Faster algorithms for the characteristic polynomial

Clément Pernet

Arne Storjohann

David R. Cheriton School of Computer Science
University of Waterloo, Ontario,
Canada N2L 3G1
{cpernet,astorjoh}@uwaterloo.ca

ABSTRACT

A new randomized algorithm is presented for computing the characteristic polynomial of an $n \times n$ matrix over a field. Over a sufficiently large field the asymptotic expected complexity of the algorithm is $O(n^\theta)$ field operations, improving by a factor of $\log n$ on the worst case complexity of Keller–Gehrig's algorithm [8].

1. INTRODUCTION

Computing the characteristic polynomial of an $n \times n$ matrix A over a field F is a classical problem. Keller-Gehrig [8] gave three reductions of the problem to matrix multiplication. Let θ be an admissible exponent for the complexity of matrix multiplication: $O(n^\theta)$ operations from F are sufficient to multiply together two $n \times n$ matrices over F. In this paper all complexity bounds are in terms of field operations from F and we make the common assumption that $\theta > 2$.

Keller-Gehrig's third algorithm has cost $O(n^\theta)$ but only works for input matrices with restrictive genericity requirements. His first algorithm, a simplified version of the second, also only works for input matrices satisfying certain requirements. Of primary interest here is his second algorithm which works for all input matrices and has a worst case cost of $O(n^\theta \log n)$. The extra $\log n$ factor arises because the algorithm computes $A^2, A^4, A^8, \ldots, A^{\lceil \log_2 n \rceil}$ using binary powering

Computing the characterstic polynomial is closely related to other problem such as computing the minimal polynomial, testing two matrices for similarity, and computing the Frobenius canonical form. Known reductions to matrix multiplication for these problems, both deterministic [10, 11] and probabalistic [4, 5, 6], all have an extra $\log n$ factor in their worst case complexity bounds, arising because Keller–Gehrig's algorithm is used as a subroutine directly [5, 6, 10, 11] or because a logarithmic number of powers of A might be computed [4].

In this paper we combine ideas from [6, 8, 12] to get a new randomized algorithm for computing the characteristic polynomial. If F has at least $2n^2$ elements our Las Vegas algorithm has expected cost $O(n^\theta)$, matching the lower bound for this problem. Unlike Keller-Gehrig's $O(n^\theta \log n)$ algorithm, we proceed in phases for $k = 1, 2, 3, \ldots, n$ and thus converges arithmetically.

In Section 2 we introduce some notation and recall some facts about Krylov matrices. Section 3 gives a worked example of the new algorithm and offers an overview of Sections 4—6 which are devoted to presenting the algorithm and proving correctness. The new algorithm is not only of theoretical interest but also practical. In Section 7 we describes an implementation, present some timings, and compare with the previously most efficient implementations that we are aware of. Section 8 concludes.

2. NOTATION

We will frequently write matrices using a conformal block decomposition. A block is a submatrix comprised of a contiguous sequence of rows and columns. A block may be a single matrix entry or may have row or column dimension zero. The generic block label $*$ denotes that a block is possibly nonzero. Blocks that are necessarily zero are left unlabelled.

In this paper a companion matrix looks like

$$C_* = \begin{bmatrix} 0 & \cdots & 0 & * \\ 1 & \ddots & \vdots & \vdots \\ & \ddots & 0 & * \\ & & 1 & * \end{bmatrix} \in \mathsf{K}^{k \times k}. \tag{1}$$

and the sizes of companion blocks in the Frobenius canonical form are monotonically nonincreasing. Companion blocks may have dimension zero. We use the label B_* to denote a block which has all entries zero except for possibly entries in the last column. The dimension of a block labelled B_* will be conformal with adjacent blocks.

For a square matrix $A \in \mathsf{K}^{n \times n}$ and vector $v \in \mathsf{K}^{n \times 1}$, let $K_A(v, d)$ denote the Krylov matrix

$$\left[\; v \mid Av \mid \cdots \mid A^{d-1}v \;\right] \in \mathsf{K}^{n \times d}.$$

For $V \in \mathsf{K}^{n \times j}$ we denote by $\mathrm{Orb}_A(V)$ the subspace of K^n spanned by all the column vectors in $\left[V \mid AV \mid A^2V \mid \ldots\right]$.

FACT 1. *Let $A \in \mathsf{K}^{n \times n}$ be arbitrary and $U \in \mathsf{K}^{n \times n}$ be nonsingular. Then*

1. *$U = \left[\ K_A(v_1, d_1) \,\middle|\, \cdots \,\middle|\, K_A(v_n, d_m)\ \right]$ for some vectors $v_1, \ldots, v_m \in \mathsf{K}^{n\times 1}$ and positive integers $d_1, \ldots, d_m$ if and only if*

$$U^{-1}AU = \begin{bmatrix} C_1 & B_* & \cdots & B_* \\ B_* & C_2 & \cdots & B_* \\ \vdots & \vdots & \ddots & \vdots \\ B_* & B_* & \cdots & C_m \end{bmatrix} \tag{2}$$

with C_i a companion matrix of dimension d_i, $1 \le i \le m$.

2. *For any j, $1 \le j \le m$, the matrix (2) can be written as*

$$\left[\begin{array}{cccc|cccc} C_1 & B_* & \cdots & B_* & B_* & B_* & \cdots & B_* \\ B^* & C_2 & \cdots & B_* & B_* & B_* & \cdots & B_* \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ B_* & B_* & \cdots & C_j & B_* & B_* & \cdots & B_* \\ \hline & & & & C_{j+1} & B_* & \cdots & B_* \\ & & & & B_* & C_{j+2} & \cdots & B_* \\ & & & & \vdots & \vdots & \ddots & \vdots \\ & & & & B_* & B_* & \cdots & C_m \end{array}\right]$$

if and only the dimension of $\mathrm{Orb}_A([\, v_1 \mid \cdots \mid v_j\,])$ is equal to $d_1 + \cdots + d_j$.

The matrix in (2) is a *shifted Hessenberg form* with degree sequence $(d_1, d_2, \ldots, d_m)$, corresponding to the dimensions of the diagonal blocks. A shifted Hessenberg form with the shape

$$\begin{bmatrix} C_1 & B_* & \cdots & B_* \\ & C_2 & \cdots & B_* \\ & & \ddots & \vdots \\ & & & C_m \end{bmatrix}$$

is simply called a Hessenberg form.

Let e_i denote the i'th column of the identity matrix of the appropriate dimension.

3. OVERVIEW

The key to our algorithm is what we call a *k-uniform shifted Hessenberg form*: each diagonal companion block has dimension k, except for possibly the last which may have dimension less than k. For brevity we will refer to such a matrix as a k-shifted form. The algorithm proceeds in phases for increasing k. Phase k involves the transformation of a k-shifted form to a $(k+1)$-shifted form. We begin directly with a worked example of one phase of the algorithm and then fill in the details in the subsequent sections.

Consider the following 3-shifted form of order 14 over $\mathbb{Z}/(97)$, with diagonal blocks corresponding to the degree sequence $(3,3,3,3,2)$:

$$A = \begin{bmatrix}
 & & 57 & & & 93 & & & 63 & & & 32 & & 29 \\
1 & & 15 & & & 13 & & & 78 & & & 92 & & 33 \\
 & 1 & 26 & & & 88 & & & 53 & & & 70 & & 35 \\
 & & 0 & & & 22 & & & 4 & & & 23 & & 78 \\
 & & 21 & 1 & & 64 & & & 16 & & & 18 & & 43 \\
 & & 76 & & 1 & 12 & & & 77 & & & 56 & & 73 \\
 & & 62 & & & 50 & & & 92 & & & 57 & & 30 \\
 & & 13 & & & 6 & 1 & & 27 & & & 31 & & 65 \\
 & & 22 & & & 41 & & 1 & 76 & & & 9 & & 2 \\
 & & 64 & & & 59 & & & 47 & & & 67 & & 55 \\
 & & 39 & & & 19 & & & 83 & 1 & & 46 & & 36 \\
 & & 91 & & & 19 & & & 64 & & 1 & 82 & & 6 \\
 & & 55 & & & 73 & & & 49 & & & 66 & & 86 \\
 & & 42 & & & 24 & & & 48 & & & 31 & 1 & 12
\end{bmatrix}.$$

The striped Krylov matrix

$$\left[\, K_A(e_1,3) \,\middle|\, K_A(e_4,3) \,\middle|\, K_A(e_7,3) \,\middle|\, K_A(e_{10},3) \,\middle|\, K_A(e_{13},2) \,\right]$$

will be the identity matrix since the dimensions of the slices corresponding to the basis vectors $(e_1, e_4, e_7, e_{10}, e_{13})$ match the degree sequence $(3,3,3,3,2)$ of the diagonal blocks. Our idea, made precise in Section 4, is to compute what we call the Krylov extension of A by increasing the rank of each Krylov slice by at most one in a lexicographically maximal fashion. In this example the Krylov extension is $(4,4,3,2,1)$, corresponding to the full column rank matrix K:

$$\left[\, K_A(e_1,4) \,\middle|\, K_A(e_4,4) \,\middle|\, K_A(e_7,3) \,\middle|\, K_A(e_{10},2) \,\middle|\, K_A(e_{13},1) \,\right].$$

If the the Krylov extension is not monontically nonincreasing or does not correspond to a square matrix the algorithm will abort. For this example we may continue. The Krylov extension corresponds to the striped Krylov matrix

$$K = \left[\begin{array}{cccc|cccc|ccc|cc|c}
1 & & & 57 & & & & 93 & & & & & & \\
 & 1 & & 15 & & & & 13 & & & & & & \\
 & & 1 & 26 & & & & 88 & & & & & & \\
 & & & 0 & 1 & & & 22 & & & & & & \\
 & & & 21 & & 1 & & 64 & & & & & & \\
 & & & 76 & & & 1 & 12 & & & & & & \\
 & & & 62 & & & & 50 & 1 & & & & & \\
 & & & 13 & & & & 6 & & 1 & & & & \\
 & & & 22 & & & & 41 & & & 1 & & & \\
 & & & 64 & & & & 59 & & & & 1 & & \\
 & & & 39 & & & & 19 & & & & & 1 & \\
 & & & 91 & & & & 19 & & & & & & \\
 & & & 55 & & & & 73 & & & & & & 1 \\
 & & & 42 & & & & 24 & & & & & &
\end{array}\right].$$

As demonstrated by this example, it is always the case that K will be comprised entirely of identity vectors and columns of A. This follows from the fact that A is in k-shifted form and we are extending each Krylov slice to dimension at most $k+1$. Applying the similarity transform K to A we obtain the shifted Hessenberg form

$$K^{-1}AK = \left[\begin{array}{c|c} \bar{A} & B \\ \hline C & D \end{array}\right]$$

$$= \left[\begin{array}{ccccccccccc|ccc}
 & & & 19 & & & & 21 & & & 72 & & 56 & 69 \\
1 & & & 89 & & & & 65 & & & 3 & & 69 & 58 \\
 & 1 & & 55 & & & & 1 & & & 69 & & 82 & 85 \\
 & & 1 & 32 & & & & 10 & & & 50 & & 84 & 79 \\
 & & & 5 & & & & 77 & & & 42 & & 33 & 74 \\
 & & & 14 & 1 & & & 9 & & & 73 & & 78 & 73 \\
 & & & 54 & & 1 & & 41 & & & 19 & & 36 & 68 \\
 & & & 24 & & & 1 & 68 & & & 60 & & 47 & 76 \\
 & & & 29 & & & & 45 & & & 6 & & 8 & 32 \\
 & & & 39 & & & & 7 & 1 & & 84 & & 81 & 69 \\
 & & & 29 & & & & 45 & & 1 & 8 & & 8 & 93 \\
\hline
 & & & & & & & & & & & & 96 & 63 \\
 & & & & & & & & & & & 1 & 2 & 34 \\
 & & & & & & & & & & & & & 1
\end{array}\right].$$

Because the Krylov extension $(4, 4, 3, 2, 1)$ is monontonically nonincreasing we may partition it into two parts: $(4, 4, 3)$ corresponsing the the principal block $\bar{A}$ of $K^{-1}AK$ which is necessarily in 4-shifted form; and $(2, 1)$ corresponding to the trailing 3×3 block D. For this example the Krylov extension is what we call normal: the southwest block C of the matrix $K^{-1}AK$ is filled with zeroes and the trailing block D is in block upper triangular shifted Hessenberg form (also called simply Hessenberg form). If these conditions are not satisfied the algorithm will abort. For this example we may continue: the algorithm recursively computes the characteristic polynomial of the principal block $\bar{A}$ and multiplies the result by the characteristic polynomial of D, thus obtaining the characteristic polynomial of A.

In Section 4 we define precisely what we mean by the Krylov extension of a k-shifted form and give an algorithm for its computation that has cost $O(k(n/k)^\theta)$. As discussed in the example above, for the algorithm to be able to continue the Krylov extension must be *normal*, that is, must satisfy certain conditions (see Definition 1). In Section 6 we show how to precondition the input matrix so that all the Krylov extensions computed during the course of the algorithm will be normal with high probability.

In Section 5 we present an algorithm that takes as input a square matrix $A \in \mathsf{K}^{n \times n}$ over a field, and either returns the characteristic polynomial or reports "fail." The algorithm transform the principal block of the work matrix from k-shifted to $(k+1)$-shifted form for $k = 2, 3, \ldots, n$ in succession. The running time of the algorithm is bounded by $O(\sum_{k=1}^{n-1} k(n/k)^\theta)$, which can be shown to be $O(n^\theta)$ under the assumption that $\theta > 2$.

4. NORMAL KRYLOV EXTENSION

Note that the number of (non–trivial) diagonal blocks in a k-shifted form $A \in \mathsf{K}^{n \times n}$ is given by $m := \lceil n/k \rceil$, and that the dimension of the trailing block is $n - (m-1)k$. If we let $v_i = e_{(i-1)k+1}$ for $1 \leq i \leq m$, then the block Krylov matrix

$$\left[K_A(v_1, k) \,\middle|\, \cdots \,\middle|\, K_A(v_{m-1}, k) \,\middle|\, K_A(v_m, n-(m-1)k) \right] \quad (3)$$

will be equal to I_n.

Definition 1. *The* Krylov extension *of a k-shifted form $A \in \mathsf{K}^{n \times n}$ with $m := \lceil n/k \rceil$ diagonal blocks is the lexicographically maximal sequence $(d_1, \ldots, d_m)$ of nonnegative integers that satisfies the following restrictions:*

- *$d_i \leq k+1$ for all $1 \leq i \leq m$;*
- *$K = \left[\; K_A(v_1, d_1) \,\middle|\, \cdots \,\middle|\, K_A(v_m, d_m) \; \right]$ has full column rank;*

where $v_i = e_{(i-1)k+1}$ for $1 \leq i \leq m$. The Krylov extension is said to be normal *if the following additional conditions are satisfied:*

1. *$d_1 + \cdots + d_m = n$;*
2. *$(d_1, \ldots, d_m)$ is monotonically nonincreasing;*
3. *$d_m \leq n - (m-1)k$;*
4. *The shifted Hessenberg form $K^{-1}AK$ has the shape*

$$K^{-1}AK = \left[\begin{array}{c|c} \bar{A} & B \\ \hline & D \end{array} \right],$$

where D is a Hesssenberg form (possibly of dimension zero) and $\bar{A}$ is $(k+1)$-shifted form of dimension $\bar{n} = d_1 + \cdots + d_{\bar{m}}$, where $\bar{m}$ is the minimal index such that $d_{\bar{m}} < k+1$.

We now describe an algorithm that computes the Krylov extension. Actually, the algorithm is only guaranteed to work if the Krylov extension is normal. If any of conditions 1, 2 or 3 of Definition 1 are not satisfied the algorithm will detect this and report failure. The idea of the algorithm is straighforward. Consider the $n \times (n+m-1)$ matrix E obtained from the matrix in (3) by extending the dimension of each Krylov slice from k to $k+1$, except for the last. Then E has all the columns of I_n plus an additional $m-1$ columns from A.

The following result follows from Fact 1.2 by considering the shape of $K^{-1}AK$ in case the Krylov extension is normal.

Lemma 1. *If the Krylov extension $(d_1, \ldots, d_m)$ of a k-shifted form $A \in \mathsf{K}^{n \times n}$ is normal, then the submatrix of E comprised of the rank profile columns is equal to the matrix K of Defintion 1.*

We next describe how to compute the column rank profile of the matrix E taking advantage of its structure.

Computing the column rank profile

Consider the matrix $F = E^T J$, where J is the anti-diagonal matrix such that $J_{i,j} = 1$ if $i + j = n + 1$ and 0 otherwise. The column rank profile of E is the row rank profile of F.

For example for the matrix

$$E = \begin{bmatrix} 1 & & & 10 & & & & 20 & & & & 30 \\ & 1 & & 11 & & & & 21 & & & & 31 \\ & & 1 & 12 & & & & 22 & & & & 32 \\ & & & 13 & 1 & & & 23 & & & & 33 \\ & & & 14 & & 1 & & 24 & & & & 34 \\ & & & 15 & & & 1 & 25 & & & & 35 \\ & & & 0 & & & & 26 & 1 & & & 36 \\ & & & 0 & & & & 27 & & 1 & & 37 \\ & & & 0 & & & & 28 & & & 1 & 38 \end{bmatrix}$$

the corresponding F is the matrix

$$F = \begin{bmatrix} & & & & & & & & 1 \\ & & & & & & & 1 & \\ & & & & & & 1 & & \\ 0 & 0 & 0 & 15 & 14 & 13 & 12 & 11 & 10 \\ & & & & & 1 & & & \\ & & & & 1 & & & & \\ & & & 1 & & & & & \\ 28 & 27 & 26 & 25 & 24 & 23 & 22 & 21 & 20 \\ & & 1 & & & & & & \\ & 1 & & & & & & & \\ 1 & & & & & & & & \\ 38 & 37 & 36 & 35 & 34 & 33 & 32 & 31 & 30 \end{bmatrix}$$

The row rank profile of F can be computed using gaussian elimination, processing each row in turn, starting from the first row to the last. Processing of a row involves either determining that the row has already been zeroed out, and hence is not included in the rank profile, or performing gaussian elmination to zero entries below the first non-zero entry in the row (the pivot). Processing of the first three rows consists in zeroing the coefficients below the ones. After processing the fourth row the matrix has the following shape:

$$F=\begin{bmatrix} &&&&&&&&1\\ &&&&&&&1&\\ &&&&&&1&&\\ 0&0&0&15&14&13&&&\\ &&&&&1&&&\\ &&&&1&&&&\\ &&&&83&84&&&\\ 28&27&26&&33&66&&&\\ &&1&&&&&&\\ &1&&&&&&&\\ 1&&&&&&&&\\ 38&37&36&&66&35&&& \end{bmatrix}$$

The key observation now is that afer processing rows 5 and 6, row 7 will be zeroed out and is therefore not in the rank profile. After the elimination is completed, the matrix has the form

$$F=\begin{bmatrix} &&&&&&&&1\\ &&&&&&&1&\\ &&&&&&1&&\\ 0&0&0&1&85&72&&&\\ &&&&&1&&&\\ &&&&1&&&&\\ &&&&&&&&\\ 28&27&26&&&&&&\\ &&1&&&&&&\\ &1&&&&&&&\\ 1&&&&&&&&\\ &&&&&&&& \end{bmatrix}.$$

The rank profile is $(1,2,3,4,5,6,8,9,10)$.

To take advantage of the structure of the matrix, we will perform the elimination on the $n\times m$ submatrix G formed by the dense rows of index $k+1, 2(k+1), \ldots, (m-2)(k+1), n$ of the matrix F.

In the previous example,

$$G=\begin{bmatrix} 0&0&0&15&14&13&12&11&10\\ 28&27&26&25&24&23&22&21&20\\ 38&37&36&35&34&33&32&31&30 \end{bmatrix}$$

It is then sufficient to keep track of the structured rows by the vector ℓ of their indices: if H is the submatrix formed by these n rows, $\ell[i]=j \Leftrightarrow H_{i,j}=1$. At the beginning of the elimination $H=J$ and so $\ell=[n,n-1,\ldots,1]$.

Now consider the processing of the ith row of G if we include pivoting.

- The coefficients $F_{i,\ell[j]}\ \forall j\le k\times i$ are set to zero to simulate the elimination of the corresponding structured rows above.
- The vector ℓ has to be updated with the permutation that may be used to find the first non zero pivot on the current row.

The elimination on G can be performed in time $O(n(n/k)^\theta)$ using the LQUP algorithm of Ibarra, Moran & Hui [7]. The only modification is to incorporate the operations listed above into the last recursion level of the algorithm (for $m=1$). In the following algorithm we will denote the subroutine just described by `StructuredRankProfile`.

THEOREM 1. *Algorithm* `Extension` *is correct. The cost of the algorithm is* $O(k(n/k)^\theta)$.

5. CHARACTERISTIC POLYNOMIAL VIA ARITHMETIC PROGRESSION

Algorithm 1 `Extension`(A,n,k)

Require: A k-shifted form $A\in\mathsf{K}^{n\times n}$.
Ensure: The Krylov extension $(d_1,\ldots,d_m)$ of A, or fail.

```
/* Fail will be returned if any of conditions 1,
2 and 3 of Definition 1 are not satisifed.  Fail
will not be returned if the Krylov extension is
normal.  */
```

Form the $n\times(n+m)$ matrix E from (3) by extending the dimension of each Krylov slice by one.
$[j_1,\ldots,j_r] :=$ `StructuredRankProfile`(E,k).
if there exists a monotonicall nonincreasing sequence $(d_1,\ldots,d_m)$ increasing such that $[j_1,\ldots,j_r]$ is equal to $[1,\ldots,d_1,\ (k+1)+1,\ldots,(k+1)+d_2,\ \ldots,\ (m-1)(k+1)+1,\ldots,(m-1)(k+1)+1+d_m]$ **then**
  return $(d_1,\ldots,d_m)$
else
  return `Fail`.
end if

Let $A\in\mathsf{K}^{n\times n}$ be a k-shifted form with a normal Krylov extension $(d_1,\ldots,d_m)$. Let K be the striped Krylov matrix associated to the extension. A key step of the algorithm is perform the change of basis $K^{-1}AK$. To perform this efficiently the structure of the matrices A, K and $K^{-1}AK$ have to be taken into account.

Note that all the columns of $K^{-1}AK$ will be known columns of I_n except for the at most m columns $\{d_1, d_1+d_2,\ldots,d_1+d_2+\cdots+d_m\}$. Let Y be the submatrix of K corresponding to these columns. To recover $K^{-1}AK$ we need to compute $K^{-1}AY$.

Let $*_p$ denote a permutation matrix. Up to a row and column permutations, which may be deduced from the degree sequence of diagonal blocks in A, we have

$$A=*_p\left[\begin{array}{c|c} I_{n-m} & * \\ \hline & * \end{array}\right]*_p\,.$$

Similarly, since K will have fewer than $\lfloor n/(k+1)\rfloor$ columns which are not identity vectors, and $\lfloor n/(k+1)\rfloor < m$, up to row and column permutations, which may be deduced from $(d_1,\ldots,d_m)$, we have

$$K=*_p\left[\begin{array}{c|c} I_{n-m} & * \\ \hline & * \end{array}\right]*_p\,.$$

Note that K^{-1} can be expressed similarly to K. This shows

$$K^{-1}AY=*_p\left[\begin{array}{c|c} I_{n-m} & * \\ \hline & * \end{array}\right]^{-1}*_p\left[\begin{array}{c|c} I_{n-m} & * \\ \hline & * \end{array}\right]*_p\,Y.$$

This gives the following result.

LEMMA 2. *Let $K\in\mathsf{K}^{n\times n}$ be the striped Krylov matrix corresponding to the uniform Krylov extension $(d_1,\ldots,d_m)$ of a k-shifted form $A\in\mathsf{K}^{n\times n}$. There exists an algorithm* `Transform` *that takes as input $(A,k,(d_1,\ldots,d_m))$ and returns $K^{-1}AK$. The cost of the algorithm is $O(k(n/k)^\theta)$ field operations from* K.

Assembling these components together gives Algorithm 2 (`CharPolyRec`) that recursively computes the characteristic polynomial of the input matrix or returns `fail`. Each recursive step correspond to the transformation from a k-shifted form to a $k+1$-shifted form.

Algorithm 2 CharPolyRec(A, n, k, x)

Require: A k-shifted form $A \in \mathsf{K}^{n\times n}$, an indeterminate x.
Ensure: return $\det xI - A$, or `fail`.
  if $n = k$ **then**
    Return $\det(xI - A)$
  else
    $(d_1, \ldots, d_m) :=$ Extension(A, k)
    `/* If the call to Extension fails then abort and return fail */`
    $\bar{m} :=$ minimal index with $d_{\bar{m}} < k + 1$
    $\bar{n} := d_1 + \cdots + d_{\bar{m}}$
    $\left[\begin{array}{c|c} \bar{A} & B \\ \hline C & D \end{array}\right] :=$ Transform$(A, k, (d_1, \ldots, d_m))$
    `/* If` C `is not the zero matrix then abort and return fail */`
    Return CharPolyRec$(\bar{A}, \bar{n}, k + 1, x) \times \det(xI - D)$
  end if

THEOREM 2. *Algorithm 2 (CharPolyRec) returns the characteristic polynomial of the input matrix or* `fail`. *The cost of the algorithm is $O(n^\theta)$.*

PROOF. The complexity is deduced from the following arithmetic progression:

$$\sum_{k=1}^{n} k(n/k)^\theta = n^\theta \sum_{k=1}^{n} (1/k)^{\theta-1} = O(n^\theta)$$

since $\theta - 1 > 1$. □

To ensure that the algorithm will only fail with a bounded probability, the input matrix A has to be preconditioned by a random similarity transformation. This gives the following algorithm.

Algorithm 3 CharPoly(A, n, x)

Require: A matrix $A \in \mathsf{K}^{n\times n}$, an indeterminate x.
Ensure: return $\det(xI - A)$, or fail.
  `/* Fail will be returned with probability at most` $1/2$. `We require` $\#\mathsf{K} \geq 2n^2$. `*/`
  $\Lambda :=$ a subset of K with $\#\Lambda \geq 2n^2$
  Choose $V \in \mathsf{K}^{n\times n}$ with entries uniformly and randomly from Λ.
  $B := V^{-1}AV$ `/* If` V `is singular then abort and return fail */`
  Return CharPolyRec$(B, n, 1, x)$

The probability analysis of Algorithm 3 (CharPoly) will be detailed in Section 6; the cost of the algorithm is obviously still $O(n^\theta)$ field operations.

6. PRECONDITIONING

Let $A \in \mathsf{K}^{n\times n}$ be an arbitrary matrix. In this subsection we prove that Algorithm 2 (CharPolyRec) will not fail when given as input the tuple $(B, n, 1, x)$, where $B = V^{-1}AV$ and V is filled with algebraically independent indeterminates. Upon specialization of the indeterminates with random field elements, as is done by Algorithm 3 (CharPoly), a bound of $1/2$ on the probability of failure will follow due to the Schwartz-Zippel Lemma [9, 13].

The proof of the following theorem is similar to and inspired by [12, Proof of Proposition 6.1]. Note that for convenience we assume that the Frobenius form of A has n blocks, some of which may trivial (i.e., 0×0). In the statement of the theorem this means that some of the f_* and d_* may be zero.

THEOREM 3. *Let $A \in \mathsf{K}^{n\times n}$ have Frobenius form with blocks of dimension $f_1 \geq \cdots \geq f_n$, and let $v_1, \ldots, v_n$ be the columns of a matrix V filled with algebraically independant indeterminates. Suppose $(d_1, \ldots, d_n)$ is monotonically nonincreasing sequence of nonnegative integers. Then*

$$K = \left[\; K_A(v_1, d_1) \mid \cdots \mid K_A(v_n, d_n) \;\right]$$

has full column rank if and only if $\sum_{j=1}^{i} d_j \leq \sum_{j=1}^{i} f_j$ for all $1 \leq i \leq n$.

PROOF. The "only if" direction follows because for any block X of i vectors, even a generic block $X = [v_1 \mid \cdots \mid v_i]$, the dimension of $\mathrm{Orb}_A(X)$ is at most $\sum_{j=1}^{i} f_i$.

To prove the other direction we specialize the indeterminates in the vectors v_i. In particular, it will be sufficient to construct a full column rank matrix

$$K = \left[\; K_1 \mid \cdots \mid K_n \;\right]$$

over K such that each K_i is in Krylov form and has dimension d_i, $1 \leq i \leq n$. Consider a change of basis matrix $U \in \mathsf{K}^{n\times n}$ such that $U^{-1}AU$ is in Frobenius form. Then

$$U = \left[\; K_A(u_1, f_1) \mid \cdots \mid K_A(u_n, f_n) \;\right]$$

is nonsingular. Let

$$\bar{K} = \left[\; \bar{K}_1 \mid \cdots \mid \bar{K}_n \;\right]$$

be the submatrix of U such that each $\bar{K}_i$ has the form

$$\bar{K}_i = \left[\; K_A(u_i, \min(f_i, d_i)) \mid E_i \;\right],$$

where E_i has dimension $d_i - \min(f_i, d_i)$, and the columns of $E_1, E_2, \ldots, E_n$ are filled with unused columns of U, using the columns in order from left to right. Then $\bar{K}$ has full column rank and each $\bar{K}_i$ has the correct dimension. Our goal now is to demonstrate the existence of an invertible matrix T such that $K = \bar{K}T$ has the desired form. We will construct $T = I + \sum_{i=1}^{n}(T_i - I)$ where each T_i is unit upper triangular. For all i with $d_i \leq f_i$ no transformation of $\bar{K}_i$ is required: set $T_i = I$. If $f_i < d_i$ then

$$\bar{K}_i = \left[\; K_A(u_i, f_i) \mid K_A(A^{s_1}u_{j_1}, t_1) \mid \cdots \mid K_A(A^{s_k}u_{j_k}, t_k) \;\right]$$

where, by construction of the E_i, we have $j_1 < j_2 < \cdots < j_k$, $t_l = f_{j_l} - s_l$ for $1 \leq l \leq k - 1$, and $t_k \leq f_k$. Using the property $\sum_{j=1}^{i} d_j \leq \sum_{j=1}^{i} f_j$ we have $j_k < i$. Since $(d_1, \ldots, d_n)$ is monotonically nondecreasing and $K_A(v_l, d_l)$ is a submatrix of $\bar{K}_i$ for $1 \leq l \leq k$, it follows that

$$s_l \geq d_i \quad \text{for} \quad 1 \leq l \leq k. \tag{4}$$

We can write $\bar{K}_i$ as the sum of the following $k+1$ matrices:

$$\bar{K}_i = \left[\; K_A(u_i, f_i) \mid 0, \ldots, 0 \;\right] \tag{5}$$

$$+ \sum_{l=1}^{k-1} \left[\; 0, \ldots, 0 \mid K_A(A^{s_l}u_{j_l}, f_{j_l} - s_l) \mid 0, \ldots, 0 \;\right] \tag{6}$$

$$+ \left[\; 0, \ldots, 0 \mid K_A(A^{s_k}u_{j_k}, t_k) \;\right] \tag{7}$$

To bring the matrix in (5) to Krylov form we may add suitable linear combinations of the first f_i columns to the last $d_i - f_i$ columns to obtain

$$\left[\; K_A(u_i, f_i) \mid K_A(A^{f_i}u_i, d_i - f_i) \;\right].$$

This is possible since the i'th invariant subspace has dimension f_i. Denote by $T_i^{(1)}$ the unit upper triangular matrix which effects this transformation on $\bar{K}$.

Now consider the matrix in (7). The Krylov space needs to be extended on the left to fill in the zero columns as follows:

$$\left[\; K_A(A^s u_{j_k}, s_k - s) \;\middle|\; K_A(A^{s_k} u_{j_k}, t_k) \;\right].$$

From (4) we may conclude that $s \geq 0$. Since $K_A(A^s u_{j_k}, s_k - s)$ is a submatrix of $\left[\; \bar{K}_1 \mid \cdots \mid \bar{K}_{i-1} \;\right]$, we need only copy former to latter columns. Denote by $T_i^{(2)}$ the unit upper triangular matrix which effects the copying on these columns. Similarly, there exists a unit upper triangular matrix $T_i^{(3)}$ which extends the Krylov sequence of the matrix in (6) to the left and right. Let $T_i = T_i^{(1)} + T_i^{(2)} + T_i^{(3)}$. $\square$

In the following corollary the matrix A and V are as in Theorem 3, that is, $A \in \mathsf{K}^{n\times n}$ has Frobenius form with blocks of dimension $f_1 \geq f_2 \geq \cdots \geq f_n$ and V is an $n \times n$ matrix filled with indeterminates. The corollary follows as a result of Fact 1.2.

Corollary 1. *Let $B := V^{-1}AV$ and k satisfy $2 \leq k \leq n$. The lexicographically maximal sequence $(d_1, \ldots, d_n)$ of nonnegative integers such that:*

- *$d_i \leq k$ for all $1 \leq i \leq m$, and*
- *$K = \left[\; K_B(e_1, d_1) \mid \cdots \mid K_B(e_n, d_n) \;\right]$ has full column rank,*

will satisfy $d_1 + \cdots + d_n = n$ and can be written as

$$(d_1, \ldots, d_n) = (k, \ldots, k, d_{\bar{m}}, f_{\bar{m}+1}, f_{\bar{m}+2}, \ldots, f_n)$$

with $k > d_{\bar{m}} \geq f_{\bar{m}+1}$. Moreover,

$$K^{-1}BK = \left[\begin{array}{c|c} \bar{A} & B \\ \hline & D \end{array}\right]$$

is in shifted Hessenberg form, where $\bar{A}$ is in $(k+1)$-shifted form of dimension $\bar{n} = d_1 + \cdots + d_{\bar{m}}$, and D is in Hessenberg form.

Each entry of $VK = \left[\; K_A(v_1, d_1) \mid \cdots \mid K_A(v_n, d_n) \;\right]$ is a linear combination of indeterminates of V. If follows that the determinant of VK is a nonzero polynomial in the indeterminates of V with total degree at most n.

Let $K_1 = I_n$ and K_i be the matrix of Corollary 1 for $k = i$, $2 \leq i \leq n$. Given as input $(B, n, 1, x)$, Algorithm 2 (`CharPolyRec`) will perform a change of basis at each step and computes the structured Krylov extension $K_{i-1}^{-1}K_i$ for $i = 2, 3, \ldots, n$. Let Δ be the product of the determinant of V and each matrix VK_i. Then Δ is a nonzero polynomial of total degree bounded by n^2. The next result now follows from the Schwartz-Zippel lemma.

Theorem 4. *Algorithm 2 (`CharPoly`) will return fail with probability at most $1/2$.*

We remark that the randomized Frobenious form algorithms in [5, 6] rely on the fact that the diagonal blocks in the Hessenberg form $K_n^{-1}BK_n$ will be those of the Frobenious form of A, and thus require that the determinant of the single matrix VK_n not vanish upon specialization of V with random field elements.

7. IMPLEMENTATION

In this section we discuss an implementation of the new characteristic polynomial algorithm that is modified to perform the preconditinoing step more efficiently in practice. Actually, the algorithm is adaptive and involves a parameter that is highly architecture-dependant and must be set experimentally. We present experiments comparing the practical performance of our implementation with several others softwares.

The implementation we describe here makes use of the FFLAS-FFPACK library[1]. This C++ library provides the efficient basic routines such as matrix multiplications and LQUP decomposition that make use of the level 3 BLAS numerical routines [1, 2].

7.1 Efficient preconditioning

Although it does not affect the asymptotic complexity, the preconditioning phase $V^{-1}AV$ of Algorithm 3 (`CharPoly`) is expensive in practice. This preconditioning phase can also be achieved by modifying Algorithm 2 (`CharPolyRec`) to compute the first Krylov extension using random vectors from Λ instead of identity vectors.

Our heuristic for this preconditioning step is to compute a block Krylov matrix $M = [U|AU|\ldots|A^{c-1}U]$ where U is formed by $\lceil n/c \rceil$ random vectors, for some paramter c. If this matrix is non singular, then the matrix $M^{-1}AM$ will be in c-shifted form (up to row and column permutations) and Algorithm 2 (`CharPolyRec`) can be called with shift parameter $k = c$ instead of $k = 1$. If $r = \text{rank}(M) < n$ then the linearly independent columns of M can be completed into a non singular matrix $\overline{M}$ by adding $n - r$ columns at the end, and we obtain the block upper triangular matrix

$$\overline{M}^{-1}A\overline{M} = \begin{bmatrix} H_c & * \\ & R \end{bmatrix}$$

where the $r \times r$ matrix H_c is in c-shifted form (up to row and column permutations). The characteristic polynomial of this matrix is computed by two recursive calls on the diagonal blocks H_c and R. Algorithm 4 (`CharPoly`) gives the algorithm with this modified preconditioning step.

Further explanations on the completion of M into $\overline{M}$ using the LQUP decomposition can be found in [3]. Note that again, only c columns of the matrix H_c have to be computed, which makes the computation of B much cheaper.

As c gets larger, the slices of the block Krylov matrix K become smaller. In the extreme case $c = n$, the algorithm computes the usual Krylov matrix of only one vector. In this case, the algorithm is equivalent to the algorithm `LU-Krylov` presented in [3, algorithm 2.2]. Assuming $\theta = 3$ the leading constant of algorithm `LU-Krylov` is competitive $(2.66n^3)$ but the algorithm does not fully exploit matrix multiplication. At the opposite, the case $c = 2$ corresponds to Algorithm 3 (`CharPoly`): it reduces the problem fully to matrix multiplication. The preconditioning parameter c makes it possible to balance the computation between these two algorithms.

Figure 1 displays the computation time of the algorithm for different values of c. Three matrices of order 5000 are used: they differ in the number of blocks in their Frobenius form. For $c < 55$, the timings are decreasing when

[1]This library is available online at `http://www-ljk.imag.fr/membres/Jean-Guillaume.Dumas/FFLAS` or within the LinBox library `http://www.linalg.org`

Algorithm 4 CharPoly(A, n, x)

Require: A matrix $A \in \mathsf{K}^{n\times n}$, an indeterminate x, a preconditioning parameter c.

Ensure: $\det(xI - A)$, or fail.

`/* Fail will be returned with probability at most` $1/2$ `if` $\#\mathsf{K} > 2n^2$ `*/`

$\Lambda :=$ a subset of K with $\#\Lambda \geq 2n^2$

$m := \lceil n/c \rceil$

Choose $V \in \mathsf{K}^{n\times m}$ with entries uniformly and randomly from Λ.

Compute the $n \times (c\lceil n/c \rceil)$ matrix

$$M = [V|AV|\dots|A^{c-1}V]$$

Compute (L, Q, U, P), the LQUP decomposition of M^T.

Let $r = \text{rank}(M^T)$

$\overline{M} := \left[MQ \left[\; I_r \; | \; 0 \; \right] \middle| P^T \begin{bmatrix} 0 \\ I_{n-r} \end{bmatrix} \right]$

$B := \overline{M}^{-1} A \overline{M} = \begin{bmatrix} H_c & * \\ & R \end{bmatrix}$

Return CharPolyRec$(H_c, n, c, x) \times$ CharPolyRec$(R, n, 0, x)$

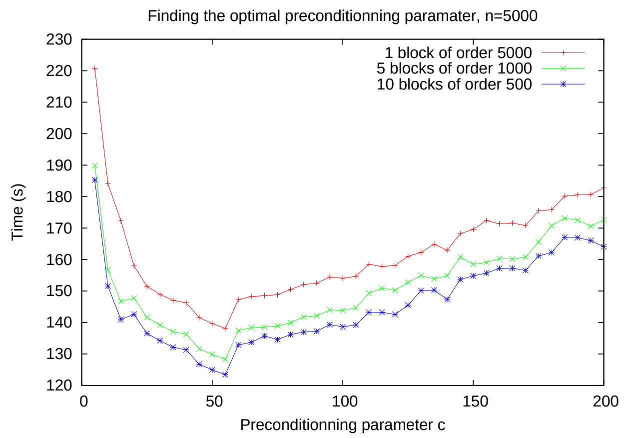

Figure 1: Finding the optimal preconditioning parameter c for matrices of order 5000, Itanium2-64 1.3Ghz, 192Gb

c increases, which shows the advantage of using the block Krylov preconditioning for a large enough value for c. Then the timings increase again for larger c. In these cases, the dominant operation is the computation of the block Krylov matrix M by many matrix multiplications of uneven dimensions. The matrix multiplication routine used will be more efficient for computing one $n \times n$ by $n \times n$ product rather than c $n \times n$ by $n \times n/c$ products, due to both the level 3 BLAS behaviour and the use of sub-cubic matrix multiplication. The optimal value $c = 55$ gives here the best timings. This value is not only depending the matrix dimension, but also on the architecture and the BLAS that are used, since it is linked with the ratio between the efficiency of the matrix vector product and the matrix matrix multiplication.

Note that the algorithm gets faster as the dimension of the largest block decreases.

7.2 Timing comparisons

We now compare the running time of our implementation of Algorithm 4 `CharPoly` with that of other state of the art implementations of characteristic polynomial algorithms. The routine `LU-Krylov`, available in the FFLAS-FFPACK and LinBox, libraries was shown to be the most efficient implementation in most cases [3].

For all the following experiments, we used the finite field $\mathbb{Z}/(547\,909)$. On one hand, it is large enough to ensure a high probability of success; none of the computations returned `fail`. On the other hand, the field size is small enough so that the FFLAS-FFPACK routines can make efficient use of the level 3 BLAS subroutines, using delayed modular reductions with the 53 bits of the `double` mantissa.

n	`LU-Krylov`	`New algorithm`
200	**0.024**	0.032
300	**0.06s**	0.088s
500	**0.248s**	0.316s
750	**1.084s**	1.288s
1000	2.42s	**2.296s**
5000	267.6s	**153.9s**
10 000	1827s	**991s**
20 000	14 652s	**7097s**
30 000	48 887s	**24 928s**

Table 1: Computation time for 1 Frobenius block matrices, Itanium2-64 1.3Ghz, 192Gb

Table 1 presents the timings for the computation of the characteristic polynomial of matrices having only one block on their Frobenius form. The preconditioning parameter c has been set to 100 for these experiments. The new algorithm improves the computation time of `LU-Krylov` for matrices of order not less than 1000. For matrices of order 30 000, the improvement factor is about 47.6%, due to the fact that the new algorithm fully reduces to matrix multiplication and can better exploit the level 3 BLAS efficiency. Figure 2 presents these timings in a log scale graph. The slopes of the two lines, which corresponds to the exponent of their complexity, are both close to 3. However, the slope of

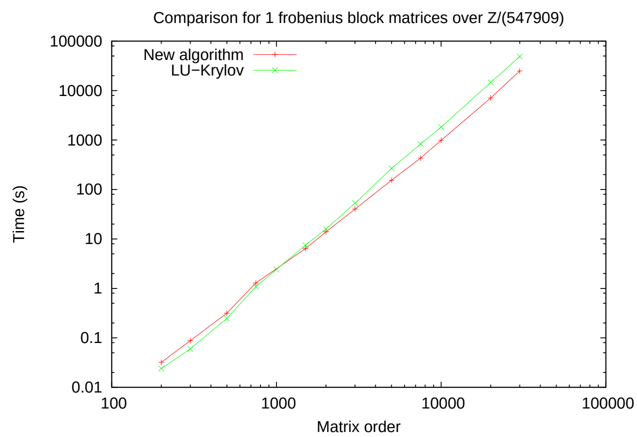

Figure 2: Timing comparison between the new algorithm and LU-Krylov, logarithmic scales, Itanium2-64 1.3Ghz, 192Gb

the `new algorithm` is slightly lower, indicating the effective use of sub-cubic matrix multiplication for this computation.

n	`magma-2.11`	`LU-Krylov`	`New algorithm`
100	0.010s	**0.005s**	0.006s
300	0.830s	0.294s	**0.105s**
500	3.810s	1.316s	**0.387s**
800	15.64s	4.663s	**1.387s**
1000	29.96s	10.21s	**2.755s**
1500	102.1s	33.36s	**7.696s**
2000	238.0s	79.13s	**17.91s**
3000	802.0s	258.4s	**61.09s**
5000	3793s	1177s	**273.4s**
7500	MT	4209s	**991.4s**
10 000	MT	8847s	**2080s**

Table 2: Computation time for 1 Frobenius block matrices, Athlon 2200, 1.8Ghz, 2Gb

MT: Memory thrashing

Lastly table 2 gives a comparison with `magma-2.11`[2]. Here again, our new implementation improves the computation time of this software, with a gain factor of about 13.8 for $n = 5000$. Moreover, its better memory management makes it possible to compute with larger matrices. On this machine, the efficiency ratio between matrix-vector and matrix multiplication is much lower than on the Itanium2. Therefore the new algorithm gets already faster for dimensions over 300.

8. CONCLUSIONS

We remark that the characteristic polynomial algorithm we have presented can easily be modified to compute the entire Frobenius form by checking some divisibility conditions of the polynomials induced by the blocks in the computed Hessenberg form. The additional cost is bounded by $O(n^\theta)$ since $\theta > 2$. Thus, we obtain a Las Vegas algorithm for computing the Frobenius form of a matrix over field that has expected cost $O(n^\theta)$.

To ensure a probability of success at least 1/2, we require that the ground field have at least $2n^2$ elements. If the field is too small we can work over an extension but a better solution (currently) would be the apply an alternative algorithm such as `LU-Krylov` discussed in the previous section for computing the characteristic polynomial, or the Frobenius form algorithm of Eberly [4].

For comparison, Eberly's Las Vegas Frobenius form algorithm has expected cost $O(n^\theta \log n)$, no restrictions on the field size, and it computes a similarity transform matrix as well as the form itself. Our algorithm has expected cost $O(n^\theta)$, requires the ground field to have size at least $2n^2$, and does not recover a similarity transform matrix in the same time.

On the one hand, recovery of a similarity transform matrix is undoubtedly useful for various applications [5]. On the other hand, for problems such as computing the minimal polynomial or testing two matrices for similarity the Frobenius form itself will suffice.

The main open problem we identify is to eliminate the condition on the field size while maintaining the cost bound $O(n^\theta)$: ideally the algorithm could be derandomized entirely. The currently fastest deterministic algorithm has cost $O(n^\theta(\log n)(\log\log n))$ [10, 11].

[2]We are grateful to the Medicis computing center hosted by the CNRS STIX lab : `medicis.polytechnique.fr/medicis` for the possibility of running magma on their machines

9. REFERENCES

[1] J.-G. Dumas, T. Gautier, and C. Pernet. Finite field linear algebra subroutines. In *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '02*, pages 63–74. ACM Press, New York, 2002.

[2] J.-G. Dumas, P. Giorgi, and C. Pernet. Finite field linear algebra package. In J. Gutierrez, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '04*, pages 119–126. ACM Press, New York, 2004.

[3] J.-G. Dumas, C. Pernet, and Z. Wan. Efficient computation of the characteristic polynomial. In M. Kauers, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '05*, pages 140–147. ACM Press, New York, 2005.

[4] W. Eberly. Asymptotically efficient algorithms for the Frobenius form. Technical report, Department of Computer Science, University of Calgary, 2000.

[5] M. Giesbrecht. *Nearly Optimal Algorithms for Canonical Matrix Forms.* PhD thesis, University of Toronto, 1993.

[6] M. Giesbrecht. Nearly optimal algorithms for canonical matrix forms. *SIAM Journal of Computing*, 24:948–969, 1995.

[7] O. Ibarra, S. Moran, and R. Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of Algorithms*, 3:45–56, 1982.

[8] W. Keller-Gehrig. Fast algorithms for the characteristic polynomial. *Theoretical Computer Science*, 36:309—317, 1985.

[9] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27:701–717, 1980.

[10] A. Storjohann. *Algorithms for Matrix Canonical Forms.* PhD thesis, Swiss Federal Institute of Technology, ETH–Zurich, 2000.

[11] A. Storjohann. Deterministic computation of the Frobenius form (Extended Abstract). In *Proc. 42nd Annual Symp. Foundations of Comp. Sci.*, pages 368–377, Los Alamitos, California, 2001. IEEE Computer Society Press.

[12] G. Villard. A study of Coppersmith's block Wiedemann algorithm using matrix polynomials. Technical Report RR 975-I-M, IMAG Grenoble France, April 1997.

[13] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proc. EUROSAM 79*, pages 216–226, Marseille, 1979.