Exact Linear Algebra for SAGE

William Stein

Department of Mathematics
University of Washington

August 7, 2006 / MSRI Workshop

Outline

The Wiki Page

`http://sage.math.washington.edu:9000/linalg`

Coding sprint this week: Me, Robert Bradshaw, Soroosh Yazdani.

Extremely difficult: We need more help.

Reduced Row Echelon Form

Definition (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 E such that E is obtained from A by left multiplication by an invertible matrix and E is in reduced row echelon form.
- This matrix E 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.

Echelon form example 1

Example

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

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

Echelon form example 2

Example

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

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

Linear Algebra Problems and Echelon Form

Linear algebra class: Convert some standard algorithmic problems into computing echelon forms.

- **Kernel of A:** Kernel of the reduce row echelon form E of A is the same as the kernel of A. Easy to write down kernel of a matrix in echelon form.
- **Intersection of Subspaces:**
  1. W_1 and W_2 subspace of V.
  2. Let A_1 and A_2 be matrices whose columns form a basis for W_1 and W_2, respectively.
  3. Let $A = [A_1|A_2]$ be the augmented matrix formed from A_1 and A_2.
  4. 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 $\dim(W_1) \leq \dim(W_2)$ and do the following: For each b in a basis for K, write down the linear combination of a basis for W_1 got by taking the first $\dim(W_1)$ entries of the vector b. So $\sum a_i w_{1,i} + \sum b_j w_{2,j} = 0$, i.e., an element of that kernel is $\sum a_i w_{1,i} = \sum -b_j w_{2,j}$.

Matrix Multiplication

Strassen-Winograd: Multiply two $n \times n$ matrices (with n a 2-power).
Complexity is $O(n^{\log_2(7)}) = O(n^{2.807})$.

```
Input matrices [A00 A01]    [B00 B01]
               [A10 A11]    [B10 B11]
# 8 Pre-Additions
S0 = A10 + A11,  T0 = B01 - B00
S1 = S0  - A00,  T1 = B11 - T0
S2 = A00 - A10,  T2 = B11 - B01
S3 = A01 - S1,   T3 = T1 - B10

# 7 (Potentially) Recursive Multiplications
P0 =  A00*B00
P1 =  A01*B10
P2 =  S0*T0
P3 =  S1*T1
P4 =  S2*T2
P5 =  S3*B11
P6 =  A11*T3

# 7 Post Additions
U0 = P0 + P1
U1 = P0 + P3
U2 = U1 + P4
U3 = U2 + P6
U4 = U2 + P2
U5 = U1 + P2
U6 = U5 + P5

Answer is U0 U6
          U3 U4
```

Example: Fast Matrix Multiplication Over $\mathbb{F}_5$ (this weekend)

```
sage: from sage.ext.dense_matrix_pyx import Matrix_modint
sage: n = 512
sage: A = Matrix_modint(5, n,n, range(n^2))
sage: B = Matrix_modint(5, n,n, list(reversed(range(n^2))))
sage: C = B._mul_submatrices_strassen(A, 0, 0, n, 0,0,64) # CPU time:
sage: C = A*B     # CPU time: 1.51

sage: n = 1024
sage: A = Matrix_modint(5, n,n, range(n^2))
sage: B = Matrix_modint(5, n,n, list(reversed(range(n^2))))
sage: C = B._mul_submatrices_strassen(A, 0, 0, n, 0,0,64) # CPU time:
sage: C = A*B                                              # CPU time:

sage: n = 2048
sage: A = Matrix_modint(5, n,n, range(n^2))
sage: B = Matrix_modint(5, n,n, list(reversed(range(n^2))))
sage: C = B._mul_submatrices_strassen(A, 0, 0, n, 0,0,64)  # CPU time:
sage: time C = A*B                                          # CPU time:

sage: n = 4096
sage: A = Matrix_modint(5, n,n, range(n^2))
sage: B = Matrix_modint(5, n,n, list(reversed(range(n^2))))
sage: time C= B._mul_submatrices_strassen(A, 0, 0, n, 0,0,64)
CPU time: 155.61 s
```

Notes

Remark

- There is also an approach called FFLAS-FFPACK that works just like above, but for the small finite field matrices, does the matrices using a general **floating point** BLAS (=Basic Linear Algebra System), e.g., ATLAS.
- This can result in additional speedups, but requires that a BLAS be installed.
- This is not yet implemented in SAGE, because SAGE does not include an optimized BLAS.

Rational Reconstruction

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

Algorithm (Rational Reconstruction)

Given an integer $a \geq 0$ and an integer $m > 1$, this algorithm **efficiently** computes the numerator n and denominator d of the unique rational number n/d, if it exists, with

$$|n|, d \leq \sqrt{\frac{m}{2}} \qquad \text{and} \qquad n \equiv ad \pmod{m}, \tag{2.1}$$

or reports that there is no such number.

1. [Reduce mod m] Let 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 rational number as in (2.1).

Example

We compute an example using SAGE.

```
sage: p = 389
sage: k = GF(p)
sage: a = k(7/13)
210
sage: a.rational_reconstruction()
7/13
```

Example

```
sage: R = Integers(432392082039)
sage: a = R(-1983/17)
sage: print a
178043798370
sage: a.rational_reconstruction()
-1983/17
```

Cool!!

Multimodular Algorithm (Good for Sparse Matrices)

Algorithm (Multimodular Echelon Form)

1. Rescale input matrix A to have 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., any other echelon algorithm.
5. Discard any B_i whose pivot column list is not maximal among pivot lists of all B_j found so far.
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 Algorithm 5 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, use a few more primes. Let E be the matrix over $\mathbb{Q}$ so obtained.
8. Compute the denominator d of E, i.e., the smallest positive integer such that dE has integer entries. If $H(dE) \cdot H(A) \cdot n < \prod p_i$, then E is the reduced row echelon form of A. If not, repeat the above steps with a few more primes.

Jen B. and I: This also all generalizes to number fields.

Echelon Form Example

Example

```
sage: A = MatrixSpace(QQ,2000,2000, sparse=True).random_element

sage: time B = A.echelon_form()
CPU time: 1.19 s,  Wall time: 1.42 s

sage: A = MatrixSpace(QQ,2000,2100, sparse=True).random_element

sage: set_verbose(2)

sage: time B = A.echelon_form(height_guess=100)
verbose 2 (2969: matrix.py, echelon_form) height_guess = 100
verbose 2 (2969: matrix.py, echelon_form) echelon modulo p=200
verbose 2 (2969: matrix.py, echelon_form) time to reduce matri
verbose 1 (2969: matrix.py, sparse_matrix_pyx matrix_modint ec
verbose 2 (2969: matrix.py, echelon_form) time to put reduced r
....
CPU time: 15.79 s
```

Matrix Inverse

Algorithm (Inverse Using Matrix Multiplication)

Given an $m \times m$ matrix M this algorithm computes the inverse of M (for m a 2-power).

1. [Extend] Replace M with an augmented matrix $\begin{pmatrix} M & 0 \\ 0 & I_{2^n-m} \end{pmatrix}$ where I_{2^n-m} is the $(2^n - m) \times (2^n - m)$ identity matrix. Note that $\begin{pmatrix} M & 0 \\ 0 & I_{2^n-m} \end{pmatrix}^{-1} = \begin{pmatrix} M^{-1} & 0 \\ 0 & I_{2^n-m} \end{pmatrix}$, so it suffices to compute the inverse of this new matrix (which we call M for the rest of the algorithm).
2. [Submatrices] Write $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ with each of A, B, C, D a $2^{n-1} \times 2^{n-1}$ matrix.
3. [Multiply and Invert] 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 2 (or fall back to a standard matrix inverse algorithm).
4. [Compute Inverse] Compute

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

5. [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.)

Echelon Forms via Matrix Multiplication

Algorithm (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 p,q compute the echelon form of $A \pmod{p}$ (e.g., using Gauss elimination). Let $c_0, \ldots, c_{n-1}$ be the pivot columns of $A \pmod{p}$. When computing the echelon form save the positions $r_0, \ldots, r_{n-1}$ of the rows used to clear each column.
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 Algorithm 10 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 on page 7. 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.

The Problem

Suppose T is an $n \times n$ matrix with entries in a field K (typically a number field or finite field) and that the minimal polynomial of T is square free and has degree n. View T as acting on $V = K^n$.

Problem

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.

Application: From this one can efficiently compute systems of eigenvalues, i.e., newforms.

Solution

1. *Compute* $f = \mathrm{charpoly}(T)$
2. *Factor* $f = \prod g_i(x)$.
3. *Compute a nonzero* $v_i \in \ker(g_i(T))$.
4. *Compute images of* v_i *under powers of* T *(Repeat on summands with multiplicity* > 1.*)*

Examples!

```
sage: m = ModularSymbols(389,2,sign=1).cuspidal_submodule()
sage: t2 = m.T(2).matrix()

sage: t2.charpoly().factor()
(x + 2) * (x^2 - 2) * (x^3 - 4*x - 2) *
  (x^6 + 3*x^5 - 2*x^4 - 8*x^3 + 2*x^2 + 4*x - 1) * (x^20 - ...

sage: kernel(t2^2-2)
Vector space of degree 32 and dimension 2 over Rational Field
Basis matrix:
[ 0  0  1  0  0  0  0  0  1 -1 -1  0  1  0  0  0 -1  0  0 -1  0
     0  0  0  0  0  0  0  1 -1  0 -1]
[ 0  0  0  0  1 -1 -1  1 -2  2  1  0 -2  1 -1  0  2  2 -1  1  0
     0 -1  2 -1 -1  1  1 -2  2  0  2]

sage: t2.decomposition()
...
```

Polynomial Factorization

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 the LLL lattice basis reduction algorithm in a novel way to solve the optimization problems that come up in trying to lift factorizations mod p to $\mathbb{Z}$. It has been generalized by Belebas, Hoeij, Klüners, and Steel to number fields.

This is in SAGE via both NTL and PARI.

Wiedemann's Minimal Algorithm

This gives a flavor for advanced charpoly algorithms.
Choose a random vector v and compute the iterates

$$v_0 = v, \quad v_1 = A(v), \quad v_2 = A^2(v), \quad \ldots, \quad v_{2n-1} = A^{2n-1}(v). \tag{3.1}$$

If $f = x^m + c_{m-1}x^{m-1} + \cdots + c_1 x + c_0$ 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

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

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 finds the minimal polynomial of a linear recurrence sequence $\{a_r\}$, which is a factor of the minimal polynomial of A.

This algorithm is especially good for sparse matrices.

p-adic Nullspace

Dixon's algorithm computes p-adic approximations to a solution to a linear equations over $\mathbb{Q}$. Rational reconstruction modulo p^n then allows us to recover the corresponding solutions over $\mathbb{Q}$.

Algorithm (p-adic Nullspace)

Given a matrix A with integer entries and nonzero kernel, this algorithm computes a nonzero element of $\ker(A)$.

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 5) 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.