JOURNAL OF COMPUTATIONAL PHYSICS **140,** 453–458 (1998)
ARTICLE NO. CP985909

An Accurate and Efficient Algorithm for the Computation of the Characteristic Polynomial of a General Square Matrix

S. Rombouts*,[1] and K. Heyde†

Vakgroep Subatomaire en Stralingsfysica, Institute for Theoretical Physics and Institute for Nuclear Physics, Gent, Belgium
E-mail: *Stefan.Rombouts@rug.ac.be and †Kris.Heyde@rug.ac.be

Received October 3, 1997

An algorithm is presented for the efficient and accurate computation of the coefficients of the characteristic polynomial of a general square matrix. The algorithm is especially suited for the evaluation of canonical traces in determinant quantum Monte-Carlo methods.

Key Word: characteristic polynomial.

I. INTRODUCTION

The characteristic polynomial $P_U(x)$ of a general N-by-N matrix U is given by

$$P_U(x) = \det(xI - U) \tag{1}$$

(I is the unit matrix). Though the characteristic polynomial of a matrix is a basic concept in linear algebra, its numerical computation is scarcely documented in the literature. There are, however, a number of applications for which an accurate and efficient algorithm for the calculation of the coefficients of $P_U(x)$ would be useful, e.g., for the study of random matrices [1]. It would also be useful for determinant quantum Monte-Carlo (QMC) methods [2–4]: the application of determinant QMC method in the canonical ensemble, especially the shell-model Monte-Carlo method [4], requires the evaluation of the coefficients of the polynomial

$$\bar{P}_U(x) = \det(I + xU). \tag{2}$$

This polynomial is closely related to $P_U(x)$:

$$\bar{P}_U(x) = (-x)^N P_U(-1/x). \tag{3}$$

[1]Postdoctoral fellow of the Fund for Scientific Research—Flanders (Belgium).

0021-9991/98 $25.00

In the determinant QMC method, the A-particle trace of a one-body evolution matrix U is given by the coefficient of x^A in $\bar{P}_U(x)$. This coefficient is equal to $(-1)^A$ times the coefficient of $x^{(N-A)}$ in $P_U(x)$. It is in the light of these canonical QMC methods that we developed an algorithm that is presented in the following sections. Accuracy is important here because for calculations at low temperature, QMC methods tend to become very sensitive to numerical instabilities in the evaluation of the matrix elements and the coefficients of $\bar{P}_U(x)$ [5]. Speed is important here because for one Monte-Carlo run, several thousand matrices have to be evaluated.

The algorithms that can be found in the literature, such as, e.g., the Faddeev–Leverrier method [6], are far from optimal on the point of view of numerical computation. Let a_A denote the coefficient of x^A in $P_U(x)$ and let $t_k = Tr(U^k)$. The coefficients t_k correspond to the power sums of the roots of $P_U(x)$. Their computation is straightforward. A relation between the a_A and the t_k can be obtained by equating the coeffients of the powers of x in the series expansion of both sides of

$$\det(xI - U) = x^N \exp\left[\mathrm{Tr}\ln\left(I - \frac{1}{x}U\right)\right]. \tag{4}$$

This leads to a relation already established by Newton [7], which formally can be written as

$$a_A = \frac{1}{A!}\det\begin{pmatrix} t_1 & 1 & 0 & 0 & \cdots & 0 \\ t_2 & t_1 & 2 & 0 & \cdots & 0 \\ t_3 & t_2 & t_1 & 3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ t_A & t_{A-1} & t_{A-2} & t_{A-3} & \cdots & t_1 \end{pmatrix}. \tag{5}$$

The Faddeev–Leverrier method [6] is a recurrence relation that implicitly generates the same relations. Though mathematically elegant, this formula is impractical: it is inaccurate because it is very sensitive to roundoff errors, especially if the eigenvalues of U differ by several orders of magnitude, which is common in determinant QMC methods. It is also inefficient because it requires $A/2$ matrix multiplications. Therefore the method is only useful for small A. Another method, suggested by Ormand *et al.* [8], amounts to the expression

$$a_A = \frac{1}{N}\sum_{m=0}^{N-1} e^{-i\frac{2\pi}{N}mA}\det\left(e^{i\frac{2\pi}{N}m}I - U\right). \tag{6}$$

In order to evaluate this expression efficiently, it is suggested to diagonalize U first. However, if U is diagonalized, the coefficients of $P_U(x)$ can be evaluated more easily by explicit construction of the polynomial

$$\det(xI - U) = \prod_{i=1}^{N}(x - \epsilon_i). \tag{7}$$

If this polynomial in x is constructed from the smallest up to the largest eigenvalue, a_A can be computed in an easy and stable way. The main computational effort is in the diagonalization of the matrix U. The polynomial can be constructed even more efficiently without diagonalization, as is explained in the next section.

In Section II we present an algorithm for calculating the coefficients of the characteristic polynomial of a general square matrix. In Section III we present the results of the numerical tests of the speed and accuracy of the algorithm.

II. ALGORITHM FOR THE CALCULATION OF THE CHARACTERISTIC POLYNOMIAL OF A GENERAL SQUARE MATRIX

The basic idea of the algorithm is to consider $I + xU$ as a matrix of polynomials in x. We then calculate the polynomial $\bar{P}_U(x)$ by evaluating the determinant in Eq. (2) using Gaussian elimination, with polynomials instead of scalars as matrix elements. As mentioned above, the coefficients of $\bar{P}_U(x)$ are closely related to the coefficients a_A. Because the multiplication of two polynomials of degree A requires about $2A^2$ flops and calculation of a determinant about $N^3/3$ polynomial multiplications, the calculation would require a number of the order of N^5 flops, which is too much for an efficient implementation. This number can be drastically reduced if U is transformed to an upper-Hessenberg form by a similarity transformation (a Householder reduction to Hessenberg form requires approximatly $\frac{10}{3}N^3$ flops [9]). This leaves the coefficients of $\bar{P}_U(x)$ unchanged. In order to calculate the determinant we transform $I + xU$ to upper diagonal form by Gaussian elimination, requiring now only N^2 polynomial multiplications. The Gaussian elemination is performed from the right bottom corner of the matrix up to the top left corner because in determinant QMC methods, the right bottom corner often contains the smallest elements, so that the summations involved are performed from small to large terms, which is less sensitive to roundoff errors than the summation the other way round. We start with $T^N = I + xU$. Now we bring column after column in upper triangular form. Suppose that T^j has column j to N already in upper triangular form, i.e.,

$$T^j_{ik} = 0, \tag{8}$$

for $i > k$ and $k \geq j$. Now we calculate

$$T^{j-1} = T^j G^j, \tag{9}$$

where

$$G^j_{ik} = \delta_{i,k}, \tag{10}$$

except for

$$\begin{aligned} G^j_{j-1\,j-1} &= T^j_{j\,j}, \\ G^j_{j\,j-1} &= -T^j_{j\,j-1}. \end{aligned} \tag{11}$$

In the end we obtain the upper triangular matrix $T^1 = T^N G^N G^{N-1} \cdots G^2$ so that

$$\bar{P}_U(x) = \det(I + xU) \tag{12}$$

$$= \det(T^N) \tag{13}$$

$$= \frac{\det(T^1)}{\det(G^N G^{N-1} \cdots G^2)} \tag{14}$$

$$= \frac{\prod_{i=1}^{N} T_{ii}^{1}}{\prod_{i=2}^{N} T_{ii}^{i}} \tag{15}$$

$$= T_{11}^{1} \tag{16}$$

because $T_{ii}^{1} = T_{ii}^{i}$. The operations can be ordered to minimize memory use. This leads to the following algorithm (t_{ki} corresponds with the coefficient of x^k in T_{ij}^{j}):

algorithm for calculating the coefficients of the characteristic polynomial of a N-by-N matrix U

reduce U to upper Hessenberg form

```
DO j = N, 1, -1
    DO i = 1, j
        DO k = N - j, 1, -1
            t_{k+1 i} = U_{i j} t_{k j+1} - U_{j+1 j} t_{k i}
        ENDDO
        t_{1 i} = U_{i j}                                   (17)
    ENDDO
    DO k = 1, N - j
        t_{k j} = t_{k j} + t_{k j+1}
    ENDDO
ENDDO
```

In the end t_{k1} is the coefficient of x^k in $\bar{P}_U(x)$. Then a_A is given by

$$a_A = (-1)^{N-A} t_{N-A1}. \tag{18}$$

This algorithm cannot break down and requires $N^3/2 + N^2 - N/2$ flops. If one needs only the coefficient of x^A in $\bar{P}_U(x)$ e.g., for the calculation of an A-particle trace in determinant QMC methods, the number of flops can be reduced further by calculating the polynomials only up to degree A. This is done by restricting the loop over k to values smaller than or equal to A. The sixth line in (17) then becomes

$$\text{DO } k = \text{MAX}(N - j, A), 1, -1. \tag{19}$$

Together with the Householder reduction to the upper-Hessenberg form this makes less than $4N^3$ flops. Diagonalization of the matrix U with the QR algorithm (suggested in [9] as the obvious method for the diagonalization of general square matrices) would require about $10N^3$ flops.

III. NUMERICAL TESTS

We have tested our algorithm numerically on its speed and accuracy. All the tests were done in Fortran 77 (DEC Fortran V3.8) on a Digital Alphastation 255/300 MHz workstation

running Digital Unix 3.2D. For the reduction to Hessenberg form and the diagonalization optimized Lapack routines were used [10]. For the part of the algorithm listed in the previous section only the standard optimizations of the Fortran compiler were used.

The speed was tested by calculating, for several matrix sizes, all the coefficients of the characteristic polynomial of 100 matrices with random elements (uniformely distributed between $-1/\sqrt{N}$ and $1/\sqrt{N}$). This was done with our algorithm and with complete diagonalization. The speed was measured by counting the number of cycles executed by the procedures of the algorithms (fewer cycles means faster calculation) using the "prof pixie" command. Table 1 lists the results. It is clear that our algorithm is much faster than complete diagonalization: from a factor 4.5 for small matrices to a factor 1.8 for large matrices. The decrease of this factor for large matrices can be understood by the fact that the routines for the reduction to Hessenberg form and diagonalization are strongly optimized while the routine for the algorithm of Section II is not and that these optimizations become more and more efficient with larger matrix sizes. Applying similar optimizations to our algorithm would significantly increase the ratio. Typical matrix sizes in determinant QMC methods range from 20-by-20 to 100-by-100. So there our unoptimized algorithm is a factor 2 to 4 faster than complete diagonalization.

TABLE 1

Comparison of the Number of Cycles Needed for the Calculation of the Coefficients of the Characteristic Polynomial of 100 Matrices with Random Elements, for Several Matrix Dimensions

Matrix dimension	Our algorithm	Complete diagonalization	Ratio
4	451400	1983818	4.39
6	1009400	4413843	4.37
8	1760300	8062663	4.58
10	2870100	12511637	4.36
15	7261300	29676436	4.09
20	14224100	55696656	3.93
25	25524300	93696774	3.67
30	41735900	144177197	3.45
35	63852100	209670202	3.28
40	90395400	290658105	3.22
45	126513800	388488344	3.07
50	171095900	512056714	2.99
60	284484900	794032492	2.79
70	447113900	1163945220	2.60
80	652709400	1630550332	2.50
90	926006900	2207209655	2.38
100	1251268900	2923248380	2.34
150	4268580600	8925077120	2.09
200	10018384500	20050929483	2.00
300	32993383700	63384810388	1.92
400	77249914100	145321243773	1.88
500	149825926400	278218705522	1.86
600	257427888100	474763616745	1.84
700	407433443000	745631287828	1.83
800	607094132500	1104878051129	1.82
900	863225666500	1564619645628	1.81

For testing the accuracy, we calculated 200,000 random samples with a determinant quantum Monte-Carlo method for the 4×4 Hubbard model with 8 up and 8 down electrons, with $U = 4$ and $\beta = 6$, following the method of Ref. [11], but taking the canonical trace instead of the grand-canonical one. For each sample, the canonical trace is given by the square of the coefficient of x^8 in the characteristic polynomial of a 16-by-16 matrix. The canonical trace was calculated in double precision and in single precision using our algorithm and complete diagonalization. As a measure for the accuracy we used the average absolute value of the difference between the single- and double-precision result divided by the double-precision result. For our algorithm we found a value of 0.00186 ± 0.00005 and for the complete diagonalization we found 0.00607 ± 0.00010 (error limits at 95% confidence level), indicating that our algorithm is more accurate. This could be expected since it requires less operations on the data. Furthermore complete diagonalization was much more sensitive to overflow errors than our algorithm. At values of $\beta > 6$ complete diagonalization (in single precision) was no longer usable.

IV. CONCLUSION

We have presented a stable and efficient algorithm for the calculation of the coefficients of the characteristic polynomial of a general square matrix. This algorithm is especially useful for determinant quantum Monte-Carlo calculations in the canonical ensemble because it is faster (a factor 2 to 4) and more accurate than the algorithms that can be found in the literature.

ACKNOWLEDGMENT

This work was supported by the Fund for Scientific Research—Flanders (Belgium).

REFERENCES

1. F. Haake, M. Kuś, H.-J. Sommers, H. Schomerus, and K. Życzkowski, *J. Phys. A: Math. Gen.* **29**, 3641 (1996).
2. R. Blankenbecler, D. J. Scalapino, and R. L. Sugar, *Phys. Rev. D* **24**, 2278 (1981).
3. J. E. Hirsch, *Phys. Rev. B* **31**, 4403 (1985).
4. S. E. Koonin, D. J. Dean, and K. Langanke, *Phys. Rep.* **278**, 1 (1997).
5. W. von der Linden, *Phys. Rep.* **220**, 53 (1992).
6. G. Helmberg, P. Wagner, and G. Veltkamp, *Lin. Alg. Appl.* **185**, 219 (1993); and references therein.
7. A. Mostowski and M. Stark, *Introduction to Higher Algebra* (Pergamon, Oxford, 1964).
8. W. E. Ormand, D. J. Dean, C. W. Johnson, G. H. Lang, and S. E. Koonin, *Phys. Rev. C* **49**, 1422 (1994).
9. G. H. Golub and C. F. Van Loan, *Matrix Computations* (The Johns Hopkins Univ. Press, London, 1989).
10. Available at http://www.netlib.org/netlib/lapack/.
11. S. R. White, D. J. Scalapino, R. L. Sugar, E. Y. Loh, J. E. Gubernatis, and R. T. Scalettar, *Phys. Rev. B* **40**, 506 (1989).