



Also the memory needed to store the σ-revealing vector mutes the memory advantage BBSM. However the modular images of the vector are easily stored on hard disk and manipulated from there, so the memory advantage remains real. The crossover point of the runtime formulas for BBSM and DSLU is around $n = 6931$, however the formula $t = 10.02n^4$ fits the DSLU data about as well as our formula above. Using that form, the crossover point is at $n = 9150$.

6. CONCLUSION

We have demonstrated that we can compute (on current hardware) the signature of a dense $n \times n$ integer matrix having entries of bit length around $\log(n)$ in a minute if $n \leq 200$, in three hours if $n = 1000$ and (projected) in a CPU year for $J_{7168}(\nu)$. Beyond that size, using algorithm BBSM, the time grows at a rate slightly higher than n^3 and memory is not a constraint (except for storage of the sparse factors of the blackbox). However, we conclude that algorithm DSLU serves best for matrices of dimension $n <$ 7000. It is tossup between DSLU and BBSM for dimensions $7000 \leq n \leq 9000$ and BBSM is superior beyond that. DSLU time grows slightly above n^4.

For $J_{7168}(\nu)$ DSLU is requires explicit use of file storage of the expanded matrix and both algorithms should do this for intermediate results (modular images of σ-revealing vectors), because of the large size. We have not measured the cost of this file manipulation. At the crossover about one CPU year is required and DSLU needs a large memory. The run time is expected to be about a CPU year, so Parallel computation is desirable (and is quite straightforward for either algorithm), on distributed or shared memory hardware.

It is an open question whether definiteness can be determined fundamentally faster than signature. There is a fast Monte Carlo algorithm for rank, hence for distinguishing semidefinite from definite matrices. We have sketched a heuristic that sometimes can determine indefiniteness much faster than the signature computation.

To provide for the needs of Lie group representation studies, both BBSM and DSLU will be further refined and their parallel implementation developed. Also the judicious incorporation of numeric computation is a possibility.

7. REFERENCES

[1] J Adams. Integral models of representations of weyl groups. `http://atlas.math.umd.edu/weyl/integral`.

[2] D. Barbasch. Unitary spherical spectrum for split classical groups, preprint. `http://www.math.cornell.edu/~barbasch/nsph.ps`.

[3] H. Bronnimann, I.Z. Emiris, V.Y. Pan, and S. Pion. Sign determination in residue number systems. *Theoret. Comput. Sci.*, 210:173197, 1999.

[4] L. Chen, W. Eberly, E. Kaltofen, B. D. Saunders, W. J. Turner, and G. Villard. Efficient matrix preconditioners for black box linear algebra. *Linear Algebra and Applications*, 343-344:119–146, 2002.

[5] C.Pernet and Z. Wan. Lu based algorithms for the characteristic polynomial over a finite field. In *Poster, ISSAC'03*. ACM Press, 2003.

[6] Jean-Guillaume Dumas, Pascal Giorgi, and Clément Pernet. FFPACK: Finite field linear algebra package.

[7] F. R. Gantmacher. *The Theory of Matrices*. Chelsea, New York, NY, 1959.

[8] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, third edition, 1996.

[9] James E. Humphreys. Reflection groups and Coxeter groups, *volume 29 of* Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, 1990.

[10] I.Z.Emiris. A complete implementation for computing general dimensional convex hulls. *Inter. J. Comput. Geom. Appl.*, 8:223–253, 1998.

[11] E. Kaltofen and A. Lobo. On rank properties of Toeplitz matrices over finite fields. In *ISSAC 96 Proc. 1996 Internat. Symp. Symbolic Algebraic Comput.*, pages 241–249.

[12] E. Kaltofen and B. D. Saunders. On Wiedemann's method of solving sparse linear systems. In H. F. Mattson, T. Mora, and T. R. N. Rao, editors, *Proc. AAECC-9*, volume 539 of *Lect. Notes Comput. Sci.*, pages 29–38, Heidelberg, Germany, 1991. Springer Verlag.

[13] B. David Saunders and Zhendong Wan. Smith normal form of dense integer matrices fast algorithms into practice. In *Proc. ISSAC'04*, pages 274–281. ACM Press, 2004.

[14] John R. Stembridge. Explicit matrices for irreducible representations of Weyl groups. *Represent. Theory (electronic)*, 8:267–289, 2004.

[15] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theory*, IT-32:54–62, 1986.

Signature of Symmetric Rational Matrices and the Unitary Dual of Lie Groups

[Extended Abstract]

Jeffrey Adams*
Department of Mathematics
University of Maryland
College Park, MD, 20742 USA
jda@math.umd.edu

B.David Saunders†
Department of Computer and Information Sciences
University of Delaware
Newark, DE, 19716 USA
saunders@cis.udel.edu

Zhendong Wan†
Department of Computer and Information Sciences
University of Delaware
Newark, DE, 19716 USA
wan@cis.udel.edu

ABSTRACT
A key step in the computation of the unitary dual of a Lie group is the determination if certain rational symmetric matrices are positive semidefinite. The size of some of the computations dictates that high performance integer matrix computations be used. We explore the feasibility of this approach by developing three algorithms for integer symmetric matrix signature and studying their performance both asymptotically and by experimenting on a particular matrix family constructed from the exceptional Weyl group E8. We conclude that the computation is doable, with a parallel implementation needed for the largest representations.

Categories and Subject Descriptors
G.4 [**Mathematical Software**]: Algorithm Design and Analysis; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*

General Terms
Algorithms, Performance

Keywords
matrix signature, symmetric matrix, lie group

1. INTRODUCTION
We propose, analyze, and test several algorithms for computing the signature and for verifying or disproving specific definiteness properties of symmetric integer matrices. The latter is sometimes easier than the signature. Also there is often a considerable difference in the cost of verifying (certifying) a property and of certifying it's negation. This is primarily an experimental paper whose purpose is to assess the feasibility of exact integer linear algebra methods for signature and definiteness determinations that may be desirable in the study of Lie group representations. The combination of measurements and analysis of asymptotic growth rates of time and memory use we provide are for the purpose of predicting the cost of the larger computations of interest, so as to determine the most promising algorithms and the hardware resource needs. The algorithmic asymptotic complexities reported here are largely straightforward applications of known results. We have an interesting observation which allows efficient binary tree structured Chinese remaindering to be used with an early termination strategy, giving practical and asymptotic speedup when using early termination. We have measured the time and memory costs of three algorithms on some matrices arising in group representations, studied their relative merits, and developed formulas to predict the costs for large matrices.

In section 2, the motivation for this work in the study of Lie Group representations is developed. This application creates large problems straining our limits of time and memory.

In section 3 we present the proposed algorithms and discuss their mathematical basis and complexity. Also included is some discussion of unimplemented alternatives and of the space issues for large instances of the problems. In section 4, we apply these algorithms to the matrices from Lie group representation.

Finally, in section 5, experimental results are reported and, in section 6, conclusions drawn on the state of these problems.

The signature of a real symmetric matrix A is generally defined as $\sigma = p - n$, the number by which positive eigenvalues outnumber negative ones. For our purposes, define the *signature* to be the triple $\sigma(A) = (p, z, n)$, where p is the number of positive eigenvalues, z is the multiplicity of zero as an eigenvalue, and n is the number negative. The triple $(\sigma(A), \texttt{rank}(A), \dim(A))$ and $\sigma(A)$ convey the same information. By the symmetry, all eigenvalues are real. A matrix is

*Supported by NSF grant DMS 0200851.
†Supported by NSF grants CCR 0098284 and 0112807.

ISSAC '05, KLMM, Chinese Academy of Sciences, Beijing, China

positive definite if the signature has pattern is $(*, 0, 0)$ and positive semidefinite if the pattern is $(*, *, 0)$.

2. LIE GROUP REPRESENTATIONS AND UNITARY DUALITY

We assume some familiarity with root systems and Weyl groups, for example see [9]. We begin with a formal construction. Let R be a root system with Weyl group W. Thus R is a finite subset of $V = \mathbb{R}^n$ satisfying certain properties; in particular for each $\alpha \in R$ the reflection $s_\alpha \in \mathrm{Hom}(V, V)$ takes R to itself. By definition W acts on V. We may choose simple roots $S = \{\alpha_1, \ldots, \alpha_n\}$ so that W is generated by $\{s_i = s_{\alpha_i} \mid i = 1, \ldots, n\}$, with relations $\mathrm{order}(s_i s_j) = m_{i,j}$ for certain $m_{i,j} \in \{2, 3, 4, 6\}$. For $\nu \in V$ define $\langle \nu, \alpha^\vee \rangle = 2(\nu, \alpha)/(\alpha, \alpha)$. Let V^+ be the set of $\nu \in V$ which are dominant, i.e. $\langle v, \alpha_i^\vee \rangle \geq 0$ $(1 \leq i \leq n)$.

Fix a finite dimensional representation (ρ, V_ρ) of W. Thus $\rho : W \to \mathrm{Hom}(V_\rho, V_\rho)$ is a group homomorphism. Fix an invariant Hermitian form $(,)_\rho$ on V_ρ, i.e. satisfying $(v_1, v_2)_\rho = (\rho(w)v_1, \rho(w)v_2)_\rho$ for all $v_1, v_2 \in V_\rho, w \in W$. Choose a matrix J_ρ so that $(v_1, v_2)_\rho = v_1 J_\rho v_2^t$.

For $\alpha \in R$ and $\nu \in V^+$ we define

$$A_\rho(\alpha, \nu) = \frac{1 + \langle \nu, \alpha^\vee \rangle \rho(s_\alpha)}{1 + \langle \nu, \alpha^\vee \rangle} \in \mathrm{Hom}(V_\rho, V_\rho). \tag{1}$$

Let w_0 be the longest element of W and choose a reduced expression $w_0 = s_{\alpha_{i_N}} s_{\alpha_{i_{N-1}}} \ldots s_{\alpha_{i_1}}$ $(1 \leq j \leq N,\ 1 \leq i_j \leq n)$. Set $w^0 = 1$ and define $w^j = s_{\alpha_{i_j}} s_{\alpha_{i_{j-1}}} \ldots s_{\alpha_{i_1}}$ $(1 \leq j \leq N)$. For $\nu \in V$ define

$$A_\rho(\nu) = \prod_{j=1}^{n-1} A_\rho(\alpha_{i_j}, w^{j-1}\nu) \in \mathrm{Hom}(V_\rho, V_\rho)$$

$$\text{and } J_\rho(\nu) = A_\rho(\nu) J_\rho.$$

LEMMA 1.
1. *$A_\rho(\nu)$ is independent of the choice of reduced expression for w_0,*
2. *$A_\rho(0) = Id$ and $A_\rho(\delta) = 0$ (unless ρ is trivial) where δ is one-half the sum of the positive roots,*
3. *$\lim_{\nu \to \infty} A_\rho(\nu) = \rho(w_0)$,*
4. *$A_\rho(\nu)$ is invertible unless $\langle \nu, \alpha_i^\vee \rangle = 1$ for some i,*
5. *Assume $w_0 \nu = -\nu$. Then $(A_\rho(\nu)v_1, v_2)_\rho = (v_1, A_\rho(\nu)v_2)_\rho$ for all $v_1, v_2 \in V$, and $J_\rho(\nu)$ is symmetric.*

For proofs of this and other statements in this section see [2] and `atlas.math.umd.edu/papers`.

Let $\mathcal{H}$ be the affine Hecke algebra of W [9, Chapter 7]. Associated to ν is an irreducible spherical representation τ_ν.

LEMMA 2. *The representation τ_ν of $\mathcal{H}$ is unitary if and only if for every irreducible representation ρ of W, the operator $J_\rho(\nu)$ is positive semi–definite.*

It is also of interest to determine the signature of $J_\rho(\nu)$.

The denominator in (1) is a convenient normalization, which makes Lemma 1(2) and (3) hold. Since $\nu \in V^+$ it is positive and does not affect whether $J_\rho(\nu)$ is positive semi–definite.

The question of whether $J_\rho(\nu)$ is positive semi–definite only depends on whether $1 - \langle \nu, \alpha^\vee \rangle$ is positive, negative, or 0, for each $\alpha \in R$. Therefore the set of dominant parameters ν is decomposed into a finite number of facets, each one determined by an element of $\{+, 0, -\}^n$ (not every such n-tuple arises). The classification of the unitary representations τ_ν of $\mathcal{H}$ is therefore reduced to a finite calculation.

Each facet contains an element ν with rational coordinates. It is well known we may choose a basis of V_ρ so that $\rho(w)$ is a rational, or even an integral, matrix for all $w \in W$. For $n \leq 8$ the former has been carried out explicitly in [14], and the latter in [1] except for E_8. Then the matrices $A_\rho(\nu)$ and $J_\rho(\nu)$ will have rational entries. We may clear denominators when testing for positive semi–definiteness.

Now let G be a semi–simple group over $\mathbb{R}$ or a p–adic field $\mathbb{F}$, with root system R. Associated to ν is an irreducible spherical representation π_ν of G.

LEMMA 3. *If $\mathbb{F}$ is p–adic π_ν is unitary if and only if τ_ν is unitary.*

CONJECTURE 2. *If $\mathbb{F} = \mathbb{R}$ then π_ν is unitary if and only if τ_ν is unitary.*

Thus computing the unitary representations τ_ν of Lemma 2 tells us about a subset of the unitary dual of Lie groups (the "spherical" unitary dual).

We may as well assume that G is simple, or equivalently that R is irreducible. The irreducible root systems are of type A_n, B_n, C_n, D_n (classical) or E_6, E_7, E_8, F_4 or G_2 (exceptional).

The most interesting case is that of E_8. The Weyl group has order 696, 729, 600; it has 112 representations, the largest of which has dimension 7, 168. There are 1, 070, 716 facets.

In the classical case the classification of the unitary representations τ_ν, and Conjecture 2, are known [2]. Dan Barbasch and Dan Ciubotaru have also computed the unitary τ_ν in the exceptional case. Thus the problem which the calculation of $J_\rho(\nu)$ solves is already known. However this calculation is the prototype of a much more general calculation which will be needed to calculate the unitary dual of Hecke algebras, real and p–adic Lie groups.

The Atlas of Lie Groups and Representations is a project to compute the unitary dual by theoretical and computational means, see `atlas.math.umd.edu`. Information about what size computations are feasible is of great importance in the continuation of this project.

3. ALGORITHMS

For the rest of this paper we will consider the question of signatures and sign patterns of symmetric matrices in general, but constantly keeping in mind the operators generated from Lie group matrix representations.

The signature of a matrix can be determined from the characteristic polynomial or the polynomial with coeffecients of leading principal minors. For a polynomial with only real roots, Descartes' rule can be used to determinate the number of positive roots, the number of negative roots. To facilitate discussion, a n-degree polynomial $f_0 + f_1 x + \dots f_n x^n$ is presented by a $n+1$-length vector $v = (f_0, f_1, \dots, f_n)$, and also a vector may be viewed as a polynomial, and sometimes, we interexchange them. So here, for a vector v, we define $\sigma(v) = (p, z, n)$, where p is the number of alternating successive nonzero pairs, and n is the number of constant successive nonzero pairs, z is the number of successive zeros at the beginning, i.e, the number of zero roots. To be precise, a pair of entries, (v_i, v_j) of $(v_0, v_1, \dots v_n)$ is *successive* if $i < j$ and the entries between them are zero. A successive pair of nonzero entries is *alternating*(*constant*) if their signs are opposite(same). Following [11], say a matrix of rank r has *generic rank profile* if the leading principal minors of dimensions 1 through r are nonzero.

THEOREM 1 (SIGNATURE THEOREM). *Let A be a real symmetric $n \times n$ matrix. The following hold:*

1. *Signature is invariant under congruence, that is, if Q is nonsingular then $\sigma(A) = \sigma(QAQ^T)$.*
2. *$\sigma(A) = \sigma(\texttt{charpoly}(A))$.*
3. *If A is in generic rank profile,*
$$\sigma(A) = \sigma(1, -m_1, \dots, (-1)^n m_n),$$
where m_i is the ith leading principal minor of A.
4. *A matrix in generic rank profile has a unique $A = LDL^T$ decomposition with unit lower triangular L and diagonal $D = \ diag(d_1, \dots, d_n)$. If A has generic rank profile and rank r, then (with $n - r$ trailing zeroes)*
$$\sigma(A) = \sigma(1, -d_1, d_1 d_2, \dots, (-1)^i \prod_{1 \le i \le r} d_i, 0, \dots, 0).$$

Proof. A good source for these fundamental facts is [7]. In particular the third statement is a theorem of Jacobi, [7, Chapter X, §3, theorem 2]. The fourth item follows since $d_i = m_i / m_{i-1}$. See [8, Chapter 4] for a good discussion of LDL^T decomposition. □

The generic rank profile condition assures that the m_i consist of nonzero entries followed by zeros with no intermingling of zero and nonzero values, and the same applies to the diagonal D of the LDL^T decomposition. Interestingly, again see [7], the signature can be recovered even when there are some scattered zeroes among the nonzero leading minors, hence something less than generic rank profile is needed. We will not pursue this point further.

Let us say that a vector v, such as the characteristic polynomial coefficients or leading principal minors with alternative sign is a σ-revealing vector for A if $\sigma(\mathrm{A}) = \sigma(\mathrm{v})$. The algorithms we propose all work by computing images of a signature revealing vector mod a series of primes, and construction of the integer vector via the Chinese Remainder Algorithm. In this paper, we are interested in computing the characteristic polynomial, minimal polynomial and leading principal minors of an integer matrix, all of them can be represented as a vector of integers. Next, a generic Chinese remaindering algorithm for such vectors with an early termination technique is described.

Algorithm: `GenCRA`
[Chinese Remaindering Algorithm with early termination]
Input:
A, a symmetric integer matrix.
`reveal` (v, A, p), a function takes two inputs A, p, and compute integer answer mod p and store in in-out vector v. this function will return false if `reveal` can't compute integer answer mod p correctly, or true if it computes the integer answer mod p.
$\mathcal{P}$, a set of primes of size M, with minimal prime β.
Ω, the random sample size.
Output:
The revealed vector v with integer coefficients.

1. set $n := 0$, the length of output vector. set list $\mathcal{L} :=$, list of pair (good prime, revealed answer mod that prime).
2. choose a random vector x of length n with entries independently and uniformly chosen from $[0, \Omega - 1]$.
3. Uniformly choose a prime p from $\mathcal{P}$, until p is not used before. Mark p as being used. call `reveal` (v_p, A, p).
If return value is false, goto statement 3.
If the length of v_p is lower than r, reject the prime, and goto statement 3.
If the length of v_p is equal to n, append pair (p, v_p) to $\mathcal{L}$. Use Chinese remaindering algorithm to construct a certificate $c^{(i)}$ where i the size of $\mathcal{L}$, such that $c^{(i)} = x \cdot v_q \pmod q$, for each pair (q, v_q) in $\mathcal{L}$.
If the length of v_p is larger than n, set n to be this length, empty $\mathcal{L}$, append pair (p, v_p) to $\mathcal{L}$, goto statement 2.
4. If $c^{(i)}$ is equal to the $c^{(i-1)}$, then the termination condition is met, Return the vector v, constructed from pairs in *List* by the Chinese remaindering algorithm, such that $v = v_q \pmod q$, for every pair (q, v_q) in $\mathcal{L}$. This construction can be done by using a divide-and-conquer method.

Note: 1. Here, using Chinese remaindering algorithm to construct a number a from its remainders $r_1, \dots, r_n$ moduli a bunch of primes $p_1, \dots, p_n$,respectively. We need to normalize the final number a, such that a lies between $[-m/2, m/2)$, where $m = \prod_{1 \le i \le n} p_i$.
2. An early termination technique is used here to speedup the computation in practice. There is already some discussion about the early termination, for example [10]. But here, we use a different and more efficient termination technique. At each step, only one integer called "certificate" is constructed instead of the actual answer mod modulus so far, and has almost the same probability. This technique is easy

to adopt to other cases for example, solving a non-singular integer systems over the rationals.
3. More than one independently certificates can reduce the error probability. Since the error for each independently certificate are independent. In practice, we use two independently certificates.

THEOREM 2. *On the condition that the length of output is correct, the probability that output v is not correct is at most* $\frac{1}{\Omega} + \frac{\log^2(2l\Omega B)}{\log^2(\beta)(\mathsf{M}-\mathsf{M}'-\log(2l\Omega B)/\log(\beta))}$, *where l is the length of the output vector, B is infinity norm of the correct answer, M', number of primes p, such that either* `reveal` (v_p, A, p) *returns false or v_p has length of less than l.*

Note: For a typical problem, such as characteristic polynomial, minimal polynomial, it is easy to choose a reasonable set of prime, such that the second term in error probability is very tiny like one in a million.

PROOF: Let α be the correct answer, and $c = x \cdot \alpha$. Under assumption, l must be equal to length of α. Also Let pair (p_i, v_{p_i}) in the $\mathcal{L}$ in order. We define $\|t\|_\infty$ to be the maximal absolute value of entries of x, for any vector t.

Claim 1:If x is a random vector with entries independently and uniformly chosen from the integer set of $[0, \Omega - 1]$, then $\text{Prob}(|(|x \cdot \alpha) < \|\alpha\|_\infty) \leq \frac{1}{\Omega}$. This claim is true since there is at least one entry of α, whose absolute value is equal to $\|\alpha\|_\infty$, and if with loss of generosity, we assume $|\alpha_1| = \|\alpha\|_\infty$, then any $x_2, \ldots, x_l$, then there is at most one x_1 such that $|x \cdot \alpha| < \|\alpha\|_\infty$.

Claim 2: For an integer m, if a random prime p is uniformly chosen from a cardinality k subset of $\mathcal{P}$, then $\text{Prob}(p|m) \leq \frac{\log(|m|)}{\log(\beta)k}$. This claim is true since there is at most $\log(|m|)/\log(\beta)$ primes which are members of $\mathcal{P}$ and also divide m.

When the early termination first meets at $c_n = c_{n-1}$, let us look at the probability of $c \neq c_n$. We have $c_n = c_{n-1}$, that means $c - c_{n-1} = 0 \pmod{p}_n$, and moreover, $\|c - c_{n-1}\|_\infty \leq 2\|c\|_\infty \leq 2l\Omega B$.

For any fixed n, if a random prime p_n is uniformly chosen from $\mathcal{P}$, and p_n is not rejected, then $\text{Prob}(p_n|(c - c_{n-1}), c \neq c_{n-1}) \leq \frac{\log_2(2l\Omega B)}{\log(\beta)(\mathsf{M}-\mathsf{M}'-n)}$. If n is larger than $\frac{\log(2l\Omega B)}{\log(\beta)}$, then v must be equal to c. Since the constructed $v = c \pmod{m}$, and $m = \prod_{1 \leq i \leq n} p_i > 2\|c\|_\infty$,

So when the early termination condition meets, $\text{Prob}(c \neq c_{i-1}) \leq \frac{(\log(2l\Omega)^2}{(\log(\beta))^2(\mathsf{M}-\mathsf{M}'-\log(2l\Omega B)}\log(\beta))$.

If both $\|c\|_\infty \geq \|\alpha\|_\infty$ and $c = c_n$ are true, then v must be equal to the correct answer. So the probability of $v \neq \alpha$ is at most $\frac{1}{\Omega} + \frac{\log^2(2l\Omega B)^2}{\log^2(\beta)(\mathsf{M}-\mathsf{M}'-\log(2l\Omega B)}\log(\beta))$. □

The following theorem gives additional flexibility in using the strategies for signature computation made available by `GenCRA`. For $n \times n$ matrix A let `shiftminpoly`(A) denote a function which returns x^k`minpoly`$(()A)$ for k such that the degree of the resulting polynomial is n.

THEOREM 3. *Let A be an $n \times n$ real symmetric matrix and let S be a set of nonzero integers of sufficient size that $\epsilon = n^2 \log(n)^2/|S|$ is as small as desired.*

1. *Let D be a diagonal matrix whose n diagonal entries are chosen uniformly at random from S, and let $B = DAD^T$. Then*

$$Prob(\texttt{shiftminpoly}(B) = \texttt{charpoly}(B) \geq 1 - \epsilon.$$

2. *Let Q be a butterfly matrix whose $n\log(n)$ defining entries are chosen uniformly at random from S, and let $B = QAQ^T$. Then*

$$Prob(B \textit{ is in generic rank profile}) \geq 1 - \epsilon.$$

Proof: These preconditionings are discussed in detail in [4] □

For the generic rank profile condition, The butterfly is chosen as preconditioner because the specified matrix QAQ^T can be computed in $O\tilde{}(n^2)$ time. For our purposes here we can afford preconditioning complexity up to $O\tilde{}(n^3)$ time. A general random matrix could be used for the preconditioner, or Toeplitz [12] or sparse preconditioners [4, Section 6]. It is of interest to keep the size of the entries of the resulting matrix as small as possible.

We build three algorithms on this theorem, two using the minimal polynomial (for blackbox and for dense matrices) and one using LDL^T decomposition.

Algorithm: `BBSM [BlackBox Signature by Minpoly ]`
Input: Symmetric matrix A, in blackbox form.
Output: The signature $\sigma(A)$.

1. [Preconditioning may be necessary]
   Let q be a random prime.
   Let $r :=$ `rank`(A, q) and $m_q(x) :=$ `minpoly`(A, q).
   If $\deg(m_p) < n$ and $\deg(m_p) \leq r$, let $B := DAD$ [A blackbox],
   for a random diagonal matrix D with entries chosen from $[1, \ldots, s]$ (or any set of s nonzero integers).
   Otherwise, let $B := A$.
   [Now B has the same signature as $-A$ and its minpoly is a shift of its charpoly.]
   Choose a set $\mathcal{P}$ of primes, and sample size Ω, such that the error probability is as small as desired.
2. Return $\sigma(x^{n-r}\text{GenCRA}(B, \texttt{minpoly}(), \mathcal{P}, \Omega))$.

The `rank`(A, p) and
`minpoly`(A, p) algorithms used are as in [15, 12, 4], for example. Here `minpoly`$()$ and `rank`$()$ run in time $O\tilde{}(nE)$, and are probabilistic with probability of error no more than $1/p$. But `rank`$()$ will never return a value greater than the true rank and `minpoly`$()$ always computes at least a factor of the true minimal polynomial of A mod p, its return status is true. (This is in addition to the possibility of p being unfaithful.) DAD is the blackbox whose matrix-vector product is formed as $y = D(A(Dx))$.

Algorithm: `DSM [Dense Signature by Minpoly ]`
Input: Matrix A, in dense form.
Output: $\sigma(A)$.

Method:

Apply algorithm BBSM, except use $\texttt{rank}(A, p)$ [6] and $\texttt{minpoly}(A, p)$, [5], algorithms which are available for the explicit (dense) matrix representation. Then rank() and minpoly() are are deterministic eliminations running in time $O\tilde{}(n^3)$ and using $O(n^2 d)$ memory Of course, in this case the DAD preconditioning is done explicitly (and cheaply). Again, of course, p may be unfaithful.

Especially for blackbox matrices, it is useful that minpoly computation can suffice, because we have faster algorithms for minpoly than for charpoly. Also note that the minpoly suffices for determining the sign pattern (not full signature) even without preconditioning. To be precise, the sign pattern of A is determined from $s = \text{reverse } \sigma(\texttt{minpoly}(A))$ by replacing positive entries in s by $+$. When the minpoly is of low degree or has small coefficients, this is a great savings. In general, though, BBSM is not a fully memory efficient algorithm because of the size of the σ-revealing vector. It is possible that the technique of [3] could be used to determine the signs using less memory and perhaps less time. This approach deserves further examination.

However, it is not necessary to compute the characteristic or minimal polynomial to answer signature and definiteness questions. Combining Theorem 2, part 3, and theorem 3, part 2, we may work from the LDL^T decomposition of a matrix in generic rank profile. If the matrix should fail to have generic rank profile, this will be detected during the elimination because of the need for pivoting. A note on pivoting: Of course not every symmetric matrix has an LDL^T decomposition. We can use preconditioners Q to assure that QAQ^T has generic rank profile, and in some cases symmetric pivoting could be used instead to avoid the increased entry size caused by preconditioning. Also, the image mod p may fail to have generic rank profile even when the prime is faithful. We could modify the LU to successfully handle such cases, but it is a remote possibility and not worth the overhead. So we reject such primes as if unfaithful.

The LDL^T decomposition based signature algorithm assumes a procedure $\text{lpm}(A, p)$ which returns the vector of leading principal minors up to the first which is zero mod p. The procedure lpmworks by attempting, mod p, the LDL^T decomposition, stopping when a zero pivot is encountered. The procedure compute the vector

$$(1, -D_{1,1}, D_{1,1}D_{2,2}, \ldots, (-i)^k \prod_{1\le i\le k} D_{k,k}),$$

i.e, leading principal minors of A up to the kth with alternative sign. Its return status is true, if no poviting is necessary in the LU elimination, otherwise false. When p is faithful and X is nonzero (so $k < \texttt{rank}(()A)$), it would be possible to usefully continue the elimination. However, this situation is not encountered in practice (for random primes) and there is no need to incur the overhead involved. In any case, $k \le r$, so the lpmmeets the requirements of the σ-revealing function for GenCRA.

Algorithm: DSLU
[Dense matrix Signature by LU-decomposition]
Input: A, an integer symmetric $k \times k$ matrix.
Output: Signature $\sigma(A) = (p, z, n)$.

1. [Preconditioning may be necessary]
   Let q be a random prime.
   Let $v := \text{lpm}(A, q)$, $r := \texttt{rank}(A, q)$.
   If $\texttt{rank}(v) \ne r$, let $B := QAQ^T$,
   for a random integer matrix Q with entries chosen from $1..s$ (or any set of s nonzero integers).
   [Q may be a Toeplitz or Butterfly matrix for speed]
   Otherwise, let $B := A$.
   [Now B has the same signature as A and has generic rank profile.]
   Choose a set $\mathcal{P}$ of primes, and sample size Ω, such that the error probability is as small as desired.
2. [Lift]
   Return $\sigma(x^{n-k}\text{GenCRA}(B, \text{lpm}, \mathcal{P}, \Omega))$.

THEOREM 4. *Let A be an integer blackbox $n \times n$ symmetric matrix, whose matrix-vector product costs e operations and whose entries would be of bit length at most d if they were constructed.*

Algorithms BBSM, DSM, and DSLU are Las Vegas probabilistic algorithm if the Hadamard bound is used and Monte Carlo if early termination is used in the remaindering. Even with early termination, DSM is Las Vegas if the computed integer minimal polynomial is checked by evaluation at A over the integers. This costs $O\tilde{}(n^4)$.Also a minimal polynomial verification by application to the identity could be done after BBSM making it Las Vegas for the sign pattern of the signature.

The BBSM expected run time is $O\tilde{}(neh)$, where h is $O\tilde{}(nd)$ if the Hadamard bound is used and h is the largest bit length of any charpoly coefficient if early termination is used. In either case, BBSM runs in expected time $O\tilde{}(n^2 ed)$. The DSM and DSLU expected run time is $O\tilde{}(n^3 h)$, where h is $O\tilde{}(nd)$ if the Hadamard bound is used and h is the largest bit length of any charpoly coefficient if early termination is used.

In particular if $e \in O\tilde{}(n)$, and $d \in \mathcal{O}(\log(n))$, then the BBSM expected run time is $O\tilde{}(n^3)$ and the DSM, DSLU expected times in $O\tilde{}(n^4)$.

Also, with either of the σ-revealing vectors, the entries tend to grow in proportion to their index. In particular the ith entry is either an $i \times i$ minor or sum of $i \times i$ minors, so is bounded by the Hadamard bound which is in $O\tilde{}(id)$. A heuristic to determine indefiniteness compute the first few vector entries, using many fewer remaindering steps than are required for the later entries. If the sign pattern fails to be constant or strictly alternating the matrix is indefinite. It is an open question whether an conjugacy preconditioning, $A \to QAQ^T$, could give that early entries indicate definiteness with high probability.

4. APPLICATION TO LIE MATRICES

The matrices from Lie group representation are rational matrices, our algorithms in the previous section focus on integer matrices. There are at least two alternative way to adapt. One is to compute the minimal polynomial, or the leading principal minors over the rationals. The rational answers need to be reconstructed at the GenCRA, and it is easy to adopt our GenCRA with early termination technique to this case. The other way is to multiply the matrices by the lcm of denominators of all entries. For these special constructed matrices from Lie group representation, the lcm of denominators of all entries is just a little larger than each individual one, so the latter is a better way. We found for some models the gcd of all the numerators of the dense representation are not trivial, so it could be removed. Here, we present blackbox algorithms to compute the lcm of denominators of all entries, and gcd of numerators of all entries, respectively.

Algorithm: `LDE`
Input:
  A, a rational matrix
  M, sample size
  n, number of trials
Output:
  d, the lcm of denominators of all entries of dense representation of A.
1. for i from 1 to n do
   Choose a random vector x with entries independently and uniformly chosen from $[0, M]$
   $y = Ax$, apply x to A
   $d^{(i)}$= the lcm of denominators of every entry of y.
2. $d := \mathrm{lcm}(d_1, \cdots, d_n)$
3. return d.

Algorithm: `GNE`
Input:
  A, an integer matrix
  M, sample size
  n, number of trials
Output:
  g, the gcd of all entries of dense representation of A.
  Method:
  Replice *lcm* with *gcd* and denominators with numerators in algorithm *LDE.*

The algorithm *LDE* always returns a factor of the true answer. The algorithm *GNE* always returns a multiple of the true answer. For both algorithms, for a rational matrix A, if lde and gne are the lcm of all dominators and gcd all numerators of entries of A, respectively. If let $A' = \frac{d}{g}A$, then A' is an integer matrix. For each individual trial, if all entries of $A'x^{(i)}$ are coprime, then $d^{(i)}$ is equal to the correct answer. Please see [13] for probability analysis. Roughly speaking, for both the error is 2^{-n+1}.

Each matrices from Lie group represenation can be represented as $(\prod_{1\le i\le 120} \frac{1}{r_i}(r_i Ops^{(i)} + I))F$, where $Ops^{(i)}$ is one of the eight sparse matrices called operators, F is also sparse matrix. The algorithm BBSM is very suitable for this case, and we also reduce them to product of 94 sparse matrices by explicitly computing and store $r_i Ops^{(i)} + I$ as a sparse matrix, and multiplying it to left one if the computed sparse matrix is a diagonal matrix. By reducing number of sparse matrices, 20% run time was saved. After multiplying these collapsed matrices by the lcm of denominators of all entries and divding by gcd of numerators of all entries, these matrices can be handled by these algorithms discussed in previous section.

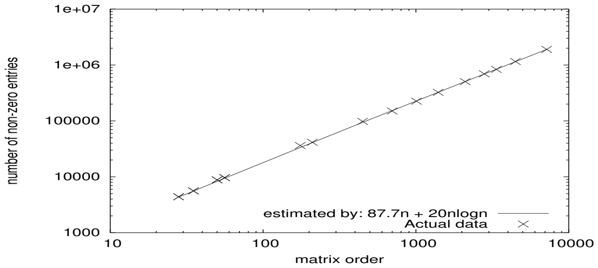
5. EXPERIMENTS

Our goal is to know which algorithm to use and how much resources it will take for computing the signature of the $J_n(f) = J_\rho(f)$, the matrix generated by the facet f for the model of dimension n of the group E8 . Let $J_n(f) = J_\rho(f)$ to emphasize the model dimension. In particular, we focus on the case when the facet is the challenge facet $\nu = 1/40(0, 1, 2, 3, 4, 5, 6, 23)$, and the goal is to verify (or disprove) that $J_n(\nu)$ is positive definite. The three signature algorithms we have discussed are BBSM, DSM, DSLU. We have computed full solutions by each model for the models Q2..Q12, Q22, Q32, Q42, Q52(dimension 1008), and we have computed single modular step costs for every 10th model on up to the largest, Q112(dimension 7168). The experiments involved individual run times up to about one day and verified, where the full computation was done, that $J_n(\nu)$ is indeed positive definite. All computations reported here were performed on a 3.2GHz Xeon processor with 6GB of main memory. Our purpose here is to determine the algorithm to use and resources needed for the full signature computation on examples of all dimensions (up to 7168) and to have tools to estimate the cost for matrices defined by other facets.

The blackbox representation of $J_n(\nu)$ is a product of 121 very sparse factors. Since the cost of computing the characteristic polynomial depends on the cost of applying A to a vector, we examined the number of nonzero elements in total in these factors.

The blackbox step cost is sensitive to the cost e of the single matrix vector product. To guess the expected growth of e with dimension, we measured the total number of nonzero entries in the factors of every 10th model up to Q112. Since the number of factors is fixed and they are extremely sparse we expect a linear or $n \log(n)$ term to dominate. The formula $e = 132.5900691n+13.12471811n \ln(n)$, a least squares fit to the data, is plotted in figure 1 along with the data values. The fit is extremely good and we see that e is only slightly superlinear. Therefore we expect the runtime of characteristic polynomial computation mod a single prime to grow at a rate only slightly above quadratic in n.

Figure 1: Blackbox number of nonzeroes

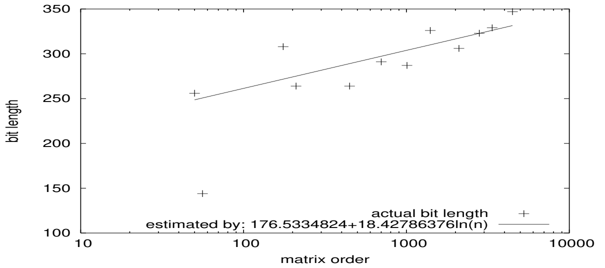


Each algorithm involves a series of steps mod primes p_i, each step being a computation of σ-revealing vector (a minpoly or diagonal of LDL^T-decomposition) mod p_i. The computation is finished when the CRA remaindering has given an image of the vector with modulus, $M = 2\prod p_i$, sufficiently large to recover the actual (signed) vector entries. If we know a bound d for the length of the entries of $J_n(\nu)$, we have by the Hadamard bound a maximal length

$b = n(\log(n)/2 + d) + 1$ for M.

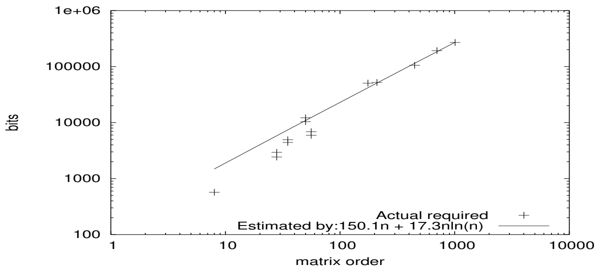
the Dense matrices are computed by applying the black box to the columns of the identity matrix (over $\mathbb{Z}$, not modularly), and so are sensitive to the length of the actual integer entries of the expanded (and dense). Let d be the length of the largest entry. We do not have a theory to predict how matrix entry bit length d may depend on the model dimension for a fixed facet. We have plotted the bit length d of the largest entry of $J_n(\nu)$ along with the fitted curve $d = 67.7 + 33.2\log(n)$ in figure 2. One sees the $\log(n)$ playing a stronger role than that for the nonzeroes in the blackbox, but the fit to the data is poorer and likely to have less predictive value. We noted that the n^2 entries of A all have about the same length, so $n^2 d$ accurately describes the storage required for A.

Figure 2: Bit length of entries of these dense matrices



It will also be useful to use d in estimating the number of remaindering steps. The needed bit length b for the modulus M is at most 1 bit (for sign) more than the length of the Hadamard bound. The rows norms are no more than $\sqrt{n(e^d)^2}$, so their product has length bounded by $n/2(\log(n) + 2d)$. Reasonably consistent with this prediction is the number of bits used when we ran the full algorithms on $J_n(\nu)$ for n up to 1008 (model Q52). The curve in figure 3 is a least squares fit, $b = 150.1n + 17.30nln(n)$, The number of bits for the charpoly coefficients (sums of minors) are expected to be slightly larger than for the entries of the leading principal minors vector, but most likely, the final term, determinant of $J_n(\nu)$, is dominant. The number of bits needed appears to be slightly superlinear, not quite as large as the worst case Hadamard bound level.

Figure 3: Bit length of final modulus

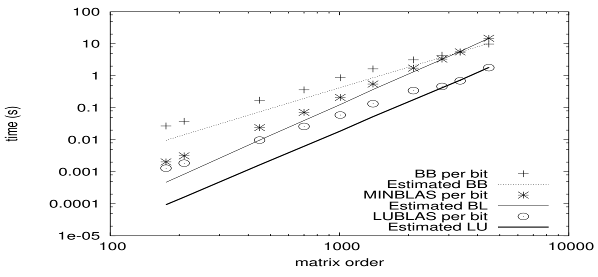


Next we consider the costs of the modular steps. Let step(s) denote the cost per step which varies with length, s, of the prime, but is essentially constant among primes of the same length. The total time is $b/s \times \text{step}(s)$ when using s bit primes. Thus we see the advantage of minimizing $\text{step}(s)/s$, the cost per bit. After discussing $\text{step}(s)/s$ and determining it's value, for each algorithm, we will know how to estimate the full runtimes of the algorithms.

We continue to tune the implementation of the modular steps. We expect to be able to reduce the LU step cost by half, as an elimination better tailored to symmetric matrices can be done, while still taking advantage of block operations using BLAS. For the blackbox algorithm, block methods may be able to reduce the costs somewhat, and the DSM may perform relatively better than here if the dense `charpoly()` algorithm [5] is used rather than preconditioning. At any rate, let us see the performance of the modular steps of the three algorithms, BBSM, DSM, DSLU as currently implemented. Asymptotically, the BBSM step runtime, s, is expected to grow in proportion to ne. There was a good fit to $s = 61n^2\log(n)$ nanoseconds per bit. (recall that e has a $n\log(n)$ dominant term). . The algorithms DSM and DSLU depend on the construction of the full integer matrix of $J_n(\nu)$, which is done once. The step times consist of computing a modular image of that and proceeding with an elimination on the image. The implementation of the Krylov matrix construction in the `minpoly()` algorithm of DSM, uses a technique of recursive doubling to better exploit fast matrix multiplication, and gets a step time s which fits the formula $t = 1.8n^3 + 0.16n^3\log(n)$ nanoseconds per bit. $t = 0.37n^3\log(n)$ nanoseconds per bit. quite well. The DSLU step fits $t = 0.3838888362n^3$ nanosec/bit. But these formulas have to be worked to the bit cost level!!!

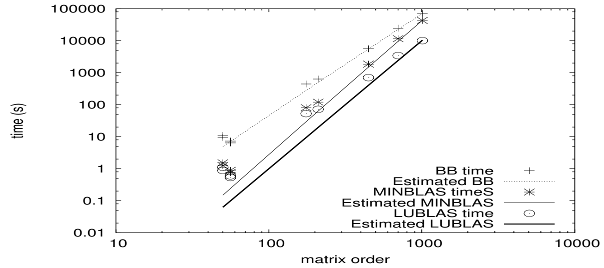
The log factor for DSM step can be removed, so it remains possible that it can become competitive or participate with BBSM in a heterogeneous distributed computation, running on the machines with larger memory. The time formula for the dense matrices should remain valid as long as the modular step fits in main memory. After than swapping would dramatically increase times. Assuming a memory capable of holding a 2GB virtual memory, this would allow for n^2 words, so $n < 2^{15}$. For all practical purposes, the blackbox step has no memory limitation.

Figure 4: Step time (per bit of M)



The total runtime then involves the time per bit in the modular steps times the number of bits in the σ-revealing vector. The Chinese remaindering adds a cost that is similar for each but large enough to mute somewhat the effect of the differing step costs. However with the early termination technique of the previous section this remaindering is a smaller factor than in earlier timings. For instance, without the asymptotic improvement in the termination strategy our time for x was y, now it is z. The two dense algorithm also incur a lower order cost for the creation of the expanded integer matrix initially. The least squares fit formulas for the overall run times in nanoseconds are BBSM: $t = 10050n^3 \log(n)$, DSM: $t = 6.08n^4 \log(n)$, and DSLU: $t = 1.45n^4 \log(n)$.Theory predicts a second log factor in the BBSM time, coming from the blackbox e. However that effect is weak and the fit was poorer.

Figure 5: time



Also the memory needed to store the σ-revealing vector mutes the memory advantage BBSM. However the modular images of the vector are easily stored on hard disk and manipulated from there, so the memory advantage remains real. The crossover point of the runtime formulas for BBSM and DSLU is around $n = 6931$, however the formula $t = 10.02n^4$ fits the DSLU data about as well as our formula above. Using that form, the crossover point is at $n = 9150$.

6. CONCLUSION

We have demonstrated that we can compute (on current hardware) the signature of a dense $n \times n$ integer matrix having entries of bit length around $\log(n)$ in a minute if $n \leq 200$, in three hours if $n = 1000$ and (projected) in a CPU year for $J_{7168}(\nu)$. Beyond that size, using algorithm BBSM, the time grows at a rate slightly higher than n^3 and memory is not a constraint (except for storage of the sparse factors of the blackbox). However, we conclude that algorithm DSLU serves best for matrices of dimension $n <$ 7000. It is tossup between DSLU and BBSM for dimensions $7000 \leq n \leq 9000$ and BBSM is superior beyond that. DSLU time grows slightly above n^4.

For $J_{7168}(\nu)$ DSLU is requires explicit use of file storage of the expanded matrix and both algorithms should do this for intermediate results (modular images of σ-revealing vectors), because of the large size. We have not measured the cost of this file manipulation. At the crossover about one CPU year is required and DSLU needs a large memory. The run time is expected to be about a CPU year, so Parallel computation is desirable (and is quite straightforward for either algorithm), on distributed or shared memory hardware.

It is an open question whether definiteness can be determined fundamentally faster than signature. There is a fast Monte Carlo algorithm for rank, hence for distinguishing semidefinite from definite matrices. We have sketched a heuristic that sometimes can determine indefiniteness much faster than the signature computation.

To provide for the needs of Lie group representation studies, both BBSM and DSLU will be further refined and their parallel implementation developed. Also the judicious incorporation of numeric computation is a possibility.

7. REFERENCES

[1] J Adams. Integral models of representations of weyl groups.

[2] D. Barbasch. Unitary spherical spectrum for split classical groups.

[3] H. Bronnimann, I.Z. Emiris, V.Y. Pan, and S. Pion. Sign determination in residue number systems. *Theoret. Comput. Sci.*, 210:173197, 1999.

[4] L. Chen, W. Eberly, E. Kaltofen, B. D. Saunders, W. J. Turner, and G. Villard. Efficient matrix preconditioners for black box linear algebra. *Linear Algebra and Applications*, 343-344:119–146, 2002.

[5] C.Pernet and Z. Wan. Lu based algorithms for the characteristic polynomial over a finite field. In *Poster, ISSAC'03*. ACM Press, 2003.

[6] Jean-Guillaume Dumas, Pascal Giorgi, and Clément Pernet. FFPACK: Finite field linear algebra package.

[7] F. R. Gantmacher. *The Theory of Matrices*. Chelsea, New York, NY, 1959.

[8] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, third edition, 1996.

[9] James E. Humphreys. Cambridge University Press, Cambridge, 1990.

[10] I.Z.Emiris. A complete implementation for computing general dimensional convex hulls. *Inter. J. Comput. Geom. Appl.*, 8:223–253, 1998.

[11] E. Kaltofen and A. Lobo. On rank properties of Toeplitz matrices over finite fields. In *ISSAC 96 Proc. 1996 Internat. Symp. Symbolic Algebraic Comput.*, pages 241–249.

[12] E. Kaltofen and B. D. Saunders. On Wiedemann's method of solving sparse linear systems. In H. F. Mattson, T. Mora, and T. R. N. Rao, editors, *Proc. AAECC-9*, volume 539 of *Lect. Notes Comput. Sci.*, pages 29–38, Heidelberg, Germany, 1991. Springer Verlag.

[13] B. David Saunders and Zhendong Wan. Smith normal form of dense integer matrices fast algorithms into practice. In *Proc. ISSAC'04*, pages 274–281. ACM Press, 2004.

[14] John R. Stembridge. Explicit matrices for irreducible representations of Weyl groups. *Represent. Theory (electronic)*, 8:267–289, 2004.

[15] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theory*, IT-32:54–62, 1986.