3. Gaussian Elimination

The standard *Gaussian elimination algorithm* takes an $m \times n$ matrix M over a field F and applies successive elementary row operations

(i) multiply a row by a field element (the inverse of the left-most non-zero)
(ii) subtract a multiple of one from another (to create a new left-most zero)
(iii) exchange two rows (to bring a new non-zero pivot to the top)

to produce M' in *reduced row echelon form*:

- any all-zero rows at bottom
- one at left-most position in non-zero row
- number of leading zeroes strictly increasing.

Comments

- Complexity $\mathcal{O}(n^3)$ field operations if $m < n$.
- With little more bookkeeping we can obtain LUP decomposition: write $M = LUP$ with L lower triangular, U upper triangular, P a permutation matrix.
- Standard applications:
  - solving system $M \cdot x = b$ of m linear equations in $n - 1$ unknowns x_i, by applying Gaussian elimination to the *augmented matrix* $M|b$;
  - finding the inverse M^{-1} of a square invertible $n \times n$ matrix M, by applying Gaussian elimination to the matrix $M|I_n$;
  - finding the determinant $\det M$ of a square matrix, for example by using $\det M = \det L \det U \det P$.
  - find bases for row space, column space (image), null space, find rank or dimension, dependencies between vectors: minimal polynomials.

Fraction-free Gaussian elimination

When computing over a domain (rather than a field) one encounters problems entirely analogous to that for the Euclidean algorithm: avoiding fractions (and passing to the quotient field) leads to intermediate expression swell.

Remedies also analogous: *modular methods* combined with Chinese remainder algorithm, or PRS-like solution: *Bareis' algorithm*.

Start with $M^{(1)} = M = (x_{i,j})$.

In the typical step $M^{(k)}$ is transformed to $M^{(k+1)}$ by changing the element x_{ij} for $i > k, j \geq k$ to

$$x_{ij}^{(k+1)} = x_{ij}^{(k)} - x_{kj}^{(k)} \frac{x_{ik}^{(k)}}{x_{kk}^{(k)}},$$

using $x_{kk}^{(k)}$ as pivot.

In the *division-free Gaussian elimination algorithm* we replace this by

$$x_{ij}^{(k+1)} = x_{kk}^{(k)} \cdot x_{ij}^{(k)} - x_{kj}^{(k)} \cdot x_{ik}^{(k)},$$

and the number of digits may double (quadruple?) in each step (in the integer case).

See example.

Adjoint

The *adjoint* $\mathrm{adj}M$ of M has $(-1)^{i+j} \det M[j,i]$ as entry at position i,j, where $M[j,i]$ is the matrix obtained by omitting row j and column i from M. Then

$$M \cdot \mathrm{adj}M = \det M \cdot I.$$

Lemma 1. *Let M be an $n \times n$ matrix over a domain D, and*

$$M = \begin{pmatrix} A & B \\ C & X \end{pmatrix},$$

with A and X square $k-1$ and $n-(k-1)$ matrices. If $\delta = \det A \neq 0 \in D$ then

$$\delta^{n-k} \det M = \det(\delta X - C \cdot \mathrm{adj}A \cdot B).$$

Define $x_{rs}^{(k)} = \det M_{rs}^{(k)}$ with $M_{rs}^{(k)}$ the k-square matrix of the top left $(k-1)$-square submatrix of M augmented by partial r-th row and s-th column of M.

Note that $M = M_{nn}^{(n)}$ and $A = M_{k-1,k-1}^{(k-1)}$ so $\delta = x_{k-1,k-1}^{(k-1)}$.

Lemma 2.

$$(\delta X - C \cdot \mathrm{adj} A \cdot B)_{r-k+1,s-k+1} = x_{rs}^{(k)}.$$

Sylvester's identity

Lemma 3. [Sylvester]

$$\delta^{n-k} \det M = \det \begin{pmatrix} x_{kk}^{(k)} & x_{kk+1}^{(k)} & \cdots & x_{kn}^{(k)} \\ x_{k+1k}^{(k)} & x_{k+1k+1}^{(k)} & \cdots & x_{k+1n}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{nk}^{(k)} & x_{nk+1}^{(k)} & \cdots & x_{nn}^{(k)} \end{pmatrix}.$$

Applying this with $M = M_{ij}^{(k+1)}$ we find

$$\delta \det M_{ij}^{(k+1)} = \det \begin{pmatrix} x_{kk}^{(k)} & x_{kj}^{(k)} \\ x_{ik}^{(k)} & x_{ij}^{(k)} \end{pmatrix}.$$

Corollary

$$x_{ij}^{(k+1)} = \frac{x_{kk}^{(k)} \cdot x_{ij}^{(k)} - x_{kj}^{(k)} \cdot x_{ik}^{(k)}}{x_{k-1,k-1}^{(k-1)}}$$

Single step Bareiss

```
bareiss := procedure(~M)
 n := NumberOfRows(M);
 for k in [1..n-1] do
  for i in [k+1..n] do
   for j in [k+1..n] do
    D := M[k][k]*M[i][j] - M[k][j]*M[i][k];
    M[i][j] := k eq 1 select D div 1
                      else D div M[k-1][k-1];
   end for;
  end for;
 end for;
 print M;
end procedure;
```

Column spaces

Consider $A, B \in \mathbb{Z}^{m \times n}$ as n column vectors of length m. To determine whether the column spaces spanned over $\mathbb{Z}$ (that is: *lattices* in $\mathbb{Z}^m$) by A and B are the same, compute their *Hermite normal forms* $H(A)$ and $H(B)$.

H is the Hermite normal form of A if

- it is obtained by elementary column operations over the integers from A;
- any zero columns appear at the end, at $r + 1, \ldots, n$;

if first non-zero element in column i is $H_{c(i),i}$:

- $1 \leq c(1) < c(2) \cdots < c(r) \leq m$;
- $0 \leq H_{c(i),j} < H_{c(i),i}$ for $1 \leq j < i \leq r$.

Theorem *A and B generate the same lattice if and only if $H(A) = H(B)$.*

Corollary *A and B generate the same lattice if and only if there is a unimodular matrix U such that $A \cdot U = B$.*