Chapter 7

Linear Algebra

This chapter is about exact matrix algebra over the rational numbers and cyclotomic fields. Algorithms for linear algebra over exact fields are necessary in order to implement the modular symbols algorithms that we will describe in Chapter 7. This chapter partly overlaps with [Coh93, §2.1–2.4].

7.1 Echelon Forms of Matrices

Definition 7.1.1 (Reduced Row Echelon Form)**.** A matrix is in *(reduced row) echelon form* if each row in the matrix has more zeros at the beginning than the row above it, the first nonzero entry of every row is 1, and the first nonzero entry of any row is the only nonzero entry in its column.

Given a matrix A, there is another matrix B such that B is obtained from A by left multiplication by an invertible matrix and B is in reduced row echelon form. This matrix B is called the echelon form of A. It is unique.

A *pivot column* of A is a column of A such that the reduced row echelon form of A contains a leading 1.

Example 7.1.2. The following matrix is not in reduced row echelon form:

$$\begin{pmatrix} 14 & 2 & 7 & 228 & -224 \\ 0 & 0 & 3 & 78 & -70 \\ 0 & 0 & 0 & -405 & 381 \end{pmatrix}$$

The reduced row echelon form of the above matrix is

$$\begin{pmatrix} 1 & \frac{1}{7} & 0 & 0 & -\frac{1174}{945} \\ 0 & 0 & 1 & 0 & \frac{152}{135} \\ 0 & 0 & 0 & 1 & -\frac{127}{135} \end{pmatrix}$$

Notice that the entries of the reduced row echelon form can be rationals with large denominators even though the entries of the original matrix A are integers.

Another example is the simple looking matrix

$$\begin{pmatrix} -9 & 6 & 7 & 3 & 1 & 0 & 0 & 0 \\ -10 & 3 & 8 & 2 & 0 & 1 & 0 & 0 \\ 3 & -6 & 2 & 8 & 0 & 0 & 1 & 0 \\ -8 & -6 & -8 & 6 & 0 & 0 & 0 & 1 \end{pmatrix}$$

whose echelon form is

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \frac{42}{1025} & -\frac{92}{1025} & \frac{1}{25} & -\frac{9}{205} \\ 0 & 1 & 0 & 0 & \frac{716}{3075} & -\frac{641}{3075} & -\frac{2}{75} & -\frac{7}{615} \\ 0 & 0 & 1 & 0 & -\frac{83}{1025} & \frac{133}{1025} & \frac{1}{25} & -\frac{23}{410} \\ 0 & 0 & 0 & 1 & \frac{184}{1025} & -\frac{159}{1025} & \frac{2}{25} & \frac{9}{410} \end{pmatrix}.$$

A basic fact is that two matrices A and B have the same reduced row echelon form if and only if there is an invertible matrix E such that $EA = B$. Also, many standard operations in linear algebra, e.g., computation of the kernel of a linear map, intersection of subspaces, membership checking, etc., can be encoded as a question about computing the (reduced row) echelon form of a matrix.

The following is a naive algorithm for computing the echelon form of a matrix.

Algorithm 7.1.3 (Gauss Elimination)**.** *Given an $m \times n$ matrix A over a field, the algorithm outputs the reduced row echelon form of A. Write $a_{i,j}$ for the i, j entry of A, where $0 \leq i \leq m-1$ and $0 \leq j \leq n-1$.*

1. [Initialize] Set $k = 0$.
2. [Clear each column] For each column $c = 0, 1, \ldots, n-1$, clear the cth column as follows:
   (a) [First nonzero] Find the smallest r such that $a_{r,c} \neq 0$, or if there is no such r go to the next column.
   (b) [Rescale] Replace row r of A by $\frac{1}{a_{r,c}}$ times row r.
   (c) [Swap] Swap row r with row k.
   (d) [Clear] For each $i = 0, \ldots, m-1$ with $i \neq k$, if $a_{i,c} \neq 0$, add $-a_{i,c}$ times row k of A to row i to clear the leading entry of the ith row.
   (e) [Increment] Set $k = k + 1$.

This algorithm takes $O(mn^2)$ arithmetic operation in the base field, where A is an $m \times n$ matrix. If the base field is $\mathbb{Q}$, the entries can become huge and arithmetic operations are then very expensive. See Section 7.3 for ways to mitigate this problem.

To conclude this section we mention how to convert a few standard problems into questions about reduced row echelon forms of matrices. Note that one can also phrase some of these answers in terms of the echelon form, which might be easier to compute, or an LUP decomposition (lower triangular times upper triangular times permutation matrix), which the numerical analysts use.

1. **Kernel of A:** Since passing to the reduced row echelon form of A is the same as multiplying on the left by an invertible matrix, the kernel of the reduce row echelon form E of A is the same as the kernel of A. There is a basis vector of $\ker(E)$ that corresponds to each non-pivot column of E. That vector has a 1 at the non-pivot column, 0's at all other non-pivot columns, and for each pivot column, the negative of the entry of A at the non-pivot column in the row with that pivot element.

2. **Intersection of Subspaces:** Suppose W_1 and W_2 are subspace of a finite-dimensional vector space V. Let A_1 and A_2 be matrices whose columns form a basis for W_1 and W_2, respectively. Let $A = [A_1|A_2]$ be the augmented matrix formed from A_1 and A_2. Let K be the kernel of the linear transformation defined by A. Then K is isomorphic to the desired intersection. To write down the intersection explicitly, suppose that $\dim(V) \leq \dim(W)$ and do the following: For each b in a basis for K, write down the linear combination of a basis for V got by taking the first $\dim(V)$ entries of the vector b. The fact that b is in $\mathrm{Ker}(A)$ implies that the vector we just wrote down is also in W. We took V to have smaller dimension just so that the linear combinations in the intersection could be written down slightly more quickly.

7.2 Rational Reconstruction

Rational reconstruction is a process that allows one to sometimes lift an integer modulo m uniquely to a bounded rational number.

Algorithm 7.2.1 (Rational Reconstruction)**.** *Given an integer $a \geq 0$ and an integer $m > 1$, this algorithm computes the numerator n and denominator d of the unique rational number n/d, if it exists, with*

$$|n|, d \leq \sqrt{\frac{m}{2}} \qquad \textit{and} \qquad n \equiv ad \pmod{m}, \tag{7.2.1}$$

or reports that there is no such number.

1. [Reduce mod m] Replace a be the least integer between 0 and $m-1$ that is congruent to a modulo m.
2. [Trivial cases] If $a = 0$ or $a = 1$, return a.
3. [Initialize] Let $b = \sqrt{m/2}$, $u = m$, $v = a$, and set $U = (1, 0, u)$ and $V = (0, 1, v)$. Use the notation U_i and V_i to refer to the ith entries of U, V, for $i = 0, 1, 2$.
4. [Iterate] Do the following as long as $|V_2| > b$: Set $q = \lfloor U_2/V_2 \rfloor$, set $T = U - qV$, set $U = V$ and $V = T$.
5. [Numerator and Denominator] Set $d = |V_1|$ and $n = V_2$.
6. [Good?] If $d \leq b$ and $\gcd(n, d) = 1$ return n/d, otherwise report that there is no raitonal number as in (7.2.1).

Algorithm 7.2.1 for rational reconstruction is described (with proof) in [Knu, pg.656–657] as the solution to exercise 51 on page 379. See, in particular, the paragraph right in the middle of page 657, which describes the algorithm. Knuth says this rational reconstruction algorithm is due to Wang, Kornerup, and Gregory from around 1983.

We now give an indication of why Algorithm 7.2.1 computes the rational reconstruction of $a \pmod{m}$, leaving the precise details and uniqueness to [Knu, pg.656–657]. At each step in Algorithm 7.2.1, the 3-tuple $V = (v_0, v_1, v_2)$ satisfies

$$m \cdot v_0 + a \cdot v_1 = v_2, \tag{7.2.2}$$

and similarly for U. When computing the usual extended gcd, at the end $v_2 = \gcd(a, m)$ and v_0, v_1 give a representation of the v_2 as a $\mathbb{Z}$-linear combination of m and a. In Algorithm 7.2.1, we are instead interested in finding a rational number n/d such that $n \equiv a \cdot d \pmod{m}$. If we set $n = v_2$ and $d = v_1$ in (7.2.2) and rearrange, we obtain

$$n = a \cdot d + m \cdot v_0.$$

Thus at *every* step of the algorithm we find a rational number n/d such that $n \equiv ad \pmod{m}$. The problem at intermediate steps is that, e.g., v_0 could be 0, or n or d could be too large.

Example 7.2.2. We compute an example using SAGE.

```
----------------------------------------------------
p = 389
k = GF(p)
a = k(7/13); print "a = %s (mod %s)"%(a,p)
a.rational_reconstruction()
----------------------------------------------------
--> a = 210 (mod 389)
--> 7/13
```

7.3 Echelon Forms over $\mathbb{Q}$

A difficulty with computation of the echelon form of a matrix over the rational numbers is that arithmetic with large rational numbers is potentially time consuming; each addition potentially requires a gcd and numerous additions and multiplications of integers. Moreover, the entries of A during intermediate steps of Algorithm 7.1.3 can be huge even though the entries of A and the answer are small. For example, suppose A is an invertible square matrix. Then the echelon form of A is the identity matrix, but during intermediate steps the numbers involved could be quite large. One technique for mitigating this is to compute the echelon form using a multi-modular method.

If A is a matrix with rational entries, let $H(A)$ be the *height* of A, which is the maximum of the absolute values of the numerators and denominators of all entries of A.

Algorithm 7.3.1 (Multimodular Algorithm for Computing Echelon Form)**.** *Given an $m \times n$ matrix A with entries in $\mathbb{Q}$, this algorithm computes the reduced row echelon form of A.*

1. Rescale the input matrix A to have integer entries. This does not change the echelon form and makes reduction modulo many primes easier. We may thus assume A has integer entries.
2. Let c be a guess for the height of the echelon form.
3. List successive primes $p_1, p_2, \ldots$ such that the product of the p_i is bigger than $n \cdot c \cdot H(A) + 1$, where n is the number of columns of A.
4. Compute the echelon forms B_i of the reduction $A \pmod{p_i}$ using, e.g., Algorithm 7.1.3 or any other echelon algorithm. This is where most of the work takes place, and can be optimized by using asymptotically fast algorithms, see Section 7.4.
5. Discard any B_i whose pivot column list is not maximal among pivot lists of all B_j found so far. (The pivot list associated to B_i is the ordered list of integers k such that the kth column of B_j is a pivot column. We mean maximal with respect to the following ordering on integer sequences: shorter integer sequences are smaller, and if two sequences have the same length, then order in reverse lexicographic order. Thus $[1, 2]$ is smaller than $[1, 2, 3]$, and $[1, 2, 7]$ is smaller than $[1, 2, 5]$. Think of maximal as "optimal", i.e., best possible pivot columns.)
6. Use the Chinese remainder theorem to find a matrix B with integer entries such that $B \equiv B_i \pmod{p_i}$ for all p_i.
7. Use rational reconstruction (Algorithm 7.2.1) to try to find a matrix C whose coefficients are rational numbers n/r such that $|n|, r \leq \sqrt{M/2}$, where $M = \prod p_i$, and $C \equiv B_i \pmod{p_i}$ for each prime p. If rational reconstruction fails, compute a few more echelon forms mod the next few primes (using the above steps), and attempt rational reconstruction again. Let E be the matrix over $\mathbb{Q}$ so obtained. (A trick here is to keep track of denominators found so far to avoid doing very many rational reconstructions.)
8. Compute the denominator d of E, i.e., the smallest positive integer such that dE has integer entries. If
$$H(dE) \cdot H(A) \cdot n \leq \prod p_i, \tag{7.3.1}$$
then E is the reduced row echelon form of A. If not, repeat the above steps with a few more primes.

Proof. We prove that if (7.3.1) is satisfied, then the matrix E computed by the algorithm really is the reduced row echelon form R of A. First note that E is in reduced row echelon form since the set of pivot columns of all matrices B_i used to construct E are the same, so the pivot columns of E are the same as those of any B_i and all other entries in the B_i pivot columns are 0, so the other entries of E in the pivot columns are also 0.

Recall from the end of Section 7.1 that a matrix whose columns are a basis for the kernel of A can be obtained from the reduced row echelon form R. Let K be the matrix whose columns are the vectors in the kernel algorithm applied to E, so $EK = 0$. Since the reduced row echelon form is got by left multiplying by an invertible matrix, for each i, there is an invertible matrices C_i mod p_i such that $A = C_i B_i$ so

$$A \cdot dK \equiv dC_i B_i K \equiv C_i \cdot dE \cdot K \equiv 0 \pmod{p_i}.$$

Since dK and A are integer matrices,

$$A \cdot dK \equiv 0 \quad (\text{mod } \prod p_i).$$

The integer entries of $A \cdot dK$ are all at most $H(A) \cdot H(dK) \cdot n$, where n is the number of columns of A. Since $H(K) \leq H(E)$, the bound (7.3.1) implies that $A \cdot dK = 0$. Thus $AK = 0$, so $\mathrm{Ker}(E) \subset \mathrm{Ker}(A)$. On the other hand, the rank of E equals the rank of each B_i (since the pivot columns are the same), so

$$\mathrm{rank}(E) = \mathrm{rank}(B_i) = \mathrm{rank}(A \pmod{p_i}) \leq \mathrm{rank}(A).$$

Thus $\dim(\mathrm{Ker}(A)) \leq \dim(\mathrm{Ker}(E))$, and combining this with the bound obtained above we see that $\mathrm{Ker}(E) = \mathrm{Ker}(A)$. This implies that E is the reduced row echelon form of A, since two matrices have the same kernel if and only if they have the same reduced row echelon form (the echelon form is an invariant of the row space, and the kernel is the orthogonal complement of the row space).

The reason for Step 5 is that the matrices B_i need *not* be the reduction of R modulo p_i, and indeed this reduction might not even be defined, e.g., if p_i divides the denominator of some element of R, then this reduction makes no sense. For example, set $p = p_i$ and suppose $A = \left(\begin{smallmatrix} p & 1 \\ 0 & 0 \end{smallmatrix}\right)$. Then $R = \left(\begin{smallmatrix} 1 & 1/p \\ 0 & 0 \end{smallmatrix}\right)$, which has no reduction modulo p; also, the reduction of A modulo B_i is $B_i = \left(\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)$ (mod p), which is already in reduced row echelon form. However if we were to combine B_i with the echelon form of A modulo another prime, the result could never be lifted using rational reconstruction. Thus the reason we exclude all B_i with non-maximal pivot column sequence is so that a rational reconstruction will exist. There are only finitely many primes that divide denominators of entries of R, so eventually all B_i will have maximal pivot column sequences, i.e., are the reduction of the true reduced row echelon form R, so the algorithm terminates. $\square$

Remark 7.3.2.

1. There is also an iterative p-adic method for lifting solutions modulo p to an equation $Ax = v$ to characteristic 0. This is supposed to be faster for a single solution, but slower for lifting many solutions. See [Fau99] for a discussion of sparse linear algebra methods (in the context of computation of Groebner basis).

2. Algorithm 7.3.1, with *sparse* matrices seems to work particularly well in practice. A simple but helpful modification to Algorithm 7.1.3 in the sparse case is to clear each column using a row with a minimal number of nonzero entries, so as to reduce the amount of "fill in" (denseness) of the matrix. There are much more sophisticated methods along these lines called "intelligent Gauss elimination". (Cryptographers are interested in linear algebra mod p with huge sparse matrices, since they come up in attacks on the discrete log problem and integer factorization.)

One can adapt Algorithm 7.3.1 to computation of reduced row echelon forms of matrices A over cyclotomic fields $\mathbb{Q}(\zeta_n)$. Assume A has denominator 1. Let p be a prime that splits completely in $\mathbb{Q}(\zeta_n)$. Compute the homomorphisms $f_i : \mathbb{Z}_p[\zeta_n] \to \mathbb{F}_p$ by finding the elements of order n in $\mathbb{F}_p^*$. Then compute the mod p matrix $f_i(A)$ for each i, and find its reduced row echelon form. Taken together, the maps f_i together induce an isomorphism $\Psi : \mathbb{F}_p[X]/\Phi_n(X) \cong \mathbb{F}_p^d$, where $\Phi_n(X)$ is the nth cyclotomic polynomial and d is its degree. It's easy to compute $\Psi(f(x))$ by evaluating $f(x)$ at each element of order n in $\mathbb{F}_p$. To compute Ψ^{-1} simply use linear algebra over $\mathbb{F}_p$ to invert a matrix that represents Ψ. Use Ψ^{-1} to compute the reduced row echelon form of $A \pmod{p}$, where (p) is the non-prime ideal in $\mathbb{Z}[\zeta_n]$ generated by p. Do this for several primes p, and use rational reconstruction on each coefficient of each power of ζ_n, to recover the echelon form of A.

7.4 Echelon Forms via Matrix Multiplication

In this section we show how to compute echelon forms using matrix multiplication.[1] Computing echelon forms using only matrix multiplication is valuable because there are asmptotically fast algorithms for matrix multiplication, and actual implementations of linear algebra libraries often include highly-optimized matrix multiplication algorithms.

In [Str69], Strassen also sketched the following algorithm for computing the inverse of a square matrix using matrix multiplication. Suppose the input matrix is $2^n \times 2^n$ and we write it in block form as $\left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right)$ where A, B, C, D are all $2^{n-1} \times 2^{n-1}$ matrices. Assume that any intermediate matrices below that we invert are invertible. Consider the augmented matrix

$$\left(\begin{array}{cc|cc} A & B & I & 0 \\ C & D & 0 & I \end{array}\right)$$

Multiply the top row by A^{-1} to obtain:

$$\left(\begin{array}{cc|cc} I & A^{-1}B & A^{-1} & 0 \\ C & D & 0 & I \end{array}\right),$$

[1]This section is based on a talk by Allan Steel.

and write $E = A^{-1}B$. Subtract C times the firspt row to the second row to get

$$\left(\begin{array}{cc|cc} I & E & A^{-1} & 0 \\ 0 & D-CE & -CA^{-1} & I \end{array}\right)$$

Set $F = D - CE$ and multiply by the bottom row by F^{-1} on the left to obtain

$$\left(\begin{array}{cc|cc} I & E & A^{-1} & 0 \\ 0 & I & -F^{-1}CA^{-1} & F^{-1} \end{array}\right)$$

Set $G = -F^{-1}CA^{-1}$, and subtract E times the second from the first row to arrive at

$$\left(\begin{array}{cc|cc} I & 0 & A^{-1}-EG & -EF^{-1} \\ 0 & I & G & F^{-1} \end{array}\right).$$

This algorithm can be extended to matrices A of any size by padding the matrices by 1 in the lower right to make A a $2^n \times 2^n$ matrix. If one of the intermediate matrices is not invertible, apply a random permutation of the rows of A then retry the algorithm. Applying a random permutation of the rows is the same as multiplying A on the left by an invertible matrix E. If we compute $B = (EA)^{-1}$ we recover A^{-1} by noting that $B = A^{-1}E^{-1}$, so $A^{-1} = BE$. I.e., to obtain A^{-1} we permute the *columns* of B using the permutation that we applied to the rows of A.

Algorithm 7.4.1 (Inverse Using Matrix Multiplication)**.** *Given an $m \times m$ matrix M this algorithm computes the inverse of M.*

1. [2-power] Let 2^n be the smallest power of 2 such that $m \leq 2^n$, so $n = \lceil \log_2(m) \rceil$.
2. [Extend] Replace M with an augmented matrix $\left(\begin{smallmatrix} M & 0 \\ 0 & I_{2^n-m}\end{smallmatrix}\right)$ where I_{2^n-m} is the $(2^n-m)\times(2^n-m)$ identity matrix. Note that $\left(\begin{smallmatrix} M & 0 \\ 0 & I_{2^n-m}\end{smallmatrix}\right)^{-1} = \left(\begin{smallmatrix} M^{-1} & 0 \\ 0 & I_{2^n-m}\end{smallmatrix}\right)$, so it suffices to compute the inverse of this new matrix (which we call M for the rest of the algorithm).
3. [Submatrices] Write $M = \left(\begin{smallmatrix} A & B \\ C & D\end{smallmatrix}\right)$ with each of A, B, C, D a $2^{n-1} \times 2^{n-1}$ matrix.
4. [Multiplies and Inverses] Compute each of $E = A^{-1}B$, $F = D - CE$ and $G = -F^{-1}CA^{-1}$. The inverses involve $2^{n-1} \times 2^{n-1}$ and we do them recursively using this algorithm, except if $n = 2$. If either A or F is not invertible choose a random permutation σ of 2^n, apply this permutation to the rows of M, and go to step 3.
5. [Compute Inverse] Compute

$$M' = \begin{pmatrix} A^{-1} - EG & -EF^{-1} \\ G & F^{-1} \end{pmatrix}.$$

6. [Apply Permutation] Apply the permutation σ to the columns of M'. Output the resulting matrix, which is M^{-1}. (If $m \neq 2^n$ output only the upper left $m \times m$ submatrix of this matrix.)

Next we explain how to compute echelon forms of matrices using the above algorithm.

Algorithm 7.4.2 (Asymptotically Fast Echelon Form)**.** *Given a matrix A over the rational numbers (or a number field) this algorithm computes the echelon form of A.*

1. [Find pivots] Choose a random prime p (coprime to the denominator of any entry of A) and compute the echelon form of $A \pmod p$, e.g., using Algorithm 7.1.3. Let $c_0, \ldots, c_{n-1}$ be the pivot columns of $A \pmod p$. When computing the echelon form save the numbers of the rows $r_0, \ldots, r_{n-1}$ used to clear each column. (For all but finitely many q these are also the pivot columns of A.)
2. [Extract submatrix] Extract the $n \times n$ submatrix B of A whose entries are A_{r_i, c_j} for $0 \leq i, j \leq n-1$.
3. [Compute inverse] Using, e.g., Algorithm 7.4.1 compute the inverse B^{-1} of B. Note that B must be invertible since its reduction modulo p is invertible.
4. [Multiply] Let C be the matrix whose rows are the rows $r_0, \ldots, r_{n-1}$ of A. Compute $E = B^{-1}C$. If E is not in echelon form go to step 1.
5. [Done?] Write down a matrix D whose columns are a basis for $\ker(E)$ as explained in Page 99 (this requires no arithmetic operations since E is in echelon form). Let F be the matrix whose rows are the rows of A other than rows $r_0, \ldots, r_{n-1}$. Compute the product FD. If $FD = 0$ output E, which is the echelon form of A. If $FD \neq 0$ go to step 1 and run the whole algorithm again.

Proof. We prove both that the algorithm terminates and that when it terminates the matrix E is the echelon form of A. First, suppose that the algorithm terminates. Then by assumption E is in echelon form and is obtained by multiply C on the left by an invertible matrix, so E must be *the* echelon form of C. The rows of C are a subset of those of A, so the rows of E are a subset of the rows of the echelon form of A. Thus $\ker(A) \subset \ker(E)$. To show that E equals the echelon form of A, we just need to verify that $\ker(E) \subset \ker(A)$, i.e., that $AD = 0$, where D is as in step 5. Since E is the echelon form of C, we know that $CD = 0$. By step 5 we also know that $FD = 0$. Thus $AD = 0$, since the rows of A are the union of the rows of F and C.

It remains to prove that the algorithm terminates. Let E be the echelon form of A. For all but finitely many primes p (i.e., any prime where $A \pmod p$ has full rank) the echelon form of $A \pmod p$ equals $E \pmod p$. For any such prime p the pivot columns of $E \pmod p$ are the pivot columns of E, so the algorithm will terminate for that choice of p. $\square$

7.5 Decomposing Spaces Under the Action of Matrix

Efficiently solving the following problem is a crucial step in computing a basis of eigenforms for a space of modular forms (e.g., using modular symbols).

Problem 7.5.1. Suppose T is an $n \times n$ matrix with entries in K and that the minimal polynomial of T is square free and has degree n. View T as acting on $V = K^n$. Find a simple module decomposition $W_0 \oplus \cdots \oplus W_m$ of V as a direct sum of simple $K[T]$-modules. Equivalently, find an invertible matrix A such that $A^{-1}TA$ is a block direct sum of matrices $T_0, \ldots, T_m$ such that the minimal polynomial of each T_i is irreducible.

Remark 7.5.2. A natural generalization of Problem 7.5.1 to arbitrary matrices is to find the *rational Jordan form* (or rational canonical form) of T. This is like the usual Jordan form, but the resulting matrix is over $\mathbb{Q}$ and the summands of the matrix corresponding to eigenvalues are replaced by matrices whose minimal polynomials are the minimal polynomials (over $\mathbb{Q}$) of the eigenvalues. The rational Jordan form was extensively studied by Geisbrecht in his Ph.D. thesis and many successive papers, where he analyzes the complexity of his algorithms, and observes that the limiting step is factoring polynomials over K. The reason is that given a polynomial $f \in K[x]$, one can easily write down a matrix T such that one can can read off the factorization of f from the rational Jordan form of T (see also [Ste97]).

7.5.1 Polynomials

There are several linear algebra algorithms that involve polynomials and are important to modular forms algorithms.

Computation of characteristic polynomials of matrices is crucial to modular forms computations. There are many approaches to this problems: compute $\det(xI - A)$ symbolically (bad), compute the traces of the powers of A (bad), or compute the Hessenberg form modulo many primes and use CRT (bad, see [Coh93, §2.2.4]). A more sophisticated method is to compute the rational canonical form of A using Giesbrecht's algorithm[2] (see [GS02]), which involve computing Krylov subspaces (i.e., cyclic spaces spanned by a single vector), and building up the whole space on which A acts. This latter method is a generalization of Wiedemann's algorithm for computing minimal polynomials (see Section 7.5.2), but with more structure to handle the case when the characteristic polynomial is not equal to the minimal polynomial.

Factorization of polynomials in $\mathbb{Q}[X]$ (or over number fields) is an important step in computing an explicit basis Hecke eigenforms for spaces of modular forms. The best algorithm is the van Hoeij method, which uses LLL in a novel way to solve the optimization problems that come up in trying to lift factorizations mod p to $\mathbb{Z}$. See [KB06].

[2] Allan Steel also independently invented a similar algorithm.

7.5.2 Wiedemann's Minimal Polynomial Algorithm

In this section we describe an algorithm due to Wiedemann for computing the minimal polynomial of an $n \times n$ matrix A over a field.

Choose a random vector v and compute the iterates

$$v_0 = v, \quad v_1 = A(v), \quad v_2 = A^2(v), \quad \ldots, \quad v_{2n-1} = A^{2n-1}(v).$$

If $f = x^m + c_{m-1}x^{m-1} + \cdots + c_1x + c_0$ is the minimal polynomial of A, then

$$A^m + c_{m-1}A^{m-1} + \cdots + c_0 I_n = 0,$$

where I_n is the $n \times n$ identity matrix. For any $k \geq 0$, by multiplying both sides on the right by the vector $A^k v$, we see that

$$A^{m+k}v + c_{m-1}A^{m-1+k}v + \cdots + c_0 A^k v = 0,$$

hence

$$v_{m+k} + c_{m-1}v_{m-1+k} + \cdots + c_0 v_k = 0, \qquad \text{all } k \geq 0.$$

Wiedemann's idea is to observe that any single component of the vectors $v_0, \ldots, v_{2n-1}$ satisfies the linear recurrence with coefficients $1, c_{m-1}, \ldots, c_0$. The Berlekamp-Massey algorithm (see Algorithm 7.5.4 below) was introduced in the 1960s in the context of coding theory to find the minimal polynomial of a linear recurrence sequence $\{a_r\}$. The minimal polynomial of this linear recurrence is by definition the unique monic polynomial g, such that if $\{a_r\}$ satisfies a linear recurrence $a_{j+k} + b_{j-1}a_{j-1+k} + \cdots + b_0 a_k = 0$ (for all $k \geq 0$), then g divides the polynomial $x^j + \sum_{i=0}^{j-1} b_i x^i$. If we apply Berlekamp-Massey to the top coordinates of the v_i, we obtain a polynomial g_0, which divides f. We then apply it to the second to the top coordinates and find a polynomial g_1 that divides f, etc., Taking the least common multiple of the first few g_i, we find a divisor of the minimal polynomial of f. One can show that with "high probability" one quickly finds f, instead of just a proper divisor of f.

Remark 7.5.3. In the literature, techniques that involve iterating a vector are often called Krylov methods. The subspace generated by the iterates of a vector under a matrix is called a Krylov subspace.

Algorithm 7.5.4 (Berlekamp-Massey)**.** *Given the first $2n$ terms $a_0, \ldots, a_{2n-1}$ of a linear sequence that satisfies a linear recurrence of degree at most n, this algorithm computes the minimal polynomial f of the sequence.*

1. *Let $R_0 = x^{2n}$, $R_1 = \sum_{i=0}^{2n-1} a_i x^i$, $V_0 = 0$, $V_1 = 1$.*
2. *While $\deg(R_1) \geq n$ do the following:*
   (a) *Compute Q and R such that $R_0 = QR_1 + R$.*
   (b) *Let $(V_0, V_1, R_0, R_1) = (V_1, V_0 - QV_1, R_1, R)$.*
3. *Let $d = \max(\deg(V_1), 1 + \deg(R_1))$ and set $P = x^d V_1(1/x)$.*

4. *Let c be the leading coefficient of P and output $f = P/c$.*

The above description of Berlekamp-Massey is taken from [ADT04], which contains some additional ideas for improvements.

Now suppose T is an $n \times n$ matrix as in Problem 7.5.1. We find the minimal polynomial of T by computing the minimal polynomial of $T \pmod{\wp}$, using Wiedemann's algorithm, for many primes $\wp$ and using the Chinese remainder theorem. (One has to bound the number of primes that must be considered; see, e.g., [Coh93].)

One can also compute the characteristic polynomial of T directly from the Hessenberg form of T, which can be computed in $O(n^4)$ field operations, as described in [Coh93]. This is simple, but slow. Also, the T we consider will often be sparse, and Wiedemann is particularly good when T is sparse.

Example 7.5.5. We compute the minimal polynomial of the Hecke operator $A = T_2$ on $\mathbb{M}_2(\Gamma_0(23))^+$ using Wiedemann's algorithm. We have

$$A = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 0 & 2 \\ -1 & 1/2 & -1 \end{pmatrix}$$

Let $v = (1, 0, 0)^t$. Then

$$v = (1,0,0)^t, \quad Av = (3,0,-1)^t, \quad A^2v = (9,-2,-2)^t,$$
$$A^3v = (27,-4,-8)^t, \quad A^4v = (81,-16,-21)^t, \quad A^5v = (243,-42,-68)^t.$$

The linear recurrence sequence coming from the first entries is

$$1, 3, 9, 27, 81, 243.$$

This sequence satisfies the linear recurrence

$$a_{k+1} - 3a_k = 0, \qquad \text{all } k > 0$$

so its minimal polynomial is $x - 3$. This implies that $x - 3$ divides the minimal polynomial of the matrix A. Next we use the sequence of second coordinates of the iterates of v, which is

$$0, 0, -2, -4, -16, -42.$$

The recurrence that this sequence satisfies is slightly less obvious, so we apply the Berlekamp-Massey algorithm to find it, with $n = 3$.

1. We have $R_0 = x^6$, $R_1 = -42x^5 - 16x^4 - 4x^3 - 2x^2$, $V_0 = 0, V_1 = 1$.

2. (a) Dividing R_0 by R_1, we find

$$R_0 = R_1\left(-\frac{1}{42}x + \frac{4}{441}\right) + \left(\frac{22}{441}x^4 - \frac{5}{441}x^3 + \frac{8}{441}x^2\right)$$

(b) The new V_0, V_1, R_0, R_1 are

$$\begin{aligned}
V_0 &= 1 \\
V_1 &= \frac{1}{42}x - \frac{4}{441} \\
R_0 &= -42x^5 - 16x^4 - 4x^3 - 2x^2 \\
R_1 &= \frac{22}{441}x^4 - \frac{5}{441}x^3 + \frac{8}{441}x^2
\end{aligned}$$

Since $\deg(R_1) \geq n = 3$, we have to do the above three steps again.

3. We repeat the preceeding three steps.

   (a) Dividing R_0 by R_1, we find

$$R_0 = R_1\left(-\frac{9261}{11}x - \frac{123921}{242}\right) + \left(\frac{1323}{242}x^3 + \frac{882}{121}x^2\right)$$

   (b) The new V_0, V_1, R_0, R_1 are:

$$\begin{aligned}
V_0 &= \frac{1}{42}x - \frac{4}{441} \\
V_1 &= \frac{441}{22}x^2 + \frac{2205}{484}x + \frac{441}{121} \\
R_0 &= \frac{22}{441}x^4 - \frac{5}{441}x^3 + \frac{8}{441}x^2 \\
R_1 &= \frac{1323}{242}x^3 + \frac{882}{121}x^2
\end{aligned}$$

4. Unfortunately we have to repeat the steps yet again:

$$\begin{aligned}
V_0 &= \frac{441}{22}x^2 + \frac{2205}{484}x + \frac{441}{121} \\
V_1 &= -\frac{242}{1323}x^3 + \frac{968}{3969}x^2 + \frac{484}{3969}x - \frac{242}{3969} \\
R_0 &= \frac{1323}{242}x^3 + \frac{882}{121}x^2 \\
R_1 &= \frac{484}{3969}x^2
\end{aligned}$$

5. We have $d = 3$, so $P = -\frac{242}{3969}x^3 + \frac{484}{3969}x^2 + \frac{968}{3969}x - \frac{242}{1323}$.

6. Multiply through by $-3969/242$ and output

$$x^3 - 2x^2 - 4x + 3 = (x-3)(x^2+x-1).$$

The minimal polynomial of T_2 is $(x-3)(x^2+x-1)$, since the minimal polynomial has degree at most 3 and is divisible by $(x-3)(x^2+x-1)$.

7.5.3 p-adic Nullspace

The following algorithm is due to Dixon[[**Todo: add ref; see Faugere.**]]

Algorithm 7.5.6 (p-adic Nullspace)**.** *Given a matrix A with integer entries, this algorithm computes a nonzero element of* $\ker(A)$ *using successive p-adic approximations, if* $\ker(A)$ *is nonzero.*

1. [Prime] Choose a random prime p.
2. [Echelon] Compute the echelon form of A modulo p.
3. [Done?] If A has full rank modulo p it has full rank, so we terminate the algorithm.
4. [Setup] Let $b_0 = 0$.
5. [Iterate] For each $m = 0, 1, 2, \ldots, k$, use the echelon form of A modulo p to find a vector y_m with integer entries such that $Ay_m \equiv b_m \pmod{p}$, then set
$$b_{m+1} = \frac{b_m - Ay_m}{p}.$$
6. [p-adic Solution] Let $x = y_0 + y_1 p + y_2 p^2 + y_3 p^3 + \cdots + y_k p^k$.
7. [Lift] Use rational reconstruction (Algorithm 7.2.1) to find a vector z with rational entries such that $z \equiv x \pmod{p^{k+1}}$, if such a vector exists. If the vector does not exist, increase k or use a different p. Otherwise, output z.

Proof. We verify the case $k = 2$ only. We have $Ay_0 = 0 \pmod{p}$ and $Ay_1 = -\frac{Ay_0}{p} \pmod{p}$. Thus

$$Ay_0 + pAy_1 \equiv Ay_0 + (-Ay_0) \pmod{p^2}.$$

□

7.5.4 Decomposition Using Kernels

We now know enough to give an algorithm to solve Problem 7.5.1.

Algorithm 7.5.7 (Decomposition Using Kernels)**.** *Given an $n \times n$ matrix T over a field K as in Problem 7.5.1, this algorithm computes the decomposition of V as a direct sum of simple $K[T]$ modules.*

1. *[Minimal Polynomial] Compute the minimal polynomial f of T, e.g., using the multi-modular Wiedemann algorithm.*
2. *[Factorization] Factor f using the Belebas, Hoeij, Klüners, and Steel algorithm.*
3. *[Compute Kernels] For each irreducible factor g_i of f:*
   (a) *Compute the matrix $A_i = g_i(T)$. (This is difficult, and A_i will have huge coefficients.)*

   (b) *Compute* $W_i = \ker(A_i)$ *using, e.g., a multi-modular kernel algorithm.*

4. *[Output Answer] Then* $V = \oplus W_i$.

Remark 7.5.8. In practice Step 3 is very time consuming. As mentioned in Remark 7.5.2, if one can compute such decompositions $V = \oplus W_i$, then one can easily factor polynomials f, hence the difficulty of polynomial factorization is a lower bound on the complexity of writing V as a direct sum of simples.

7.5.5 Multimodular Decomposition Algorithm

The following algorithm is a modification of Algorithm 7.5.7, which provides an alternative to the difficult Step 3.

Algorithm 7.5.9 (Multimodular Decomposition)**.** *Given an* $n \times n$ *matrix* T *over a field* K *as in Problem 7.5.1, this algorithm outputs a decomposition of* V *as a direct sum of simple* $K[T]$ *modules.*

1. *[Minimal Polynomial] Compute the minimal polynomial* f *of* T*, e.g., using the multi-modular Wiedemann algorithm.*
2. *[Factorization] Factor* $f = \prod g_i$ *using the Belebas, Hoeij, Klüners, and Steel algorithm.*
3. *[Cofactors] For each* i*, let* $h_i = f/g_i$.
4. *[Find Kernels] For several primes* $\wp$ *(how many?), compute reduced row echelon forms for basis of all the kernels* $\overline{W}_i = \ker(g_i(\overline{T}))$ *as follows:*
   (a) *Choose a random vector* $v \in \overline{V}$.
   (b) *Compute the iterates*
   $$v_0 = v, \quad v_1 = \overline{T}v, \quad \ldots, \quad v_{n-1} = \overline{T}^{n-1}v.$$
   (c) *For each* i *do the following:*
      i. *Compute* $w = h_i(\overline{T})v \in \ker(g_i(\overline{T}))$ *by taking the linear combination of the* v_i *given by the coefficients of* h_i.
      ii. *Generate a subspace of* $\ker(g_i(\overline{T}))$ *using* $w, \overline{T}w, \ldots, \overline{T}^i w$*, keeping the subspace basis in Echelon form at each step. If this subspace does not equal the full* $\ker(g_i(\overline{T}))$*, repeat the above steps with another* v*, and add the resulting iterates of the new* w *to this subspace. Repeat this process until we obtain a basis for* $\ker(g_i(\overline{T}))$*, in reduced row echelon form.*
5. *[Lift] Using the Chinese remainder theorem and rational reconstruction, lift the* $\overline{W}_i$ *to* K*-vector spaces* W_i *such that* $V = \oplus W_i$ *is the desired decomposition. (WARNING: It is probably necessary to throw away "bad" primes, just as we did in the multi-modular echelon algorithm.)*

7.6 Exercises

7.1 Figure out how in Algorithm 7.4.2 to obtain the invertible transformation matrix T such that $TA = E$. You should use matrix multiplications in the spirit of that algorithm.