Digital signal processing methods for biosequence comparison

Donald C.Benson
Department of Mathematics, University of California, Davis, CA 95616, USA

Received November 29, 1989; Revised and Accepted April 6, 1990

ABSTRACT

A method is discussed for DNA or protein sequence comparison using a finite field fast Fourier transform, a digital signal processing technique; and statistical methods are discussed for analyzing the output of this algorithm. This method compares two sequences of length N in computing time proportional to N log N compared to N^2 for methods currently used. This method makes it feasible to compare very long sequences. An example is given to show that the method correctly identifies sites of known homology.

INTRODUCTION

Major research effort in recent years in the field of nucleic acid and protein sequencing has resulted in massive databases of nucleic acid and protein sequences. Users of these databases often wish to determine the degree of similarity between pairs of these biosequences. This article is concerned with the application of digital signal processing (DSP) methods, especially fast convolution algorithms and the fast Fourier transform, to the problem of searching biosequence databases for similarity. For an exposition of DSP algorithms see (1). DSP methods are currently applied, for example, in the fields of sonar, radar, seismology, tomography, and computerized photographic image enhancement. We believe that DSP methods are also a rich source of techniques for biosequence analysis. To illustrate the possibilities, this paper puts forth a particular biosequence application.

The most rapid current methods of sequence comparison are very rapid, and there is probably not a generally perceived need among molecular biologists for faster methods. However, this situation is likely to change as the size of the databases increases. It seems likely that either there will be need to screen sequences which are orders of magnitude longer than those currently studied, or there will be a need to do simultaneous screening of a large number of sequences.

DSP methods do not deliver exactly the same kind of information as standard methods. We believe that it is worthwhile to explore what kind of information can be obtained and to develop statistical methods to analyze this information. DSP methods are *not* capable of simply accelerating the speed of algorithms that are currently used for sequence analysis. There are steps in the currently used algorithms which seem impossible to replicate using DSP methods. For example, DSP methods seem to be unable to distinguish between consecutive and non-consecutive matches. On the other hand, certain information can be obtained much more rapidly by DSP methods than by standard methods. For example, the method described here produces a list of the total number of exact matches in each alignment of two sequences. This abundance of information requires statistical analysis in order to determine which alignments are significant. While we cannot compare DSP methods with standard methods in accomplishing precisely the same tasks, we shall give an example which verifies that the DSP method discussed below identifies the 'correct' alignments of sequences.

Using DSP methods, we have developed an experimental computer program for DNA sequence comparison. We do not have a full-featured, completely documented program suitable for public distribution which we can offer as an alternative for current sequence comparison programs. The current capabilities of the program are too special to serve as a tool for molecular biologists. The program is intended to research the potential of DSP methods. The program compares two DNA sequences, each of length 1024 or less. A DSP method of sequence comparison is inherently faster than current methods because an N log N method inevitably gains a speed advantage over an N^2 method if N, i. e. the length of the sequences, is large enough; however, $N = 1024$ may not be large enough to demonstrate an advantage over current methods. Presently, our greatest concern is not speed but correctness. We wish to show that it is possible to devise a DSP program that satisfactorily identifies sites of similarity.

The application of the fast Fourier transform to biosequence similarity searches has been previously discussed by Felsenstein et al. in (2). We believe that they underestimate the potential usefulness of this method. They state certain problems which we discuss below.

a. *The method cannot detect insertions or deletions.* If insertions or deletions are present, more than one alignment exhibits an unusual number of matches. In this article, we introduce statistical methods which answer this objection by developing statistical tests to determine if a significant number of alignments exhibit an unusually large number of matches.

b. *The method cannot determine whether matches are consecutive or not.* On the other hand, methods which select on the basis of contiguity miss significant alignments in which matches are separated by substitutions. DSP methods have not been researched sufficiently to say that there is no way around this problem.

c. *The method involves cumbersome calculations with complex floating-point numbers.* This difficulty is avoided by using *finite field* fast Fourier transforms. This avoids the use of complex numbers. Only integer arithmetic is involved, and a large proportion of the multiplications in the algorithm can be reduced to bit shifts which are much more rapidly executed on the computer than true multiplications.

The problem of searching for similarities in strings of characters is of general interest in the field of computer science. The problem of searching for *exact* matches is considered in (3) and (4). Methods of biosequence comparison seek a degree of similarity rather than an exact match. See (5) for a review of current methods for biosequence comparison. The most powerful methods, the dynamic programming algorithms, model the process of biological evolution by constructing an optimal series of transformations that carry the one sequence through a chain of intermediate sequences ending finally with the second sequence (6). Each transformation contributes to a similarity score. Dynamic programming methods are capable of uncovering subtle relationships; however, these methods are too slow for large-scale database searches.

Rapid methods for sequence comparison generally count the number of matches for each alignment of the sequences. For example, the *dot-matrix* method (7) produces a rectangular array of dots; a dot appears in the ith column of the jth row if item i in the first sequence matches item j in the second sequence. Dots on a particular NW to SE diagonal line represent matches in the corresponding alignment of the sequences.

Some rapid methods count matches with a suitable weighting factor if a certain matching criterion is met. The matching criterion, for example, may require that, for some fixed integer k, k contiguous characters of the two sequences are identical (8). A weighting factor may give a higher score to matches that are considered less likely (9). Dynamic programming can be used after an initial screening by means of more rapid methods (9).

In order to use DSP methods, the character sequences are translated into suitable numerical sequences. We obtain similarity information from the numerical sequences by convolution, an arithmetic process. The convolutions are then computed by fast DSP methods. This similarity information is then interpreted by statistical methods.

For comparison of two sequences of length N, the best current methods use algorithms that require, at least in the worst case, computation time proportional to N^2. We present here a method with computation time proportional to N log N. This greatly extends our ability to compare very long sequences.

The key to our method is the use of a variant of the *fast Fourier transform* (FFT). See (1), especially chapter 6. In this application the standard FFT has the disadvantage of replacing mere character-by-character comparisons with rather complex floating-point arithmetic. This problem is eliminated by doing the arithmetic in a suitable finite field of integers instead of the field of complex numbers. Improvement results because integer arithmetic is faster than floating-point arithmetic and because in a suitably chosen finite field most of the multiplications of the FFT algorithm become bit shifts which are computed much faster than true multiplications.

Although our methods also apply to protein sequences, for simplicity we refer below only to DNA sequences.

Our method can be considered a variant of the dot-matrix method described above. However, the output of our method consists of the total number of matches for each character (A, C, G, and T for DNA sequences) for each alignment of the two sequences (without regard for contiguity). For each alignment, the sum of the total number of matches is equal to the number of dots in the corresponding diagonal of the dot-matrix. For large sequences, this information can be obtained very much faster than the dot-matrix.

Our technique differs from those commonly used in that it does not distinguish between contiguous and noncontiguous matching characters. The advantage is that our method is more sensitive in detecting string transformations by substitution. The disadvantage is that the method is not able to give greater significance to matches when they are contiguous. Our technique shares with all rapid methods a difficulty in detecting string transformations by insertion or deletion. Nevertheless, when an insertion or deletion occurs, similarity may be detected by observing a large number of matches in two or more separate alignments. The overriding advantage of the method is speed. This method is intended as a screening method. Supplemental dynamic programming methods can be used to confirm and elaborate probable sites of similarity.

In the next section, we define the FFT and develop the basic comparison algorithm.

In the last section, we discuss the statistical analysis of the output of the comparison algorithm. In particular, we analyze a test of similarity using extreme values.

THE COMPARISON ALGORITHM

Let

$$\mathbf{v} = \{v_j,\ j = 0,\ldots,\ n-1\} \tag{2.1}$$

be an vector of complex or real numbers. The *discrete Fourier transform* of $\mathbf{v}$ is defined to be the complex vector $\mathbf{V}$ of length n given by

$$V_k = v_0 w^{0\cdot k} + v_1 w^{1\cdot k} + \ldots + v_{n-1} w^{(n-1)\cdot k},\ 0 \le k \le n-1 \tag{2.2}$$

where w is the primitive nth root of unity, $\exp(-2\pi i/n)$. The number n is called the *block-length*.

We define the *inverse Fourier transform* of a vector $\mathbf{U}$ to be the vector $\mathbf{u}$ given by

$$u_j = (U_0 w^{0\cdot j} + U_1 w^{1\cdot j} + \ldots + U_{n-1} w^{(n-1)\cdot j})/n,\ 0 \le j \le n-1 \tag{2.3}$$

The inverse Fourier transform is a true inverse in the sense that for any vectors $\mathbf{v}$ and $\mathbf{V}$ related according to (2.1) and (2.2), $\mathbf{v}$ is equal to the inverse transform of $\mathbf{V}$.

Our application of the Fourier transform uses the following fact, known as the convolution theorem. For any pair of real or complex n-dimensional vectors $\mathbf{f}$ and $\mathbf{g}$ define their convolution to be the vector $\mathbf{e}$ given by

$$e_j = f_{\{j-0\}} g_0 + f_{\{j-1\}} g_1 + \ldots + f_{\{j-n+1\}} g_{n-1},\ 0 \le j \le n-1 \tag{2.4}$$

where

$$\{k\} = \begin{cases} k \text{ if } k \ge 0 \\ k + n \text{ otherwise.} \end{cases}$$

The convolution of $\mathbf{f}$ and $\mathbf{g}$ is denoted $\mathbf{f} * \mathbf{g}$.

Let $\mathbf{F}$ and $\mathbf{G}$ be vectors of length n. We define $\mathbf{F} \cdot \mathbf{G}$ to be the vector $\mathbf{E}$ given by

(2.5) $$E_k = F_k G_k,\ 0 \leq k \leq n-1$$

The following assertion is called the *convolution theorem*. Let **F** and **G** be the Fourier transforms of **f** and **g** respectively. Then the Fourier transform of **f** * **g** is **F** · **G**. Equivalently, the inverse Fourier transform of **F** · **G** is **f** * **g**.

For sufficiently large n, use of the convolution theorem is faster than the direct method for computing the convolution of **f** and **g**. Note that it is easier to compute **F** · **G** than **f** * **g**. In fact, **F** · **G** requires n multiplications and one addition whereas **f** * **g** requires n^2 multiplications and n additions. Moreover, the fast Fourier transform (FFT) and the inverse FFT require of the order of n log n multiplications and additions. More precisely, there are constants M and A not depending on n such that the number of multiplications is less than Mn log n for all n, and the number of additions is less than An log n for all n. The standard way of expressing this fact is to say that the number of multiplications and additions are each O(n log n). As a consequence of the above, for sufficiently large n, it is faster to compute the FFT's **F** and **G**, then **F** · **G**, and finally the inverse FFT of **F** · **G**, than it is to compute **f** * **g** directly.

It is advantageous to replace the complex number field in the definitions above with a finite field, the field of integers modulo a prime number p. Let w be a number between 0 and p. The order of w is defined to be the smallest integer n such that $w^n \equiv 1 \bmod p$. For vectors of size n of integers, we may define a Fourier transform, and an inverse Fourier transform by reinterpreting the formulas (2.1−2.3). All equalities are interpreted as congruence modulo p. We use the w we have just discussed instead of the complex nth root of unity. The convolution theorem holds exactly as before. We obtain exactly the same results computing convolutions using this version of the convolution theorem provided all the numbers involved are nonnegative integers less than p. We shall call this type of Fourier transform a *finite field Fourier transform* (F^3T). There are also fast counterparts of these called *finite field fast Fourier transforms* (F^4T). We have written a computer program for a F^4T using p equal to the Fermat prime $2^{16} - 1 = 65{,}537$ and n equal to $2^{10} = 1024$. See (1) pp. 180−182. With apologies for alliterative excess (*Fermat* finite field fast Fourier transform), we will name this transform F^5T. For F^5T, most of the multiplications reduce to bit shifts which are much faster than true multiplications. Using an IBM-AT microcomputer our program computes F^5T's of four numerical sequences of length 1024 simultaneously in about two seconds.

We now give a method of translating sequences of characters into numerical vectors. We will then show how convolutions of these vectors bear on the problem of sequence comparison. For definiteness we consider the case of nucleic acid sequences which consist of strings of length L ≤ n of the four characters A, C, G, T. Note that it is permissible for the length L of the sequence to be less than the block-length n. Let **a** be the vector given by

(2.6) $$a_j = \begin{cases} 1 \text{ if } j \leq L \text{ and the jth character is an A} \\ 0 \text{ otherwise, } j \leq n. \end{cases}$$

Note that if the length of the sequence L is greater than the block-length n, then $a_j = 0$ for j such that $L < j \leq n$.

Similarly, define the vectors **c**, **g**, and **t** with respect to the incidence of the characters C, G, and T. Let **A**, **C**, **G**, and **T** be the respective FFT's.

Consider the problem of comparing this sequence with a second sequence of length M. *The sequences are compared by totaling the number of matching paired characters under every possible alignment.*

First consider the case that L = M = n. When we compare the first sequence with an offset of the second sequence, there are characters at the beginning of the first sequence and at the end of the second sequence which are not paired with any character. For the alignment shown below n is 15 and the number of matches is 4.

(2.7)
```
ATCACAAGTACCTTA
   TTGTTAACTAACGTA
```

This type of comparison is called *linear comparison*. There is another type called *cyclic comparison* in which we consider the first character of each sequence to be successor of the last character. In other words, in cyclic comparison we consider each of the sequences to be circular. Applying this type of comparison to the above alignment (2.7) adds one more match making a total of 5 instead of 4.

(2.8)
```
ATCACAAGTACCTTA
ACGTATTGTTAACTA
```

In effect, we are lumping two different alignments in the original formulation, namely the alignment (2.7) and

(2.9)
```
              ATCACAAGTACCTTA
TTGTTAACTAACGTA
```

If one or both of the sequences is shorter than the block-length, then the sequences must be padded with blanks. Consider altering the above example so that n = 15, L = 9, and M = 7. Blanks are indicated with '*'.

(2.7′)
```
ATCACAAGT******
   TTGTTAA*******
```

For this example, cyclic comparison and linear comparison produce exactly the same number of matches. This is true not only for this particular alignment, but for any alignment of these two sequences, because all of the newly introduced pairs in the cyclic comparison have at least one member which is a blank. (Pairing a blank with a blank does not count as a match.) In general the cyclic comparison is the same as the linear comparison provided that L + M ≤ n + 1. Although cyclic comparison is more suited to the convolution method and has a simpler statistical analysis, linear comparison is more natural for the biosequence application. The convolution method can be used for linear comparisons provided that the above inequality is satisfied.

Suppose L = M = n and that we wish to make a cyclic comparison of the sequence described by the vectors **a**, **c**, **g**, and **t**, with a second sequence of the same length. Define **a′**, **c′**, **g′**, and **t′** to be the character vectors for A, C, G, and T, as above, but for the second sequence of characters *in reverse order*.

Note that the jth components (0 ≤ j ≤ n−1) of the vectors

(2.10) $$\mathbf{A} = \mathbf{a} * \mathbf{a}',\ \mathbf{C} = \mathbf{c} * \mathbf{c}',\ \mathbf{G} = \mathbf{g} * \mathbf{g}',\ \mathbf{T} = \mathbf{t} * \mathbf{t}'$$

are equal the number of times an A (respectively C, G, T) in the first sequence matches an A (respectively C, G, T) in the alignment with the second sequence in which the first character of the first sequence is paired with the (n − j − 1)th character of the second sequence. In other words, the convolution vectors consist of the total number of single-character matches. We will

call the convolutions **A, C, G,** and **T** *match vectors*. The FFT is used to compute the match vectors.

These methods can be easily adapted to the computation of weighted averages instead of mere totals. Particularly for protein comparisons, weights may be taken large or small according as the matches are scarce or widespread, and, for non-identical pairs, weights are assigned to evaluate the replacement likelihood. The PAM250 matrix (10) provides weights for protein comparisons.

STATISTICAL ANALYSIS

The match vectors discussed in the previous paragraph can be computed quite rapidly using, for example, F^5T. We now discuss the problem of the statistical analysis of the match vectors. In particular, we want to determine if a particular set of match vectors represents a significant similarity between the two sequences from which they were derived.

Statistical tests can be devised to detect similarity. We discuss two types of tests, the first using *measures of dispersion*, the second using *extreme values*. A measure of dispersion tests for *average* similarity over all alignments. Extreme values identify the most significant alignments. Current search techniques generally adopt the second view of similarity.

Mean and Variance

For simplicity we first assume that the two sequences to be compared are of the same length. The mean of a match vector is determined completely by the number of characters in each of the two sequences and the length. In fact, let a and a′ be the number of A's in the first and second sequences, respectively, and let N be the common length of the sequences. Note that given a character in the first sequence and a character in the second sequence, there is exactly one alignment in which these two characters are matched. There are N alignments, and the number of AA pairs possible is aa′. Thus the mean number of AA pairs per alignment is

$$\mu_A = aa'/N. \tag{3.1a}$$

Similarly, the mean number of CC's, GG's, and TT's are, respectively,

$$\mu_C = cc'/N,\ \mu_G = gg'/N,\ \mu_T = tt'/N. \tag{3.1b-d}$$

Every sample has the same mean. This surprising property is possible because the components of the match vector are not statistically independent.

On the other hand, if a measure of dispersion, such as the variance, is large, it means that some alignments have a relatively large number of matches. This justifies taking a measure of dispersion as a measure of similarity of the sequences.

There is more than one possible probability model for non-similarity. Such a model could be based empirically on properties observed for actual biosequences. Consequences of various models of non-similarity are discussed in (11). Here we will base our further development on a simple model which represents a first order approximation. Our model of non-similarity is that sequences with a given proportion of A's, C's, G's, and T's are generated by shuffling randomly the order of the characters while keeping fixed the total number of each character. Under this hypothesis, the statistical distribution of the number of matches for any particular alignment is hypergeometric. (See, for example, (12) pp. 179 and 218) However, the number of matches for different alignments of the same two sequences are obviously *not* statistically independent. Nevertheless, the means are given by (3.1a-d), and the variances are given by

$$\sigma^2_A = \frac{aa'(N-a)(N-a')}{N^2(N-1)} \quad \sigma^2_C = \frac{cc'(N-c)(N-c')}{N^2(N-1)}$$

$$\sigma^2_G = \frac{gg'(N-g)(N-g')}{N^2(N-1)} \quad \sigma^2_T = \frac{tt'(N-t)(N-t')}{N^2(N-1)} \tag{3.2a-d}$$

In order to compare sample variances of sequences with differing proportions of A, C, G, and T, we use (3.1) and (3.2) to standardize the match vectors variables **A, C, G,** and **T**.

$$\begin{aligned} A^*_i &= (A_i - \mu_A)/\sigma_A \\ C^*_i &= (C_i - \mu_C)/\sigma_C \\ G^*_i &= (G_i - \mu_G)/\sigma_G \\ T^*_i &= (T_i - \mu_T)/\sigma_T \\ &(0 \le i \le n-1) \end{aligned} \tag{3.3a-d}$$

A complication arises if, as in linear comparison, one or both of the sequences are shorter than the block-length. In that case formula (3.1) needs to be reinterpreted because, for any alignment, matches can occur only in the region in which the sequences overlap. The length N in formula (3.1) is the length of the overlap. Similarly, a, a′, c, c′, etc., represent the numbers of A's, C's, etc. for each of the sequences *in the overlapping region*. This means that the variables N, a, a′, etc., may have different values for different alignments. This additional information can be obtained using the convolution method. Formulas (3.1), (3.2), and (3.3) have obvious reinterpretations if the sequences are shorter than the block-length.

A significantly high variance of the standardized variables can be taken as an indication of significant similarity. A chi-square test could be used in the case of independent trials. If the non-similarity hypothesis specified that each alignment were shuffled independently, then we would have independence and we could use chi-square. Unfortunately, this hypothesis is inconsistent with the manner in which our data is actually generated. Nevertheless, ignoring this difficulty, we have used chi-square on Genbank sequence data and have found that the values of chi-square correlate with known cases of sequence similarity, but it is difficult to interpret the meaning of chi-square confidence levels.

Screening by Extreme Values

Researchers are currently more interested in pinpointing the few most significant alignments rather than analyzing the average significance of a batch of alignments. In this section we consider a particular screening procedure for similarity which implements this concern. If there are alignments for which the number of matches exceeds a certain critical value, Z, then we flag those alignments for further analysis, say using a Needleman-Wunsch algorithm (6). In using the FFT we examine alignments in batches of a certain size which we denote n. We say that a *type 1 error* occurs if in screening a batch of n alignments we overlook significant matches, and a *type 2 error* occurs if we flag one or more alignments which are not significant. The probabilities of the two types of errors depend on the choice of critical value, the lengths of the sequences, etc. In this section we analyze the dependence of type 1 and type 2 errors on the various parameters.

We begin with a discussion of type 2 errors. Our analysis depends on extreme value theory. Let $Z_1, Z_2, \ldots$ be a sequence of independent and identically distributed random variables.

Classical extreme value theory (13) determines the asymptotic distribution of

$$M_n = \max(Z_1, Z_2, \ldots, Z_n)$$

as n tends to infinity. It has recently been shown (14) that the principal results of this theory remain valid under certain conditions of stationarity and mild dependence. These results have implications for the biosequence comparison problem, at least in the case in which one of the sequences being compared is much longer than the other.

In order to make use of asymptotic results, we must idealize the comparison problem. We suppose that one of the sequences being compared is infinite in length and the other has length M.

In order to study type 2 errors, we test a model of non-similarity in which the underlying biosequences are sequences independent random variables. Let z_i be the total number of matches for the ith alignment, and let $Z_i = (z_i - \mu_i)/\sigma_i$ where μ_i and σ_i^2 are the mean and variance of z_i; i. e. Z_i is the standardization of z_i. The assumption of *stationarity* needed for the extension (14) of extreme value theory requires that the joint distributions of

$$Z_{i_1}, Z_{i_2}, \ldots, Z_{i_k} \tag{3.4}$$

and

$$Z_{i_1+m}, Z_{i_2+m}, \ldots, Z_{i_k+m}$$

are the same for any choice of the positive integers k, i_1, i_2, ... , i_k, m. This assumption is natural for the application under consideration. We make the approximation that (3.4) has a k-dimensional normal distribution for any choice of k, i_1, i_2, ... i_k. Since the sequence has length M, it is not unreasonable to assume further that the covariance $\mathrm{cov}(Z_i, Z_j)$ is equal to zero for all i and j such that $i < j + M$. This covariance assumption and the assumption of stationarity imply (13, p. 84) that for any real x, the probability of the inequality

$$a_n(M_n - b_n) \leq x \tag{3.5}$$

tends to $\exp(-e^{-x})$ as $n \to \infty$, where

$$a_n = (2 \log n)^{1/2}, \tag{3.6}$$

$$b_n = (2 \log n)^{1/2} - \tfrac{1}{2}(2 \log n)^{-1/2}(\log \log n + \log 4\pi).$$

The foregoing can be interpreted as giving the asymptotic behavior of the probability of a type 2 error. In fact, if z is the critical value, we simply put $x = a_n(Z - b_n)$ in (3.5).

To illustrate the analysis of type 1 errors, we make much more restrictive assumptions. Assume that there is a 'significant' alignment in which L of the M characters match. We assume that the remaining M − L pairs may be considered independent Bernoulli trials with probability p of a successful match. The distribution of these matches is binomial with mean $\mu = p(M - L)$ and variance $\sigma^2 = p(1 - p)(M - L)$. Suppose that the critical value for the number of matches is z, and $Z = (z - \mu)/\sigma$ is the standardization of z. The type 1 error is the eventuality that the screening will miss the significant alignment of length L, i. e. that the number of matches among the remaining M − L is less than z − L. Approximating the binomial with the normal distribution we have that the probability of the type 1 error is equal to $Q(Z - L/\sigma)$ where Q is the left tail of the normal probability distribution:

$$Q(x) = (2\pi)^{1/2} \int_{-\infty}^{x} \exp(-\tfrac{1}{2}t^2)\,dt.$$

Table. 1. Type 1 and type 2 errors for sequence of length 256.

Critical Value	Probabilities of type 1 error Match length=L				Type 2 error No. of alignments=n	
z	L=30	L=40	L=50	L=60	n=1024	n=4096
81	0.1991	0.0205	0.0005	0.0000	1.0000	1.0000
82	0.2447	0.0297	0.0009	0.0000	0.9992	1.0000
83	0.2954	0.0420	0.0015	0.0000	0.9839	1.0000
84	0.3505	0.0580	0.0024	0.0000	0.9105	1.0000
85	0.4089	0.0786	0.0040	0.0000	0.7559	0.9991
86	0.4694	0.1044	0.0063	0.0001	0.5613	0.9796
87	0.5306	0.1357	0.0098	0.0001	0.3821	0.8848
88	0.5911	0.1729	0.0149	0.0003	0.2452	0.6987
89	0.6495	0.2160	0.0222	0.0005	0.1515	0.4861
90	0.7046	0.2648	0.0321	0.0009	0.0915	0.3089
91	0.7553	0.3187	**0.0456**	0.0015	**0.0546**	0.1854
92	0.8009	0.3767	0.0632	0.0025	0.0322	0.1076
93	0.8410	0.4376	0.0857	0.0042	0.0190	0.0612
94	0.8754	0.5000	0.1137	0.0067	0.0111	0.0345
95	0.9042	0.5624	0.1478	0.0105	0.0065	0.0193
96	0.9278	0.6233	0.1881	0.0160	0.0038	0.0107
97	0.9466	0.6813	0.2345	0.0239	0.0022	0.0060
98	0.9614	0.7352	0.2867	0.0348	0.0013	0.0033
99	0.9726	0.7840	0.3438	0.0495	0.0008	0.0018
100	0.9809	0.8271	0.4046	0.0688	0.0004	0.0010

These results are illustrated in Table 1 which lists type 1 and type 2 errors for p = ¼, M = 256, and for various choices of L and n. For the type 2 error model we assume $\mu = pM$ and $\sigma^2 = p(1 - p)M$ for all z_i, and we use the asymptotic formulas (3.5) and (3.6) to estimate the probabilities. For example, using a critical value of 91 we incur type 1 and type 2 errors of about 5% in searching for matching subsequences of length 50 of a sequence of length 256 using batches of size 1024.

	BASES OF H	MAX MATCHES	OFFSET	ERROR PROBABILITY
1.	1− 1024	94	91	0.0111
2.	1001− 2024	92	777	0.0322
3.	**2001− 3024**	**141**	**374**	$\mathbf{1.2\times10^{-13}}$
4.	3001− 4024	98	209	0.0013
5.	4001− 5024	88	960	0.2452
6.	5001− 6024	88	793	0.2452
7.	**6001− 7024**	**131**	**136**	$\mathbf{2.6\times10^{-11}}$
8.	**7001− 8024**	**131**	**112**	$\mathbf{2.6\times10^{-11}}$
9.	8001− 9024	90	709	0.0915
10.	9001−10024	93	300	0.0190
11.	**10001−11024**	**188**	**325**	$\mathbf{1.3\times10^{-24}}$
12.	11001−12024	95	71	0.0065

Table 2. Comparison of M (MUSHBA.ROD 596−851) with H (HUMHBA4.PRI). BASES OF H lists successive subsequences of H. MAX MATCHES is the maximum number of exact matches over all alignments of M with the indicated subsequence of H. OFFSET is the number of bases of offset that defines the alignment with the maximum number of matches. ERROR PROBABILITY is the probability of a type 2 error, i. e. the probability of the listed number of matches assuming Bernoulli trials with p = ¼. The boldface entries are alignments which exhibit the expected homology between mouse and human alpha globin exon 2. See text.

As an example of this type of comparison, we compare two GenBank sequences. The sequence MUSHBA.ROD (1441 bp) according to GenBank documentation is 'Mouse alpha globin gene, complete cds' and HUMHBA4.PRI (12847 bp) is 'Human alpha globin psi-alpha-1, alpha-2 and alpha-1 genes, complete cdns.' The subsequence MUSHBA.ROD 622−826 consists of

alpha globin, exon 2. Enlarging this subsequence slightly, we compare MUSHBA.ROD 596–851, which we denote M, with all of HUMHBA4.PRI, which we denote H. As our computer implementation handles sequences of length at most 1024, we compare M successively with H 1–1024, H 1001–2024, ..., H 11001–12024. GenBank documentation lists, in part, the following subsequences of H:

a. 6915– 7119 hba2 alpha globin, exon2
b. 10726–10930 hba1 alpha globin, exon2
c. 2697– 2881 pseudo-hba1 alpha globin, exon2.

Evidence of the expected homology to M of these subsequences is shown very clearly in this experiment. We summarize the results in the Table 2. For each of the 12 subsequences of H we list the alignment with the greatest number of matches. The alignments are identified by their offsets. The H subsequences **3**, **7**, **8**, and **11** contain all or part of the above variants **c**, **a**, **a**, and **b**, respectively, of human alpha globin exon 2. Note that the above statistical test shows *extremely* small probabilities of type 2 errors associated with these subsequences. In other words it is almost certain that these subsequences contain significant matches with exon 2 of mouse alpha globin. Note that variant **a** of exon 2 overlaps subsequences **7** and **8**. This fact does not diminish the effectiveness of the statistic in identifying the homology; both subsequences are flagged with a very low type 2 error. Note that the offsets indicated for subsequences **7** and **8** differ by 24 which is exactly the amount of overlap of the two subsequences. This confirms the fact that, as expected, the alignments **7** and **8** are actually different portions of the same alignment, namely the alignment which matches human alpha globin exon 2 variant **a** with mouse alpha globin exon 2.

Table 2 exhibits other error probabilities that are indicative of sites of homology. In fact, all of the probabilities are less than ¼. In any case, these probabilities are indicative of more subtle homology. For example, the probable homology in the alignment listed for sequence **4** may be due, at least in part, to bases 3949–4023 of HUMHBA4.PRI which match 39/75 (= 52%) of the corresponding bases in the specified alignment with MUSHBA.ROD. In the following listing of HUMHBA4.PRI 3949–4023, capital letters indicate matches in the specified alignment with MUSHBA.PRI:

(3.7)

```
3949 CCtGTGCTGC   caGCaACtTC   tggaAaCgtC   CCtGTcCCcg   GTgctgaagt
3999 cctGgaaTcC   ATGCtgggAA   GtTGCa
```

GenBank documentation does not give any reason to suspect homology at this site. In order to estimate the probability that this much matching (39/75 = 52%) occurred by chance, we can reuse the same extreme value theory that we used above in computing the probabilities of type 2 errors. Suppose that we divide H and M into subsequences of the same length as the subsequence exhibited above. The number of subsequences of H is 12847/75 = 171.29 and of M is 256/75 = 3.41. The total number of possible comparisons is the product of these numbers which rounds off to 584. If we count the number of matches for each of the 584 alignments of subsequences, each of length 75, and if we assume that each alignment of the subsequences consists of Bernoulli trials with p = ¼, what is the probability that at least one of the alignments will contain at least as many matches (39) as (3.7)? If this probability is low enough, then we conclude that the matching exhibited in (3.7) did not occur by chance. Using (3.5) and (3.6), we find that the probability of at least one subsequence of length 75 with 39 or more matches is only 0.00016. We conclude that (3.7) probably represents a significant match.

ACKNOWLEDGEMENTS

We thank Dr. William J. Black, Stanford University, for helpful suggestions and for providing GenBank DNA sequences. We also thank the referees for valuable suggestions.

REFERENCES

1. Blahut, Richard E. (1985) Fast Algorithms for Digital Signal Processing. Addison-Wesley, Reading, Massachusetts.
2. Felsenstein, J., S. Sawyer, and R. Kochin, (1982) Nucleic Acids Research **10**, 133–139.
3. Knuth, D. E., J. H. Morris, V. R. Pratt (1977) SIAM J. Comp. **6:2** (June), 323–350.
4. Boyer, R. S., and J. S. Moore, (1977) CACM **20:10** (October), 762–772.
5. Waterman, Michael S. (1984) Bulletin of Mathematical Biology **46**, 473–500.
6. Needleman, S. B., and C. D. Wunch (1970) J. Mol. Biol. **48**, 444–453.
7. Maizel, J. V., and R. P. Lenk (1981) Proc. Natl. Acad. Sci. USA **78**, 7665–7669.
8. Wilbur, W. J. and D. J. Lipman (1983) Proc. Natl. Acad. Sci. USA **80**, 726–730.
9. Lipman, D. J., and W. R. Pearson (1985) Science **227**, 1435–1441.
10. Dayhoff, M. (1978) Atlas of Protein Sequence and Structure. National Biomedical Research Foundation, Silver Spring, MD.
11. Lipman, D. J., W. J. Wilbur, T. F. Smith, and M. S. Waterman (1984) Nucleic Acids Research **12**, 215–226.
12. Parzen, Emanuel (1960) Modern Probability Theory and its Applications. John Wiley, New York.
13. Gumbel, E. J. (1958) Statistics of Extremes. Columbia University Press, New York.
14. Leadbetter, M. R. (1978) In Rosenblatt, M. (ed.) Studies in Mathematics, Vol. 18 (Studies in Probability Theory), The Mathematical Association of America, Washington, D. C., pp. 46–110.