A Simple Program for Computing Characteristic Polynomials with Mathematica†

Gordon G. Cash

New Chemicals Screening and Assessment Branch, Risk Assessment Division (7403), Office of Pollution Prevention and Toxics, U.S. Environmental Protection Agency, Washington, DC 20460

Received December 5, 1998

This paper presents a program for computing characteristic polynomials written for the commercial software package Mathematica. The program utilizes several features that are relevant to problems encountered in computing characteristic polynomials by other means, especially limitations on problem size due to integer overflow.

INTRODUCTION

Uses of the characteristic polynomial and methods for computing it have been recently reviewed.[1] A particularly troublesome problem in computing graph polynomials for large structures is that the number of digits in the coefficients exceeds the size of registers on most available computers. For example, the characteristic polynomial of a $C_{96}H_{24}$ benzenoid structure published by Herndon et al.[2] has several coefficients of 23 digits. While that computation was carried out in quadruple precision on a mainframe computer, the ever increasing speed of desktop machines brings larger structures within the reach of these machines as well. The floating-point unit of the desktop CPU in most common use today has a significand of only 64 bits, or about 19 decimal digits long,[3] too few for the 96-vertex problem cited above. Furthermore, there is no guarantee that any given software will preserve all 19 digits. Methods exist for working around this limitation,[4] but their application requires both a compiler and a programmer. A very recent publication[5] described a method of computing the factors of some characteristic polynomials that avoided the large coefficients of the complete polynomials. The largest structure treated in ref 5 was $C_{48}H_{18}$, with just half the vertexes of the structure in ref 2. The method described in ref 5 also depended on the presence of certain symmetry elements, but it was not applicable to general graphs. The present work describes an implementation that is applicable to any unweighted graph, requires almost no programming skills, handles coefficients of arbitrary size, and produces characteristic polynomials even for structures as large as C_{96} in a time frame of minutes on a midrange desktop computer.

METHODS

Several methods for computing the characteristic polynomial are described in ref 1, namely, direct evaluation of the determinant, $\det(x\mathbf{I} - \mathbf{A})$, where **A** is the adjacency matrix and **I** is the identity matrix; multiplication of the zeroes of the characteristic equation

$$\prod_{i=1}^{N}(x - \lambda_i)$$

where the λ_i are the eigenvalues of the adjacency matrix; and an iterative method, the Le Verrier−Faddeev−Frame method, which involves successive multiplications of square matrixes of the same order as **A**. Another iterative method, called Krylov's method,[6] avoids the multiplication of $N \times N$ matrixes but demands finding determinants of $N \times N$ matrixes with very large elements. The present work evaluates and compares implementations of all these methods in Mathematica.

The version of Mathematica used in this study was 2.2.1, which has since been superseded. The statements made herein have not been evaluated with prior or subsequent versions, and all statements referring to calculations performed by Mathematica refer to version 2.2.1. Mathematica has two features that make it particularly useful for calculating characteristic polynomials: in integer calculations, it keeps all the digits, and in floating-point calculations, it works with an arbitrary, user-specified number of significant figures. Numbers of significant figures, whether integer or floating-point, that are larger than machine precision are presumably handled through software emulations. However, the documentation[7] does not describe the methods. In any case, increasing the number of significant figures exacts a noticeable penalty in execution time.

(1) Direct Evaluation of the Determinant. Unlike high-level language compilers, Mathematica is capable of symbolic manipulation and, at least in principle, could evaluate $\det(x\mathbf{I} - \mathbf{A})$ directly without first assigning a numerical value to x. A procedure for performing this evaluation is explicitly provided on page 12 of ref 7. In practice, however, this approach is several orders of magnitude slower than any of the numerical methods, and for structures of any nontrivial size the computation time required is prohibitive. The symbolic manipulation feature is useful, however, in that the characteristic polynomial can be stored and used as a polynomial in explicit powers of x instead of being

† **Disclaimer:** The suitability of other software packages was not evaluated. Therefore, the author does not represent that Mathematica is the only commercial software product capable of performing the calculations described, nor necessarily even the best one. Mention of this or other products by name should not be construed as recommendation or endorsement by either the author or the U.S. Environmental Protection Agency.

10.1021/ci9801668 CCC: $18.00
Published on Web 05/13/1999

```
a=ReadList["filename", Number, RecordLists->True];
c = Range[Length[a]+1];
c[[1]] = 1;
c[[2]] = 0;
ax = a;
For[i = 2, i <= Length[a], i++,
   bx = ax + c[[i]] * IdentityMatrix[Length[a]];
   ax = a . bx;
   c[[i+1]] = -Sum[ax[[j,j]], {j,Length[a]}] / i;
]
c = Reverse[c];
Table[c[[i]] x^(i-1), {i,Length[c]}];
charpoly = Sum[%[[i]], {i,Length[%]}]
```

Figure 1. Mathematica implementation of the Le Verrier−Faddeev−Frame method. The first statement reads the adjacency matrix from a file. Alternatively, the matrix could simply be typed into the program. Mathematica treats a vector as a list of numbers and a matrix as a list of lists. The command "Reverse" reverses the order of elements in a list, in this instance the coefficients of the polynomial. The "Table" statement multiplies each coefficient by its respective power of *x*, which is carried through as a symbol rather than being assigned a numeric value. The "Sum" statement uses the symbol for the result of the previous statement (%) to build the polynomial as the sum of its terms.

represented as a list of coefficients. The form **Factor[polynomial_name]** produces the factors of such a polynomial, and built-in procedures exist for finding exact or approximate roots.

(2) Product of the Zeroes of the Equation. The zeroes of the characteristic equation (characteristic polynomial = 0) are $x - \lambda_i$, where the λ_i are the eigenvalues of the adjacency matrix. In Mathematica, the eigenvalues are available directly from **Eigenvalues[N[a]]**. The shortcoming of this method is that most of the eigenvalues are not integers, and a sufficient number of significant figures must be retained to produce all the integer digits in the coefficients. For the $C_{96}H_{24}$ structure mentioned above, the eigenvalues must have 30-digit precision to produce the correct 23-digit result. **Eigenvalues[N[a],30]** produces eigenvalues with 30-digit precision, but execution time degrades to the point where this method is somewhat slower than the all-integer Le Verrier−Faddeev−Frame method described below.

(3) Le Verrier−Faddeev−Frame Method. The Mathematica program shown in Figure 1 is a direct implementation of the Le Verrier−Faddeev−Frame iterative method as described on pages 78−79 of ref 1. The coefficients of the characteristic polynomial are the traces of matrixes built up iteratively by multiplication of $N \times N$ matrixes. For large matrixes, the off-diagonal elements can be considerably smaller than the coefficients, thus reducing computation time. For integer calculations in Mathematica, no precision need be specified; Mathematica simply keeps all the digits. In addition, the dot operator (.) allows the product of two matrixes of any compatible size to be expressed simply as, for example, m1.m2, avoiding the need to construct iteration loops and pass size parameters. Of the four methods tested, this is clearly the fastest, although it is slightly less than twice as fast as finding the eigenvalues and multiplying the zeroes of the equation. For example, on a 233 MHz Pentium-class computer, calculation of the characteristic polynomial for the $C_{96}H_{24}$ polyhex from ref 2 took 9 min 23 s by the Le Verrier−Faddeev−Frame method. Finding the eigenvalues to 30 significant figures and multiplying the zeroes of the equation took 11 min 15 s.

As an example of concrete results, the characteristic polynomial coefficients for all 40 of the C_{40} fullerenes are included as Supporting Information. On the same machine described above, each of these calculations takes about 12 s. Balasubramanian showed[8] that the first eight coefficients of the characteristic polynomials of fullerenes depend only on the number of carbon atoms. For C_{40} these are $x^{40} - 60x^{38} + 1650x^{36} - 24x^{35} - 27600x^{34} + 1200x^{33}$. The x^{n-1} coefficient is always zero because the trace of the adjacency matrix is zero. Since fullerenes contain no three-membered rings, the x^{n-3} coefficient is also always zero. The isomers are listed in Table 1 of the Supporting Information in the order they are produced by the spiral algorithm of Fowler and Manolopoulos.[9]

(4) Krylov's Method. This is yet another iterative method which was developed a number of years ago before computing machinery had attained anything like its current capabilities. The advantage of Krylov's method is that it never requires multiplying two $N \times N$ matrixes together—only iterative multiplication of an $N \times N$ matrix by an $N \times 1$ vector. As in the Le Verrier−Faddeev−Frame method, only integer arithmetic is needed. The drawback of Krylov's method is that the size of the integers in the vector grows much faster than the size of the coefficients they are generated to calculate. As a result, the speed gained by reducing the number of arithmetic operations is more than lost because of the large number of digits in the integers. For the 96-vertex problem, this method was much slower than the Le Verrier−Faddeev−Frame method.

CONCLUSIONS

Several methods for computing the characteristic polynomial were programmed in Mathematica and compared, and the published result for a $C_{96}H_{24}$ structure was successfully reproduced. Of the methods tested, the fastest one was an implementation of the iterative Le Verrier−Faddeev−Frame procedure. As implemented in Mathematica, this method produces characteristic polynomials with exact coefficients for structures of arbitrary size in a few minutes on a midrange desktop computer.

Supporting Information Available: Coefficients of the characteristic polynomials of the 40 C_{40} fullerenes. This material is available free of charge via the Internet at http://pubs.acs.org.

REFERENCES AND NOTES

(1) Trinajstić, N. *Chemical Graph Theory*, 2nd ed.; CRC Press: Boca Raton, FL, 1992; Chapter 5.
(2) Herndon, W. C.; Radhakrishnan, T. P.; Živković, T. P. Characteristic and Matching Polynomials of Chemical Graphs. *Chem. Phys. Lett.* **1988**, *152*, 233−238.
(3) Intel Corporation, *Intel486 Processor Family Programmer's Reference Manual*; Intel Literature: Mt. Prospect, IL, 1995; page 16-2.
(4) Bailey, D. H. Algorithm 719. Multiprecision Translation and Execution of FORTRAN Programs. *ACM Trans. Math. Software* **1993**, *19*, 288−319.
(5) Mishra, R. K.; Patra, S. M. Numerical Determination of the Kekulé Structure Count of Some Symmetrical Polycyclic Aromatic Hydrocarbons and Their Relationship with π-Electronic Energy (A Computational Approach). *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 113−124.
(6) Ralston, A. *A First Course in Numerical Analysis*; McGraw-Hill: New York, 1965; Chapter 10, p 466.
(7) Wolfram, S. *Mathematica: A System for Doing Mathematics by Computer*; Addison-Wesley: Reading, MA, 1991.
(8) Balasubramanian, K. Graph-theoretical Characterization of Fullerene Cages. *Polycyclic Aromatic Compd.* **1993**, *3*, 247−259.
(9) Fowler, P. W.; Manolopoulos, D. E. *An Atlas of Fullerenes*; Clarendon Press: Oxford, 1995; pp 165−176 and 186−189.

CI9801668