

Math 642:550 — Summer 2008
MTTh 6:00–8:30 PM Hill 425
Prof. Bumby

Supplement 9, A robust method for finding characteristic polynomials

1. Introduction It is easy to read too much into the definition of the characteristic polynomial as a determinant. As soon as the n by n matrix has $n > 3$, it becomes extremely difficult to evaluate that determinant reliably. Furthermore, none of the methods for evaluating determinants work well when the matrix is allowed to contain an indeterminate. In this supplement, we describe an alternative method that can reliably obtain the characteristic polynomial of an integer matrix using hand computation for $n = 5$. It is also easy to program. A calculator version will be illustrated in lecture, and studies of the method from the middle of the twentieth century note that $n = 10$ was within reach on the digital computers of that time. However, if you are looking for eigenvalues and eigenvectors, an **approximate** method that concentrates on finding **some** eigenvectors to **high accuracy** is likely to be more useful in practice than this method, which can find eigenvalues only as roots of the characteristic polynomial.

The need for such a method was impressed on me when I wanted to compute a certain 5 by 5 integer determinant by hand and every method I tried gave me a different answer because of mistakes made while exercising the large number of arithmetic steps. Although the determinant is only one coefficient in the characteristic polynomial, it doesn't hurt to find more than is required. Since the calculations are less demanding and the method includes error-detection, this method is ideal for hand computation with integer matrices.

More details on the method can be found in F. R. Gantmacher, *The Theory of Matrices*, Chelsea, 1990, § IV.5. This follows the description by D. K. Faddeev given in the 1940s, although the method is attributed to work of Leverrier in 1840. In the 1940s, there were several independent announcements of a more efficient version. However, when those authors learned that their discovery was more than a century old, attempts to expand the announcement to an article seemed to have been set aside. A few articles were published that gave further explorations of the method. The expositions in Alton S. Householder, *The Theory of Matrices in Numerical Analysis*, Dover, New York, 1975. (ISBN 0-486-61781-5) and J. S. Frame, "Matrix functions and applications", *IEEE Spectrum* **1** (1964) (five articles in numbers 3–7) have good lists of published rediscoveries.

We will find that, while Frame's proof that the algorithm is correct, it reverses history and includes easy proofs of the important results cited in the historical development. Since it can appear unmotivated in its drive for efficiency, we begin with a historical treatment.

2. Powers and traces We begin by collecting some facts that are easily verified in the case when the characteristic polynomial has no repeated roots. The general case is also true, but the proof involves technical difficulties that are avoided in the modern proof.

The key to the method is that the sum of the elements on the diagonal of a square matrix M , called the **trace** of M is also equal to the sum of the eigenvalues of M . By itself, this isn't too useful. However, we also know that the eigenvalues of M^k are the k -th powers of the eigenvalues of M . Thus, computing the traces of powers of M gives the sums of powers of the roots of the characteristic polynomial.

A small example will be useful. Let

$$M = \begin{bmatrix} 3 & 1 & 5 \\ 3 & 3 & 1 \\ 4 & 6 & 4 \end{bmatrix}.$$

Then,

$$M^2 = \begin{bmatrix} \mathbf{32} & 36 & 36 \\ 22 & \mathbf{18} & 22 \\ 46 & 46 & \mathbf{42} \end{bmatrix} \quad M^3 = \begin{bmatrix} \mathbf{348} & 356 & 340 \\ 208 & \mathbf{208} & 216 \\ 444 & 436 & \mathbf{444} \end{bmatrix}.$$

If the eigenvalues of M are denoted $\lambda_1, \lambda_2, \lambda_3$, then equating traces and power sums gives

$$\begin{aligned} \lambda_1 + \lambda_2 + \lambda_3 &= 10 \\ \lambda_1^2 + \lambda_2^2 + \lambda_3^2 &= 92 \\ \lambda_1^3 + \lambda_2^3 + \lambda_3^3 &= 1000 \end{aligned}$$

Since M is recognized as 10 times a **Markov matrix**, we have $\lambda_1 = 10$, and without trying too hard, one finds $\lambda_3 = -\lambda_2$ and $2\lambda_2^2 = -8$. Thus, $\lambda_2 = 2i$. This gives the characteristic polynomial to be $x^3 - 10x^2 + 4x - 40 = (x - 10)(x^2 + 4)$ — **if the calculations are correct**. Since we have the characteristic polynomial, we can verify the the other eigenvalues of the Markov matrix $(1/10)M$ are of absolute value less than 1, although, in this case they are purely imaginary.

3. The Newton identities The coefficients of a polynomial are **elementary symmetric polynomials** of the roots found by expanding the product $\prod(x - \lambda_i)$. There are relations between these elementary symmetric polynomials and the power sums used in our example that were put in a systematic form by Newton. The expressions take slightly different forms depending on how one chooses the signs of the coefficients. We choose to emphasize the coefficients of the polynomial rather than the symmetric functions. To be consistent with Faddeev, we write the characteristic polynomial in the form

$$x^n - p_1x^{n-1} - p_2x^{n-2} - \dots - p_n,$$

with negative signs in front of all terms except the leading term, which has coefficient +1. In particular, the determinant turns out to be $(-1)^{n-1}p_n$. We also introduce s_k to stand for the sum of the k -th powers of the roots. The first few formulas are

$$\begin{aligned} p_1 &= s_1 \\ 2p_2 &= s_2 - p_1s_1 \\ 3p_3 &= s_3 - p_1s_2 - p_2s_1 \end{aligned}$$

and, for each k ,

$$kp_k = s_k - \sum \{ p_i s_j : i > 0, j > 0, i + j = k \}$$

If these identities are used in order, then everything needed on the right side of an identity will be either a given s_n or a p_k obtained at a previous step. In particular, the definitions of p_1 and s_1 clearly agree, and $p_1s_1 = s_1^2$ is the sum of all products of **ordered** pairs of roots $\lambda_i\lambda_j$. This sum splits into three parts according to whether $i < j$, $i = j$, or $i > j$. The middle part is s_2 , while each of the remaining parts is the sum of products of distinct pairs of roots (the second elementary symmetric polynomial) which is $-p_2$ because of the way we have written the coefficients. This gives the second Newton identity. A similar proof could be given of the remaining identities, but this style of algebra is no longer fashionable, so for now, we will accept these identities on good authority. They will follow from Frame's proof.

It is worth noting that these identities don't give direct **formulas** for the p_k in terms of the s_j , but rather describe an iterative **program** for computing them in individual cases. Even for hand computation, this approach is superior to writing a complicated algebraic expression and searching for ways to simplify it. The **structure** imposed by these identities is superior to any that might be discovered after a well-meaning, but misguided, attempt to simplify the final formula. When done my machine, the benefits of the identities also include ease of programming and a more transparent proof of correctness.

4. The Twentieth Century Improvement Instead of computing the powers of M whose traces give the s_k and applying Newton's identities to these numbers, we construct the p_k and a sequence of matrices $F_k(M)$ such that

$$F_1(M) = M \quad (B)$$

$$p_k = \frac{1}{k} \operatorname{tr}(F_k(M)) \quad (T_k)$$

$$B_k(M) = F_k(M) - p_k I \quad (P_k)$$

$$F_{k+1}(M) = MB_k(M) \quad (M_k)$$

Here, (B) is the basis of the inductive construction. It tells us what $F_1(M)$ is. Then for $k = 1, 2, 3, \dots$ we apply first (T_k) to use $F_k(M)$ to identify p_k and then (P_k) and (M_k) to determine $F_{k+1}(M)$. The only part of the algorithm that changes with k is the division by k in (T_k). This inductive definition is used to assure that

$$B_k(M) = M^k - p_1 M^{k-1} - p_2 M^{k-2} - \dots - p_{k-1} M - p_k I.$$

The Newton identities show that (T_k) gives the correct value of p_k . It is convenient to identify both $F_k(M)$ and $B_k(M)$. In hand computation, both must be written to show a full record of the algorithm to allow the source of any errors to be identified, although we will see that the algorithm has the ability to detect some errors. In machine computation, we will see that only the current $F_k(M)$ is used, but the $B_k(M)$ should be saved to allow the computation of eigenvectors.

One nice feature of this algorithm is that it is self-checking, leading to the error-detecting claim made earlier. The Cayley-Hamilton theorem (mentioned in our text only in exercises — see the index) says that a matrix satisfies its characteristic polynomial. In the context of this method, this says that $B_n(M)$ must turn out to be identically zero. **If you fail to obtain a zero matrix at this stage, then you know that you have made a mistake in your calculation.** However, the operations in the algorithm consist only of modifying the diagonal of a matrix and multiplying by a fixed matrix. Frame gave a direct proof that $B_n(M)$ is zero, which will appear later in these notes, so that the Cayley-Hamilton Theorem appears as a consequence. Thus, although designed as a computational method, the construction has the theoretical benefit of including proofs of both the Newton identities and the Cayley-Hamilton theorem. It may be possible to make a combination of mistakes that end with $B_n(M) = 0$, but a small number of algebraic blunders usually make large changes in quantities being computed.

5. Revisiting the first example Here is how the previous example looks when the newer method is used.

$$F_1(M) = \begin{bmatrix} \mathbf{3} & 1 & 5 \\ 3 & \mathbf{3} & 1 \\ 4 & 6 & \mathbf{4} \end{bmatrix} \quad (B)$$

$$p_1 = (3 + 3 + 4)/1 = 10 \quad (T_1)$$

$$B_1(M) = \begin{bmatrix} \mathbf{-7} & 1 & 5 \\ 3 & \mathbf{-7} & 1 \\ 4 & 6 & \mathbf{-6} \end{bmatrix} \quad (P_1)$$

$$F_2(M) = \begin{bmatrix} \mathbf{3} & 1 & 5 \\ 3 & \mathbf{3} & 1 \\ 4 & 6 & \mathbf{4} \end{bmatrix} \begin{bmatrix} \mathbf{-7} & 1 & 5 \\ 3 & \mathbf{-7} & 1 \\ 4 & 6 & \mathbf{-6} \end{bmatrix} = \begin{bmatrix} \mathbf{2} & 26 & -14 \\ -8 & \mathbf{-12} & 12 \\ 6 & -14 & \mathbf{2} \end{bmatrix} \quad (M_1)$$

$$p_2 = (2 - 12 + 2)/2 = -4 \quad (T_2)$$

$$B_2(M) = \begin{bmatrix} \mathbf{6} & 26 & -14 \\ -8 & \mathbf{-8} & 12 \\ 6 & -14 & \mathbf{6} \end{bmatrix} \quad (P_2)$$

$$F_3(M) = \begin{bmatrix} \mathbf{3} & 1 & 5 \\ 3 & \mathbf{3} & 1 \\ 4 & 6 & \mathbf{4} \end{bmatrix} \begin{bmatrix} \mathbf{6} & 26 & -14 \\ -8 & \mathbf{-8} & 12 \\ 6 & -14 & \mathbf{6} \end{bmatrix} = \begin{bmatrix} \mathbf{40} & 0 & 0 \\ 0 & \mathbf{40} & 0 \\ 0 & 0 & \mathbf{40} \end{bmatrix} \quad (M_2)$$

$$p_3 = (40 + 40 + 40)/3 = 40 \quad (T_3)$$

and $B_3(M) = 0$, as expected.

A convenient way to organize the work for hand computation is to arrange the n matrices $F_k(M)$ as the **second** row of a table, using the first row for the index k . The third row should contain the p_k , including the computation shown in the lines (T_k) . The fourth row should contain the matrices $B_k(M)$. This fourth row turn out to have the most important applications. In particular, if M is invertible, $F_n(M)$ will be $\det(M)I$, so the previous matrix in the fourth row must be the **adjugate** of M . On the other hand, if M is singular, $F_n(M) = \mathbf{0}$, so the columns of the previous matrix in the fourth row belong to the **nullspace** of M . If it is not itself a zero matrix, its nonzero columns give eigenvectors of M for $\lambda = 0$. If it is the zero matrix, then so is the matrix above it in the second row and the preceding $B_k(M)$ must have columns belonging to the nullspace of M . Continuing in this fashion, the last nonzero $B_k(M)$ has columns belonging to the nullspace of M .

6. A direct proof We have now completely described the algorithm. It only remains to show that it is correct. The appearance of the adjugate in the fourth row of our computational scheme becomes a **central feature** of the method in Frame's approach. This approach identifies the bottom row of our table with coefficient matrices when the adjugate matrix of $xI - M$ is written as a polynomial in x with matrix coefficients. Note that we are working with the matrix $xI - M$, with an **indeterminate** x , which is nonsingular as a matrix with polynomial entries.

The entries of the adjugate of $xI - M$ are $(n - 1)$ by $(n - 1)$ subdeterminants of $xI - M$, so each entry is a polynomial of degree at most $n - 1$. Alternatively, it is a polynomial of degree $n - 1$ with coefficients that are constant n by n matrices. We initially write the adjugate of $xI - M$ as

$$\sum_{j=0}^{n-1} x^{n-1-j} B_j.$$

with $B_0 = I$. Indices of summation may be suppressed by adopting the convention that j ranges over **all** integers, with the convention that $B_j = \mathbf{0}$ unless $0 \leq j \leq n - 1$. In particular, $B_n = \mathbf{0}$ **by definition**.

Since this matrix is the adjugate of $xI - M$,

$$(xI - M) \left(\sum x^{n-1-j} B_j \right) = f(x)I$$

where $f(x)$ is the characteristic polynomial of M . Equating the coefficients of x^{n-k} on each side of this equation gives

$$B_k - MB_{k-1} = -p_k I \quad (*)$$

for $k > 0$ ($k = 0$ requires $p_0 = -1$, but p_0 is part of the **prehistory** of the algorithm). This shows that B_k is obtained from B_{k-1} by multiplying by M and subtracting a scalar matrix — exactly the process that we just described. To prove the algorithm correct **and to prove the Cayley-Hamilton theorem** (since we have agreed that $B_n = \mathbf{0}$), it suffices to verify that the trace of MB_{k-1} is kp_k for all $k > 0$. This uses a more general result.

Lemma. If $M(x)$ is any matrix function, then

$$\frac{d}{dx} \det(M(x)) = \text{tr} \left(\text{adj}(M) \frac{d}{dx} M(x) \right)$$

Proof. Consider $\det(M(x))$ as a function of its entries $m_{ij}(x)$ and apply the chain rule to get

$$\frac{d}{dx} \det(M(x)) = \sum_{i,j} \frac{\partial}{\partial m_{ij}} \det(M(x)) \frac{d}{dx} m_{ij}(x).$$

The sum ranges over entries in the matrix. However, the partial derivatives of the determinant with respect to an entry m_{ij} is just the **cofactor** of that entry. This is written in the (j, i) position in the adjugate, so the sum over i gives the (j, j) entry of the product in the statement of the lemma. Summing over j gives the trace of this matrix, as claimed.

To apply to the characteristic polynomial, note that the derivative of $xI - M$ with respect to x is the identity matrix. Thus, the lemma shows that the coefficient of $-x^{n-k-1}$ (the minus sign appears because of our convention about naming coefficients) in the derivative of the characteristic polynomial is

$$(k - n)p_k = \text{tr } B_k, \quad (T_k^*)$$

which remains true when $k = 0$.

Since F_k and B_k differ by a scalar multiple of I , whether that scalar p_k is **defined** by (T_k) or (T_k^*) does not affect the validity of **both** equations. This demonstrates the validity of the algorithm.

Note that this leads to $F_0(M)$ being the zero matrix, which it must be if our convention about B_k is to be valid for negative k .

It is also useful to note that all B_k are polynomials in M , so that $MB_k = B_kM$ for all k .

7. Nullspaces If zero is an eigenvalue of M , the constant term of the characteristic polynomial, $-p_n$, is zero, so $MB_{n-1} = \mathbf{0}$. Thus, the column space of B_{n-1} is contained in the nullspace of M .

If $x = 0$ is a **simple root** of the characteristic polynomial (i.e., $x = 0$ has **algebraic multiplicity** 1), then $p_{n-1} \neq 0$. In this case $B_{n-1} = MB_{n-2} - p_{n-1}I = B_{n-2}M - p_{n-1}I$, so for every v in the nullspace of M , $B_{n-1}v = -p_{n-1}v$, so v lies in the column space of B_{n-1} . Thus, the nullspace of M is contained in the column space of B_{n-1} . In this, we have used $MB_{n-2} = B_{n-2}M$, which holds because B_{n-1} is a polynomial in M , as noted at the end of the previous section..

Combining these results, we find that these two spaces are the same when $x = 0$ is a simple root of the characteristic polynomial. Moreover, we have $B_{n-1} \neq \mathbf{0}$, which implies that M has rank $n - 1$, since the entries of B_{n-1} are cofactors of M . Thus, the nullspace of M is one dimensional (i.e., the **geometric multiplicity** of $x = 0$ is 1). In particular, any nonzero column of B_{n-1} generates the nullspace of M .

If the **algebraic** multiplicity of $\lambda = 0$ is $m > 1$, then the last m coefficients of the characteristic polynomial are zero. Thus, for $k > n - m$, $p_k = 0$ and $B_k = MB_{k-1}$. It follows that $M^m B_{n-m} = B_n = \mathbf{0}$. Thus, the column space of B_{n-m} is contained in the nullspace of M^m . The nullspace of M itself is not easily characterized if the algebraic multiplicity is bigger than 1: all of its properties are obtained from the action of M on this larger space.

Meanwhile B_{n-m} is a polynomial in M with a nonzero constant term. The Euclidean algorithm for polynomials leads to another polynomial in M , $K(M)$, whose product with B_{n-m} (on either side) is the identity plus a multiple of M^m . Now, if v is in the nullspace of M^m , $B_{n-m}K(M)v = v$. The components of $K(M)v$ give the coefficients for expressing v in terms of the columns of B_{n-m} . Thus, the nullspace of M^m is contained in the column space of B_{n-m} .

Thus, the nullspace of M^m and the column space of B_{n-m} are the same. A study of the action of M on the B_k for $k \geq n - m$ can be used to construct the **Jordan canonical form** and show that the dimension of the column space of B_{n-m} is m . However, this algorithm does not appear to simplify proofs of these results.

8. Eigenvectors Another use of the sequence of coefficients B_k is the **synthetic division** algorithm for finding the coefficients of a polynomial $P(x)$ given in terms of powers of x as a polynomial given in terms of powers of $x - c$. The constant term of such an expression is found by dividing the polynomial by writing

$$P(x) = (x - c) \cdot Q(x) + R$$

where R is a constant. The familiar **long division** algorithm could be used to obtain $Q(x)$ as a polynomial written in powers of x , continuing as long as terms of positive degree remain. For polynomials, this process can be described as

$$a_k x^k = (x - c) \cdot a_k x^{k-1} + c a_k x^{k-1}$$

This allows a neater way, known as **synthetic division**, to organize the computation. The first term on the right contributes to the quotient, while the second term is **added to** the original term in x^{k-1} to get the $a_{k-1} x^{k-1}$ needed to continue the process. The resulting sequence a_k are the coefficients of $Q(x)$ **in powers of x** , with the last term being the remainder, which is the constant term when $P(x)$ is written in terms of powers of $(x - c)$. When applied to polynomials with numerical coefficients, this process is written in only three lines. The top line contains the original coefficients, from highest degree to lowest. The second line contains $c a_k$ in the column of coefficients of x^{k-1} , and the third line contains the sum of the first two lines. At the end of the process, the third line contains the coefficients of $Q(x)$ and the remainder that will be the constant term of the new expression.

To obtain more terms in the expression, $Q(x)$ should be divided by $(x - c)$. The division should again use the synthetic division process

This can be applied to matrix polynomials in exactly the same way. To find eigenvectors, it is only necessary to find **the last nonzero term** when expressed in powers of $(x - c)$. If the eigenvalue is simple, this will be the constant term found by one application of synthetic division. The rule is to add c times the **new** B_{k-1} to B_k , for $k = 1, 2, 3, \dots$ to obtain the new B_k . When working with matrices, the multiplication of B_{k-1} by c should be relegated to scratch work and only $B_k^{(\text{new})} = B_k^{(\text{old})} + c B_{k-1}^{(\text{new})}$ written below the previous B_k . In this new row, the last element, which is the remainder, should be separated from the others, which are the coefficients in the quotient.

To return to our example, the original sequence B_0, B_1, B_2 is

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -7 & 1 & 5 \\ 3 & -7 & 1 \\ 4 & 6 & -6 \end{bmatrix} \quad \begin{bmatrix} 6 & 26 & -14 \\ -8 & -8 & 12 \\ 6 & -14 & 6 \end{bmatrix}.$$

When expressed in terms of powers of $(x - 10)$, this becomes

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 3 & 1 & 5 \\ 3 & 3 & 1 \\ 4 & 6 & 4 \end{bmatrix} \quad \begin{bmatrix} 36 & 36 & 36 \\ 22 & 22 & 22 \\ 46 & 46 & 46 \end{bmatrix}.$$

The last of these has columns that are eigenvectors, as predicted.

The same calculation can be done to translate by $2i$. The details a little messier, but the columns of the last matrix turn out to be multiples of $[1 + i, -i, -1]^T$.

This calculation can be done one column at a time. Most of the time, all columns are nonzero, so the first column is all that is needed to find an eigenvector. In the case of algebraic multiplicity 1, which also happens most of the time, the eigenspace is one dimensional, so a single eigenvector is all that is required.

An upper triangular matrix gives a useful way to illustrate the theoretical aspects of this calculation. All matrices appearing are triangular — including the matrix of eigenvectors. In particular, the eigenvector for the **last** eigenvalue (i.e., the one in the **last** column) has a nonzero entry in its last position. An upper triangular matrix whose columns are multiples of this vector will have all but its last column zero. In this case, one should **immediately** restrict attention to the last column in the search for an eigenvector.

9. Computer implementation These algorithms have been implemented in MATLAB and added to a [new Teaching Codes](#) directory. There are three functions:

- (1) `function [v, B3]=polyf(M)` starts with an n by n matrix M and produces the vector v that is the $p_k(M)$ for $1 \leq k \leq n$ and a **three dimensional array** $B3$ of **all** B_k for $0 \leq k \leq n-1$. Although output is generally suppressed, the matrix that would be B_n is shown to check that it is zero (since the program was tested in octave, but not MATLAB, this inelegant (though reliable) test was used in place of one that would be more informative. Both outputs should be requested and output suppressed to allow B_n to be seen while preventing a long display of the output arrays.
- (2) `function [Q3, V] = syndivf(X3, c)` starts with a three dimensional array $X3$ that contains the coefficients of a polynomial $P(x)$ and a scalar constant c . The output is a three dimensional array $Q3$ representing the **quotient** and a matrix V that is the **remainder** on division by $x - c$. This may be used directly in the case of **algebraic multiplicity 1** to get the eigenvector for a known eigenvalues c as the columns of V , but it is also used recursively in the function to be described next. If used directly, output should be suppressed since display of $Q3$ is undesirable.
- (3) `function Y3 = shiftf(X3, c)` also starts with a three dimensional array $X3$ that contains the coefficients of a polynomial $P(x)$ and a scalar constant c . The output is a three dimensional array $Q3$ representing the coefficients of powers of $y = x - c$.

These routines are intended to be applied only to **small** matrices, so the inefficiency of generating entire n by n by n arrays to find a single eigenvector in \mathbb{R}^n can be tolerated because of the simplicity of the programs. Examination of **suitable parts** of these large arrays can be useful in the case of eigenvalues of high algebraic multiplicity.

10. A triangular matrix example The eigenvectors of

$$M = \begin{bmatrix} 5 & -1 & 2 \\ 0 & 3 & -1 \\ 0 & 0 & 2 \end{bmatrix}.$$

have been found by other methods in [supplement 4](#) and [supplement 5](#), so it is useful to compare those calculations with this method. Since the roots of the characteristic are known to be 5, 3 and 2, the characteristic polynomial is known, so the algorithm mainly serves to find the matrices B_k . In particular, traces of the MB_k , computed using the elementary symmetric polynomials in the roots, must be 10, -31 and 30. The B_k are

$$B_1 = \begin{bmatrix} -5 & -1 & 2 \\ 0 & -7 & -1 \\ 0 & 0 & -8 \end{bmatrix} \quad \text{and} \quad B_2 = \begin{bmatrix} 6 & 2 & -5 \\ 0 & 10 & 5 \\ 0 & 0 & 15 \end{bmatrix}$$

Synthetic division for the eigenvalue 2 gives

$$\text{(quotient)} \begin{bmatrix} -3 & -1 & 2 \\ 0 & -5 & -1 \\ 0 & 0 & -6 \end{bmatrix} \quad \text{and} \quad \text{(remainder)} \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 3 \\ 0 & 0 & 3 \end{bmatrix}.$$

Synthetic division for the eigenvalue 3 gives

$$\text{(quotient)} \begin{bmatrix} -2 & -1 & 2 \\ 0 & -4 & -1 \\ 0 & 0 & -5 \end{bmatrix} \quad \text{and} \quad \text{(remainder)} \begin{bmatrix} 0 & -1 & 1 \\ 0 & -2 & 2 \\ 0 & 0 & 0 \end{bmatrix}.$$

Synthetic division for the eigenvalue 5 gives

$$\text{(quotient)} \begin{bmatrix} 0 & -1 & 2 \\ 0 & -2 & -1 \\ 0 & 0 & -3 \end{bmatrix} \quad \text{and} \quad \text{(remainder)} \begin{bmatrix} 6 & -3 & 5 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

The **remainder** matrices are multiples of the projection matrices appearing in the matrix exponential since they must also give both row and column eigenvectors.

11. Exercises In order to check that you have organized the steps of this algorithm correctly, begin with some examples where you know the characteristic polynomial

- A.** A 3 by 3 identity matrix.
- B.** The 4 by 4 matrix

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For this matrix, apply the section on eigenvectors to identify the eigenvectors that can be obtained from this computation.

Then try a more general 4 by 4 matrix.

- C.** The matrix

$$\begin{bmatrix} 3 & 1 & 5 & -2 \\ 3 & 3 & 0 & 1 \\ 4 & 6 & -4 & 3 \\ 2 & -1 & -2 & 0 \end{bmatrix}$$

End of Supplement