Computing characteristic polynomials.

The characteristic polynomial of a matrix is defined to be the determinant of a matrix with polynomial entries, each of which is zero or has degree ≤ 1; as such the charactristic polynomial is a sum of monomials, each of wich is a product of scalars and first degree polynomials. As in the case of matrices with scalar entries, computation of such determinants by the explicit formula quickly becomes unmanageable, and therefore one needs some alternate means for computing such expressions. One such method is outlined below. It requires something like n^4 arithmetic computations; in contrast, computing a determinant with scalar entries from the explicit formula requires about $n! + (n+1)!$ arithmetic computations, and for matrices with polynomial coefficients the required number of computations is even larger.

This method is based upon some results concerning polynomials that are symmetric in h variables; *i.e.*, we have

$$f(x_1, \cdots, x_n) \quad = \quad f(x_{\sigma(1)}, \cdots, x_{\sigma(n)})$$

for all permutations σ of $\{1, \cdots, n\}$. The results provide formulas for computing the coeeficients of the characteristic polynomial $\chi_A(t)$ of a matrix A in terms of the traces of the power matrices A^k. and it is easy to write down compuer programs which use these formulas to compute the coefficients of $\chi_A(t)$ for matrices that are too large to permit efficient calculation of characteristic polynomials using elementary techniques, including expansion by minors.

Newton's Formula

If $k \leq n$ then the polynomial in n variables given by

$$N_k(t_1, \cdots, t_n) \quad = \quad \sum t_i^k$$

is symmetric and therefore can be expressed as a polynomial in the elementary symmetric functions σ_i that are the given by the identity

$$\prod_i (t_i + x) \quad = \quad \sum \sigma_i(t_1, \ldots, t_n)\, x^{n-i} \; .$$

This polynomial is called the k^{th} *Newton polynomial* and is generally denoted by $s_k(\sigma_1, \cdots, \sigma_n)$. One can compute this polynomial recursively using the following result:

NEWTON'S FORMULA.

$$s_n \; - \; \sigma_1 s_{n-1} \; + \; \sigma_2 s_{n-2} \; - \; \cdots \; \mp \sigma_{n-1} s_1 \; \pm \; n\, \sigma_n \quad = \quad 0$$

Sketch of proof. Make the substitution $s = -t_i$ in the defining identity for the symmetric functions and sum from $i = 1$ to n.■

Applications to characteristic polynomials

Suppose that A is a triangular matrix. Then its characteristic polynomial is equal to the product

$$\prod_{j=1}^{n} (a_{j.j} \; - \; t)$$

and using the elementary symmetric polynomials σ_i we may rewrite this as

$$\chi_A(t) \quad - \quad \sum_{k=0}^{n} (-1)^{n-k}\, \sigma_k(a_{1,1}, \ldots, a_{n,n})\, t^{n-k} \ .$$

Furthermore, if A is a triangluar matrix we also know that

$$s_k(\sigma_1, \ \cdots \ , \sigma_n) \quad = \quad N_k(a_{1,1}, \ldots, a_{n,n}) \quad = \quad \mathrm{trace}\, A^k \ .$$

We may now use Newton's Formula to find σ_k recursively using the identity

$$\sigma_k \quad = \quad \frac{(-1)^{k+1}}{k} \cdot \left(\sum_{j=0}^{k-1} (-1)^j \, s_{k-j}\, \sigma_j \right) \ .$$

Using the formula for s_j given above, we may write everything in the form

$$\chi_A(t) \quad = \quad \sum_{k=0}^{n} (-1)^{n-k}\, b_k\, t^{n-k}$$

where we have $b_0 = (-1)^n$ and

$$b_k \quad = \quad \frac{(-1)^{k+1}}{k} \cdot \left(\sum_{j=0}^{k-1} (-1)^j \, b_j \, \mathrm{trace}(A^{k-j}) \right) \ .$$

THEOREM. *The same formula is valid for arbitrary matrices.*

Proof. If C is an arbitrary matrix over the complex numbers, then C is similar to some triangular matrix A. Since similar matrices have the same characteristic polynomial, we know that

$$\chi_C(t) \quad = \quad \chi_A(t) \quad = \quad \sum_{k=0}^{n} (-1)^{n-k}\, b_k\, t^{n-k}$$

where b_k is given as above. Therefore it suffices to verify that thr traces of the m^{th} power matrices A^m and B^m satisfy $\mathrm{trace}(C^m) = \mathrm{trace}(A^m)$ for all m. But this is true because if A is similar to C then A^m is similar to C^m for all m and similar matrices always have equal traces.∎

This leads to a straightforward algorithm. Start off with $b_0 = (-1)^n$ and $\mathrm{trace}(A^0) = n$, assume that after Step k we know b_0 through b_k, the matrices A^0 through A^k and the traces of A^0 through A^k. At Step $k+1$ one first computes A^{k+1}, then computes its trace, and finally computes b_{k+1} using the recursive formula.∎