Is There an Error Correcting Code in the Base Sequence in DNA?

Larry S. Liebovitch, Yi Tao, Angelo T. Todorov, and Leo Levine
Center for Complex Systems, Florida Atlantic University, Boca Raton, Florida 33431 USA

ABSTRACT Modern methods of encoding information into digital form include error check digits that are functions of the other information digits. When digital information is transmitted, the values of the error check digits can be computed from the information digits to determine whether the information has been received accurately. These error correcting codes make it possible to detect and correct common errors in transmission. The sequence of bases in DNA is also a digital code consisting of four symbols: A, C, G, and T. Does DNA also contain an error correcting code? Such a code would allow repair enzymes to protect the fidelity of nonreplicating DNA and increase the accuracy of replication. If a linear block error correcting code is present in DNA then some bases would be a linear function of the other bases in each set of bases. We developed an efficient procedure to determine whether such an error correcting code is present in the base sequence. We illustrate the use of this procedure by using it to analyze the *lac* operon and the gene for cytochrome *c*. These genes do not appear to contain such a simple error correcting code.

INTRODUCTION

Modern technology utilizes numbers to represent and transmit information. The information in these numbers is contained in a string of digits. Additional error check digits are added to ensure that these numbers are correctly transmitted. These error check digits are functions of the other information digits. If an error does occur in the transmission, the error check digits computed from the altered information digits may differ from the error check digits transmitted. Thus, it is possible to detect common types of errors in transmission and to correct them. These error correcting codes are used in almost all forms of digital information in our society, such as the ISBN book codes (International Standard Book Numbers), the UPC scanner codes (Universal Product Codes), drivers' licenses, credit card numbers, airline ticket numbers, and bank account numbers (Gallian, 1991).

For example, an ISBN number such as 0-19-508013-0 consists of the nine information digits 0-19-508013 that encode the publisher and the title of a book. The 10th digit, 0, is an error check digit that is computed from the other digits. For the ISBN code, the first nine digits are multiplied, respectively, by the numbers 10, 9, 8, 7, 6, 5, 4, 3, and 2. The error check digit is equal to the negative of this sum modulus 11. In this case

$$-(0\times 10+1\times 9+9\times 8+5\times 7+0\times 6+8\times 5+0\times 4+1\times 3+3\times 2)\equiv 0 \pmod{11}, \quad (1)$$

and thus the error check digit $e = 0$. (The notation $a \equiv b \pmod{m}$, invented by Gauss, means that $a - b = km$, where k is an integer.) If an error were to occur in the transmission of this ISBN number, so that the fourth digit, which is a 5, was transformed to a 4, then

$$-(0\times 10+1\times 9+9\times 8+\mathbf{4}\times 7+0\times 6+8\times 5+0\times 4+1\times 3+3\times 2)\equiv 7 \pmod{11}, \quad (2)$$

and the value computed for the error check digit $e' = 7$. Because the value 7 computed from the information digits differs from the value 0 of the error check digit e, we know that there was an error in transmission. Different types of error code schemes can detect different types of transmission errors. The ISBN error code can detect 100% of all single digit changes and 100% of all transpositions of adjacent digits.

Biology also uses a digital code to represent and transmit information. Genetic information is encoded by a sequence of digits, each with one of four possible values: adenine, cytosine, guanine, and thymine (A, C, G, and T). This digital code is replicated to transmit genetic information to the next generation. This digital code is translated into an analog code in the shape of proteins that carry out the functions of living cells (Dawkins, 1987).

We wanted to know whether the base sequence in DNA also contains a digital error correcting code. Such a code would make it possible for repair enzymes to maintain the fidelity of the base sequence. It would also increase the accuracy of replication of DNA.

Previous authors have thought about the existence of such digital error correcting codes in the base sequence in DNA. Forsdyke (1981) proposed that the intervening base sequences (introns) that are not translated into proteins could contain error check information about the expressed base sequences (exons) that are translated into proteins. Rzeszowska-Wolny (1983) proposed that an appropriate arrangement of the DNA on nucleosomes may be crucial for this system to operate.

Received for publication 26 January 1996 and in final form 30 May 1996.
Address reprint requests to Dr. Larry S. Liebovitch, Center for Complex Systems, Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431. Tel.: 407-367-2239; Fax: 407-367-2223; E-mail: liebovitch@walt.ccs.fau.edu.

0006-3495/96/09/1539/06 $2.00

Our approach here is different from these previous approaches. Rather than constructing models of possible error correcting code mechanisms, we sought to develop an efficient procedure that could be used to search through sequence data to determine whether such an error code is present, or not present, in DNA. This paper presents the procedure that we developed. This procedure can detect a class of error correcting codes called linear block error correcting codes. It is capable of detecting the existence of such a code even when the number and placement of the error check digits are not known a priori. We illustrate how this procedure can be carried out by using it to analyze the base sequence of the *lac* operon and the structural gene for cytochrome *c*.

ASSUMPTIONS

In essence, we are asking, "Are the choices of some bases in DNA derivable from a function of other bases upstream or downstream in the native sequence?" If we consider functions of all possible types, then this is a far too general question to answer. We thus restricted our attention to searching for the existence of linear block error correcting codes (Guiasu, 1977). Almost all of the error correcting codes now in use in our technology are of this form. These codes have the form

$$(a_1x_1 + a_2x_2 + a_3x_3 + \ \ldots\ + a_nx_n) \equiv 0 \pmod{k}, \quad (3)$$

where x_i are the factors of the error check code and the a_i are the information and/or error check digits. For example, for the ISBN error code

$$(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = (10, 9, 8, 7, 6, 5, 4, 3, 2, 1), \quad (4)$$

and the modulus $k = 11$. The first nine digits a_i ($1 \le i \le 9$) are the information digits, and the 10th digit, a_{10}, is the error check digit.

In our case, the a_i are the bases of DNA, that is A, C, G, and T. It was easier to work with numbers rather than letters. Thus, we transformed the symbol sequence of bases into a sequence of integer digits by encoding A, C, G, and T, respectively, as 0, 1, 2, and 3. We chose the modulus $k = 4$. The use of modulus 4 means that the a_i are restricted to values of either 0, 1, 2, or 3, and thus the error check digits determined from Eq. 3 correspond to one of the four DNA bases.

We assume that the DNA consists of "words," each of which has n bases. Because three bases form a codon that encodes a specific amino acid or a termination signal, one might restrict $n = 3$. However, bases in DNA serve other functions besides their role in codons. For example, base sequences provide sites for DNA-binding proteins that regulate gene function, and the sequence of introns plays a role in the subsequent splicing of RNA. There is no a priori reason to suggest that the error correcting code would be necessarily limited to the same structure as that of the codons. Thus, to keep our method as general as possible, we let n be an adjustable parameter. In analyzing sequence data we let n vary over the range $3 \le n \le 8$.

Each word will consist of both information digits and error check digits. There will be one or more error check digits in each word. We assume that the same number of error check digits occurs in each word and that these digits are in the same location in each word. However, we do not need to assume how many error check digits are present in each word or where they are located.

Thus, we are now asking, "Are the choices of some bases in each word of n bases of DNA derivable from a linear combination (modulus 4) of other bases in that word?" To test whether this is the case, we choose a set of m words. We let b_{ij} be the value of the jth base in the ith word. We now apply Eq. 3 to each word. Thus, we now have the following set of m equations:

$$\begin{aligned} b_{11}x_1 + b_{12}x_2 + b_{13}x_3 + \ \ldots\ + b_{1n}x_n &\equiv 0 \pmod 4 \\ b_{21}x_1 + b_{22}x_2 + b_{23}x_3 + \ \ldots\ + b_{2n}x_n &\equiv 0 \pmod 4 \\ &\vdots \\ b_{m1}x_1 + b_{m2}x_2 + b_{m3}x_3 + \ \ldots\ + b_{mn}x_n &\equiv 0 \pmod 4. \end{aligned} \quad (5)$$

The values b_{ij} ($1 \le i \le m$, $1 \le j \le n$) are determined from the sequence of bases. Thus, we have a set of m equations with n unknowns x_i ($1 \le i \le n$). If there is an error check code present, we will find that there is a set of n values of the error code factors x_i that satisfy these m equations. We expect that the sequence of bases will be much longer than the length n of each word. Thus, there will be many words and hence m will be greater than n. That is, we have a set of m equations with n unknowns, where $m > n$. This set of equations can be satisfied by the trivial solution that $x_i \equiv 0$. Besides this trivial solution, we must determine whether this set of equations has no other solutions, one other solution, or a set of many other solutions.

PROCEDURE

First method: testing all possible solutions

Our first approach was to test all possible sets of solutions for the variables $x_1, x_2, \ldots x_n$ for the given sequence of bases. There are 4^n sets of possible solutions. We took the first word $m = 1$ consisting of the bases $b_{11}, b_{12}, \ldots b_{1n}$, and tested each solution to determine whether it satisfied the condition

$$b_{11}x_1 + b_{12}x_2 + \ \ldots\ + b_{1n}x_n \equiv 0 \pmod 4. \quad (6)$$

We kept a list of these solutions. We then tested each of these solutions to determine whether it satisfied the condition imposed by the base pairs $b_{21}, b_{22}, \ldots b_{2n}$ in the second word $m = 2$. That is, we tested each remaining solution to

determine whether it satisfied the condition

$$b_{21}x_1 + b_{22}x_2 + \ldots + b_{2n}x_n \equiv 0 \pmod 4. \quad (7)$$

We continued this process of testing the remaining solutions on additional words until either no solutions remained, or a set of one or more solutions was found for $x_1, x_2, \ldots x_n$ that were consistent for all m words. If solutions were found, then the values of $x_1, x_2, \ldots x_n$ specify the factors of the error check code. Only the solutions surviving from all of the previous tests needed to be tested for each new word.

To test the rate of convergence of this method we constructed different sets of test data. In the first set of test data we used words of $n = 6$ digits, where the first five digits, a_1, a_2, a_3, a_4, a_5, were the information digits with values 0, 1, 2, or 3, and the last digit a_6 was the error check digit computed from the solution to the condition

$$(a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 + a_5x_5 + a_6x_6) \equiv 0 \pmod 4, \quad (8)$$

where the factors $x_1, x_2, x_3, x_4, x_5, x_6$ of the error check code were given by

$$(x_1, x_2, x_3, x_4, x_5, x_6) \equiv (1, 3, 1, 3, 1, 1) \pmod 4. \quad (9)$$

The input data consisted of all $4^5 = 1024$ possible words. From the initial set of $4^6 = 4096$ of all possible solutions for $x_1, x_2, \ldots x_6$, this method found the following solutions:

$$\begin{aligned}
&(x_1, x_2, x_3, x_4, x_5, x_6) \equiv (1, 3, 1, 3, 1, 1) \pmod 4 \\
&(x_1, x_2, x_3, x_4, x_5, x_6) \equiv (0, 0, 0, 0, 0, 0) \pmod 4 \\
&(x_1, x_2, x_3, x_4, x_5, x_6) \equiv (2, 2, 2, 2, 2, 2) \pmod 4 \\
&(x_1, x_2, x_3, x_4, x_5, x_6) \equiv (3, 1, 3, 1, 3, 3) \pmod 4.
\end{aligned} \quad (10)$$

The first solution is the one that was used to generate the test data. Note that Eq. 3 is also satisfied if the variables $(x_1, x_2, x_3, x_4, x_5, x_6)$ are multiplied by the same common factor. The additional solutions are equivalent to the first solution multiplied, respectively, by the factors 0, 2, and 3 using arithmetic operations modulus 4. Because there are four possible common factors, 0, 1, 2, and 3, each actual solution has four different representations.

In the second set of test data we constructed words of all combinations of the six digits $a_1, a_2, a_3, a_4, a_5, a_6$. Thus, in this case there is no possible solution for $x_1, x_2, \ldots x_6$, and thus no error correcting code is present.

This method converged more rapidly than we had anticipated. For example, for the first set of test data, when an error correcting code was present, each stage of testing one word eliminated about one-fourth of the remaining possible solutions. This approximately exponential rate of convergence was found when the words were tested in random order, with or without replacement. Convergence was slower when the words were tested in order of increasing value. For the second set of test data, where no error correcting code was present, convergence was also exponentially fast. In 10 separate runs, only six to nine words were needed to be tested before all possible 4096 solutions were eliminated. However, as the number of bases n in each word increases, the total number of solutions also increases exponentially as 4^n, and thus, even for modest n, the computer memory requirements of this method became impractical. Hence we developed a more efficient procedure.

Second method: modified Gaussian elimination

The more efficient procedure is based on the standard Gaussian algorithm of elimination to solve m equations with n unknowns (Fraleigh and Beauregard, 1987; Kolman and Shapiro, 1990). However, the standard algorithm assumes that the coefficients and variables have real values and that the operations use standard arithmetic operations. In our case, the facts that the coefficients and variables are restricted to integer values and that the operations use modular arithmetic introduce a number of changes throughout the procedure. The properties of modular arithmetic in the solution of simultaneous equations are described in the textbook by Stewart (1952). The steps of this modified form of Gaussian elimination are as follows.

The coefficients of the variables in the set of equations in Eq. 5 can be represented as the matrix

$$\begin{array}{ccccc}
b_{11} & b_{12} & b_{13} & \ldots & b_{1n} \\
b_{21} & b_{22} & b_{23} & \ldots & b_{2n} \\
b_{31} & b_{32} & b_{33} & \ldots & b_{3n} \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
b_{m1} & b_{m2} & b_{m3} & \ldots & b_{mn}
\end{array} \quad (11)$$

The solutions for the unknowns x_i are unchanged if one row of the matrix is multiplied by a factor and added to the other rows or if two rows are interchanged. The Gaussian elimination method uses such operations to transform the matrix into the Gaussian form

$$\begin{array}{ccccc}
b'_{11} & b'_{12} & b'_{13} & \ldots & b'_{1n} \\
0 & b'_{22} & b'_{23} & \ldots & b'_{2n} \\
0 & 0 & b'_{33} & \ldots & b'_{3n} \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
0 & 0 & 0 & \ldots & b'_{nn} \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
0 & 0 & 0 & \ldots & 0
\end{array} \quad (12)$$

Note that some of the matrix elements b'_{ij} may be equal to zero. The form of this matrix is that there are one or more rows at the bottom that consist of elements that are all equal to zero. Above those rows of zero elements, each row has more contiguous zero elements on the left than the row above it. For example, if the initial matrix (Eq. 11) has the form

$$\begin{array}{ccccc}
2 & 2 & 0 & 1 & 1 \\
3 & 1 & 3 & 1 & 1 \\
1 & 3 & 3 & 2 & 1 \\
1 & 3 & 1 & 3 & 1 \\
0 & 0 & 2 & 3 & 3 \\
2 & 2 & 2 & 2 & 0
\end{array} \quad (13)$$

then a Gaussian form (Eq. 12) can be shown to be

$$\begin{matrix} 1 & 3 & 1 & 3 & 3 \\ 0 & 0 & 2 & 3 & 3 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix} \tag{14}$$

The values of x_i ($1 \leq i \leq n$) can be computed in a straightforward way from the Gaussian form (Eq. 12). The value of x_n can be directly determined from the simple equation represented by the first row above the rows that are identically zero. Then the value of x_{n-1} is directly determined from the simple equation represented from the row above it. This procedure is iterated until all of the values of x_i are determined. Thus, the Gaussian elimination method consists of first transforming the matrix of the coefficients of the equations into Gaussian form, and then the solutions of the values of x_i can be easily found by back-substitution, working upward through the matrix.

To transform the matrix (Eq. 11) into the Gaussian form (Eq. 12), we begin by examining the element b_{11}. If this element is equal to zero, we select another row with a nonzero element in the first column and interchange it with the original first row. Let us denote this new matrix by $\mathbf{B}^{(0)} = (b_{ij}^{(0)})$. The nonzero element $b_{11}^{(0)}$ is called the "pivot." We then subtract multiples of the top row from those rows further down until all those rows have a zero element in the first column. In the standard Gaussian elimination method, any nonzero element can serve as the pivot. In our case, if possible, we choose a pivot whose value is equal to 1. Note that an element whose value is equal to 3 can be converted to one whose value is equal to 1 by multiplying the row by 3, because $3 \times 3 \equiv 1 \pmod 4$. We choose such an element as the pivot because we can multiply 1 by a factor to produce 0, 1, 2, or 3 and thus use it to zero out any element in the rows further down. We do not want to choose an element whose value is equal to 2 as a pivot, because all of the products of 2 (mod 4) are equal to 0 or 2, and thus they cannot be used to zero out the elements further down that are equal to 1 or 3. If there are no rows that contain elements equal to 1 or 3 in the column position we need for the pivot, then all of those rows must consist of elements that are equal to 0 or 2 in that column, and thus we can choose any row with its element equal to 2 as the pivot because it can be used to zero out the elements in all of the rows further down.

This procedure leaves the first row and the first column unchanged and changes the values of the other elements $b_{ij}^{(1)}$, where

$$b_{ij}^{(1)} = b_{ij}^{(0)} b_{11}^{(0)} - b_{1j}^{(0)} b_{i1}^{(0)} \tag{15}$$

for all $i = 2, 3, \ldots, m$; $j = 1, 2, \ldots, n$; so that the matrix (Eq. 11) has the form

$$\begin{matrix} b_{11}^{(0)} & b_{12}^{(0)} & b_{13}^{(0)} & \ldots & b_{1n}^{(0)} \\ 0 & b_{22}^{(1)} & b_{23}^{(1)} & \ldots & b_{2n}^{(1)} \\ 0 & b_{32}^{(1)} & b_{33}^{(1)} & \ldots & b_{3n}^{(1)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & b_{m2}^{(1)} & b_{m3}^{(1)} & \ldots & b_{mn}^{(1)} \end{matrix} \tag{16}$$

Some of the elements b_{ij} in the matrix (Eq. 16) may be equal to zero. We then iterate this procedure, starting with the second row and second column, interchanging rows if necessary, so that $b_{22}^{(1)}$ is a nonzero element, and then pivoting on that element. As noted above, if possible, an element whose value is equal to 1 is chosen as the pivot. The new elements are given by

$$b_{ij}^{(2)} = b_{ij}^{(1)} b_{11}^{(1)} - b_{1j}^{(1)} b_{i1}^{(1)} \tag{17}$$

for all $i = 3, 4, \ldots, m$; $j = 1, 2, \ldots, n$; so that the matrix (Eq. 11) has the form

$$\begin{matrix} b_{11}^{(0)} & b_{12}^{(0)} & b_{13}^{(0)} & \ldots & b_{1n}^{(0)} \\ 0 & b_{22}^{(1)} & b_{23}^{(1)} & \ldots & b_{2n}^{(1)} \\ 0 & 0 & b_{33}^{(2)} & \ldots & b_{3n}^{(2)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & b_{m3}^{(2)} & \ldots & b_{mn}^{(2)} \end{matrix} \tag{18}$$

We continue to iterate this procedure until the original matrix (Eq. 11) now has the Gaussian form:

$$\begin{matrix} b_{11}^{(0)} & b_{12}^{(0)} & b_{13}^{(0)} & \ldots & b_{1n}^{(0)} \\ 0 & b_{22}^{(1)} & b_{23}^{(1)} & \ldots & b_{2n}^{(1)} \\ 0 & 0 & b_{33}^{(2)} & \ldots & b_{3n}^{(2)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & b_{mn}^{(n-1)} \end{matrix} \tag{19}$$

In this form, all of the elements are equal to zero for the rows $i > n$. There may also be additional rows $i \leq n$ that consist of all elements that are equal to zero. Such rows imply that some variables x_i take on all four possible values of 0, 1, 2, and 3 in the set of solutions. Thus, the matrix (Eq. 19) can be rewritten as

$$\begin{matrix} b_{11}^{(0)} & b_{12}^{(0)} & b_{13}^{(0)} & \ldots & b_{1n}^{(0)} \\ 0 & b_{22}^{(1)} & b_{23}^{(1)} & \ldots & b_{2n}^{(1)} \\ 0 & 0 & b_{33}^{(2)} & \ldots & b_{3n}^{(2)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & b_{nn}^{(n-1)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 0 \end{matrix} \tag{20}$$

where some of the elements b_{ij} in the matrix (Eq. 20) may be equal to zero. In addition, there may be additional rows with elements equal to zero at the bottom and rows above them with leftward contiguous elements equal to zero. Thus, using the elements from the matrix (Eq. 20), the set of

equations is given by

$$b_{11}^{(0)}x_1 + b_{12}^{(0)}x_2 + b_{13}^{(0)}x_3 + \ldots b_{1n}^{(0)}x_n \equiv 0 \pmod 4$$

$$b_{22}^{(1)}x_n + b_{23}^{(1)}x_3 + \ldots b_{2n}^{(1)}x_n \equiv 0 \pmod 4$$

$$b_{33}^{(2)}x_3 + \ldots b_{3n}^{(2)}x_n \equiv 0 \pmod 4$$

$$\ldots \quad \ldots$$

$$b_{nn}^{(n-1)}x_n \equiv 0 \pmod 4. \tag{21}$$

It is now much easier to determine the values of the variables x_i from the transformed set of equations (Eq. 21) than from the original set of equations (Eq. 5). We start with the last equation $b_{nn}^{(n-1)}x_n \equiv 0 \bmod (4)$. If $b_{nn}^{(n-1)} \equiv 1$ or $b_{nn}^{(n-1)} \equiv 3$, then the only solution is $x_n \equiv 0$. If $b_{nn}^{(n-1)} \equiv 2$, then $x_n \equiv 0$ or $x_n \equiv 2$. If $b_{nn}^{(n-1)} = 0$, then $x_n \equiv 0$, $x_n \equiv 1$, $x_n \equiv 2$, or $x_n \equiv 3$. We then use these all of these values of x_n and the penultimate equation to determine the values of x_{n-1}. We continue this process, using back-substitution to determine the solutions for all x_i, keeping track of all the different solution sets.

When one or more sets of x_i ($1 \leq i \leq n$) (other than the trivial solution $x_i \equiv 0$) are found that satisfy the m equations, it means that within each word some bases are linear combinations of other bases and thus a linear block error correcting code may be present. This procedure for detecting linear error correcting codes can tell us whether a linear relationship is present, but it cannot tell us which are the "information" bases and which are the "error check" bases.

APPLICATION OF THE PROCEDURE

As an illustration of the use of this method we used the modified Gaussian elimination method described above to determine whether a linear block error correcting code was present in the base sequence in two genes: the *lac* operon and the structural gene for cytochrome *c*. Sequence data were downloaded from GenBank. We encoded the linear sequence of bases into words of length n, where $3 \leq n \leq 8$. We constructed these words in a number of different ways. We used the method of Mantegna et al. (1994), where consecutive words of length n were obtained by progressively shifting a window of length n by one base for each new word. We also formed words by progressively shifting the window by two to n base pairs. A shift of n bases produces a set of nonoverlapping consecutive words over the DNA sequence. We also varied the starting location on the DNA for the first base, from the first to the $(n - 1)$th base in the gene. We performed the analysis both on the native sequence data and on the same bases shuffled into a random order.

RESULTS

When the number of words, m, analyzed is small, the constraints imposed by the equations (Eq. 5) are not very restrictive. Thus, by chance alone solutions for the variables $x_1, x_2, \ldots x_n$ will be found. To determine whether these solutions arose by chance or represented an error checking code, we therefore determined how the number of nontrivial solutions varied as a function of n and m. These results are presented in Tables 1 and 2. For both large n and m, we found that there were no solutions for $x_1, x_2, \ldots x_n$, and thus an exact error correcting code consistent over large sections of DNA was not present. To determine whether an approximate error correcting code was present over more limited regions of DNA, we compared the number of solutions found for the native sequence and its surrogate formed by randomizing the order of the bases of the native sequence. As shown in Tables 1 and 2, the number of solutions was similar in both the native and randomized sequences for similar values of n and m. This implies that an approximate error correcting code was not present in the genes that we analyzed.

TABLE 1 Solutions from the native base sequence

m	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$
4	0.383					
5	0.209	0.438				
6	0.128	0.253	0.473			
8	0.065	0.088	0.017	0.208	0.539	
9	0.042	0.022	0.157	0.243	0.271	0.361
10	0.013	0.039	0.035	0.012	0.087	0.230
11	0.027	0.032	0.013	0.052	0.077	0.100
12	0.013	0.000	0.025	0.053	0.025	0.052
13	0.000		0.013	0.033	0.000	0.017
14			0.000	0.017		
15				0.000		

The modified Gaussian elimination method was used to determine the solution set of factors $x_1, x_2, \ldots x_n$ of an error correcting code present in m words of n bases each. The fraction of such n by m segments constructed from the native base sequence of the cytochrome *c* gene that were found with nonzero solutions is presented here as a function of n and m.

TABLE 2 Solutions from the randomized base sequence

m	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$
4	0.403					
5	0.223	0.505				
6	0.179	0.327	0.438			
7	0.090	0.225	0.246	0.438		
8	0.051	0.122	0.192	0.312	0.544	
9	0.017	0.021	0.117	0.119	0.287	0.439
10	0.034	0.023	0.035	0.108	0.066	0.206
11	0.000	0.035	0.016	0.019	0.033	0.059
12		0.000	0.021	0.006	0.041	0.028
13			0.000	0.017	0.000	0.014
14				0.000		

The modified Gaussian elimination method was used to determine the solution set of factors $x_1, x_2, \ldots x_n$ of an error correcting code present in m words of n bases each. The fraction of such n by m segments constructed from the bases of the cytochrome *c* gene shuffled into a random sequence that were found with nonzero solutions is presented here as a function of n and m.

DISCUSSION

Error checking codes make it possible to detect and correct errors in digital information. An error correcting code in the base sequence in DNA would make it possible to repair corrupted sequences and increase the fidelity of replicating DNA. Rather than investigating theoretical possibilities of such codes, we have presented here a procedure that makes it possible to determine whether one type of error correcting code is or is not present in the sequence of bases in real DNA. We have only illustrated the use of this procedure by applying it to analyzing the base sequence of two genes. A comprehensive test for the existence of an error correcting code would require an exhaustive application of this procedure to large segments of DNA, including structural genes, introns, promoters, and protein-binding regions. That work is beyond the scope of the present paper. The goal of this paper has been to establish a procedure by which such a search can be conducted and illustrate how it can be carried out.

The procedure presented here is limited to determining the presence of the type of error correcting code called a linear block code. This type of error correcting code is itself quite general and has been used extensively in our present technology. However, there are more general codes based on finite groups (Gallian, 1991) or nonlinear functions. A lack of evidence for the presence of such a linear block code does not rule out the possibility that more complex error correcting schemes may be present in DNA. Such complex codes imply that some bases are complicated functions of the bases in upstream or downstream sequences. This raises the difficult mathematical question: How does one determine whether some symbols in a sequence of symbols are functions of other symbols?

The biological community has thought of DNA as primarily a set of codons that encode amino acids, perhaps with a few regulatory regions here and there. In a more general sense, the issue we raise in this paper is whether there are additional ways that information is encoded in DNA. Perhaps, rather than thinking of DNA as a list of proteins and analyzing the statistical properties of genes and their base sequences, we should think of DNA as a sequence of digital symbols to be analyzed more as a language (Mantegna et al., 1994) and solved as a code breaking problem.

This work was supported in part by National Institutes of Health grant EY06234.

REFERENCES

Dawkins, R. 1987. The Blind Watchmaker. W. W. Norton and Company, New York.

Forsdyke, D. R. 1981. Are introns in-series error-detecting sequences? *J. Theor. Biol.* 93:861–866.

Fraleigh, J. B., and R. A. Beauregard. 1987. Linear Algebra. Addison-Wesley, Reading, MA.

Gallian, J. A. 1991. The mathematics of identification numbers. *Coll. Math. J.* 22:194–202.

Guiasu, S. 1977. Information Theory with Applications. McGraw-Hill, New York.

Kolman, B., and A. Shapiro. 1990. Precalculus: Functions and Graphs, 2nd Ed. Harcourt Brace Jovanovich, New York.

Mantegna, R. N., S. V. Buldyrev, A. L. Goldberger, S. Havlin, C.-K. Peng, S. Simons, and H. E. Stanley. 1994. Linguistic features of noncoding DNA sequences. *Phys. Rev. Lett.* 73:3169–3172.

Rzeszowska-Wolny, J. 1983. Is genetic code error-correcting? *J. Theor. Biol.* 104:701–702.

Stewart, B. M. 1952. Theory of Numbers, 2nd Ed. Macmillan, New York.